SCALABLE AND RELIABLE INFERENCE FOR PROBABILISTIC MODELING

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Ruqi Zhang

August 2021

SCALABLE AND RELIABLE INFERENCE FOR PROBABILISTIC MODELING

Ruqi Zhang, Ph.D.

Cornell University 2021

Probabilistic modeling, as known as probabilistic machine learning, provides a principled framework for learning from data, with the key advantage of offering rigorous solutions for uncertainty quantification. In the era of big and complex data, there is an urgent need for new inference methods in probabilistic modeling to extract information from data effectively and efficiently.

This thesis shows how to do theoretically-guaranteed scalable and reliable inference for modern machine learning. Considering both theory and practice, we provide foundational understanding of scalable and reliable inference methods and practical algorithms of new inference methods, as well as extensive empirical evaluation on common machine learning and deep learning tasks.

Classical inference algorithms, such as Markov chain Monte Carlo, have enabled probabilistic modeling to achieve gold standard results on many machine learning tasks. However, these algorithms are rarely used in modern machine learning due to the difficulty of scaling up to large datasets. Existing work suggests that there is an inherent trade-off between scalability and reliability, forcing practitioners to choose between expensive exact methods and biased scalable ones. To overcome the current trade-off, we introduce general and theoretically grounded frameworks to enable fast and asymptotically correct inference, with applications to Gibbs sampling, Metropolis-Hastings and Langevin dynamics.

Deep neural networks (DNNs) have achieved impressive success on a variety

of learning problems in recent years. However, DNNs have been criticized for being unable to estimate uncertainty accurately. Probabilistic modeling provides a principled alternative that can mitigate this issue; they are able to account for model uncertainty and achieve automatic complexity control. In this thesis, we analyze the key challenges of probabilistic inference in deep learning, and present novel approaches for fast posterior inference of neural network weights.

BIOGRAPHICAL SKETCH

Ruqi Zhang was born and raised in Tianjin, China. She received her B.S. in Mathematics from Renmin University of China in 2016. After that, she started her Ph.D. in Statistics at Cornell University under the supervision of Chris De Sa. During her Ph.D., she worked at Microsoft Research Cambridge and Microsoft Research New England as a research intern.

To my parents.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor Chris De Sa, for giving me so much encouragement, advice and support. He always believed in me and helped me in all the ways he could. As one of his first students, we often discussed tirelessly to mold a research idea. His enthusiasm for doing great research significantly inspired me to continue my career in academia. Without his guidance, this thesis would not be possible.

I am very grateful to my mentor Andrew Gordon Wilson, who introduced me to Bayesian deep learning. Talking with him was always inspiring and his view on Bayesian methods still has great influence on me.

I would like to express my gratitude to my amazing collaborators. Yingzhen Li was always available to help, providing me so much feedback and support. Thanks to Cheng Zhang and Chunyuan Li for their valuable guidance in research. I would also like to acknowledge my co-authors including A. Feder Cooper, Changyou Chen, Sam Devlin, Jianyi Zhang; and my internship mentors at Microsoft Research Nicolo Fusi and Rishit Sheth.

I am thankful to my committee members, Thorsten Joachims and Giles Hooker, for their time and effort to serve on my committee.

Finally, I would like to thank my friends for making this journey enjoyable, and my parents for their unwavering support and love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Part I

Motivation and Background

CHAPTER 1

INTRODUCTION

The power of data propels the development of society, science and technology. The newest generation of machine learning systems is largely driven by the explosion of data in recent decades. While data are key to solving problems, they are useless until we turn them into predictions and decisions. Therefore, there is an urgent need for innovative modeling tools that can effectively and efficiently extract information from big data.

Probabilistic modeling takes the intrinsic uncertainty of the real world into consideration, offering a principled way to quantify uncertainty about predictions and decisions. It has been widely used across many different domains, ranging from social science [114, 54] and neuroscience [100], to natural language processing [51] and image generation [130].

In probabilistic modeling, the key algorithmic problem is probabilistic *inference*. Inference refers to the process of using information from data to reason unknown properties, which transfers our prior knowledge to posterior knowledge given data evidence. Inference from big data faces several challenges: (1) the posterior distribution is always complex or multimodal, adding difficulties to inferring accurately; (2) most inference methods become unacceptably slow or even intractable when used with big data, due to the complexity dependency of the dataset size; (3) inference with non-convex objective functions—as is the case when working with big data—is poorly understood, leading to unexplained and unpredictable empirical results.

We will analyze these challenges and provide our solutions in this thesis. First, we will introduce general frameworks to enable theoretically-guaranteed scalable

and reliable inference, with applications to Gibbs sampling, Metropolis-Hastings and Langevin dynamics. In addition to proposing new algorithms, we provide theoretical analysis of the proposed algorithms, as well as theoretical understanding about how well the existing methods are doing and how well we can possibly hope to do. Then we focus on deep probabilistic modeling whose posterior is complicated and highly multimodal. We analyze the key challenges of probabilistic inference in deep learning, and presents novel approaches for fast posterior inference of neural network weights.

1.1 Thesis Outline

Theoretically-Guaranteed Inference

Classical inference algorithms, such as Markov chain Monte Carlo (MCMC), have enabled probabilistic modeling to achieve gold standard results on many machine learning tasks [107]. However, these algorithms are rarely used in modern machine learning due to the difficulty of scaling up to large datasets. Previous methods have mostly attempted to scale inference using stochasticity, which uses a subset to approximate the whole dataset. This strategy comes with a significant drawback: stochasticity often sacrifices exactness, that is, it introduces asymptotic bias into inference results. Exact inference is crucial for reliable uncertainty quantification, especially for high-risk tasks such as healthcare and autonomous driving. Prior work suggests that there is an inherent trade-off between scalability and exactness, forcing practitioners to choose between expensive exact methods and biased scalable ones.

In this part, we will discuss why it is important to do theoretically-guaranteed inference and how we can do it fast in practice. Chapter 3 is based on the work at NeurIPS 2019 [156] and NeurIPS 2020 [155], where we introduce a general framework to make classical inference algorithms both theoretically-guaranteed correct and scalable, with the application to Gibbs sampling and Metropolis-Hastings. In Chapter 4, we present the paper at AISTATS 2020 [154] where we propose a way to do theoretically-guaranteed *gradient-based* inference methods by using amortized Metropolis correction.

Efficient Inference for Reliable Deep Learing

Deep neural networks (DNNs) have achieved impressive success on a variety of learning problems in recent years. However, DNNs have been criticized for being unable to estimate uncertainty accurately. Probabilistic modeling provides a principled alternative that can mitigate this issue; they are able to account for model uncertainty and achieve automatic complexity control. Because of these properties, Bayesian inference methods on small neural networks have achieved great success. However, due to scalability issue, they remained unused on modern machine learning problems concerning deep neural networks.

In this part, we develop several methods towards advancing deep probabilistic modeling. In Chapter 5, which presents the paper at ICLR 2020 [157], we talk about a new stochastic MCMC method to efficiently explore the highly multimodal parameter space of DNNs given a practical computational budget. In Chapter 6, which presents the paper at AISTATS 2021 [158], we advance variational inference for cheap approximations of the posterior of Bayesian neural networks.

CHAPTER 2

BACKGROUND

Probabilistic modeling is a very general framework to learn from data. The pipeline of it can be summarized as in Figure 2.1. We first collect the data about the problem we are interested in, and then define a model with a prior distribution over the parameters to describe how the data can be observed from this system. After doing these, we are ready to infer the parameters of the model from the data by *Bayes rule*. In this inference step, we extract the knowledge from the data to update our model. Finally, we use the learned model to do downstream tasks such as prediction, decision making and scientific discovery.

2.1 Bayesian Inference

This thesis focuses on the algorithmic step of the probabilistic modeling pipeline: Bayesian inference. Specifically, given a dataset $\mathcal{D} = \{x_i\}_{i=1}^N$ and a θ-parameterized model, it aims to compute the posterior distribution

$$\pi(\theta) \propto \exp\left(-\sum_{i=1}^N U_i(\theta)\right), \text{where } U_i(\theta) = -\log p(x_i|\theta) - \tfrac{1}{N}\log p(\theta).$$

Here $p(\theta)$ is the prior and the $p(x_i|\theta)$ give the likelihood of observing x_i given the parameter θ. We assume the data are conditionally independent given θ. The U_i have a natural interpretation as component *energy functions* with π acting as a Gibbs measure. In practice, computing $\pi(\theta)$ is often intractable and thus requires using *approximate* methods, such as Markov chain Monte Carlo (MCMC) and variational inference (VI). Below we will review the foundations of these two popular approximate inference methods.

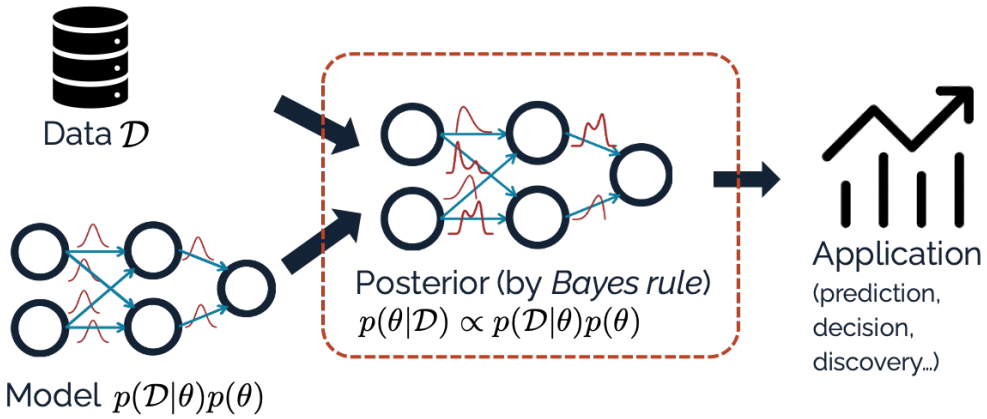


Figure 2.1: Probabilistic modeling pipeline, with inference step in the red square.

2.1.1 Markov chain Monte Carlo

Markov chain Monte Carlo (MCMC) methods play an important role in Bayesian inference. They work by constructing a Markov chain with the desired distribution as its equilibrium distribution; one samples from the chain and, as the algorithm converges to its equilibrium, the samples drawn reflect the desired distribution [102, 42, 71, 108]. We introduce several popular MCMC methods which also serve as the foundations for the following sections.

Metropolis-Hastings (MH)

The Metropolis-Hastings (MH) algorithm [65, 102] is one of the most commonly used MCMC methods. In each step, MH generates a proposal θ' from a proposal distribution $q(\cdot|\theta)$, and accepts it with probability

$$a(\theta, \theta') = \min\left(1, \frac{\pi(\theta')q(\theta|\theta')}{\pi(\theta)q(\theta'|\theta)}\right) = \min\left(1, \exp\left(\sum_{i=1}^{N}(U_i(\theta) - U_i(\theta'))\right) \cdot \frac{q(\theta|\theta')}{q(\theta'|\theta)}\right). \tag{2.1}$$

If accepted, the chain transitions to θ'; otherwise, it remains at the current state θ. This accept/reject step can be quite costly when N is large, since it entails

computing a sum over the entire dataset.

Gibbs Sampling

Gibbs sampling [55] works by iteratively resampling a variable from its conditional distribution with the remaining variables fixed. Suppose that the target distribution is $\pi(\theta) = \pi(\theta^1, \ldots, \theta^n)$. In each iteration, we select a variable θ^j to sample at random, and resample it from the conditional distribution

$$\theta^j \sim \pi(\theta^j | \theta^{\{1,\ldots,n\}\setminus\{j\}}).$$

Hamiltonian Monte Carlo and Second-Order Langevin Dynamics

Hamiltonian Monte Carlo (HMC) and second-order Langevin dynamics (L2MC) [42, 71, 108] construct a Markov chain by *augmenting* the state space with an additional momentum variable r, giving joint distribution

$$\pi(\theta, r) \propto \exp(-H(\theta, r)) = \exp\left(-U(\theta) - \frac{1}{2\sigma^2}\|r\|^2\right),$$

where H is the *Hamiltonian*, which measures the total energy of the system. Note we could replace the norm with any positive definite quadratic form on r. For simplicity, we only consider the case of isotropic momentum energy—where the *mass matrix* is $\sigma^2 I$. HMC then simulates Hamiltonian dynamics

$$d\theta = \sigma^{-2} r \, dt, \qquad dr = -\nabla U(\theta) \, dt. \tag{2.2}$$

The value of the Hamiltonian is preserved under these dynamics, so we must also include transitions that change the value of H to explore the whole state space. HMC does this by periodically resampling r from its conditional distribution. L2MC does so by continuously modifying r with a friction term and added Gaussian noise.

Even though (2.2) preserves H, the discrete simulation of (2.2) run by HMC or L2MC *does not* necessarily do so. Therefore both algorithms need an MH correction step to prevent bias due to discretization.

Stochastic Gradient MCMC

Stochastic Gradient MCMC (SG-MCMC) is a family of scalable sampling methods that enables inference with mini-batches of data. When $\mathcal{D}$ is too large, it is expensive to evaluate $U(\theta)$ for all the data points at each iteration. Instead, SG-MCMC methods use a minibatch to approximate $U(\theta)$: $\tilde{U}(\theta) = -\frac{N'}{N}\sum_{i=1}^{N'} \log p(x_i|\theta) - \log p(\theta)$, where $N' \ll N$ is the size of minibatch. We recommend [94] for a general review of SG-MCMC algorithms. We describe two SG-MCMC algorithms considered in this thesis.

SGLD & SGHMC Stochastic Gradient Langevin Dynamics (SGLD) [146] uses stochastic gradients with Gaussian noise and can be regarded as stochastic version of the first order Langevin Monte Carlo (LMC). Posterior samples are updated at the k-th step as: $\theta_k = \theta_{k-1} - \alpha_k \nabla \tilde{U}(\theta_k) + \sqrt{2\alpha_k}\epsilon_k$, where α_k is the stepsize and ϵ_k has a standard Gaussian distribution.

To improve mixing over SGLD, Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) [27] introduces an auxiliary momentum variable v. SGHMC is built upon HMC, with an additional friction term to counteract the noise introduced by a mini-batch. Thus it can also be thought of as a stochastic version of L2MC. The update rule for posterior samples is: $\theta_k = \theta_{k-1} + v_{k-1}$, and $v_k = v_{k-1} - \alpha_k \nabla \tilde{U}(\theta_k) - \eta v_{k-1} + \sqrt{2(\eta - \hat{\gamma})\alpha_k}\epsilon_k$, where $1 - \eta$ is the momentum term and $\hat{\gamma}$ is the estimate of the noise.

2.1.2 Variational Inference

Another common approximate inference method is variational inference (VI) [76, 152], which approximates the true posterior with a tractable posterior $q_\phi(\theta) \approx p(\theta|\mathcal{D})$. Typically the approximate posterior $q_\phi(\theta)$ is obtained by minimizing a divergence, e.g. variational inference (VI) often minimizes $\mathrm{KL}(q_\phi(\theta)\|p(\theta|\mathcal{D}))$. This turns Bayesian inference into an optimization task (divergence minimization). In practice, due to the intractability of $p(\mathcal{D})$, VI alternatively maximizes an equivalent objective called the *variational lower bound*:

$$\mathcal{L}_{\mathrm{VI}} = \mathbf{E}_{\theta\sim q_\phi}\left[\log \frac{p(\mathcal{D},\theta)}{q_\phi(\theta)}\right] = \log p(\mathcal{D}) - \mathrm{KL}(q_\phi\|p). \tag{2.3}$$

Renyi's α-divergence α-divergence is a generalization of KL divergence [68, 90, 104]. There are different definitions of α-divergence and their equivalences are shown in [31]. Here we focus on Renyi's definition [90, 124] instead of others [4, 138] as it allows our meta-learning framework to be differentiable in α (Section 6.2.1). Renyi's α-divergence is defined on $\alpha > 0, \alpha \neq 1$

$$D_\alpha(p\|q) = \frac{1}{\alpha - 1}\log \int p(\theta)^\alpha q(\theta)^{1-\alpha} d\theta, \tag{2.4}$$

and for $\alpha = 1$ it is defined by continuity: $D_1(p\|q) = \lim_{\alpha\to 1} D_\alpha(p\|q) = \mathrm{KL}(p\|q)$. Similar to the variational lower bound, one can maximize the *variational Renyi bound* (VR bound) [90]:

$$\begin{aligned}\mathcal{L}_\alpha(q_\phi;\mathcal{D}) &= \frac{1}{1-\alpha}\log \mathbf{E}_{\theta\sim q_\phi}\left[\left(\frac{p(\theta,\mathcal{D})}{q_\phi(\theta)}\right)^{1-\alpha}\right] \\ &= \log p(\mathcal{D}) - D_\alpha(q_\phi\|p).\end{aligned} \tag{2.5}$$

The expectation is usually computed by Monte Carlo (MC) approximation. To allow gradient backpropagation, the VR bound uses the reparameterization trick [79, 131],

where sampling $\theta \sim q_\phi(\theta)$ is conducted by first sampling $\epsilon \sim p(\epsilon)$ from a simple distribution independent with the variational distribution (e.g. Gaussian) then parameterizing $\theta = r_\phi(\epsilon)$. It follows that the gradient of the VR bound w.r.t. the variational parameter ϕ after MC approximation with K particles is

$$\nabla_\phi L_\alpha(q_\phi; x) = \sum_{k=1}^{K} \left[w_{\alpha,k} \nabla_\phi \log \frac{p(r_\phi(\epsilon_k), x)}{q(r_\phi(\epsilon_k))} \right], \tag{2.6}$$

where $w_{\alpha,k} = \left(\frac{p(r_\phi(\epsilon_k),x)}{q(r_\phi(\epsilon_k))}\right)^{1-\alpha} \Big/ \sum_{k=1}^{K} \left[\left(\frac{p(r_\phi(\epsilon_k),x)}{q(r_\phi(\epsilon_k))}\right)^{1-\alpha}\right]$. When $\alpha = 1$ the weights $w_{\alpha,k} = 1/K$ and the gradient Eq.(2.6) becomes an unbiased estimate of the gradient of the variational lower bound Eq.(2.3).

f-divergence f-divergence defines a more general family of divergences [33, 105]. Given a twice differentiable convex function $f : \mathbb{R}_+ \to \mathbb{R}$, the f-divergence is defined as [33]:

$$D_f(p\|q_\phi) = \mathbf{E}_{\theta\sim q_\phi}\left[f\left(p(\theta)/q_\phi(\theta)\right) - f(1)\right]. \tag{2.7}$$

This family includes KL-divergences in both directions, by taking $f(t) = -\log t$ for $\mathrm{KL}(q\|p)$ and $f(t) = t\log t$ for $\mathrm{KL}(p\|q)$. It also includes α-divergences by setting $f(t) = t^\alpha/(\alpha(\alpha-1))$ for $\alpha \in \mathbb{R}\backslash\{0,1\}$. Although the f-divergence family is very rich due to its parameterization by an arbitrary twice-differentiable convex function, it requires significant expertise to design a suitable f function for a specific task. Thus the potential of f-divergence has not been fully leveraged.

2.2 Scalability-Reliability Trade-Off

The existing inference tools can be evaluated across two important features: the ability to work with big data and large models, and the ability to provide reliable

and trustworthy inference results. As shown in Figure 2.2, on the one hand, we have optimization or *Maximum A Posteriori*, which approximates the posterior distribution by just a point mass. It is highly scalable and can work well with deep neural networks on millions of data points, but it is not reliable because it completely discards uncertainty and a point estimate is very sensitive to disturbance. On the other hand, we have Markov chain Monte Carlo, which is guaranteed to converge to the true posterior regardless the posterior's shape, but it is just too slow to work with modern models and big data. In between, we have variational inference which provides a simple approximation, such as a Gaussian approximation, to the true posterior. Recently, a class of inexact MCMC methods [146, 81, 11] emerges. In order to gain scalability, they introduce asymptotic bias to the inference results, thus significantly sacrifice reliability. Clearly there exists a trade-off in current inference methods, forcing practitioners to choose between expensive exact methods and biased scalable ones.

However modern probabilistic modeling needs better trade-off, because we have a much higher requirement for scalability and reliability and the current trade-off cannot satisfy it. Many real applications' scale grows exponentially and the results have to be reliable enough to avoid disastrous outcomes. Examples include medical diagnosis, autopilot driving, climate forecasting and policy enacting. In order to solve these problems, We must push the frontier of the trade off to satisfy the urgent need for new inference methods.

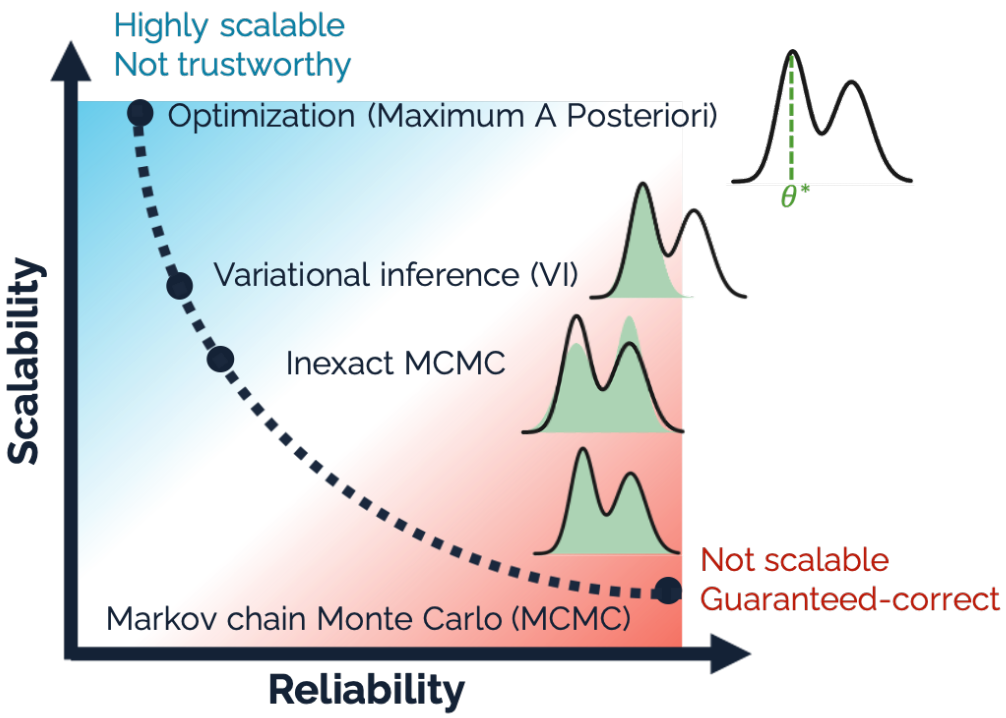


Figure 2.2: Scalability-reliability trade-off in current inference methods.

Part II

Theoretically-Guaranteed Inference

CHAPTER 3

POISSON-MINIBATCHING

In this chapter, we will introduce a general framework called *Poisson-Minibatching* to enable theoretically-guaranteed scalable and reliable inference. Poisson-Minibatching significantly scales classical inference methods while still keeping their asymptotic correctness. We will discuss two applications of this framework, one to Gibbs sampling and another to Metropolis-Hastings.

3.1 Gibbs sampling with Poisson-Minibatching

Gibbs sampling is a Markov chain Monte Carlo (MCMC) method which is widely used for inference on graphical models [80]. Although Gibbs sampling is a powerful method, its utility can be limited by its computational cost when the model is large. One way to address this is to use *stochastic methods*, which use a subsample of the dataset or model—called a minibatch—to approximate the dataset or model used in an MCMC algorithm. Minibatched variants of many classical MCMC algorithms have been explored [146, 97, 37, 87], including the MIN-Gibbs algorithm for Gibbs sampling [37].

In this section, we propose a new minibatched variant of Gibbs sampling on factor graphs called *Poisson-minibatching Gibbs* (Poisson-Gibbs). Like other minibatched MCMC methods, Poisson-minibatching Gibbs improves Gibbs sampling by reducing its computational cost. In comparison to prior work, our method improves upon MIN-Gibbs in two ways. First, it eliminates the need for a potentially expensive Metropolis-Hastings (MH) acceptance step, giving it a better asymptotic per-iteration time complexity than MIN-Gibbs. Poisson-minibatching Gibbs is able

to do this by choosing a minibatch in a way that depends on the current state of the variables, rather than choosing one that is independent of the current state as is usually done in stochastic algorithms. We show that such state-dependent minibatches can still be sampled quickly, and that an appropriately chosen state-dependent minibatch can result in a reversible Markov chain with the correct stationary distribution even without a Metropolis-Hastings correction step.

The second way that our method improves upon previous work is that it supports sampling over continuous state spaces, which are common in machine learning applications (in comparison, the previous work only supported sampling over discrete state spaces). The main difficulty here for Gibbs sampling is that resampling a continuous-valued variable from its conditional distribution requires sampling from a continuous distribution, and this is a nontrivial task (as compared with a discrete random variable, which can be sampled from by explicitly computing its probability mass function). Our approach is based on fast inverse transform sampling method, which works by approximating the probability density function (PDF) of a distribution with a polynomial [113].

In addition to these two new capabilities, we prove bounds on the convergence rate of Poisson-minibatching Gibbs in comparison to plain (i.e. not minibatched) Gibbs sampling. These bounds can provide a recipe for how to set the minibatch size in order to come close to the convergence rate of plain Gibbs sampling. If we set the minibatch size in this way, we can derive expressions for the per-iteration computational cost of our method compared with others; these bounds are summarized in Table 3.1. In summary, the contributions of this section are as follows:

- We introduce Poisson-minibatching Gibbs, a variant of Gibbs sampling which

State Space	Algorithm	Computational Cost/Iter
Discrete	Gibbs sampling	$O(D\Delta)$
	MIN-Gibbs [37]	$O(D\Psi^2)$
	MGPMH [37]	$O(DL^2 + \Delta)$
	DoubleMIN-Gibbs [37]	$O(DL^2 + \Psi^2)$
	Poisson-Gibbs	$O(DL^2)$
Continuous	Gibbs with rejection sampling	$O(N\Delta)$
	PGITS: Poisson-Gibbs with ITS	$O(L^3)$
	PGDA: Poisson-Gibbs with double approximation	$O(L^2 \log L)$

Table 3.1: Computational complexity cost for a single-iteration of Gibbs sampling. Here, N is the required number of steps in rejection sampling to accept a sample, and the rest of the parameters are defined in Section 3.1.1.

can reduce computational cost without adding bias or needing a Metropolis-Hastings correction step.

- We extend our method to sample from continuous-valued distributions.
- We prove bounds on the convergence rate of our algorithm, as measured by the spectral gap, on both discrete and continuous state spaces.
- We evaluate Poisson-minibatching Gibbs empirically, and show that its performance can match that of plain Gibbs sampling while using less computation at each iteration.

3.1.1 Preliminaries and Definitions

In this section, we present some background about Gibbs sampling and graphical models and give the definitions which will be used throughout the section. In this section, we consider Gibbs sampling on a factor graph [80], a type of graphical model that defines a probability distribution in terms of its *factors*. Explicitly, a

factor graph consists of a set of variables $\mathcal{V}$ (each of which can take on values in some set $\mathcal{X}$) and a set of factors Φ, and it defines a probability distribution π over a state space $\Omega = \mathcal{X}^{\mathcal{V}}$, where the probability of some $x \in \Omega$ is

$$\pi(x) = \frac{1}{Z} \cdot \exp\left(\sum_{\phi \in \Phi} \phi(x)\right) = \frac{1}{Z} \cdot \prod_{\phi \in \Phi} \exp\left(\phi(x)\right).$$

Here, Z denotes the scalar factor necessary for π to be a distribution. Equivalently, we can think of this as the *Gibbs measure* with energy function

$$U(x) = \sum_{\phi \in \Phi} \phi(x), \qquad \text{where} \qquad \pi(x) \propto \exp(U(x));$$

this formulation will prove to be useful in many of the derivations later in the section. (Here, the $\propto$ notation denotes that the expression on the left is a distribution that is proportional to the expression on the right with the appropriate constant of proportionality to make it a distribution.) In a factor graph, the factors ϕ typically only depend on a subset of the variables; we can represent this as a bipartite graph where the nodesets are $\mathcal{V}$ and Φ and where we draw an edge between a variable $i \in \mathcal{V}$ and a factor $\phi \in \Phi$ if ϕ depends on i. For simplicity, in this section we assume that the variables are indexed with natural numbers $\mathcal{V} = \{1, \ldots, n\}$. We denote the set of factors that depend on the ith variable, as

$$A[i] = \{\phi | \phi \text{ depends on variable } i,\ \phi \in \Phi\}.$$

An important property of a factor graph is that the conditional distribution of a variable can be computed using only the factors that depend on that variable. This lends to a particularly efficient implementation of Gibbs sampling, in which only these adjacent factors are used at each iteration (rather than needing to evaluate the whole energy function U): this is illustrated in Algorithm 1.

The performance of our algorithm will depend on several parameters of the graphical model, which we will now restate, from previous work on MIN-Gibbs [37].

Algorithm 1 Gibbs Sampling

Input: initial point x
loop
  sample variable $i \sim \text{Unif}\{1, \ldots, n\}$
  for all $v \in \mathcal{X}$ **do**
    $x(i) \leftarrow v$
    $U_v \leftarrow \sum_{\phi \in A[i]} \phi(x)$
  end for
  construct distribution ρ where

$$\rho(v) \propto \exp(U_v)$$

  sample v from ρ
  update $x(i) \leftarrow v$
  output sample x
end loop

If the variables take on discrete values, we let $D = |\mathcal{X}|$ denote the number of values each can take on. We let $\Delta = \max_i |A[i]|$ denote the maximum degree of the graph. We assume that the magnitudes of the factor functions are all bounded, and for any ϕ we let M_ϕ denote this bound

$$M_\phi = (\sup_{x \in \Omega} \phi(x)) - (\inf_{x \in \Omega} \phi(x)) .$$

Without loss of generality (and as was done in previous works [37]), we will assume that $0 \leq \phi(x) \leq M_\phi$ because we can always add a constant to any factor ϕ without changing the distribution π. We define the *local maximum energy* L and *total maximum energy* Ψ of the graph as bounds on the sum of M_ϕ over the set of the factors associated with a single variable i and the whole graph, respectively,

$$L = \max_{i \in \{1,2,\ldots,N\}} \sum_{\phi \in A[i]} M_\phi \qquad \text{and} \qquad \Psi = \sum_{\phi \in \Phi} M_\phi.$$

If the graph is very large and has many low-energy factors, the maximum energy of a graph can be much smaller than the maximum degree of the graph. All runtime analyses in this section assume that evaluating a factor ϕ and sampling from a small discrete distribution can be done in constant time.

3.1.2 Poisson-Minibatching Gibbs Sampling

In this section, we will introduce the idea of Poisson-minibatching under the setting in which we assume we can sample from the conditional distribution of $x(i)$ exactly. One such example is when the state space of x is discrete. We will consider how to sample from the conditional distribution when exact sampling is impossible in the next section.

In plain Gibbs sampling, we have to compute the sum over all the factors in $A[i]$ to get the energy in every step. When the graph is large, the computation of getting the energy can be expensive; for example, in the discrete case this cost is proportional to $D\Delta$. The main idea of Poisson-minibatching is to augment a desired distribution with extra Poisson random variables, which control how and whether a factor is used in the minibatch for a particular iteration. [97] used a similar idea to control whether a data point will be included in the minibatch or not with augmented Bernoulli variables. However, this method has been shown to be very inefficient when only updating a small fraction of Bernoulli variables in each iteration [117]. Our method does not suffer from the same issue due to the usage of Poisson variables which we will explain further later in this section.

We define the conditional distribution of additional variable s_ϕ for each factor ϕ as

$$s_\phi | x \sim \text{Poisson}\left(\frac{\lambda M_\phi}{L} + \phi(x)\right)$$

where $\lambda > 0$ is a hyperparameter that controls the minibatch size. Then the joint distribution of variables x and s, where s is a variable vector including all s_ϕ, is

$\pi(x,s) = \pi(x) \cdot \mathbf{P}(s|x)$ and so

$$\pi(x,s) \propto \exp\left(\sum_{\phi\in\Phi}\left(s_\phi \log\left(1+\frac{L}{\lambda M_\phi}\phi(x)\right) + s_\phi \log\left(\frac{\lambda M_\phi}{L}\right) - \log\left(s_\phi!\right)\right)\right). \tag{3.1}$$

Using (3.1) allows us to compute conditional distributions (of the variables x_i) using only a subset of the factors. This is because the factor ϕ will not contribute to the energy unless s_ϕ is greater than zero. If many s_ϕ are zero, then we only need to compute the energy over a small set of factors. Since

$$\mathbf{E}\left[|\{\phi \in A[i] \mid s_\phi > 0\}|\right] \leq \mathbf{E}\left[\sum_{\phi\in A[i]} s_\phi\right] = \sum_{\phi\in A[i]}\left(\frac{\lambda M_\phi}{L} + \phi(x)\right) \leq \lambda + L,$$

this implies that $\lambda + L$ is an upper bound of the expected number of non-zero s_ϕ. When the graph is very large and has many low-energy factors, $\lambda + L$ can be much smaller than the factor set size, in which case only a small set of factors will contribute to the energy while most factor terms will disappear because s_ϕ is zero.

Using Poisson auxiliary variables has two benefits. First, compared with the Bernoulli auxiliary variables as described in FlyMC [97], there is a simple method for sampling n Poisson random variables in total expected time proportional to the sum of their parameters, which can be much smaller than n [37]. This means that sampling n Poisson variables can be much more efficient than sampling n Bernoulli variables, which allows our method to avoid any inefficiencies caused by sampling Bernoulli variables as in FlyMC. Second, compared with a fixed-minibatch-size method such as the one used in [146], Poisson-minibatching has the important property that the variables s_ϕ are independent. Whether a factor will be contained in the minibatch is independent to each other. This property is necessary for proving convergence rate theorems in the section.

In Poisson-Gibbs, we will sample from the joint distribution alternately. At each iteration we can (1) first re-sample all the s_ϕ, then (2) choose a variable index

i and re-sample $x(i)$. Here, we can reduce the state back to only x, since the future distribution never depends on the current value of s. Essentially, we only bother to re-sample the s_ϕ on which our eventual re-sampling of $x(i)$ depends: statistically, this is equivalent to re-sampling all s_ϕ. Doing this corresponds to Algorithm 2.

However, minibatching by itself does not mean that the method must be more effective than plain Gibbs sampling. It is possible that the convergence rate of the minibatched chain becomes much slower than the original rate, such that the total cost of the minibatch method is larger than that of the baseline method even if the cost of each step is smaller. To rule out this undesirable situation, we prove that the convergence speed of our chain is not slowed down, or at least not too much, after applying minibatching. To do this, we bound the convergence rate of our algorithm, as measured by the *spectral gap* [83], which is the gap between the largest and second-largest eigenvalues of the chain's transition operator. This gap has been used previously to measure the convergence rate of minibatched MCMC [37].

Theorem 1. *Poisson-Gibbs (Algorithm 2) is reversible and has a stationary distribution π. Let $\bar{\gamma}$ denote its spectral gap, and let γ denote the spectral gap of plain Gibbs sampling. If we use a minibatch size parameter $\lambda \geq 2L$, then*

$$\bar{\gamma} \geq \exp\left(-\frac{4L^2}{\lambda}\right) \cdot \gamma.$$

This theorem guarantees that the convergence rate of Poisson-Gibbs will not be slowed down by more than a factor of $\exp(-4L^2/\lambda)$. If we set $\lambda = \Theta(L^2)$, then this factor becomes $O(1)$, which is independent of the size of the problem. We proved Theorem 1 and the other theorems in this section using the technique of Dirichlet forms, which is a standard way of comparing the spectral gaps of two chains by comparing their transition probabilities (more details are in the supplemental material).

Next, we derive expressions for the overall computational cost of Algorithm 2, supposing that we set $\lambda = \Theta(L^2)$ as suggested by Theorem 1. First, we need to evaluate the cost of sampling all the Poisson-distributed s_ϕ. While a naïve approach to sample this would take $O(\Delta)$ time, we can do it substantially faster. For brevity, and because much of the technique is already described in the previous work [37], we defer an explicit analysis to the supplementary material, and just state the following.

Statement 1. *Sampling all the auxiliary variables s_ϕ for $\phi \in A[i]$ can be done in average time $O(\lambda + L)$, resulting in a sparse vector s_ϕ.*

Now, to get an overall cost when assuming exact sampling from the conditional distribution, we consider discrete state spaces, in which we can sample from the conditional distribution of $x(i)$ exactly. In this case, the cost of a single iteration of Poisson-Gibbs will be dominated by the loop over v. This loop will run D times, and each iteration will take $O(|S|)$ time to run. On average, this gives us an overall runtime $O((\lambda + L) \cdot D) = O(L^2 D)$ for Poisson-Gibbs. Note that due to the fast way we sample Poisson variables, the cost of sampling Poisson variables is negligible compared to other costs.

In comparison, the cost of the previous algorithms MIN-Gibbs, MGPMH and DoubleMIN-Gibbs [37] are all larger in big-O than that of Poisson-Gibbs, as showed in Table 3.1. MGPMH and DoubleMIN-Gibbs need to conduct an MH correction, which adds to the cost, and the cost of MIN-Gibbs and DoubleMIN-Gibbs depend on Ψ which is a global statistic. By contrast, our method does not need additional MH step and is not dependent on global statistics. Thus the total cost of Gibbs sampling can be reduced more by Poisson-minibatching compared to the previous methods.

Application of Poisson-Minibatching to Metropolis-Hastings. Poisson-minibatching method can be applied to other MCMC methods, not just Gibbs sampling. To illustrate the general applicability of Poisson-minibatching method, we applied Poisson-minibatching to Metropolis-Hastings sampling and call it *PoissonMH* (details of this algorithm and a demonstration on a mixture of Gaussians are given in the supplemental material). We get the following convergence rate bound.

Theorem 2. *PoissonMH is reversible and has a stationary distribution π. If we let $\bar{\gamma}$ denote its spectral gap, and let $\bar{\gamma}$ denote the spectral gap of plain MH sampling with the same proposal and target distributions, then*

$$\bar{\gamma} \geq \tfrac{1}{2} \exp\left(-\tfrac{L^2}{\lambda+L}\right) \cdot \gamma.$$

Please note that PoissonMH has the same assumptions as stated in Section 3.1.1. These assumptions are strong for an MH algorithm and thus prevent PoissonMH's appplication to many common tasks of MH. In Section 3.2, we introduce another version of MH with Poisson-minibathching that removes these strong assumptions.

3.1.3 Poisson-Gibbs on Continuous State Spaces

In this section, we consider how to sample from a continuous conditional distribution, i.e. when $\mathcal{X} = [a, b] \subset \mathbb{R}$, without sacrificing the benefits of Poisson-minibatching. The main difficulty is that sampling from an arbitrary continuous conditional distribution is not trivial in the same way as sampling from an arbitrary discrete conditional distribution is. Some additional sampling method is required. In principle, we can combine any sampling method with Poisson-minibatching, such as rejection sampling which is commonly used in Gibbs sampling. However, rejection

Algorithm 2 Poisson-Gibbs

given: initial state $x \in \Omega$
loop
  sample variable $i \sim \text{Unif}\{1, \ldots, n\}$.
  for all ϕ **in** $A[i]$ **do**
    sample $s_\phi \sim \text{Poisson}\left(\frac{\lambda M_\phi}{L} + \phi(x)\right)$
  end for
  $S \leftarrow \{\phi | s_\phi > 0\}$
  for all $v \in \mathcal{X}$ **do**
    $x(i) \leftarrow v$
    $U_v \leftarrow \sum_{\phi \in S} s_\phi \log\left(1 + \frac{L}{\lambda M_\phi}\phi(x)\right)$
  end for
  construct distribution ρ where

$$\rho(v) \propto \exp(U_v)$$

  sample v from ρ
  update $x(i) \leftarrow v$
  output sample x
end loop

sampling needs to evaluate the energy multiple times per sample, so even if we reduce the cost of evaluating the energy by minibatching, the total cost can still be large, besides which there is no good guarantee on the convergence rate of rejection sampling.

In order to sample from the conditional distribution efficiently, we propose a new sampling method based on inverse transform sampling (ITS) method. The main idea is to approximate the continuous distribution with a polynomial; this requires only a number of energy function evaluations proportional to the degree of the polynomial. We provide overall cost and theoretical analysis of convergence rate for our method.

Algorithm 3 PGDA: Poisson-Gibbs Double Chebyshev Approximation

given: state $x \in \Omega$, degree m and k, domain $[a, b]$
loop
  set i, s_ϕ, S, and U as in Algorithm 2.
  construct degree-m Chebyshev polynomial approximation of energy U_v on $[a, b]$: $\tilde{U}_v$
  construct degree-k Chebyshev polynomial approximation:$\tilde{f}(v) \approx \exp(\tilde{U}_v)$
  compute the CDF polynomial

$$\tilde{F}(v) = \left(\int_a^b \tilde{f}(y)\, dy \right)^{-1} \int_a^v \tilde{f}(y)\, dy$$

  sample $u \sim \text{Unif}[0, 1]$.
  solve root-finding problem for v: $\tilde{F}(v) = u$
  ▷ Metropolis-Hastings correction:
  $p \leftarrow \frac{\exp(U_v)\tilde{f}(x(i))}{\exp(U_{x(i)})\tilde{f}(v)}$
  with probability $\min(1, p)$, set $x(i) \leftarrow v$
  output sample x
end loop

Poisson-Gibbs with Double Chebyshev Approximation.

Inverse transform sampling is a classical method that generates samples from a uniform distribution and then transforms them by the inverse of cumulative distribution function (CDF) of the desired distribution. Since the CDF is often intractable in practice, Fast Inverse Transform Sampling (FITS) [113] uses a Chebyshev polynomial approximation to estimate the PDF fast and then get the CDF by computing an integral of a polynomial. Inspired by FITS, we propose Poisson-Gibbs with double Chebyshev approximation (PGDA).

The main idea of double Chebyshev approximation is to approximate the energy function first and then the PDF by using Chebyshev approximation *twice*. Specifically, we first get a polynomial approximation to the energy function U on

$[a, b]$, denoted by $\tilde{U}$, the *Chebyshev interpolant* [137]

$$\tilde{U}(x) = \sum_{k=0}^{m} \alpha_k T_k \left(\frac{2(x-a)}{b-a} - 1 \right), \ \alpha_k \in \mathbb{R}, \ x \in [a, b], \tag{3.2}$$

where $T_k(x) = \cos(k \cos^{-1} x)$ is the degree-k Chebyshev polynomial. Although the domain is continuous, we only need to evaluate U on $m+1$ Chebyshev nodes to construct the interpolant, and the expansion coefficients α_k can be computed stably in $O(m \log m)$ time. The following theorem shows that the error of a Chebyshev approximation can be made arbitrarily small with large m. (Although stated for the case of $[a, b] = [-1, 1]$, it easily generalizes to arbitrary $[a, b]$.)

Theorem 3 (Theorem 8.2 from [137])**.** *Assume U is analytic in the open Bernstein ellipse $B([-1, 1], \rho)$, where the Bernstein ellipse is a region in the complex plane bounded by an ellipse with foci at ± 1 and semimajor-plus-semiminor axis length $\rho > 1$. If for all $x \in B([-1, 1], \rho)$, $|U(x)| \leq V$ for some constant $V > 0$, the error of the Chebyshev interpolant on $[-1, 1]$ is bounded by*

$$|\tilde{U}(x) - U(x)| \leq \delta_m \qquad \textit{where} \qquad \delta_m = \frac{4V\rho^{-m}}{\rho - 1}.$$

After getting the approximation of the energy, we can get the PDF by $\exp(\tilde{U})$. However, it is generally hard to get the CDF now since the integral of $\exp(\tilde{U})$ for polynomial $\tilde{U}$ is usually intractable. So, we use *another* Chebyshev approximation $\tilde{f}$ to estimate $\exp(\tilde{U})$. Constructing the second Chebyshev approximation requires no additional evaluations of energy functions; its total computational cost is $\tilde{O}(mk)$ because we need to evaluate a degree-m polynomial k times to compute the coefficients. After doing this, we are able to compute the CDF directly since it is the integral of a polynomial. With the CDF $\tilde{F}(x)$ in hand, inverse transform sampling is used to generate samples. First, a pseudo-random sample u is generated from the uniform distribution on $[0, 1]$, and then we solve the following root-finding

problem for x: $\tilde{F}(x) = u$. Since $\tilde{F}(x)$ is a polynomial, this root-finding problem can be solved by many standard methods. We use bisection method to ensure the robustness of the algorithm [113].

Importantly, the sample we get here is actually from an *approximation* of the CDF. To correct the error introduced by the polynomial approximation, we add a MH correction as the final step to make sure the samples come from the target distribution. Our algorithm is given in Algorithm 3. As before, we prove a bound on PGDA in terms of the spectral gap, given the additional assumption that the factors ϕ are analytic.

Theorem 4. *PGDA (Algorithm 3) is reversible and has a stationary distribution π. Let $\bar{\gamma}$ denote its spectral gap, and let γ denote the spectral gap of plain Gibbs sampling. Assume $\rho > 1$ is some constant such that every factor function ϕ, treated as a function of any single variable $x(i)$, must be analytically continuable to the Bernstein ellipse with radius parameter ρ shifted-and-scaled so that its foci are at a and b, such that it satisfies $|\phi(z)| \leq M_\phi$ anywhere in that ellipse. Then, if $\lambda \log(2) \geq 4L$, and if m is set large enough that $4\rho^{-m/2} \leq \sqrt{\rho} - 1$, then it will hold that*

$$\bar{\gamma} \geq \left(1 - 4\sqrt{\digamma}\right) \exp\left(\frac{-4L^2}{\lambda}\right) \cdot \gamma,$$

$$\textit{where} \quad \digamma = \frac{4 \cdot \exp(8L) \cdot \rho^{-\frac{k}{2}}}{\sqrt{\rho} - 1} + \exp\left(\frac{16L \cdot \rho^{-\frac{m}{2}}}{\sqrt{\rho} - 1}\right) - 1.$$

Similar to Theorem 1, this theorem implies that the convergence rate of PGDA can be slowed down by at most a constant factor relative to plain Gibbs. If we set $m = \Theta(\log L)$, $k = \Theta(L)$ and $\lambda = \Theta(L^2)$, then the ratio of the spectral gaps will also be $O(1)$, which is independent of the problem parameters.

Poisson-Gibbs with Fast Inverse Transform Sampling (PGITS)

Note that it is possible to combine FITS with Poisson-Gibbs directly (i.e. use only one polynomial approximation to estimate the PDF directly), and we call this method *Poisson-Gibbs with fast inverse transform sampling* (PGITS). It turns out that PGDA is more efficient than PGITS since PGDA requires fewer evaluations of U to achieve the same convergence rate. If we set the parameters as above, the total computational cost of PGDA is $O(m \cdot (\lambda + L) + m \cdot k) = O(\log L \cdot (L^2 + L)) = O(\log L \cdot L^2)$. On the other hand, the cost of PGITS to achieve the same constant-factor spectral gap ratio is $O(L^3)$. Now we will outline the algorithm and derive convergence rate results for it. These results will illustrate why PGITS can be expected to perform worse than PGDA.

PGITS operates by approximating the PDF with a Chebyshev polynomial approximation and then sampling from that polynomial approximation using inverse transform sampling. Specifically, if the PDF we want to sample from is $f(x)$, we can approximate f by $\tilde{f}$ on $[a, b]$ using Chebyshev polynomials,

$$\tilde{f} = \sum_{k=0}^{m} \alpha_k T_k \left(\frac{2(x-a)}{b-a} - 1 \right), \ \alpha_k \in \mathbb{R}, \ x \in [a, b] \tag{3.3}$$

where $T_k(x) = \cos(k \cos^{-1} x)$ is the degree k Chebyshev polynomial, and α_k are the Chebyshev coefficients of the function f [137]. We do this by interpolating f at its Chebyshev nodes, resulting in $\tilde{f}$ being the mth order *Chebyshev interpolant*. Once we have the polynomial approximation $\tilde{f}$ we can construct the corresponding CDF approximation $\tilde{F}$ by calculating the integral directly (since polynomials are straightforward to integrate). With the approximation $\tilde{F}$, we are able to use inverse transform sampling to generate samples. We call this whole algorithm *PGITS* and it is listed as Algorithm 4.

We show that PGITS is reversible and bound its spectral gap in the following theorem.

Theorem 5. *PGITS (Algorithm 4) is reversible and has a stationary distribution π. Let $\bar{\gamma}$ denote its spectral gap, and let γ denote the spectral gap of plain Gibbs sampling. Assume $\rho > 1$ is some constant such that every factor function ϕ, treated as a function of any single variable $x(i)$, must be analytically continuable to the Bernstein ellipse with radius parameter ρ shifted-and-scaled so that its foci are at a and b, such that it satisfies $|\phi(z)| \leq M_\phi$ anywhere in that ellipse. Then, if $\lambda \geq 2L$ it will hold that*

$$\bar{\gamma} \geq \left(1 - \frac{8\exp(L)\rho^{-m/2}}{\sqrt{\rho - 1}}\right) \cdot \exp\left(-\frac{4L^2}{\lambda}\right) \cdot \gamma.$$

We can set $m = \Theta(L)$ and $\lambda = \Theta(L^2)$ to make the ratio of the spectral gaps $O(1)$, which is independent of the size of the problem. If the parameters are set in this way, the total cost of PGITS is $O(m \cdot (\lambda + L)) = O(L \cdot L^2) = O(L^3)$.

3.1.4 Experiments

We demonstrate our methods on three tasks including Potts models, continuous spin models and truncated Gaussian mixture in comparison with plain Gibbs sampling and previous minibatched Gibbs sampling. We release the code at `https://github.com/ruqizhang/poisson-gibbs`.

Potts Models

We first test the performance of Poisson-minibatching Gibbs sampling on the Potts model [115] as in [37]. The Potts model is a generalization of the Ising model [75]

Algorithm 4 PGITS: Poisson-Gibbs Inverse Transform Sampling

given: state $x \in \Omega$, degree m, domain $[a, b]$
loop
set i, s_ϕ, S, and U as in Algorithm 2.
construct degree-m Chebyshev polynomial approximation of polynomial PDF on $[a, b]$

$$\tilde{f}(v) \approx \exp(U_v)$$

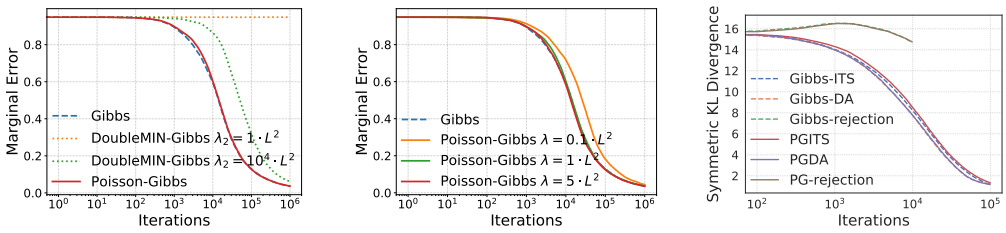
compute the CDF polynomial

$$\tilde{F}(v) = \left(\int_a^b \tilde{f}(y)\, dy \right)^{-1} \int_a^v \tilde{f}(y)\, dy$$

sample $u \sim \text{Unif}[0, 1]$.
solve root-finding problem for v: $\tilde{F}(v) = u$
▷ Metropolis-Hastings correction:

$$p \leftarrow \frac{\exp(U_v) \cdot \tilde{f}(x(i))}{\exp(U_{x(i)}) \cdot \tilde{f}(v)}$$

with probability $\min(1, p)$, set $x(i) \leftarrow v$
output sample x
end loop

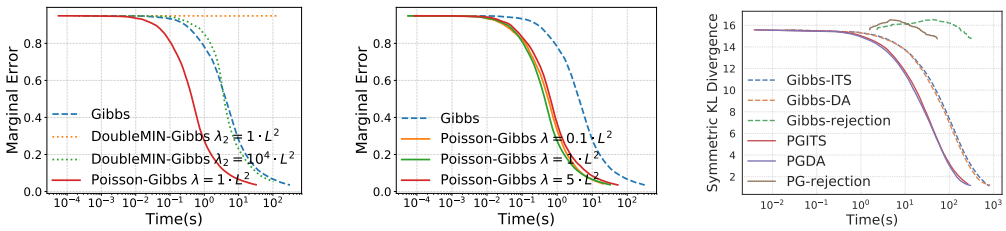


(a) (b) (c)

Figure 3.1: (a) Marginal error comparison among Poisson-Gibbs and previous methods on a Potts model. (b) Marginal error of Poisson-Gibbs on varying values of λ on a Potts model. (c) Symmetric KL divergence comparison among PGITS, PGDA and previous methods on a continuous spin model.

with domain $\{1, \ldots, D\}$ over an $N \times N$ lattice. The energy of a configuration is the following:

$$U(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} \beta \cdot A_{ij} \cdot \delta\left(x(i), x(j)\right)$$



(a) (b) (c)

Figure 3.2: Runtime comparisons with the same experimental setting as in Figure 3.1.

where the δ function equals one only when $x(i) = x(j)$ and zero otherwise. A_{ij} is the interaction between two sites i and j and β is the inverse temperature. As was done in previous work, we set the model to be fully connected and the interaction A_{ij} is determined by the distance between site i and site j based on a Gaussian kernel [37]. The graph has $n = N^2 = 400$ variables in total, $\beta = 4.6$ and $D = 10$. On this model, $L = 5.09$.

We first compare our method with two other methods: plain Gibbs sampling and the most efficient MIN-Gibbs methods on this task, DoubleMIN-Gibbs. Note that, in comparison to our method, DoubleMIN-Gibbs needs an additional MH correction step which requires a second minibatch to be sampled. We set $\lambda = 1 \cdot L^2$ for all minibatch methods. We tried two values for the second minibatch size in DoubleMIN-Gibbs $\lambda_2 = 1 \cdot L^2$ and $10^4 \cdot L^2$. We compute run-average marginal distributions for each variable by collecting samples. By symmetry, the marginal for each variable in the stationary distribution is uniform, so the ℓ_2-distance between the estimated marginals and the uniform distribution can be used to evaluate the convergence of Markov chain. We report this marginal error averaged over three runs.

Figure 3.1a shows the ℓ_2-distance marginal error as a function of iterations. We observe that Poisson-Gibbs performs comparably with plain Gibbs and it

outperforms DoubleMIN-Gibbs significantly especially when λ_2 is not large enough. The performance of DoubleMIN-Gibbs is highly influenced by the size of the second minibatch. We have to increase the second minibatch to $10^4 \cdot L^2$ in order to make it converge. This is because the variance of MH correction will be very large when the second minibatch is not large enough. On the other hand, Poisson-Gibbs does not require an additional MH correction which not only reduces the computational cost but also improves stability. In Figure 3.1b, we show the performance of our method with different values of λ. When we increase the minibatch size, the convergence speed of Poisson-Gibbs approaches plain Gibbs, which validates our theory. The number of factors being evaluated of Poisson-Gibbs varies each iteration, thus we report the average number which are 7, 28 and 132 respectively for $\lambda = 0.1 \cdot L^2$, $1 \cdot L^2$ and $5 \cdot L^2$.

The runtime comparisons with the same setup are reported in Figure 3.2a and 2b to demonstrate the computational speed-up of Poisson-Gibbs empirically. We can see that the results align with our theoretical analysis: Poisson-Gibbs is significantly faster than plain Gibbs samping and faster than previous minibatched Gibbs sampling methods. Compared to plain Gibbs, Poisson-Gibbs speeds up the computation by evaluating only a subset of factors in each iteration. Compared to DoubleMIN-Gibbs, Poisson-Gibbs is faster because it removes the need of an additional MH correction step.

Continuous Spin Models

In this section, we study a more general setting of spin models where spins can take continuous values. Continuous spin models are of interest in both the statistics and physics communities [103, 20, 41]. This random graph model can also be used to

describe complex networks such as social, information, and biological networks [111]. We consider the energy of a configuration as the following:

$$U(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} \beta \cdot A_{ij} \cdot (x(i) \cdot x(j) + 1)$$

where $x(i) \in [0, 1]$ and $\beta = 1$. Notice that the existing minibatched Gibbs sampling methods [37] are not applicable on this task since they can be used only on discrete state spaces. We compare PGITS, PGDA with: (1) Gibbs sampling with FITS (Gibbs-ITS); (2) Gibbs sampling with Double Chebyshev approximation (Gibbs-DA); (3) Gibbs with rejection sampling (Gibbs-rejection); and (4) Poisson-Gibbs with rejection sampling (PG-rejection). We use symmetric KL divergence to quantitatively evaluate the convergence. On this model, $L = 13.71$ and we set $\lambda = L^2$. The degree of polynomial is $m = 3$ for PGITS and the first approximation in PGDA. The degree of polynomial is $k = 10$ for the second approximation in PGDA. In rejection sampling, we set the proposal distribution to be wg where g is the uniform distribution on $[0, 1]$ and w is a constant tuned for best performance. The ground truth stationary distribution is obtained by running Gibbs-ITS for 10^7 iterations.

On this task, the average number of evaluated factors per iteration of Poisson-Gibbs is 190. Figure 3.1c shows the symmetric KL divergence as a function of iterations, with results averaged over three runs. Observe that our methods achieve comparable performance to Gibbs sampling with only a fraction of factors. For rejection sampling, the average steps needed for a sample to be accepted is greater than 300 which means that the cost is much larger than that of PGITS and PGDA. Given the same time budget, it can only run for many fewer iterations (we run it for 10^4 iterations). On the other hand, the two Chebyshebv approximation methods are much more efficient for both Poisson-Gibbs and plain Gibbs. The advantage

of FITS over rejection sampling has also been discussed in previous work [113]. Also notice that PGDA converges faster than PGITS given the same degree of polynomial. This empirical result validates our theoretical results that suggest PGDA is more efficient than PGITS.

We also report the symmetric KL divergence as a function of runtime in Figure 3.2c. Similar to the previous section, the two Poisson-Gibbs methods are faster than plain Gibbs sampling.

Truncated Gaussian Mixture

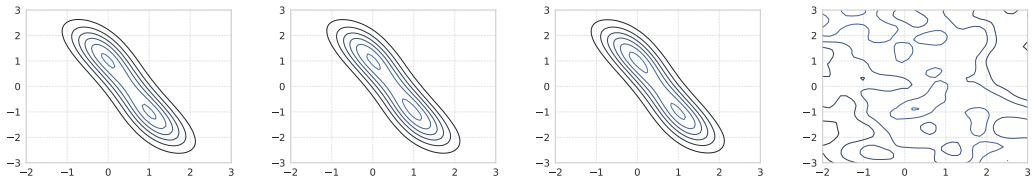
We further demonstrate PGITS and PGDA on a truncated Gaussian mixture model. We consider the following Gaussian mixture with tied means as done in previous work [146, 87]:

$$x_1 \sim \mathcal{N}(0, \sigma_1^2),\ x_2 \sim \mathcal{N}(0, \sigma_2^2),\ y_i \sim \frac{1}{2}\mathcal{N}(x_1, \sigma_y^2) + \frac{1}{2}\mathcal{N}(x_1 + x_2, \sigma_y^2).$$

We used the same parameters as in [146]: $\sigma_1^2 = 10$, $\sigma_2^2 = 1$, $\sigma_y^2 = 2$, $x_1 = 0$ and $x_2 = 1$. This posterior has two modes at $(x_1, x_2) = (0, 1)$ and $(x_1, x_2) = (1, -1)$. We truncate the posterior by bounding the variables x_1 and x_2 in $[-6, 6]$. The energy can be written as

$$U(x) = \log p(x_1) + \log p(x_2) + \sum_{i=1}^{N} \log p(y_i | x_1, x_2)$$

which can be regarded as a factor graph with N factors. We add a positive constant to the energy to ensure each factor is non-negative: this will not change the underlying distribution. As in [87], we set $N = 10^6$. $L = 1581.14$ for this model and we set $\lambda = 500$, $m = 20$ and $k = 25$. We have also considered higher values of λ and found that the results are very similar. We generate 10^6 samples for all methods. A uniform distribution in $[-6, 6]$ is used as the proposal distribution in



(a) True (b) PGITS (c) PGDA (d) Gibbs-rejection

Figure 3.3: A visualization of the estimated density on a truncated Gaussian mixture model.

Gibbs with rejection sampling. We try varying values for w but none of them results in reasonable density estimate which may be due to the inefficiency of rejection sampling [113]. We report the results when the average needed steps for a sample to be accepted is around 1000. The average number of factors being evaluated per iteration of Poisson-Gibbs is 1802. Our results are reported in Figure 3.3, where we observe visually that the density estimates of PGITS and PGDA are very accurate. In contrast, rejection sampling completely failed to estimate the density given the budget.

3.2 Metropolis-Hastings with Poisson-Minibatching

Metropolis-Hastings is one of the most basic inference algorithms and is also the oldest MCMC method. Recall that the Metropolis-Hastings (MH) algorithm accepts the candidate with probability

$$a(\theta, \theta') = \min\left(1, \frac{\pi(\theta')q(\theta|\theta')}{\pi(\theta)q(\theta'|\theta)}\right) = \min\left(1, \exp\left(\sum_{i=1}^{N}(U_i(\theta) - U_i(\theta'))\right) \cdot \frac{q(\theta|\theta')}{q(\theta'|\theta)}\right). \tag{2.1}$$

This accept/reject step can be quite costly when N is large, since it entails computing a sum over the entire dataset.

Prior work has proposed many approaches to mitigate the cost of this decision

step [10]. One popular approach involves introducing stochasticity: instead of computing over the entire dataset, a subsample, or *minibatch*, is used to compute an approximation. These minibatch MH methods can be divided into two classes, *exact* and *inexact*, depending on whether or not the target distribution π is necessarily preserved. Inexact methods introduce asymptotic bias to the target distribution, trading off correctness for speedups [11, 81, 132, 116, 117]. Exact methods either require impractically strong constraints on the target distribution [98, 156], limiting their applicability in practice, or they negatively impact efficiency, counteracting the speedups that minibatching aims to provide in the first place [9, 32]. Moreover, all existing exact methods operate on the belief that there is a trade-off between batch size and convergence rate—between scalability and efficiency. Yet no prior work formally exposes this trade-off, and most prior work gives no convergence rate guarantees. Given these various considerations, it is not entirely clear how to evaluate which minibatch MH method to use.

In this section we forge a path ahead to untangle this question. While inexact methods have been prominent recently due to their efficiency, they are not reliable: we show that the stationary distribution of any inexact method can be arbitrarily far from the target π. This means they can yield disastrously wrong inference results in practice, and it is difficult to tell just how bad those results can be.

We therefore turn our attention to exact methods and introduce *TunaMH*.[1] Compared to prior work, we make milder assumptions, which enables TunaMH to apply to a wider variety of inference tasks. More specifically, we require local rather than global bounds on the target distribution [98, 156] and do not rely on the Bernstein-von Mises approximation [32, 10, 13]. TunaMH is guaranteed to retain

[1] TunaMH since it *tunes* the efficiency-scalability trade-off and uses a Poisson (French for "fish") variable.

sample efficiency in the presence of minibatching: its convergence rate (measured by the spectral gap) is within a constant factor of standard, non-minibatch MH. More importantly, TunaMH also enables us to rigorously characterize the trade-off between scalability and efficiency. It has a hyperparameter χ, which enables tuning the trade-off between expected batch size and convergence rate.

By exposing this trade-off, our analysis raises the natural question: *is TunaMH optimal for this trade-off?* That is, could another exact algorithm use an asymptotically smaller average batch size while having the same convergence rate guarantees? We explore this in Section 3.2.3; under the same mild assumptions we use to derive TunaMH, we prove a lower bound on the expected batch size for *any* exact minibatch MH method that can keep a reasonable convergence rate. To our knowledge, we are the first to prove a lower bound of this nature for minibatch MH. Moreover, TunaMH is *asymptotically optimal* in balancing the expected batch size and convergence rate. It remains exact and efficient while on average using the smallest possible number of samples. In summary:

- We demonstrate that any inexact minibatch MH method can be arbitrarily inaccurate (Section 3.2.1).
- We introduce a new exact method, TunaMH (Section 3.2.2), with a lower bound on its convergence rate (in terms of the spectral gap) and a tunable hyperparameter to balance the trade-off between convergence rate and batch size.
- We prove a lower bound on the batch size for any exact minibatch MH method given a target convergence rate—the first such lower bound in this area. This result indicates that the expected batch size of TunaMH is asymptotically optimal in terms of the problem parameters (Section 3.2.3).
- We show empirically that TunaMH outperforms state-of-the-art exact minibatch

MH methods on robust linear regression, truncated Gaussian mixture, and logistic regression (Section 3.2.4).

3.2.1 Preliminaries and Drawbacks of Prior Minibatch MH Methods

We first formally define the class of methods that we study theoretically in this section: minibatch MH methods of the form of Algorithm 5. This class contains methods that sample a proposal from distribution q (which we always assume results in the chain being ergodic), and choose to accept or reject it by calling some randomized subroutine, `SubsMH`, which outputs 1 or 0 for "accept" or "reject," respectively. Algorithms in this class have several notable properties. First, `SubsMH` is *stateless*: each acceptance decision is made independently, without carrying over local state associated with the MH procedure between steps. Many prior methods are stateless [81, 11, 132, 32]. We do not consider *stateful* methods, in which the decision depends on previous state; they are difficult to analyze due to running on an extended state space [6, 116]. Second, `SubsMH` takes a function that computes energy *differences* $U_i(\theta) - U_i(\theta')$ and outputs an acceptance decision. We evaluate efficiency in terms of how many times `SubsMH` calls this function, which we term the *batch size* the method uses. Third, `SubsMH` takes parameters that bound the maximum magnitude of the energy differences. Specifically, as in [32], we assume:

Assumption 1. *For some constants $c_1, \ldots, c_N \in \mathbb{R}_+$, with $\sum_i c_i = C$, and symmetric function $M : \Theta \times \Theta \to \mathbb{R}_+$, for any $\theta, \theta' \in \Theta$, the energy difference is bounded by $|U_i(\theta) - U_i(\theta')| \leq c_i M(\theta, \theta')$.*

One can derive such a bound, which can be computed in $O(1)$ time, for many

Algorithm 5 Stateless, Energy-Difference-Based Minibatch Metropolis-Hastings

given: state space Θ, energy functions $U_1, \ldots, U_N : \Theta \to \mathbb{R}$, proposal dist. q, initial state $\theta \in \Theta$
given: parameters $c_1, \ldots, c_N$, C, M from Assumption 1, randomized algorithm `SubsMH`
loop
  sample $\theta' \sim q(\cdot|\theta)$
  define function $\Delta U : \{1, \ldots, N\} \to \mathbb{R}$, such that $\Delta U(i) = U_i(\theta) - U_i(\theta')$
  call subroutine $o \leftarrow$ `SubsMH`$(\Delta U, N, q(\theta|\theta')/q(\theta'|\theta), c_1, \ldots, c_N, C, M(\theta, \theta'))$
  if $o = 1$, **update** $\theta \leftarrow \theta'$
end loop

common inference problems: for example, if each energy function U_i is L_i-Lipschitz continuous, then it suffices to set $c_i = L_i$ and $M(\theta, \theta') = \|\theta - \theta'\|$ (See Appendix B.4 for examples of c_i and M on common problems). Note that the `SubsMH` method may choose *not* to use these bounds in its decision. We allow this so the form of Algorithm 5 can include methods that do not require such bounds. Most existing methods can be described in this form [81, 11, 132, 32, 9]. For example, standard MH can be written by setting `SubsMH` to a subroutine that computes the acceptance rate a as in (2.1) and outputs 1 (i.e., accept) with probability a.

Such minibatch MH methods broadly come in two flavors: *inexact* and *exact*. We next establish the importance of being exact and demonstrate how TunaMH resolves drawbacks in prior work.

The Importance of Being Exact

Inexact methods are popular due to helping scale MH to new heights [11, 81, 132, 116]. They approximate the MH acceptance ratio to within an error tolerance (> 0), trading off exactness for efficiency gains. Surprisingly, the bias from inexactness can be arbitrarily large even when the error tolerance is small.

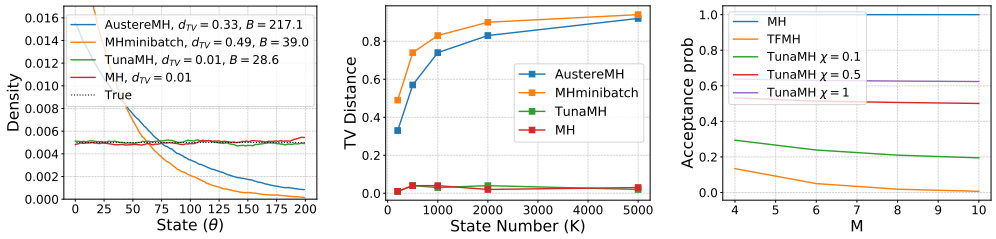
Theorem 6. *Consider any minibatch MH method of the form in Algorithm 5 that is inexact (i.e. does not necessarily have π as its stationary distribution for all π satisfying Assump. 1). For any constants $\delta \in (0, 1)$ and $\rho > 0$, there exists a target distribution π and proposal distribution q such that if we let $\tilde{\pi}$ denote a stationary distribution of the inexact minibatch MH method on this target, it satisfies*

$$\mathrm{TV}(\pi, \tilde{\pi}) \geq \delta \text{ and } \mathrm{KL}(\pi, \tilde{\pi}) \geq \rho.$$

where TV is the total variation distance and KL is the Kullback–Leibler divergence.

Theorem 6 shows that when using any inexact method, there always exists a target distribution π (factored in terms of energy functions U_i) and proposal distribution q such that it will approximate π arbitrarily poorly. This can happen even when individual errors are small; they can still accumulate a very large overall error. We prove Theorem 6 via a simple example—a random walk along a line, in which the inexact method causes the chain to step towards one direction more often than the other, even though its steps should be balanced (Appendix B.1.1). Note that it may be possible to avoid a large error by using some specific proposal distribution, but such a proposal is hard to know in general.

We use AustereMH [81] and MHminibatch [132] to empirically validate Theorem 6. For these inexact methods, we plot density estimates with the number of states $K = 200$ in Figure 3.4a (see Appendix B.4 for using other K); the stationary distribution diverges from the target distribution significantly. Moreover, the TV distance between the density estimate and the true density increases as K increases on this random walk example (Figure 3.4b). By contrast, our exact method (Section 3.2.2) keeps a small TV distance on all K and estimates the density accurately with an even smaller average batch size. We also tested AustereMH on robust linear



(a) (b) (c)

Figure 3.4: Existing MH method issues. (a)-(b) Inexact methods can diverge a lot from true distribution. "d_{TV}" and "B" denote the TV distance and the batch size respectively. (c) SMH has low and TunaMH with different values of hyperparameter χ has high acceptance rates.

regression, a common task, to show that the error of inexact methods can be large on standard problems (Appendix B.4).

Issues with Existing Exact Methods

This observation suggests that we should be using exact methods when doing minibatch MH. However, existing approaches present additional drawbacks, which we discuss below.

Factorized MH and Scalable MH are stateless, exact minibatch methods. Factorized MH (FMH) decomposes the acceptance rate into a product of factors, which allows for rejecting a proposal based on a minibatch of data [23, 30, 9]. Truncated FMH (TFMH) is a FMH variant that maintains geometric ergodicity; it falls back on standard MH in a step when the bound on the factors reaches a certain threshold [32]. No matter how this threshold is set, we can construct tasks where TFMH is either arbitrarily inefficient (rejecting arbitrarily often, slowing convergence), or degrades entirely to standard MH.

Statement 2. *For any constant $p \in (0, 1)$, there exists a target distribution such that TFMH either has an acceptance rate which is less than p times that of standard*

MH, or it completely degrades to standard MH (summing over the whole dataset at each step).

We prove this statement in Appendix B.1.2 using an example of a uniform distribution along a line, where we let x_i take one of two values, $\{-M/N, M/N\}$ with $M > 0$. We show that the acceptance rate of TFMH can be arbitrarily low by increasing M, which we also empirically verify in Figure 3.4c.

To improve the acceptance rate of TFMH, Scalable MH (SMH) introduces control variates, which approximate U_i with a Taylor series around the mode [32]. However, it only works with unimodal posteriors and high-quality Bernstein-von Mises approximations—conditions that do not hold for many common inference tasks.

PoissonMH is a stateless minibatch MH method adapted from Poisson-Gibbs on factor graphs as we discussed in Section 3.1.2. However, unlike the method in this section, it requires strong assumptions—specifically, a global upper bound on the energy. Such an upper bound usually does not exist and, even if it does, can be very large, resulting in an impractically large batch size.

FlyMC is a stateful method, which means it uses auxiliary random variables to persist state across different MH steps [98]. It requires a lower bound on the likelihood function, which is typically more demanding than Assumption 1 and does not have theoretical performance guarantees.

Other exact methods exist based on Piecewise Deterministic Markov Processes [18, 13]. They require regularity conditions only available for some problems, so their practical utility is limited.

3.2.2 TunaMH: Asymptotically Optimal Exact MH

In this section, we present our method, TunaMH, which evades the issues of prior exact methods discussed in Section 3.2.1. Like SMH [32], our method works on distributions for which an *a priori* bound on the energy differences is known (Assumption 1).

Our algorithm, presented in Algorithm 6, takes as parameters $c_1, \ldots, c_N$, C, and M from Assumption 1, along with an additional hyperparameter, $\chi > 0$. It proceeds in four steps. First, like any MH method, it generates a proposal θ' from given distribution q. Second, it samples a batch size B from a Poisson distribution. This makes the expected number of energy functions U_i evaluated by our method at each step $\mathbf{E}[B] = \chi C^2 M^2(\theta, \theta') + CM(\theta, \theta')^2$. Importantly, this means the batch size may vary from iteration to iteration, and the expected size depends on θ and θ'. For example, TunaMH may tend to set B larger for larger-distance proposals with a higher $M(\theta, \theta')$. Third, it samples (with replacement) a minibatch of size B, but for each data point it samples, it has some probability of *ejecting* this point from the minibatch. Finally, it accepts the proposed θ' with some probability, computed using a sum over the post-ejection minibatch. Our method can be derived by carefully replacing the auxiliary variables in PoissonMH with *local* Poisson variables whose distributions change each iteration depending on the pair (θ, θ') (Appendix B.2). By construction TunaMH is exact; it preserves the target distribution π as its stationary distribution. This is because TunaMH is *reversible*, meaning its transition operator T satisfies $\pi(\theta)T(\theta, \theta') = \pi(\theta')T(\theta', \theta)$ for any $\theta, \theta' \in \Theta$. This is a common condition that guarantees that a MCMC method has π as its stationary distribution [83, 19].

[2]Note that $\mathbf{E}[B]$ is typically $<< N$ and can be decreased using small step sizes. If, however, $\mathbf{E}[B] > N$, then we can simply use standard MH in that iteration, similar to TFMH.

Algorithm 6 TunaMH

given: initial state $\theta \in \Theta$; proposal dist. q; hyperparameter χ; Asm. 1 parameters c_i, C, M

loop

propose $\theta' \sim q(\cdot|\theta)$ and **compute** $M(\theta, \theta')$

▷ Form minibatch $\mathcal{I}$

sample $B \sim \text{Poisson}\left(\chi C^2 M^2(\theta,\theta') + CM(\theta,\theta')\right)$

initialize minibatch indices $\mathcal{I} \leftarrow \emptyset$ (an initially empty multiset)

for $b \in \{1, \ldots, B\}$ **do**

sample i_b such that $\mathbf{P}(i_b = i) = c_i/C$, for $i = 1 \ldots N$

with probability $\frac{\chi c_{i_b} C M^2(\theta,\theta') + \frac{1}{2}(U_{i_b}(\theta') - U_{i_b}(\theta) + c_{i_b} M(\theta,\theta'))}{\chi c_{i_b} C M^2(\theta,\theta') + c_{i_b} M(\theta,\theta')}$ **add** i_b to $\mathcal{I}$

end for

▷ Accept/reject step based on minibatch $\mathcal{I}$

compute MH ratio $r \leftarrow \exp\left(2\sum_{i\in\mathcal{I}} \text{artanh}\left(\frac{U_i(\theta) - U_i(\theta')}{c_i M(\theta,\theta')(1 + 2\chi C M(\theta,\theta'))}\right)\right) \cdot \frac{q(\theta'|\theta)}{q(\theta|\theta')}$

with probability $\min(1, r)$, set $\theta \leftarrow \theta'$

end loop

Compared to previous exact methods, a significant benefit of TunaMH is that we can prove theoretical guarantees on its efficiency. Specifically, its convergence speed is guaranteed to be close to standard MH and χ allows us to control how close. To show this, we lower bound the convergence rate of TunaMH in terms of the *spectral gap*, which is commonly used to characterize convergence speed in the MCMC literature [129, 63, 83, 156, 154]. The larger the spectral gap, the faster the chain converges.

Definition 1. *The* spectral gap *of a reversible Markov chain is the distance between the largest and second-largest eigenvalues of its transition operator. That is, if the eigenvalues of the transition operator are* $1 = \lambda_1 > \lambda_2 \geq \lambda_3 \cdots$*, then the spectral gap is* $\gamma = 1 - \lambda_2$.

Theorem 7. *TunaMH (Algorithm 6) is reversible with stationary distribution* π*. Let* $\bar{\gamma}$ *denote the spectral gap of TunaMH, and let* γ *denote the spectral gap of*

standard MH with the same target distribution and proposal distribution. Then,

$$\bar{\gamma} \geq \exp\left(-\tfrac{1}{\chi} - 2\sqrt{\tfrac{\log 2}{\chi}}\right) \cdot \gamma.$$

Intuitively, this theorem (proof in Appendix B.1.3) suggests the convergence rate of TunaMH is at most a constant slower than that of standard MH, and can be increased by adjusting the hyperparameter χ. Recall that χ also controls the batch size of TunaMH. Effectively, this means χ is a *dial* that allows us to directly tune the trade-off between convergence rate and batch size. When χ is large, the batch size B is large and the spectral gap ratio, $\bar{\gamma}/\gamma$, is close to 1: the larger batch size is less scalable but keeps a high convergence rate. Conversely, when χ is small, the batch size is small and the spectral gap ratio is close to 0: we trade off slow-downs in convergence rate for scalability. For example, for any $0 < \kappa < 1$, to guarantee the spectral gap ratio $\bar{\gamma}/\gamma \geq \kappa$ it suffices to set (Appendix B.1.6)

$$\chi = \tfrac{4}{(1-\kappa)\log(1/\kappa)}, \text{ giving an average batch size of } \mathbf{E}[B] = \tfrac{4C^2M^2(\theta,\theta')}{(1-\kappa)\log(1/\kappa)} + CM(\theta,\theta'). \tag{3.4}$$

In practice, we usually want to minimize the wall-clock time to achieve a certain estimate error, which requires tuning χ to optimally balance scalability and efficiency. We attempt to derive a theoretically optimal value of χ in Appendix B.3 by minimizing the product of the relaxation time—a measure of the number of steps needed—and the expected wall-clock time per step. Note that this product may be loose in bounding the total wall-clock time (we leave tightening this bound to future work), making the derived χ larger than necessary. In Section 3.2.4 we give a simple heuristic to tune χ, which works well and is generally better than the derived value.

Theorem 7 only requires the mild constraints of Assumption 1 on the target distribution, so applies in many scenarios and compares well to other exact methods.

SMH further requires a Bernstein-von Mises approximation to have guarantees on its batch size and acceptance rate. PoissonMH provides convergence rate guarantees, but demands the strong assumption that the target distribution has a global upper bound on the energy. FlyMC does not have any theoretical guarantees on performance.

3.2.3 Towards Optimal Exact Minibatch MH

In Theorem 7, we expose the trade-off between convergence rate and batch size in TunaMH. Here, we take this analysis a step further to investigate the limits of how efficient an exact minibatch MH method can be. To tackle this problem, we derive a lower bound on the batch size for any minibatch MH method that retains exactness and fast convergence. We then show that TunaMH is asymptotically optimal in terms of its dependence on the problem parameters C and M. In other words, it is not possible to outperform TunaMH in this sense with a method in the class described by Algorithm 5.

Theorem 8. *Consider any stateless exact minibatch MH algorithm described by Algorithm 5, any state space Θ (with $|\Theta| \geq 2$), any $C > 0$, and any function $M : \Theta \times \Theta \to \mathbb{R}^+$. Suppose that the algorithm guarantees that, for some constant $\kappa \in (0, 1)$, for any distribution, the ratio between the spectral gap of minibatch MH $\hat{\gamma}$ and the spectral gap of standard MH γ is bounded by $\hat{\gamma} \geq \kappa\gamma$. Then there must exist a distribution π over Θ and proposal q such that the batch size B of that algorithm, when deciding whether to accept any transition $\theta \to \theta'$, is bounded from below by*

$$\mathbf{E}[B] \geq \zeta \cdot \kappa \cdot \left(C^2 M^2(\theta, \theta') + CM(\theta, \theta')\right) \tag{3.5}$$

for some constant $\zeta > 0$ independent of algorithm and problem parameters.

To prove this theorem, we construct a random walk example over two states, then consider the smallest batch size a method requires to distinguish between two different stationary distributions (Appendix B.1.4). The impact of Theorem 8 is three-fold:

First, it provides an upper bound on the performance of algorithms of Algorithm 5's form: in each iteration, the average batch size of any exact minibatch MH method of the form of Algorithm 5 must be set as in (3.5) in order to maintain a reasonable convergence rate. To the best of our knowledge, this is the first theorem that rigorously proves a ceiling for the possible performance of minibatch MH.

Second, TunaMH achieves this upper bound. In fact, Theorem 8 suggests that TunaMH is *asymptotically optimal* in terms of the problem parameters, C and M. To see this, observe that when we ignore κ, both expressions that bound $\mathbf{E}[B]$ in (3.4) and (3.5) are $\mathcal{O}(C^2M^2(\theta, \theta') + CM(\theta, \theta'))$. Thus TunaMH reaches the lower bound, achieving asymptotic optimality in terms of C and M. (Of course, this sense of "optimality" does not rule out potential constant-factor improvements over TunaMH or improvements that depend on κ.)

Lastly, this result suggests directions for developing new exact minibatch MH algorithms: to be significantly faster than TunaMH, we either need to introduce additional assumptions to the problem or to develop new stateful algorithms.

In prior work, when assuming a very concentrated posterior, some methods' batch size can scale in $\mathcal{O}(1)$ [10, 13, 32] or $\mathcal{O}(1/\sqrt{N})$ [32] in terms of the dataset size N while maintaining efficiency. Theorem 8 is compatible with these results, further demonstrating this is essentially the *best* dependency on N an exact minibatch MH

method can achieve. We show this by explicitly assuming the dependency of C and M on N, as in SMH [32], yielding the following corollary (proof in Appendix B.1.5):

Corollary 1. *Suppose that C increases linearly with N ($C = \Theta(N)$) and $M(\theta, \theta')$ scales in $\mathcal{O}(N^{-(h+1)/2})$ for some constant $h > 0$. Then the lower bound in Theorem 8 becomes $\mathcal{O}(N^{(1-h)/2})$. In particular, it is $\mathcal{O}(1)$ when $h = 1$, and $\mathcal{O}(1/\sqrt{N})$ when $h = 2$.*

That is, TunaMH matches the state-of-the-art's dependency on N, and this dependency is optimal. Similarly, since C and M are the only problem parameters in the lower bound in Theorem 8, we can also get the optimal dependency on the other problem parameters by explicitly assuming the relation of them with C and M.

3.2.4 Experiments

We compare TunaMH to MH, TFMH, SMH (i.e. TFMH with MAP control variates) and FlyMC. We only include PoissonMH in the Gaussian mixture experiment, as it is not applicable in the other tasks. All of these methods are unbiased, so they have the same stationary distribution. To ensure fair wall-clock time comparisons, we coded each method in Julia; our implementations are at least as fast as, if not faster than, prior implementations. For each trial, we use Gaussian random walk proposals. We tune the proposal stepsize separately for each method to reach a target acceptance rate, and report averaged results and standard error from the mean over three runs. We set χ to be roughly the largest value that keeps $\chi C^2 M^2(\theta, \theta') < 1$ in most steps; we keep χ as high as possible while the average batch size is around its lower bound $CM(\theta, \theta')$. We found this strategy works well

in practice. We released the code at `https://github.com/ruqizhang/tunamh`.

Robust Linear Regression

We first test TunaMH on robust linear regression [32, 98]. We use a Student's t-distribution with degree of freedom $v = 4$ and set data dimension $d = 100$ (Appendix B.4). We tune each method separately to a 0.25 target acceptance rate. To measure efficiency, we record effective sample size (ESS) per second—a common MCMC metric for quantifying the number of effectively independent samples a method can draw from the posterior each second [19]. Figure 3.5a shows TunaMH is the most efficient for all dataset sizes N; it has the largest ESS/second. For minibatch MH methods, Figure 3.5b compares the average batch size. TunaMH's batch size is significantly smaller than FlyMC's—about 35x with $N = 10^5$. TFMH has the smallest batch size, but this is because it uses a very small step size to reach the target acceptance rate (Table B.1 in Appendix B.4). This leads to poor efficiency, which we can observe in its low ESS/second.

MAP variants Since TFMH and FlyMC have variants that use the *maximum a posteriori* (MAP) solution to boost performance, we also test TunaMH in this scheme. SMH uses MAP to construct control variates for TFMH to improve low acceptance rates. We consider both first- and second-order approximations (SMH-1 and SMH-2). FlyMC uses MAP to tighten the lower bound (FlyMC-MAP). For our method (TunaMH-MAP) and MH (MH-MAP), we simply initialize the chain with the MAP solution. Figure 3.5c shows that TunaMH performs the best even when previous methods make use of MAP. With control variates, SMH does increase the acceptance rate of TFMH, but this comes at the cost of a drastically increased batch size (Figure 3.5d) which we conjecture is due to the control variates scaling poorly in

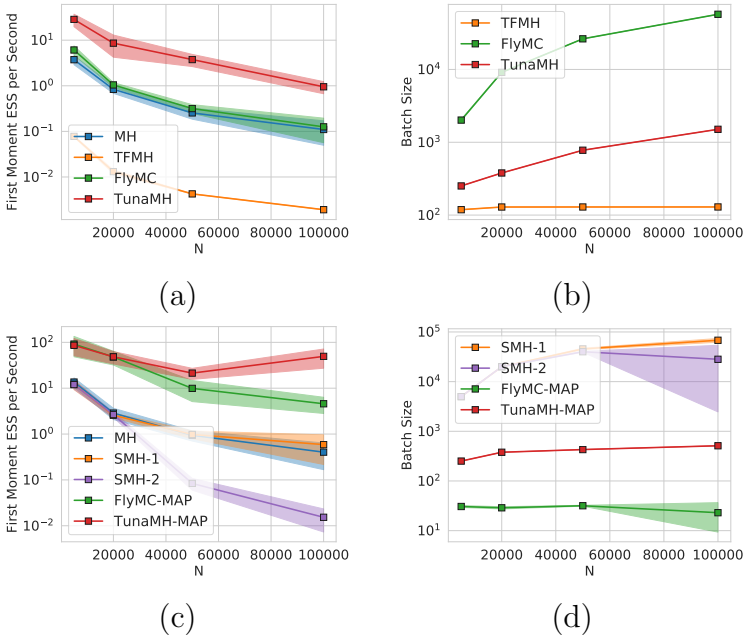


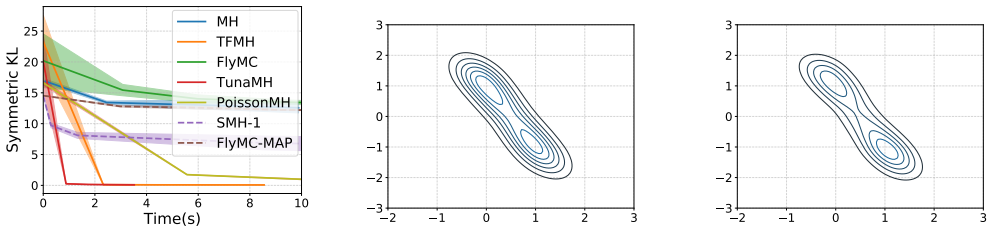
Figure 3.5: Robust linear regression, $d = 100$. (a) ESS/second without MAP. (b) Average batch size without MAP. (c) ESS/second with MAP. (d) Average batch size with MAP.

high dimensions ($d = 100$).[3] FlyMC-MAP tightens the bounds, entailing a decrease in the batch size. However, as clear in the difference in ESS/second, it is still less efficient than TunaMH due to its strong dependence between auxiliary variables and the model parameters—an issue that previous work also documents [116].

Truncated Gaussian Mixture

Next we test on a task with a multimodal posterior, a very common problem in machine learning. This demonstrates the advantage of TunaMH not relying on MAP, because MAP is a single solution and therefore is unable to reflect all possible modes in multimodal distributions. As a result, methods that rely on MAP tuning or MAP-based control variates are unable to perform well on such problems.

[3]Control variates worked well in the SMH paper [32] because all experiments had small dimension ($d = 10$).



(a) (b) (c)

Figure 3.6: Truncated Gaussian mixture. (a) Symmetric KL comparison. (b) True distribution. (c) Denstity estimate of TunaMH after 1 second.

We consider a Gaussian mixture. To get bounds on TunaMH, TFMH, SMH, and FlyMC, we truncate the posterior, bounding $\theta_1, \theta_2 \in [-3, 3]$ similar to [156]. We can include PoissonMH because its required bound exists after truncation. As in [132], we use a tempered posterior $\pi(\theta) \propto \exp\left(-\beta \sum_i U_i(\theta)\right)$ with $N = 10^6$ and $\beta = 10^{-4}$. Figure 3.6a compares performance, showing symmetric KL versus wall-clock time. TunaMH is the fastest, converging after 1 second, whereas the others take much longer. As expected, SMH-1 performs worse than TFMH, verifying the control variate is unhelpful for multimodal distributions. FlyMC and FlyMC-MAP are also inefficient; their performance is on par with standard MH, indicating negligible benefits from minibatching.

TunaMH also performs significantly better in terms of batch size, especially in comparison to PoissonMH (Table 3.2). This is due to TunaMH's local bound on the energy, as opposed to PoissonMH's global bound. This also allows TunaMH to run on more problem types, such as robust linear (Section 3.2.4) and logistic (Section 3.2.4) regression. To illustrate the estimate quality, we also visualize the density estimate after 1 second; TunaMH's estimate (Figure 3.6c) is very close to the true distribution (Figure 3.6b), while the other methods do not provide on-par estimates within the same time budget (Appendix B.4).

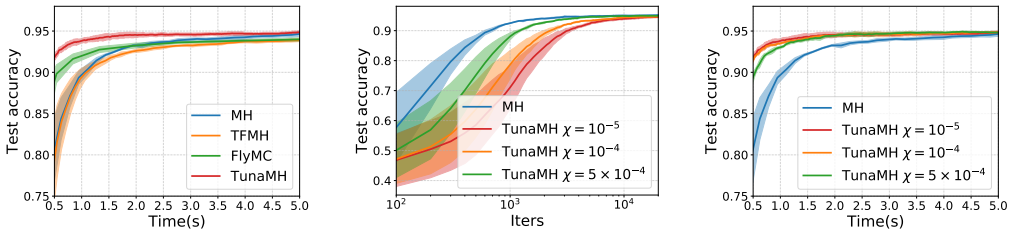
Table 3.2: Avg. batch size ± SE from the mean on 3 runs. PoissonMH not applicable to logistic reg.

Tasks	TFMH	FlyMC	PoissonMH	TunaMH
Gaussian Mixture	13.91 ± 0.016	811.52 ± 234.16	3969.67 ± 327.26	86.45 ± 0.04
Logistic Regression	39.28 ± 0.12	1960.19 ± 150.96	—	504.07 ± 0.33

Logistic Regression on MNIST

Lastly we apply TunaMH to logistic regression on the MNIST image dataset of handwritten number digits. Mirroring the work of FlyMC [98], we aim to classify 7s and 9s using the first 50 principal components as features. We set $\chi = 10^{-5}$ following our heuristic. In Figure 3.7a we see that TunaMH is the fastest of all methods to converge, as measured by wall-clock time. We also compare average batch size in Table 3.2. TunaMH's average batch size is 4x smaller than FlyMC's. TFMH again has the smallest batch size, but sacrifices efficiency by using a small step size in order to achieve the target acceptance rate. Thus, overall, TFMH is again inefficient in these experiments.

Effect of Hyperparameter χ To understand the effect of χ in TunaMH, we report results with varying χ. Figure 3.7b plots test accuracy as a function of the number of iterations. As χ increases, TunaMH's convergence rate approaches standard MH. This verifies our theoretical work: χ acts like a dial to control convergence rate and batch size trade-off—mapping to the efficiency-scalability trade-off. Figure 3.7c shows TunaMH's wall-clock time performance is not sensitive to χ, as the performance is superior to standard MH regardless of how we set it. However, χ needs to be tuned in order to achieve the best performance. Previous methods do not have such a dial, so they are unable to control this trade-off to improve the sampling efficiency.



(a) (b) (c)

Figure 3.7: MNIST logistic regression. (a) Test accuracy comparison. (b)-(c) TunaMH's test accuracy for various χ. Batch size for $\chi = 10^{-5}, 10^{-4}, 5 \times 10^{-4}$ is 504.07, 810.35 and 2047.91 respectively.

CHAPTER 4

AMORTIZED METROPOLIS ADJUSTMENT

The runtime efficiency benefits of Stochastic Gradient MCMC (SG-MCMC) methods come with a drawback: stochastic gradients introduce bias. Bias can cause convergence to a stationary distribution that differs from the one we wanted to sample from, and usually comes from two sources: converting the continuous-time process into discrete gradient updates, and noise from stochastic gradient estimates. These sources of error are in a sense unavoidable because they are also the source of SG-MCMC's runtime efficiency. Discretization with a large step size ϵ (instead of diminishing $\epsilon \to 0$) allows SG-MCMC to quickly move around its state space, and using stochastic gradients is key for scalability.

The standard approach for removing bias from a Markov chain is to introduce a *Metropolis-Hastings (MH) correction* [102]. This involves rejecting some fraction of the chain's transitions to restore the correct stationary distribution. Naïvely applying MH to SG-MCMC algorithms is often computationally prohibitive because the MH step typically needs to sum over the entire dataset. Performing this expensive computation every iteration would defeat the purpose of using stochastic gradients to improve performance. Thus, more sophisticated techniques are needed to achieve efficient, unbiased sampling for SG-MCMC.

In this section, we show that asymptotic exactness is possible *without being prohibitively expensive*. Specifically, we propose *Amortized Metropolis-Adjusted stochastic Gradient second-Order Langevin Dynamics (AMAGOLD)*. It achieves asymptotic exactness for SGHMC by using an MH correction step and does so without obliterating the performance gains provided by stochasticity. Table 4.1 provides a clarifying summary of the algorithms considered in this section. Our

Algorithm	Exact?	Stochastic Gradient?
AMAGOLD	Yes	Yes
L2MC	Yes	No
HMC	Yes	No
SGHMC	No	Yes

Table 4.1: Comparing 2nd order MCMC methods.

key insight is to apply the MH step *infrequently*. Rather than computing it every update, AMAGOLD performs it every T steps ($T > 0$). We prove this is sufficient to remove bias while also improving performance by amortizing the MH correction cost over T steps. We develop both reversible and non-reversible AMAGOLD variants and prove both converge to the desired distribution. We also prove a convergence rate relative to using full-batch gradients, which cleanly captures the effect of using stochastic gradients. This result provides insight about the trade-off between minibatching speed-ups and the convergence rate. Our results also show the noise from stochastic gradients has a provably bounded effect on convergence. In summary, our contributions are as follows:

- We introduce AMAGOLD, an efficient, asymptotically-exact SGHMC variant that *infrequently* applies an MH correction. We give reversible and non-reversible versions.
- We guarantee AMAGOLD converges to the target distribution, and does so without requiring step size $\epsilon \to 0$ or precise noise variance estimation.
- We prove a bound on AMAGOLD's convergence rate with mild assumptions, measured by the spectral gap. This bound is relative to how fast the algorithm *would have* converged if full-batch gradients were used. This is the first such relative convergence bound we are aware of for SG-MCMC.
- We validate our convergence guarantees empirically. Comparing to SGHMC,

Algorithm 7 SGHMC

given: Energy U, initial state $\theta \in \Theta$, step size ϵ, momentum variance σ^2, friction β
loop
  optionally, resample momentum:
  $r \sim \mathcal{N}(0, \sigma^2)$
  initialize position and momentum:
  $r_{\frac{1}{2}} \leftarrow r$, $\theta_0 \leftarrow \theta$
  for $t = 1$ **to** T **do**
    position update: $\theta_t \leftarrow \theta_{t-1} + \epsilon\sigma^{-2} r_{t-\frac{1}{2}}$
    sample noise $\eta_t \sim \mathcal{N}(0, 4\epsilon\beta\sigma^2)$
    sample random energy component $\tilde{U}_t$
    update momentum:

$$r_{t+\frac{1}{2}} \leftarrow r_{t-\frac{1}{2}} - \epsilon\nabla\tilde{U}_t(\theta_t) - 2\epsilon\beta r_{t-\frac{1}{2}} + \eta_t$$

  end for
  new values: $(\theta, r) \leftarrow (\theta_T, r_{T+\frac{1}{2}})$
  ▷ no MH step
end loop

AMAGOLD is more robust to hyperparameters. Regarding performance, AMAGOLD is competitive with full-batch baselines on synthetic and real-world datasets, and outperforms SGHMC on various tasks.

4.1 Preliminaries and Related Work

We write SGHMC as shown it Algorithm 7. Notably, SGHMC does not include an MH correction; to reduce the bias, it requires small ϵ.

Related Work

Our work is situated within a rich literature of SG-MCMC variants that take advantage of stochastic gradient techniques. These methods have demonstrated

success on deep neural networks (DNNs) for various tasks [88, 51, 157]. In particular, second-order SG-MCMC methods like SGHMC, which have a momentum term, have been shown to outperform first-order methods like SGLD on many applications [27, 25]. [52] proves SGHMC's convergence can be faster than SGLD's on non-convex problem due to its momentum-based acceleration. SGHMC can also be thought of as a stochastic version of L2MC [71] or HMC; we therefore use both L2MC and HMC as experimental full-batch baselines.

Prior work has also studied SGHMC's convergence properties. [27] examines its convergence for "asymptotically" small step sizes, in which a continuous-time system governs the dynamics (in contrast, our algorithm is asymptotically exact with a constant step size). Other work proves convergence with high-order integrators [25] and obtains non-asymptotic convergence bounds for SGHMC on non-convex optimization tasks [52].

Additional work has studied the properties of first-order MH adjusted Langevin methods, such as MALA [61, 128, 126, 127, 135]. [43] derives the mixing time of MALA for strongly log-concave densities, showing it has a better convergence rate than unadjusted Langevin (in comparison, AMAGOLD does not require the assumption of strongly log-concave densities). [81] developed a minibatch MH approach, which uses subsampling in the MH correction step, and applied it to correct bias in SGLD. They show cases where SGLD diverges from the target distribution, while SGLD with a minibatched MH correction performs well.

The work above involves first-order methods. To the best of our knowledge, we are the first to develop an unbiased, efficient second-order SG-MCMC algorithm. We are also the first in this space to use the spectral gap, a traditional metric to evaluate MCMC convergence [63, 83, 37]. It requires milder assumptions than

techniques in prior SG-MCMC work, such as 2-Wasserstein [119, 34], mean squared error [142, 25], and empirical risk [52].

4.2 Amortized Metropolis Adjustment

Reversible Markov chains are a particularly well-studied and well-behaved class of Markov chains. A Markov chain with transition probability operator G is reversible (also called satisfying the *detailed balance condition*) if for any pair of states x and y

$$\pi(x)G(x,y) = \pi(y)G(y,x). \tag{4.1}$$

It is well-known that a chain satisfying (4.1) has stationary distribution π. An MH correction constructs a reversible chain G with stationary distribution π from any Markov chain P (called the *proposal distribution*) by doing the following at each iteration. First, starting from state x, sample y from the proposal distribution $P(x,y)$. Second, compute the *acceptance probability*

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

Finally, with probability τ transition to state y; otherwise, remain in state x. This correction results in a reversible chain with stationary distribution π; however, computing τ at every step can be costly.

The natural way to amortize the cost of running MH is to replace the single proposal of baseline MH with T proposal-chain steps. This divides its cost among T iterations of the underlying chain, effectively decreasing it by a factor of T. For stochastic MCMC, each proposal step can be written as $P(x,y;\zeta)$, which denotes the probability of going from state x to state y given stochastic sample ζ taken

from some known distribution. (In minibatched MCMC, ζ captures information about which data we sample at that step.) Using this, we can run the following algorithm, starting at x. First, set $x_0 = x$, and run for t from 0 to $T-1$

$$\text{sample noise } \zeta_t, \text{ then sample } x_{t+1} \sim P(x_t, x_{t+1}; \zeta_t).$$

Next, set $y = x_T$: this is the proposal run an MH correction on. Finally, compute the acceptance probability

$$\tau = \min\left(1, \frac{\pi(y)}{\pi(x)} \prod_{t=0}^{T-1} \frac{P(x_{t+1}, x_t; \zeta_t)}{P(x_t, x_{t+1}; \zeta_t)}\right) \tag{4.2}$$

$$= \min\left(1, \prod_{t=0}^{T-1} \frac{\pi(x_{t+1})P(x_{t+1}, x_t; \zeta_t)}{\pi(x_t)P(x_t, x_{t+1}; \zeta_t)}\right). \tag{4.3}$$

and transition to state y with probability τ; otherwise, remain in state x.

It is straightforward to see this algorithm results in a reversible chain with stationary distribution π.[1] Additionally, this amortized MH step (4.2) is easily computed as long as the probabilities $P(\cdot, \cdot; \zeta)$ are tractable.

We expect this approach will be effective when the MH step does "not reject too often". This will certainly be the case when the terms inside the product in (4.3) all tend to be close to 1, which happens when the proposals $P(x, y; \eta)$ are "close" to being reversible with stationary distribution π. This is a good heuristic: our amortization approach should be effective when the proposals are close to being reversible.

Unfortunately, this heuristic does not apply to SGHMC's proposal step since SGHMC and other Hamiltonian-like steps are not close to satisfying the reversibility condition (4.1). Instead, the natural "reverse" trajectory for a Hamiltonian step reverses the order of the states and *negates the momentum*. The analog of

[1] A detailed proof appears in Appendix C.1.1.

reversibility for this sort of step is *skew-reversibility* [139]. Given some measure-preserving involution over the state space denoted $x^\perp$, a chain G is skew-reversible if $\pi(x) = \pi(x^\perp)$ and

$$\pi(x)G(x, y) = \pi(y^\perp)G(y^\perp, x^\perp). \tag{4.4}$$

Concretely, for Hamiltonian dynamics we use the involution that negates the momentum, i.e. $(\theta, r)^\perp = (\theta, -r)$. It is straightforward to show that a skew-reversible Markov chain also has π as its stationary distribution.[1] Such non-reversible chains have attracted a great deal of recent attention because they are more efficient than reversible ones in some situations [139, 74, 95].

A natural consequence of this setup is that we can amortize MH in the same manner as before, using skew-reversibility in place of reversibility. This gives the same multi-step-proposal algorithm as before, except that the acceptance probability is replaced with

$$\tau = \min\left(1, \frac{\pi(y^\perp)}{\pi(x)} \prod_{t=0}^{T-1} \frac{P(x_{t+1}^\perp, x_t^\perp; \zeta_t)}{P(x_t, x_{t+1}; \zeta_t)}\right). \tag{4.5}$$

The resulting corrected chain will be skew-reversible with stationary distribution π.[1] Intuitively, this chain will "not reject too often" as long as the proposals P are "close" to being skew-reversible. Since SGHMC steps are close to being skew-reversible, this is the more natural approach for amortizing MH, rather than using (4.2). If one wants to use the well-developed theoretical tools for a reversible chain, it is known that we can recover a reversible chain from a skew-reversible one by simply resampling the momentum at the beginning of the outer loop.[1] Note this reversible chain can be different from the one obtained by using condition (4.2).

4.3 AMAGOLD

We now apply the amortized Metropolis adjustment (AMA) method of Section 4.2 to second-order SG-MCMC. As a proposal, we use the stochastic leapfrog step that starts in (θ, r) and proposes (θ^*, r^*) by running

$$\begin{aligned}
\theta_0 &= \theta + \frac{1}{2}\epsilon\sigma^{-2}r \\
r^* &= ((1-\epsilon\beta)r - \epsilon\nabla\tilde{U}_t(\theta_0) + \mathcal{N}(0, 4\epsilon\beta\sigma^2 I))/(1+\epsilon\beta) \\
\theta^* &= \theta_0 + \frac{1}{2}\epsilon\sigma^{-2}r^*.
\end{aligned}$$

Applying our amortized MH correction to this proposal step using the acceptance probability (4.5) results in AMAGOLD (Algorithm 8). AMAGOLD is, by construction, skew-reversible, and we have the option of making it reversible by resampling the momentum.

AMAGOLD has three key differences compared to SGHMC. First, motivated by the time-reversal-symmetric nature of conditions (4.1) and (4.4), we use a clearly time-symmetric update step in the inner loop (compare Line 12 of Algorithm 8, which can be written as $r_{t+\frac{1}{2}} \leftarrow r_{t-\frac{1}{2}} - \epsilon\nabla\tilde{U}_t(\theta_t) - \epsilon\beta(r_{t-\frac{1}{2}} + r_{t+\frac{1}{2}}) + \eta_t$, with the less clearly symmetric Line 11 of Algorithm 7). Note this is just a different way of writing the algorithm: the update steps could be made equivalent by appropriately setting the hyperparameters. Second, we use a type of leapfrog integration that starts and ends the outer loop with a half-position-update (Lines 5 and 15). This too is done in the interest of time-reversal-symmetry. Third, there is an additional term ρ in AMAGOLD, which we call the *energy accumulator*, which accumulates the log of the product in (4.5). Computing ρ requires little extra cost since all its terms are already obtained in the standard update. AMAGOLD is thus unbiased without adding too much cost over SGHMC. The following theorem summarizes AMAGOLD's asymptotic accuracy. (This follows from the construction; an explicit

Algorithm 8 AMAGOLD

1: **given:** Energy U, initial state $\theta \in \Theta$, step size ϵ, momentum variance σ^2, friction β

2: **loop**

3: **optionally, resample momentum:**
$r \sim \mathcal{N}(0, \sigma^2 I)$

4: **initialize momentum, energy acc:**
$r_{-\frac{1}{2}} \leftarrow r$, $\rho_{-\frac{1}{2}} \leftarrow 0$

5: **half position update:** $\theta_0 \leftarrow \theta + \frac{1}{2}\epsilon\sigma^{-2} r_{-\frac{1}{2}}$

6: **for** $t = 0$ **to** $T - 1$ **do**

7: **if** $t \neq 0$ **then**

8: **position update:** $\theta_t \leftarrow \theta_{t-1} + \epsilon\sigma^{-2} r_{t-\frac{1}{2}}$

9: **end if**

10: **sample noise** $\eta_t \sim \mathcal{N}(0, 4\epsilon\beta\sigma^2 I)$

11: **sample random energy component** $\tilde{U}_t$

12: **update momentum:**
$r_{t+\frac{1}{2}} \leftarrow ((1 - \epsilon\beta) r_{t-\frac{1}{2}} - \epsilon\nabla\tilde{U}_t(\theta_t) + \eta_t)/(1 + \epsilon\beta)$

13: **update energy acc:**
$\rho_{t+\frac{1}{2}} \leftarrow \rho_{t-\frac{1}{2}} + \frac{1}{2}\epsilon\sigma^{-2}\nabla\tilde{U}_t(\theta_t)^T \left(r_{t-\frac{1}{2}} + r_{t+\frac{1}{2}}\right)$

14: **end for**

15: **half position update:**
$\theta_T \leftarrow \theta_{T-1} + \frac{1}{2}\epsilon\sigma^{-2} r_{T-\frac{1}{2}}$

16: **new values:** $\theta^* \leftarrow \theta_T$, $r^* \leftarrow r_{T-\frac{1}{2}}$

17: $a \leftarrow \exp\left(U(\theta) - U(\theta^*) + \rho_{T-\frac{1}{2}}\right)$

18: **with probability** $\min(1, a)$,
update $\theta \leftarrow \theta^*$, $r \leftarrow r^*$ (as long as $\theta^* \in \Theta$)

19: **otherwise update** $r \leftarrow -r_{-\frac{1}{2}}$

20: **end loop**

proof is in Appendix C.1.2.)

Theorem 9. *Consider the Markov chain described by AMAGOLD (Algorithm 8). If the momentum is resampled (on line 3), then this Markov chain is reversible. Otherwise the Markov chain is skew-reversible. In either case, its stationary distribution is* π.

Connection to previous methods AMAGOLD is related to several previous MCMC methods. When using a full-batch gradient, AMAGOLD becomes L2MC with amortized MH-adjustment. Using a full-batch, $\beta = 0$, and resampling, AMAGOLD becomes HMC. To see this, we first notice that with $\beta = 0$ the update rules of θ and r are the same as in HMC. The remaining thing is to show a is also the same as in HMC. We rewrite $\rho_{t+\frac{1}{2}}$ as

$$\begin{aligned}\rho_{t+\frac{1}{2}} &= \rho_{t-\frac{1}{2}} + \frac{1}{2}\sigma^{-2}\left(r_{t-\frac{1}{2}} - r_{t+\frac{1}{2}}\right)^T\left(r_{t-\frac{1}{2}} + r_{t+\frac{1}{2}}\right) \\ &= \rho_{t-\frac{1}{2}} + \frac{1}{2}\sigma^{-2}\left(\left\|r_{t-\frac{1}{2}}\right\|^2 - \left\|r_{t+\frac{1}{2}}\right\|^2\right)\end{aligned}$$

As a result,

$$\rho_{T-\frac{1}{2}} = \frac{1}{2}\sigma^{-2}\sum_{t=0}^{T-1}\left(\left\|r_{t-\frac{1}{2}}\right\|^2 - \left\|r_{t+\frac{1}{2}}\right\|^2\right) = \frac{1}{2}\sigma^{-2}\left(\left\|r_{-\frac{1}{2}}\right\|^2 - \left\|r_{T-\frac{1}{2}}\right\|^2\right)$$

It follows that a becomes the same as in HMC.

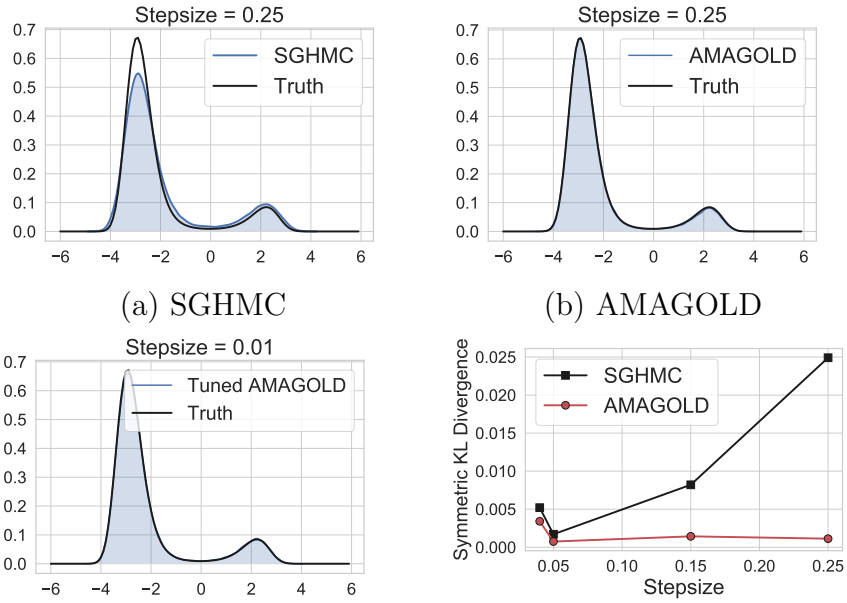
If we disable AMAGOLD's MH step (and adjust hyperparameters), it becomes SGHMC.

Illustrating AMAGOLD To illustrate AMAGOLD is able to achieve unbiased stochastic MCMC, we test our method on a double-well potential [40, 86]:

$$U(\theta) = (\theta + 4)(\theta + 1)(\theta - 1)(\theta - 3)/14 + 0.5.$$

The target distribution is proportional to $\exp(-U(\theta))$. To simulate stochastic gradients, we let $\nabla\tilde{U} = \nabla U + \mathcal{N}(0, 1)$. We show the results of SGHMC and AMAGOLD when $\beta = 0.25$, $T = 10$ and $\epsilon = 0.25$. Results for $\epsilon = \{0.05, 0.15\}$ are in Appendix C.3.

In Figure 4.1 and Appendix C.3, the estimated densities of AMAGOLD are very close to the true density on varying step sizes. In contrast, SGHMC does



(a) SGHMC (b) AMAGOLD

(c) Tuned AMAGOLD (d) KL Divergence

Figure 4.1: Estimated densities of (a) SGHMC and (b) AMAGOLD for step size 0.25 compared to the ground truth and (c) step size 0.01 for tuned AMAGOLD (see Section 4.3.2); (d) Comparison of symmetric KL divergence, varying step sizes for SGHMC and AMAGOLD.

not converge to the correct distribution asymptotically. This is especially the case when the step size is large: SGHMC diverges from the true distribution. These observations validate Theorem 9, as AMA guarantees convergence to the target distribution. To quantitatively measure divergence from the true distribution, we plot the symmetric KL divergence as a function of the step size in Figure 4.1d. We can see that SGHMC is very sensitive to step size, and may require careful tuning in practice, while AMAGOLD is more robust.

4.3.1 Convergence Rate Analysis

Using stochastic gradients in MCMC can reduce the cost of each iteration. However, this does not mean the overall cost of the algorithm will be less in comparison to

its non-stochastic counterpart. Rather, it is possible that the stochastic chain's convergence rate becomes much slower than the non-stochastic one. To be confident in the effectiveness of an SG-MCMC method, we must rule this out: We must show that the convergence speed of the stochastic chain is not slowed down, or at least not too much, compared to the non-stochastic chain. We do this analysis for AMAGOLD as follows.

Since AMAGOLD can be regarded as stochastic L2MC, we study reversible AMAGOLD's convergence rate relative to L2MC with an amortized MH correction. Prior work has used this type of bound to prove the convergence rate of subsampled MCMC methods [37, 156]. Unlike work that uses 2-Wasserstein, MSE, or empirical risk minimization to evaluate the convergence rate of SG-MCMC, we are the first to use the spectral gap—a traditional metric for evaluating MCMC convergence [63, 83] that is directly related to another common measurement, the mixing time [83]. Our bound only requires mild assumptions compared to prior work [142, 25], and we measure convergence to the target distribution directly, rather than empirical risk minimization [52].

The spectral gap γ of a reversible Markov chain with transition probability operator G is defined as the smallest distance between any non-principal eigenvalue of G and 1, the principal eigenvalue of G [83]. The spectral gap determines the convergence rate of a Markov chain: a chain with a smaller γ will take longer to converge. To ensure the existence of γ, we assume geometric ergodicity of the full-batch chain [129]. To bound γ, we assume the covariance of the gradient samples of AMAGOLD is bounded isotropically with

$$\mathbf{E}\left[(\nabla\tilde{U}(\theta) - \nabla U(\theta))(\nabla\tilde{U}(\theta) - \nabla U(\theta))^T\right] \preceq \frac{V^2}{d} I$$

for some constant $V > 0$. This sort of bounded-variance assumption is standard in

the analysis of stochastic gradient algorithms.

The following theorem shows that with appropriate hyperparameter settings the convergence rate of AMAGOLD will not be slowed down by more than a constant factor.

Theorem 10. *For some parameters $\epsilon > 0$, $\sigma > 0$, and $\beta > 0$, let $\bar{\gamma}$ denote the spectral gap of the L2MC chain running with parameters $(\epsilon, \sigma, \beta)$. Assume that these parameters are such that $\epsilon V^2 \leq 4\sigma^2 \beta d$. Define a constant $c = 1 + \sqrt{\frac{\epsilon V^2}{16\sigma^2 \beta T d^2}}$. Let γ denote the spectral gap of AMAGOLD running with parameters $(\epsilon, \sigma \cdot c^{-1/4}, \beta \cdot c^{-1/2})$. Then,*

$$\frac{\gamma}{\bar{\gamma}} \geq \exp\left(-\frac{\epsilon T V^2}{4\sigma^2 \beta} - \sqrt{\frac{\epsilon T V^2}{\sigma^2 \beta}}\right).$$

This requirement on parameters is easy to satisfy because d is generally large and ϵ is generally small in practice. For the same reason, c is usually close to 1, so the parameters used by the two chains are very close.

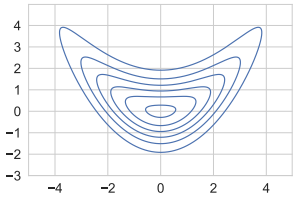
This theorem has three useful takeaways: First, AMAGOLD's convergence rate is essentially the same as L2MC up to a constant, which will approach 1 as the batch size increases (V decreases) or ϵ decreases. Second, it shows the effect of minibatching on convergence rate: if one reduces the minibatch size (i.e. V^2 increases), they can expect the convergence rate to decrease with a rate of $\exp(-O(V^2))$. Third, the theorem outlines a range of parameters (where $\epsilon T V^2 \ll \sigma^2 \beta$) over which AMAGOLD converges at a similar rate to the full-batch algorithm.

4.3.2 AMAGOLD in Practice

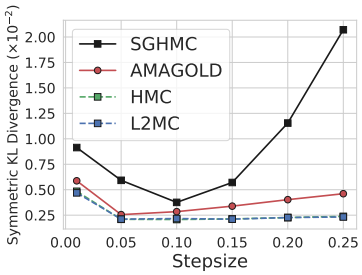
Here we describe some simple modifications that can further improve AMAGOLD's performance.

Minibatch MH Amortizing the cost of an MH correction over T steps is not always sufficient for achieving good performance on large datasets. This is because calculating the true energy U requires a scan over the whole dataset. We can further reduce the cost of a single correction by using minibatch MH to compute the acceptance probability—using a minibatch at line 17 of Algorithm 8. Prior work has estimated the MH correction using a subset of the data [81, 11, 98, 132] and also TunaMH in Section 3.2. These methods are composable with, rather than exclusive with, AMAGOLD and could provide additional speed-ups.

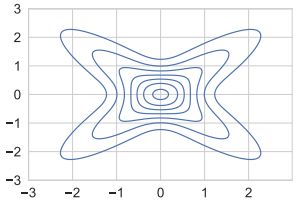
Tuning the step size Our experiments on double well potential (Figure 4.1) show step size significantly influences SGHMC's performance. Besides being more robust to step size, AMAGOLD's step size can be more easily tuned. The MH step's acceptance probability provides information about whether a step size is desirable. Based on this information, the step size can be tuned automatically to target some fixed acceptance probability during burn-in without affecting convergence. With a fixed step size $\epsilon = 0.01$, both AMAGOLD and SGHMC provide poor density estimates due to too small step size. However, when we let AMAGOLD adjust ϵ such that the average MH acceptance probability is 85%, it estimates the density accurately (Figure 4.1c, Appendix C.3).



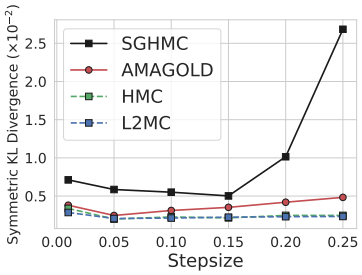
(a) Dist1



(b) KL comparison on Dist1



(c) Dist2



(d) KL comparison on Dist2

Figure 4.2: AMAGOLD's performance against baselines. In (b) and (d) the step size varies from 0.01 to 0.25; the symmetric KL divergence is a function of step size.

4.4 Experiments

Here we validate our theory empirically and explore the performance of AMAGOLD on a variety of applications. We compare to full-batch baselines HMC and L2MC to show AMAGOLD is more efficient and we compare to SGHMC because, despite exhibiting bias, it is commonly used in the literature. Unless otherwise specified, we use reversible AMAGOLD, meaning we resample the momentum, $T = 10$ and $\beta = 0.25$. We set hyperparameters for AMAGOLD in a similar way as SGHMC [27]. For simplicity, we do not use the techniques in Section 4.3.2. Additional details are in Appendix C.3. The code can be found at `https://github.com/ruqizhang/amagold`.

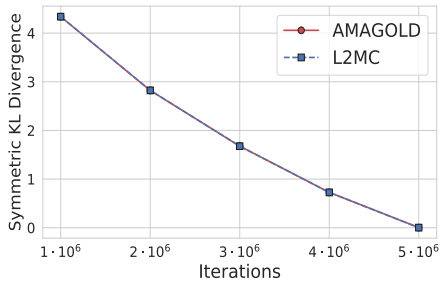


Figure 4.3: The convergence speed (symmetric KL divergence as a function of iterations) of AMAGOLD compared to L2MC on Dist1 with step size 0.15.

Synthetic Distributions

We conduct experiments on synthetic two-dimensional distributions (Figures 4.2a and 4.2c), which are adapted from [151]. The analytical expressions are in Appendix C.3. We compare our algorithm against three baselines: (1) HMC, (2) L2MC with amortized MH correction, and (3) SGHMC. HMC and L2MC serve as non-stochastic, unbiased baselines. As in [27], we replace ∇U by stochastic estimates $\nabla\tilde{U} = \nabla U + \mathcal{N}(0, I)$ for the stochastic methods. We draw 5×10^6 samples and use symmetric KL divergence as a function of step size to quantitatively evaluate the convergence of the Markov chain. On both distributions, AMAGOLD's symmetric KL divergence is close to full-batch methods and is much lower than SGHMC's, especially when the step size is large. This validates our theory that AMAGOLD is unbiased, while SGHMC's bias increases with step size. See Appendix C.3 for more details.

We then verify the theory that AMAGOLD has a comparable convergence rate to L2MC while using stochastic gradient estimates. Specifically, in Figure 4.3 AMAGOLD's convergence rate is the same as L2MC's (up to a constant factor slowdown of about 10^{-3}). We include runtime comparisons in Appendix C.3.

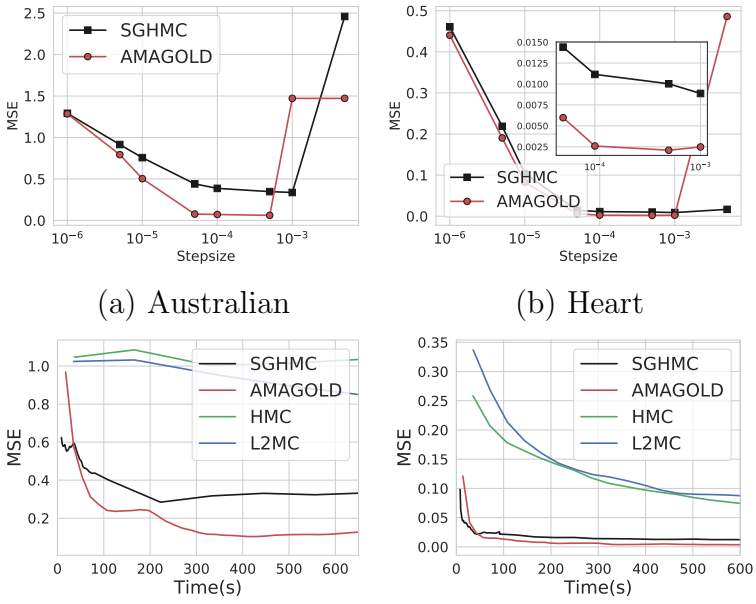
Bayesian Logistic Regression on Real-World Data

We evaluate our method on Bayesian logistic regression using two real-world datasets: *Australian* and *Heart* (Figure 4.4). We compute the MSE between the estimated and true parameters, obtained from 10^7 samples from HMC as in [85]. AMAGOLD exhibits smaller error than SGHMC on varying step sizes. We show runtime comparisons with step size 10^{-4}. Compared to full-batch HMC and L2MC, AMAGOLD is significantly faster due to minibatching. It is also not much slower than SGHMC, indicating AMA can reduce the cost of adding the MH step. AMAGOLD's large error using a large step size is due to a drop in MH acceptance probability (Appendix C.3). However, this drop can be easily avoided in practice. One can either set the step size such that it achieves a reasonable acceptance rate (usually 20–80%, depending on the application) or use the tuning technique in Section 4.3.2. With a reasonable acceptance rate, AMAGOLD achieves much lower error compared to SGHMC.

Bayesian Neural Networks

We apply AMAGOLD on Bayesian neural networks. The architecture is a MLP with two-layer with RELU non-linearities. The dataset size is 60000 and we use minibatch size 2000. We use irreversible AMAGOLD since we find it gives better results. Similar to [157], to speed up the convergence of the sampling methods, we use SGD with momentum in the first 3 epochs as burn-in and then switch to either SGHMC or AMAGOLD.

Classification We evaluate the classification accuracy of AMAGOLD and SGHMC. As in [27], we reparameterize our algorithm, setting $v = \epsilon\sigma^{-2}r, b = \epsilon\beta$ and



(a) Australian (b) Heart

(c) Australian runtime (d) Heart runtime

Figure 4.4: We use two real-world datasets (a) *Australian* (15 covariates, 690 data points) and (b) *Heart* (14 covariates, 270 data points). The minibatch size is 32 and 16, respectively. We collect 5×10^6 samples and test step size varying from 10^{-6} to 5×10^{-3}.

Algorithm	b	$h = 0.0005$	$h = 0.001$
SGHMC	0.01	3.69±0.03	3.77±0.17
SGHMC	5e-6	89.95±0.29	89.70±0.91
AMAGOLD	0.01	3.63±0.04	3.65±0.08
AMAGOLD	5e-6	3.65±0.10	3.63±0.10

Table 4.2: Comparison between AMAGOLD and SGHMC of test error (%) ± standard error. We collect 20 samples in total.

$h = \epsilon^2\sigma^{-2}$ (Appendix C.2). This equivalent two-parameter reformulated update is similar to SGD with momentum and thus more easily tuned on DNNs. Table 4.2 shows the test error on various hyperparameter settings. AMAGOLD yields consistent test error, regardless of the hyperparameter values. In contrast, the performance of SGHMC is affected significantly by the hyperparameters. When b is small, SGHMC diverges. Similar performance of SGHMC has also been reported in [40].

Uncertainty Evaluation We evaluate the sampling performance in terms of uncertainty evaluation, which is important in many ML applications [82, 16]. We test predictive uncertainty estimation on out-of-distribution samples [82]. The 8 models in Table 4.2 are tested on the notMNIST dataset [21]. Since the models have never seen the samples from notMNIST, ideally the predictive distribution should be uniform, which gives the maximum entropy. We plot the empirical CDF for the entropy of the predictive distribution (Figure 4.5). AMAGOLD provides consistent uncertainty estimations on all settings, which aligns with the classification results. In contrast, when b is small or h is large, SGHMC performance suffers; it is overconfident about its prediction.

Both of these experiments indicate that SGHMC is very sensitive to hyper-parameters. It needs to be carefully tuned to achieve desired performance on classification and uncertainty estimation. In contrast, AMAGOLD is robust to various hyperparameter settings because it is guaranteed to converge to the target distribution.

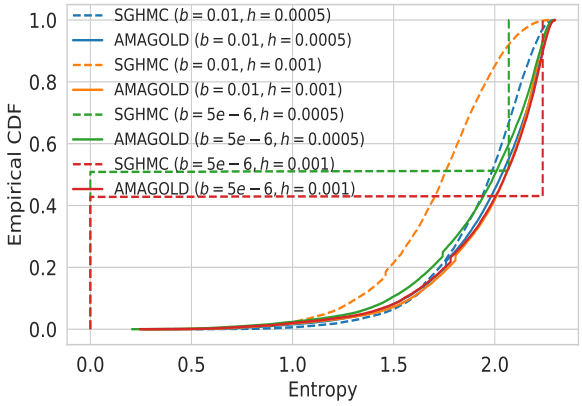


Figure 4.5: Empirical CDF on notMNIST dataset.

Part III

Efficient Inference for Reliable Deep Learing

CHAPTER 5

CYCLICAL STOCHASTIC GRADIENT MCMC

Deep neural networks are often trained with stochastic optimization methods such as stochastic gradient decent (SGD) and its variants. Bayesian methods provide a principled alternative, which account for model uncertainty in weight space [96, 106, 147], and achieve an automatic balance between model complexity and data fitting. Indeed, Bayesian methods have been shown to improve the generalization performance of DNNs [67, 16, 85, 99, 148], while providing a principled representation of uncertainty on predictions, which is crucial for decision making.

Approximate inference for Bayesian deep learning has typically focused on deterministic approaches, such as variational methods [67, 16]. By contrast, MCMC methods are now essentially unused for inference with modern deep neural networks, despite previously providing the gold standard of performance with smaller neural networks [106]. Stochastic gradient Markov Chain Monte Carlo (SG-MCMC) methods [146, 27, 40, 85] provide a promising direction for a sampling based approach to inference in Bayesian deep learning. Indeed, it has been shown that stochastic methods, which use mini-batches of data, are crucial for finding weight parameters that provide good generalization in modern deep neural networks [77].

However, SG-MCMC algorithms for inference with modern neural networks face several challenges: (*i*) In theory, SG-MCMC asymptotically converges to target distributions via a decreasing stepsize scheme, but suffers from a bounded estimation error in limited time [136, 25]. (*ii*) In practice, empirical successes have been reported by training DNNs in relatively short time [88, 27, 51, 109, 130]. For example, [130] apply SG-MCMC to generative adversarial networks (GANs) to solve the mode collapse problem and capture diverse generation styles. However, the loss

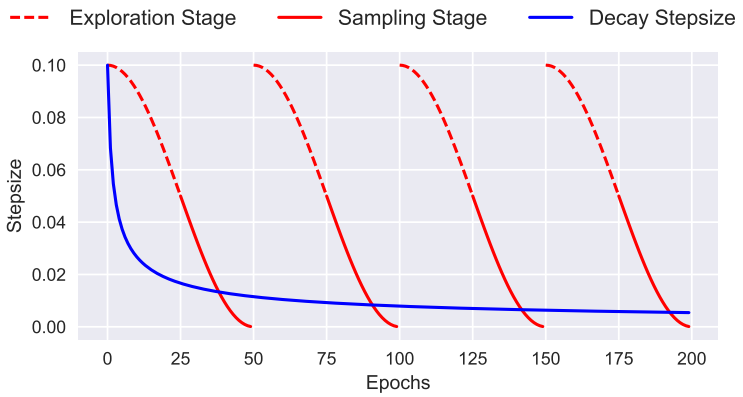


Figure 5.1: Illustration of the proposed cyclical stepsize schedule (red) and the traditional decreasing stepsize schedule (blue) for SG-MCMC algorithms.

surface for DNNs is highly multimodal [7, 29]. In order for MCMC to be effective for posterior inference in modern neural networks, a crucial question remains: how do we make SG-MCMC efficiently explore a highly multimodal parameter space given a practical computational budget?

Several attempts have been made to improve the sampling efficiency of SG-MCMC. Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) [27] introduces momentum to Langevin dynamics. Preconditioned stochastic gradient Langevin dynamics (pSGLD) [85] adaptively adjusts the sampler's step size according to the local geometry of parameter space. Though simple and promising, these methods are still inefficient at exploring multimodal distributions in practice. It is our contention that this limitation arises from difficulties escaping local modes when using the small stepsizes that SG-MCMC methods typically require. Note that the stepsize in SG-MCMC controls the sampler's behavior in two ways: the magnitude to deterministically drift towards high density regions *wrt.* the current stochastic gradient, and the level of injecting noise to randomly explore the parameter space. Therefore, a small stepsize reduces both abilities, resulting in a large numbers of iterations for the sampler to move across the modes.

In this section, we propose to replace the traditional decreasing stepsize schedule in SG-MCMC with a cyclical variant. To note the distinction from traditional SG-MCMC, we refer to this method as *Cyclical SG-MCMC* (cSG-MCMC). The comparison is illustrated in Figure 5.1. The blue curve is the traditional decay, while the red curve shows the proposed cyclical schedule. Cyclical SG-MCMC operates in two stages: (*i*) *Exploration*: when the stepsize is large (dashed red curves), we consider this stage as an effective burn-in mechanism, encouraging the sampler to take large moves and leave the local mode using the stochastic gradient. (*ii*) *Sampling*: when the stepsize is small (solid red curves), the sampler explores one local mode. We collect samples for local distribution estimation during this stage. Further, we propose two practical techniques to improve estimation efficiency: (1) a system temperature for exploration and exploitation; (2) A weighted combination scheme for samples collected in different cycles to reflect their relative importance.

This procedure can be viewed as SG-MCMC with warm restarts: the exploration stage provides the warm restarts for its following sampling stage. cSG-MCMC combines the advantages from (1) the traditional SG-MCMC to characterize the fine-scale local density of a distribution and (2) the cyclical schedule in optimization to efficiently explore multimodal posterior distributions of the parameter space. In limited time, cSG-MCMC is a practical tool to provide significantly better mixing than the traditional SG-MCMC for complex distributions. cSG-MCMC can also be considered as an *efficient* approximation to parallel MCMC; cSG-MCMC can achieve similar performance to parallel MCMC with only a fraction of cost (reciprocal to the number of chains) that parallel MCMC requires.

To support our proposal, we also prove the non-asymptotic convergence for the cyclical schedule. We note that this is the first convergence analysis of a cyclical

stepsize algorithm (including work in optimization). Moreover, we provide extensive experimental results to demonstrate the advantages of cSG-MCMC in sampling from multimodal distributions, including Bayesian neural networks and uncertainty estimation on several large and challenging datasets such as ImageNet.

In short, cSG-MCMC provides a simple and automatic approach to inference in modern Bayesian deep learning, with promising results, and theoretical support. This work is a step towards enabling MCMC approaches in Bayesian deep learning. We release code at `https://github.com/ruqizhang/csgmcmc`.

5.1 Preliminaries: SG-MCMC with a Decreasing Stepsize Schedule

To guarantee asymptotic consistency with the true distribution, SG-MCMC requires that the step sizes satisfy the following assumption:

Assumption 2. *The step sizes* $\{\alpha_k\}$ *are decreasing, i.e.,* $0 < \alpha_{k+1} < \alpha_k$*, with 1)* $\sum_{k=1}^{\infty} \alpha_k = \infty$*; and 2)* $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$.

Without a decreasing step-size, the estimation error from numerical approximations is asymptotically biased. One typical decaying step-size schedule is $\alpha_k = a(b+k)^{-\gamma}$, with $\gamma \in (0.5, 1]$ and (a, b) some positive constants [146].

5.2 Cyclical SG-MCMC

We now introduce our *cyclical SG-MCMC* (cSG-MCMC) algorithm. cSG-MCMC consists of two stages: *exploration* and *sampling*. In the following, we first introduce the cyclical step-size schedule, and then describe the exploration stage in Section 5.2 and the sampling stage in Section 5.2. We propose an approach to combining samples for testing in Section D.2.

Assumption 1 guarantees the consistency of our estimation with the true distribution in the asymptotic time. The approximation error in limited time is characterized as the risk of an estimator $R = B^2 + V$, where B is the bias and V is the variance. In the case of infinite computation time, the traditional SG-MCMC setting can reduce the bias and variance to zero. However, the time budget is often limited in practice, and there is always a trade-off between bias and variance. We therefore decrease the overall approximation error R by reducing the variance through obtaining more effective samples. The effective sample size can be increased if fewer correlated samples from different distribution modes are collected.

For deep neural networks, the parameter space is highly multimodal. In practice, SG-MCMC with the traditional decreasing stepsize schedule becomes trapped in a local mode, though injecting noise may help the sampler to escape in the asymptotic regime [159]. Inspired to improve the exploration of the multimodal posteriors for deep neural networks, with a simple and automatic approach, we propose the cyclical cosine stepsize schedule for SG-MCMC. The stepsize at iteration k is defined as:

$$\alpha_k = \frac{\alpha_0}{2}\left[\cos\left(\frac{\pi \bmod(k-1, \lceil K/M \rceil)}{\lceil K/M \rceil}\right) + 1\right], \tag{5.1}$$

where α_0 is the initial stepsize, M is the number of cycles and K is the number of

total iterations [91, 73].

The stepsize α_k varies periodically with k. In each period, α_k starts at α_0, and gradually decreases to 0. Within one period, SG-MCMC starts with a large stepsize, resulting in aggressive exploration in the parameter space; as the stepsize is decreasing, SG-MCMC explores local regions. In the next period, the Markov chain restarts with a large stepsize, encouraging the sampler to escape from the current mode and explore a new area of the posterior.

Related work in optimization. In optimization, the cyclical cosine annealing stepsize schedule has been demonstrated to be able to find diverse solutions in multimodal objectives, though not specifically different modes, using stochastic gradient methods [91, 73, 53, 49]. Alternatively, we adopt the technique to SG-MCMC as an effective scheme for sampling from multimodal distributions.

Exploration

The first stage of cyclical SG-MCMC, *exploration*, discovers parameters near local modes of an objective function. Unfortunately, it is undesirable to directly apply the cyclical schedule in optimization to SG-MCMC for collecting samples at every step. SG-MCMC often requires a small stepsize in order to control the error induced by the noise from using a minibatch approximation. If the stepsize is too large, the stationary distribution of SG-MCMC might be far away from the true posterior distribution. To correct this error, it is possible to do stochastic Metropolis-Hastings (MH) [81, 11, 26]. However, stochastic MH correction is still computationally too expensive. Further, it is easy to get rejected with an aggressive large stepsize, and every rejection is a waste of gradient computations.

Algorithm 9 Cyclical SG-MCMC.

Input: The initial stepsize α_0, number of cycles M, number of training iterations K and the proportion of exploration stage β.
  for k = 1:K **do**
    $\alpha \leftarrow \alpha_k$ according to Eq (5.1).
    if $\frac{\text{mod }(k-1,\lceil K/M \rceil)}{\lceil K/M \rceil} < \beta$ **then**
      % `Exploration stage`
      $\theta \leftarrow \theta - \alpha \nabla \tilde{U}_k(\theta)$
    else
      % `Sampling stage`
      Collect samples using SG-MCMC methods
    end if
  end for
Output: Samples $\{\theta_k\}$

To alleviate this problem, we propose to introduce a system temperature T to control the sampler's behaviour: $p(\theta|\mathcal{D}) \propto \exp(-U(\theta)/T)$. Note that the setting $T = 1$ corresponds to sampling from the untempered posterior. When $T \to 0$, the posterior distribution becomes a point mass. Sampling from $\lim_{T\to 0} \exp(-U(\theta)/T)$ is equivalent to minimizing $U(\theta)$; in this context, SG-MCMC methods become stochastic gradient optimization methods.

One may increase the temperature T from 0 to 1 when the step-size is decreasing. We simply consider $T = 0$ and perform optimization as the burn-in stage, when the completed proportion of a cycle $r(k) = \frac{\text{mod }(k-1,\lceil K/M \rceil)}{\lceil K/M \rceil}$ is smaller than a given threshold: $r(k) < \beta$. Note that $\beta \in (0, 1)$ balances the proportion of the exploration and sampling stages in cSG-MCMC.

Sampling

The *sampling* stage corresponds to $T = 1$ of the exploration stage. When $r(k) > \beta$ or step-sizes are sufficiently small, we initiate SG-MCMC updates and collect

samples until this cycle ends.

SG-MCMC with Warm Restarts. One may consider the exploration stage as automatically providing warm restarts for the sampling stage. Exploration alleviates the inefficient mixing and inability to traverse the multimodal distributions of the traditional SG-MCMC methods. SG-MCMC with warm restarts explores different parts of the posterior distribution and captures multiple modes in a single training procedure.

In summary, the proposed cyclical SG-MCMC repeats the *two* stages, with three key advantages: (*i*) It restarts with a large stepsize at the beginning of a cycle which provides enough perturbation and encourages the model to escape from the current mode. (*ii*) The stepsize decreases more quickly inside one cycle than a traditional schedule, making the sampler better characterize the density of the local regions. (*iii*) This cyclical stepsize shares the advantage of the "super-convergence" property discussed in [133]: cSG-MCMC can accelerate convergence for DNNs by up to an order of magnitude.

Connection to the Santa algorithm. It is interesting to note that our approach inverts steps of the Santa algorithm [24] for optimization. Santa is a simulated-annealing-based optimization algorithm with an exploration stage when $T = 1$, then gradually anneals $T \to 0$ in a refinement stage for global optimization. In contrast, our goal is to draw samples for multimodal distributions, thus we explore with $T = 0$ and sample with $T = 1$. Another fundamental difference is that Santa adopts the traditional stepsize decay, while we use the cyclical schedule.

We visually compare the difference between cyclical and traditional step size

schedules (described in Section 5.1) in Figure 5.1. The cyclical SG-MCMC algorithm is presented in Algorithm 9.

Connection to Parallel MCMC. Running parallel Markov chains is a natural and effective way to draw samples from multimodal distributions [140, 2]. However, the training cost increases linearly with the number of chains. Cyclical SG-MCMC can be seen as an efficient way to approximate parallel MCMC. Each cycle effectively estimates a different region of posterior. Note cyclical SG-MCMC runs along a single training pass. Therefore, its computational cost is the same as single chain SG-MCMC while significantly less than parallel MCMC.

Combining Samples. In cyclical SG-MCMC, we obtain samples from multiple modes of a posterior distribution by running the cyclical step size schedule for many periods. We provide a sampling combination scheme to effectively use the collected samples in Section D.2 in the appendix.

5.3 Theoretical Analysis

Our algorithm is based on the SDE characterizing the Langevin dynamics: $\mathrm{d}\theta_t = -\nabla U(\theta_t)\mathrm{d}t + \sqrt{2}\mathrm{d}\mathcal{W}_t$, where $\mathcal{W}_t \in \mathbb{R}^d$ is a d-dimensional Brownian motion. In this section, we prove non-asymptotic convergence rates for the proposed cSG-MCMC framework with a cyclical stepsize sequence $\{\alpha_k\}$ defined in (5.1). For simplicity, we do not consider the exploration stage in the analysis as that corresponds to stochastic optimization. Generally, there are two different ways to describe the convergence behaviours of SG-MCMC. One characterizes the sample average over

a particular test function (*e.g.*, [25, 143]); the other is in terms of the Wasserstein distance (*e.g.*, [119, 149]). We study both in the following.

Weak convergence Following [25] and [143], we define the posterior average of an ergodic SDE as: $\bar{\phi} \triangleq \int_{\mathcal{X}} \phi(\theta)\rho(\theta)\mathrm{d}\theta$ for some test function $\phi(\theta)$ of interest. For the corresponding algorithm with generated samples $(\theta_k)_{k=1}^K$, we use the *sample average* $\hat{\phi}$ defined as $\hat{\phi} = \frac{1}{K}\sum_{k=1}^K \phi(\theta_k)$ to approximate $\bar{\phi}$. We prove weak convergence of cSGLD in terms of bias and MSE, as stated in Theorem 11.

Theorem 11. *Under Assumptions 3 in the appendix, for a smooth test function ϕ, the bias and MSE of cSGLD are bounded as:*

$$\textit{BIAS: } \left|\mathbb{E}\tilde{\phi} - \bar{\phi}\right| = O\left(\frac{1}{\alpha_0 K} + \alpha_0\right), \qquad \textit{MSE: } \mathbb{E}\left(\tilde{\phi} - \bar{\phi}\right)^2 = O\left(\frac{1}{\alpha_0 K} + \alpha_0^2\right) .$$

Convergence under the Wasserstein distance Next, we consider the more general case of SGLD and characterize convergence rates in terms of a stronger metric of 2-Wasserstein distance, defined as:

$$W_2^2(\mu,\nu) := \inf_{\gamma}\left\{\int_{\Omega\times\Omega} \|\theta - \theta'\|_2^2 \mathrm{d}\gamma(\theta,\theta') : \gamma \in \Gamma(\mu,\nu)\right\}$$

where $\Gamma(\mu,\nu)$ is the set of joint distributions over (θ,θ') such that the two marginals equal μ and ν, respectively.

Denote the distribution of θ_t in the SDE as ν_t. According to [28], the stationary distribution ν_∞ matches our target distribution. Let μ_K be the distribution of the sample from our proposed cSGLD algorithm at the K-th iteration. Our goal is to derive a convergence bound on $W_2(\mu_K,\nu_\infty)$. We adopt standard assumptions as in most existing work, which are detailed in Assumption 4 in the appendix. Theorem 12 summarizes our main theoretical result.

Theorem 12. *Under Assumption 4 in the appendix, there exist constants* (C_0, C_1, C_2, C_3) *independent of the stepsizes such that the convergence rate of our proposed cSGLD with cyclical stepsize sequence* (5.1) *is bounded for all* K *satisfying* (K *mod* M *=0), as* $W_2(\mu_K, \nu_\infty) \leq$

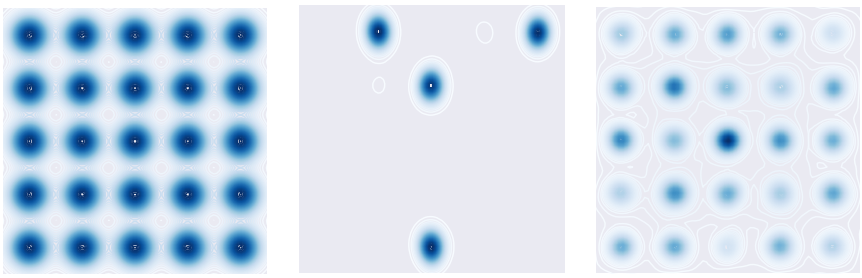
$$C_3 \exp(-\frac{K\alpha_0}{2C_4}) + \left(6 + \frac{C_2 K\alpha_0}{2}\right)^{\frac{1}{2}} \left[(C_1 \frac{3\alpha_0^2 K}{8} + \sigma C_0 \frac{K\alpha_0}{2})^{\frac{1}{2}} + (C_1 \frac{3\alpha_0^2 K}{16} + \sigma C_0 \frac{K\alpha_0}{4})^{\frac{1}{4}}\right] .$$

Particularly, if we further assume $\alpha_0 = O(K^{-\beta})$ *for* $\forall \beta > 1$, $W_2(\mu_K, \nu_\infty) \leq C_3 + \left(6 + \frac{C_2}{K^{\beta-1}}\right)^{\frac{1}{2}} [(\frac{2C_1}{K^{2\beta-1}} + \frac{2C_0}{K^{\beta-1}})^{\frac{1}{2}} + (\frac{C_1}{K^{2\beta-1}} + \frac{C_0}{K^{\beta-1}})^{\frac{1}{4}}]$.

Remark 1. *i) The bound is decomposed into two parts: the first part measures convergence speed of exact solution to the stationary distribution, i.e.,* $\nu_{\sum_k \alpha_k}$ *to* ν_∞*; the second part measures the numerical error, i.e., between* μ_K *and* $\nu_{\sum_k \alpha_k}$*. ii) The overall bound offers a same order of dependency on* K *as in standard SGLD (please see the bound for SGLD in Section D.1.3 of the appendix. See also [119]). iii) If one imposes stricter assumptions such as in the convex case, the bound can be further improved. Specific bounds are derived in the appendix. We did not consider this case due to the discrepancy from real applications.*

5.4 Experiments

We demonstrate cSG-MCMC on several tasks, including a synthetic multimodal distribution (Section 5.4), image classification on Bayesian neural networks (Section 5.4) and uncertainty estimation in Section 5.4. We also demonstrate cSG-MCMC can improve the estimate efficiency for uni-modal distributions using Bayesian logistic regression in Section D.5 in the appendix. We choose SLGD and SGHMC as the representative baseline algorithms. Their cyclical counterpart are called cSGLD and cSGHMC, respectively.

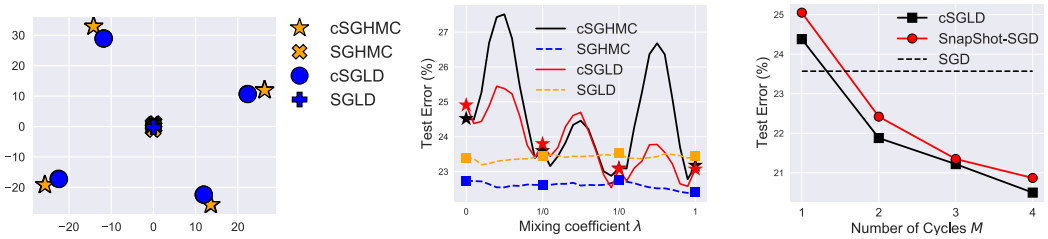


(a) Target (b) SGLD (c) cSGLD

Figure 5.2: Sampling from a mixture of 25 Gaussians shown in (a) for the parallel setting. With a budget of 50k $\times$ 4 =200k samples, traditional SGLD in (b) has only discovered 4 of the 25 modes, while our cSGLD in (c) has fully explored the distribution.

Synthetic multimodal data

We first demonstrate the ability of cSG-MCMC for sampling from a multi-modal distribution on a 2D mixture of 25 Gaussians. Specifically, we compare cSGLD with SGLD in two setting: (1) parallel running with 4 chains and (2) running with a single chain, respectively. Each chain runs for 50k iterations. The stepsize schedule of SGLD is $\alpha_k \propto 0.05k^{-0.55}$. In cSGLD, we set $M = 30$ and the initial stepsize $\alpha_0 = 0.09$. The proportion of exploration stage $\beta = \frac{1}{4}$. Fig 5.2 shows the estimated density using sampling results for SGLD and cSGLD in the parallel setting. We observed that SGLD gets trapped in the local modes, depending on the initial position. In any practical time period, SGLD could only characterize partial distribution. In contrast, cSGLD is able to find and characterize all modes, regardless of the initial position. cSGLD leverages large step sizes to discover a new mode, and small step sizes to explore local modes. This result suggests cSGLD can be a significantly favourable choice in the non-asymptotic setting, for example only 50k iterations in this case. The single chain results and the quantitative results on mode coverage are reported in Section D.5 of the appendix.



(a) MDS (b) Interpolation (c) Comparison

Figure 5.3: Results of cSG-MCMC with DNNs on the CIFAR-100 dataset. (a) MDS visualization in weight space: cSG-MCMC show larger distance than traditional schedules. (b) Testing errors (%) on the path of two samples: cSG-MCMC shows more varied performance. (c) Testing errors (%) as a function of the number of cycles M: cSGLD yields consistently lower errors.

Bayesian Neural Networks

We demonstrate the effectiveness of cSG-MCMC on Bayesian neural networks for classification on CIFAR-10 and CIFAR-100. We compare with (*i*) traditional SG-MCMC; (*ii*) traditional stochastic optimization methods, including stochastic gradient descent (SGD) and stochastic gradient descent with momentum (SGDM); and (*iii*) *Snapshot*: a stochastic optimization ensemble method method with a the cyclical stepsize schedule [73]. We use a ResNet-18 [66] and run all algorithms for 200 epochs. We report the test errors averaged over 3 runs, and the standard error ($\pm$) from the mean predictor.

We set $M = 4$ and $\alpha_0 = 0.5$ for cSGLD, cSGHMC and Snapshot. The proportion hyper-parameter β =0.8 and 0.94 for CIFAR-10 and CIFAR-100, respectively. We collect 3 samples per cycle. In practice, we found that the collected samples share similarly high likelihood for DNNs, thus one may simply set the normalizing term w_i in (D.18) to be the same for faster testing.

We found that *tempering* helps improve performance for Bayesian inference with neural networks. Tempering for SG-MCMC was first used by [85] as a

	CIFAR-10	**CIFAR-100**
SGD	5.29±0.15	23.61±0.09
SGDM	5.17±0.09	22.98±0.27
Snapshot-SGD	4.46±0.04	20.83±0.01
Snapshot-SGDM	4.39±0.01	20.81±0.10
SGLD	5.20±0.06	23.23±0.01
cSGLD	4.29±0.06	20.55±0.06
SGHMC	4.93±0.1	22.60±0.17
cSGHMC	**4.27**±0.03	**20.50**±0.11

Table 5.1: Comparison of test error (%) between cSG-MCMC with non-parallel algorithms. cSGLD and cSGHMC yields lower errors than their optimization counterparts, respectively.

practical technique for neural network training for fast convergence in limited time[1]. We simply use the prescribed temperature of [85] without tuning, but better results of the sampling methods can be achieved by tuning the temperature. More details are in Appendix D.4. We hypothesize that tempering helps due to the overparametrization of neural networks. Tempering enables one to leverage the inductive biases of the network, while representing the belief that the model capacity can be misspecified. In work on *Safe Bayes*, also known as *generalized* and *fractional* Bayesian inference, tempered posteriors are well-known to help under misspecification [12, 36, 62].

For the traditional SG-MCMC methods, we found that noise injection early in training hurts convergence. To make these baselines as competitive as possible, we thus avoid noise injection for the first 150 epochs of training (corresponding to the zero temperature limit of SGLD and SGHMC), and resume SGMCMC as usual (with noise) for the last 50 epochs. This scheme is similar to the exploration and sampling stages within one cycle of cSG-MCMC. We collect 20 samples for the MCMC methods and average their predictions in testing.

[1]https://github.com/ChunyuanLI/pSGLD/issues/2

Testing Performance for Image Classification We report the testing errors in Table 5.1 to compare with the non-parallel algorithms. Snapshot and traditional SG-MCMC reduce the testing errors on both datasets. Performance variance for these methods is also relatively small, due to the multiple networks in the Bayesian model average. Further, cSG-MCMC significantly outperforms Snapshot ensembles and the traditional SG-MCMC, demonstrating the importance of (1) capturing diverse modes compared to traditional SG-MCMC, and (2) capturing fine-scale characteristics of the distribution compared with Snapshot ensembles.

Diversity in Weight Space. To further demonstrate our hypothesis that with a limited budget cSG-MCMC can find diverse modes, while traditional SG-MCMC cannot, we visualize the 12 samples we collect from cSG-MCMC and SG-MCMC on CIFAR-100 respectively using Multidimensional Scaling (MDS) in Figure 5.3 (a). MDS uses a Euclidean distance metric between the weight of samples. We see that the samples of cSG-MCMC form 4 clusters, which means they are from 4 different modes in weight space. However, all samples from SG-MCMC only form one cluster, which indicates traditional SG-MCMC gets trapped in one mode and only samples from that mode.

Diversity in Prediction. To further demonstrate the samples from different cycles of cSG-MCMC provide diverse predictions we choose one sample from each cycle and linearly interpolate between two of them [58, 73]. Specifically, let $J(\theta)$ be the test error of a sample with parameter θ. We compute the test error of the convex combination of two samples $J(\lambda\theta_1 + (1-\lambda)\theta_2)$, where $\lambda \in [0, 1]$.

We linearly interpolate between two samples from neighboring chains of cSG-MCMC since they are the most likely to be similar. We randomly select 4 samples

Method	Cyclical+Parallel		Decreasing+Parallel		Decreasing+Parallel		Cyclical+Single	
Cost	200/800		200/800		100/400		200/200	
Sampler	SGLD	SGHMC	SGLD	SGHMC	SGLD	SGHMC	SGLD	SGHMC
CIFAR-10	4.09	3.95	4.15	4.09	5.11	4.52	4.29	4.27
CIFAR-100	19.37	19.19	20.29	19.72	21.16	20.82	20.55	20.50

Table 5.2: Comparison of test error (%) between cSG-MCMC with parallel algorithm (M=4 chains) on CIFAR-10 and CIFAR-100. The method is reported in the format of "step-size schedule (cyclical or decreasing) + single/parallel chain". The cost is reported in the format of "#epoch per chain / #epoch used in all chains". Note that a parallel algorithm with a single chain reduces to a non-parallel algorithm. Integration of the cyclical schedule with parallel algorithms provides lower testing errors.

from SG-MCMC. If the samples are from the same mode, the test error of the linear interpolation of parameters will be relatively smooth, while if the samples are from different modes, the test error of the parameter interpolation will have a spike when λ is between 0 and 1.

We show the results of interpolation for cSG-MCMC and SG-MCMC on CIFAR-100 in Figure 5.3 (b). We see a spike in the test error in each linear interpolation of parameters between two samples from neighboring chains in cSG-MCMC while the linear interpolation for samples of SG-MCMC is smooth. This result suggests that samples of cSG-MCMC from different chains are from different modes while samples of SG-MCMC are from the same mode.

Although the test error of a single sample of cSG-MCMC is worse than that of SG-MCMC shown in Figure 5.3 (c), the ensemble of these samples significantly improves the test error, indicating that samples from different modes provide different predictions and make mistakes on different data points. Thus these diverse samples can complement each other, resulting in a lower test error, and demonstrating the advantage of exploring diverse modes using cSG-MCMC.

Comparison to Parallel MCMC. cSG-MCMC can be viewed as an economical alternative to parallel MCMC. We verify how closely cSG-MCMC can approximate the performance of parallel MCMC, but with more convenience and less computational expense. We also note that we can improve parallel MCMC with the proposed cyclical stepsize schedule.

We report the testing errors in Table 5.2 to compare multiple-chain results. (1) Four chains used, each runs 200 epochs (800 epochs in total), the results are shown in the first 4 columns (Cyclical+Parallel vs Decreasing+Parallel). We see that cSG-MCMC variants provide lower errors than plain SG-MCMC. (2) We reduce the number of epochs (#epoch) of parallel MCMC to 100 epoch each for decreasing stepsize schedule. The total cost is 400 epochs. We compare its performance with cyclical single chain (200 epochs in total) in the last 4 columns (Decreasing+Parallel vs Cyclical+Single). We see that the cyclical schedule running on a single chain performs best even with half the computational cost! All the results indicate the importance of warm re-starts using the proposed cyclical schedule. For a given total cost budget, the proposed cSGMCMC is preferable to parallel sampling.

Comparison to Snapshot Optimization. We carefully compared with Snapshot, as our cSG-MCMC can be viewed as the sampling counterpart of the Snapshot optimization method. We plot the test error *wrt.*various number of cycles M in Fig. 5.3. As M increases, cSG-MCMC and Snapshot both improve. However, given a fixed M, cSG-MCMC yields substantially lower test errors than Snapshot. This result is due to the ability of cSG-MCMC to better characterize the local distribution of modes: Snapshot provides a singe minimum per cycle, while cSG-MCMC fully exploits the mode with more samples, which could provide weight uncertainty estimate and avoid over-fitting.

	NLL ↓	Top1 ↑	Top5 ↑
SGDM	0.9595	76.046	92.776
Snapshot-SGDM	0.8941	77.142	93.344
SGHMC	0.9308	76.274	92.994
cSGHMC	**0.8882**	77.114	93.524

Table 5.3: Comparison on the testing set of ImageNet. cSGHMC yields lower testing NLL than Snapshot and SGHMC.

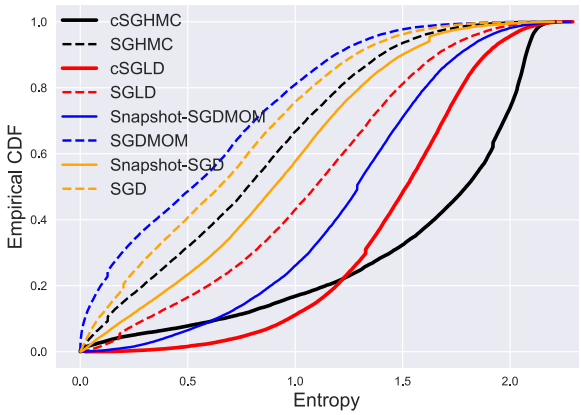


Figure 5.4: Empirical CDF for the entropy of the predictive distribution on notMNIST dataset. cSGLD and cSGHMC show lower probability for the low entropy estimate than other algorithms.

Results on ImageNet. We further study different learning algorithms on a large-scale dataset, ImageNet. ResNet-50 is used as the architecture, and 120 epochs for each run. The results on the testing set are summarized in Table 5.3, including NLL, Top1 and Top5 accuracy (%), respectively. 3 cycles are considered for both cSGHMC and Snapshot, and we collect 3 samples per cycle. We see that cSGHMC yields the lowest testing NLL, indicating that the cycle schedule is an effective technique to explore the parameter space, and diversified samples can help prevent over-fitting.

Uncertainty Evaluation

To demonstrate how predictive uncertainty benefits from exploring multiple modes in the posterior of neural network weights, we consider the task of uncertainty estimation for out-of-distribution samples [82]. We train a three-layer MLP model on the standard MNIST train dataset until convergence using different algorithms, and estimate the entropy of the predictive distribution on the notMNIST dataset [21]. Since the samples from the notMNIST dataset belong to the unseen classes, ideally the predictive distribution of the trained model should be uniform over the notMNIST digits, which gives the maximum entropy.

In Figure 5.4, we plot the empirical CDF for the entropy of the predictive distributions on notMNIST. We see that the uncertainty estimates from cSGHMC and cSGLD are better than the other methods, since the probability of a low entropy prediction is overall lower. cSG-MCMC algorithms explore more modes in the weight space, each mode characterizes a meaningfully different representation of MNIST data. When testing on the out-of-distribution dataset (notMNIST), each mode can provide different predictions over the label space, leading to more reasonable uncertainty estimates. Snapshot achieves less entropy than cSG-MCMC, since it represents each mode with a single point.

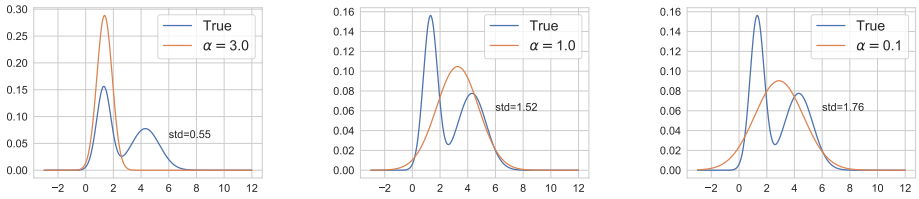
The traditional SG-MCMC methods also provide better uncertainty estimation compared to their optimization counterparts, because they characterize a local region of the parameter space, rather than a single point. cSG-MCMC can be regarded as a combination of these two worlds: a wide coverage of many modes in Snapshot, and fine-scale characterization of local regions in SG-MCMC.

CHAPTER 6

META-LEARNING DIVERGENCES FOR VARIATIONAL INFERENCE

Approximate inference is a powerful tool for probabilistic modelling of complex data. Among these inference methods, variational inference (VI) [76, 152] approximates the intractable target distribution through optimizing a tractable distribution. This optimization-based inference makes VI computationally efficient, thus suitable to large-scale models in deep learning, such as Bayesian neural networks [16] and deep generative models [79]. The objective function in VI is a divergence which measures the discrepancy between the approximate distribution and the target distribution. As an objective function, this divergence significantly affects the inductive bias of the VI algorithm. By selecting a divergence, we encode our preference to the approximate distribution, such as whether it should be mass-covering or mode-seeking. The Kullback-Leibler (KL) divergence is one of the most widely used divergence metrics. However, it has been criticized for under-estimating uncertainty, leading to poor results when uncertainty estimation is essential [14, 15, 144]. Many alternative divergences have been proposed to alleviate this issue [8, 33, 68, 90, 105, 144].

Although prior work has enriched the divergence family, the optimal divergence metric usually depends on tasks [105, 90]. As illustrated by Figure 6.1, different divergence metrics can lead to very different inference results. Unfortunately, choosing a divergence for a specific task is challenging as it requires a thorough understanding of (i) the shape of the target distribution; (ii) the desirable properties of the approximate distribution; and (iii) the bias-variance trade-off of the variational bound. A crucial question remains to be addressed in order to make VI a success:



(a) $\alpha = 3.0$ (b) $\alpha = 1.0$ (c) $\alpha = 0.1$

Figure 6.1: An illustration of approximate distributions on a Gaussian mixture by different α-divergences (defined in Eq.(2.5)). "std" is the standard deviation of the Gaussian approximation.

how can we automatically choose a suitable divergence tailored to specific types of task?

To answer this question, we propose meta-learning divergences of variational inference which utilizes meta-learning, or learning to learn, to refine VI's divergence automatically. In a nutshell, we leverage the fact that various real-world applications consist of many small tasks (e.g. personalized recommendations for different user groups in recommender systems), and it is important to design a meta-learning algorithm to learn a good inference algorithm for new tasks from previous tasks. We summarize our contributions as follows:

- We develop a general framework for meta-learning variational inference's divergence (Section 6.2.1), which chooses the desired divergence objective automatically given a type of tasks. In this way, we meta-learn the VI algorithm.
- Besides meta-learning the divergence objective, we further meta-learn the parameters for the variational distribution without additional cost (Section 6.2.2), enabling meta-learning VI in few-shot setting.
- We demonstrate VI with meta-learned divergences outperforms standard VI on Gaussian mixture distribution approximation, Bayesian neural network

regression, image generation with variational autoencoders, and recommender systems with a partial variational autoencoder (Section 6.3).

6.1 Preliminaries and Related Work

The goal of meta-learning VI algorithm is to learn, from a set of tasks, a VI algorithm that produces an approximate distribution with desired properties on new similar tasks. We approach this goal by learning the divergence in use for VI. We formalize the problem setups as follows.

Assume we have a task distribution $p(\mathcal{T})$. Each task $T_i \sim p(\mathcal{T})$ has its own dataset $\mathcal{D}_{T_i}$ and its own probabilistic model $p_{T_i}(\theta_i, \mathcal{D}_{T_i})$. Let $D_\eta(\cdot\|\cdot)$ denote a learnable divergence parameterized by η; then for each task T_i the approximate posterior $q_{\phi_i}(\theta_i)$ is computed by minimizing $D_\eta(p_{T_i}(\theta_i|\mathcal{D}_{T_i})\|q_{\phi_i}(\theta_i))$. In the rest of the section we write $D_\eta(q_{\phi_i}, T_i) = D_\eta(p_{T_i}(\theta_i|\mathcal{D}_{T_i})\|q_{\phi_i}(\theta_i))$ for brevity. To do meta-training, in each step we first sample a minibatch of tasks $T_i, i = 1, \ldots, M$ from $p(\mathcal{T})$. Then we define a meta-loss function $\mathcal{J}(q_{\phi_i}, T_i)$, and optimize the total meta-loss across all training tasks in the minibatch $\sum_{i=1}^{M} \mathcal{J}(q_{\phi_i}, T_i)$ over the divergence parameter η. This meta-loss function is designed to evaluate the desired properties of the approximate distribution for these tasks, e.g. negative log-likelihood. During meta-testing, a new task is sampled from $p(\mathcal{T})$, and the learned divergence D_η is used to optimize the variational distribution q_ϕ.

We also consider (in Section 6.2.2) a few-shot learning setup similar to the model-agnostic meta-learning (MAML) framework [44]. In this case, each task only has a few training data, therefore it is crucial to learn a good model initialization to avoid overfitting and adapt fast on unseen tasks. The goal of meta-learning VI

algorithm in this setting is to obtain a divergence as well as an initialization of the variational parameters ϕ for unseen tasks. During meta-testing, we will train the model with the learned divergence and the learned initialization of variational parameters on new tasks.

The above two meta-learning settings are practical as demonstrated in many previous works [44, 45, 57, 78], showing that attaining common knowledge from previous tasks is valuable for future tasks.

Related Work

Variational Inference Variational inference (VI) has advanced rapidly in recent years [152]. These advances can be grouped into three categories: (1) introduction of new divergences for VI [8, 68, 90]; (2) introduction of more expressive approximate families [125, 122]; (3) improvement of sampling estimates for model evidence [22] and gradient [120]; (4) stochastic optimization to scale VI [38, 69, 89]. Our work is related to the work that improves the variational objective with alternative divergence measures; the difference is that our divergence measure is learnable and can be selected in an automatic fashion for a certain type of tasks.

Meta-Learning/few-shot learning Recent work has applied Bayesian modelling techniques to enhance uncertainty estimate for meta-learning/few-shot learning [45, 59, 78, 123]. They view the framework of MAML [44] as hierarchical Bayes and conduct Bayesian inference on meta-parameters and/or task-specific parameters. [59] and [78] applied approximate Bayesian inference to task-specific parameters, while [45] kept point estimate for task-specific parameters and conducted variational inference over the meta-parameters instead. [123] obtained posteriors

over both meta and task-specific parameters with variational inference. Our focus is distinct from this line of work in that our research is in the opposite direction: leveraging the idea of meta-learning to advance Bayesian inference. Additionally, our meta-$D\&\phi$ without learning divergence (VI$\&\phi$) can be viewed as a different Bayesian MAML method other than hierarchical Bayes, which directly trains the variational parameters so that it can quickly adapt to new tasks.

Meta-Learning for loss functions Our meta-learning method is also related to meta-learning a loss function. In reinforcement learning, [72] meta-learned the loss function for policy gradients where the parameters of the loss function is updated using evolutionary strategies. [150] meta-learned the hyperparameters of the loss functions in TD(λ) and IMPALA. Our work extends the idea of a learnable loss function to Bayesian inference.

Meta-Learning for Bayesian inference algorithms A recent attempt to meta-learning stochastic gradient MCMC (SG-MCMC) is presented by [57], which proposed to meta-learn the diffusion and curl matrices of the SG-MCMC's underlying stochastic differential equation. Also [145] applied meta-learning to build efficient and generalizable block-Gibbs sampling proposals. Our work is distinct from previous work in that we apply meta-learning to improve VI, which is a more scalable inference method than MCMC. To the best of our knowledge, we are the first to study the automatic choice and design of VI algorithms.

6.2 Meta-VI

6.2.1 Meta-Learning Divergences (meta-D)

We consider the first setting of learning a divergence. We assume for now D_η is given in some parametric form; later on we will provide the details of parameterization of two divergence families (α- and f-divergence) and show how they fit in this framework. The general idea is to first optimize the approximate posterior by minimizing the current divergence, then update the divergence using the feedback from the meta-loss. Concretely, for each task T_i we perform B gradient descent steps on the variational parameters ϕ_i using VI with the current divergence D_η:

$$\phi_i \leftarrow \phi_i - \beta \nabla_{\phi_i} D_\eta(q_{\phi_i}, T_i). \tag{6.1}$$

By doing so the updated variational parameters ϕ_i are a function of the divergence parameter η, which we then update by one-step gradient descent using the meta-loss $\mathcal{J}$:

$$\eta \leftarrow \eta - \gamma \nabla_\eta \frac{1}{M} \sum_i \mathcal{J}(q_{\phi_i}, T_i). \tag{6.2}$$

We call this algorithm *meta-D* for meta-learning divergences, which is outlined in Algorithm 10. Our algorithm is different from MAML in that MAML's inner and outer loop losses are designed to be the same, prohibiting it to meta-learn the inner loop loss function which is the divergence in VI. The key insight of our approach is that the updated variational parameters are dependant on the inner loop divergence. This dependency enables meta-D to update the divergence by descending the meta-loss with back-propagation through the variational parameters.

Algorithm 10 Meta-D

Input: $p(\mathcal{T})$: distribution over tasks; β, γ: learning rate hyperparameters; initialize η
loop
  Sample M tasks $T_i \sim p(\mathcal{T})$
  for all T_i **do**
    if ϕ_i does not exist **then**
      initialize ϕ_i (can have different architectures)
    end if
    Update ϕ_i with the current divergence:
    for $b = 1 : B$ **do**
      $\phi_i \leftarrow \phi_i - \beta \nabla_{\phi_i} D_\eta(q_{\phi_i}, T_i)$
    end for
  end for
  Update $\eta \leftarrow \eta - \gamma \nabla_\eta \frac{1}{M} \sum_i \mathcal{J}(q_{\phi_i}, T_i)$
end loop
Output: η

Algorithm 11 Meta-$D\&\phi$

Input: $p(\mathcal{T})$: distribution over tasks; β, γ, τ: learning rate hyperparameters
Initialize ϕ, η
loop
  Sample M tasks $T_i \sim p(\mathcal{T})$
  for all T_i **do**
    Update ϕ_i with the current divergence:
    for $b = 1 : B$ **do**
      $\phi_i \leftarrow \phi - \beta \nabla_{\phi} D_\eta(q_{\phi}, T_i)$
    end for
  end for
  Update $\phi \leftarrow \phi - \tau \nabla_\phi \frac{1}{M} \sum_i \mathcal{J}(q_{\phi_i}, T_i)$;
  $\eta \leftarrow \eta - \gamma \nabla_\eta \frac{1}{M} \sum_i \mathcal{J}(q_{\phi_i}, T_i)$
end loop
Output: η, ϕ

Meta-learning within α-divergence family To make α-divergence learnable by the meta-D framework (in this case $\eta = \alpha$), it requires the inner-loop updates (Eq.(6.1)) to be continuous in α. This means a naive solution which relies on automatic differentiation of existing α-divergences will fail, due to the fact that these α-divergences are not twice differentiable everywhere [90, 105]. Instead,

we propose to manually compute the gradient of Renyi's α-divergence (Eq.(2.6)) which is continuous in $\alpha \in (0, +\infty)$. Specifically we parameterize α-divergence by parameterizing its gradient (Eq.(2.6)) and set $\nabla_{\phi_i} D_\eta = -\nabla_{\phi_i} \mathcal{L}_\alpha$ in Algorithm 10. We denote meta-learning a divergence within α-divergences family as *meta-α*.

Meta-learning within f-divergence family We wish to parameterize the f-divergence Eq.(2.7) by parameterizing the convex function f using a neural network, since neural networks are known to be universal approximators and thus can cover diverse f-divergences. However, it is less straightforward to specify the convexity constraint for neural networks. Fortunately, Proposition 1 below indicates that the f-divergence and its gradient can be specified through its second derivative f'' [144].

Proposition 1. *If $\nabla_\theta \log\left(\frac{p(\theta)}{q_\phi(\theta)}\right)$ exists, then by setting $g_f(t) = t^2 \cdot f''(t)$, we have (with $\theta = r_\phi(\epsilon)$)*

$$\begin{aligned} &\nabla_\phi D_f(p\|q_\phi) \\ &= -\mathbf{E}_{\epsilon\sim p(\epsilon)}\left[g_f\left(\frac{p(\theta)}{q_\phi(\theta)}\right)\nabla_\phi r_\phi(\epsilon)\nabla_\theta \log\left(\frac{p(\theta)}{q_\phi(\theta)}\right)\right]. \end{aligned} \tag{6.3}$$

Therefore it remains to specify g (or f''), and the following Proposition 2 guarantees that using non-negative functions as g is sufficient for parameterizing the f-divergence family.

Proposition 2. *For any non-negative function g on $\mathbb{R}_+$, there exists a function f such that $g(t) = g_f(t) = t^2 \cdot f''(t)$. If $g_f(1) > 0$, then $D_f(p\|q_\phi) = 0$ implies $p = q_\phi$.*

See [144] for the proofs. Given these guarantees, we propose to parameterize f implicitly by parameterizing $g(t) = g_f(t)$ which can be any non-negative function.

We turn the problem into using a neural network to express a non-negative function that is strictly positive at $t = 1$. For convenience, we further restrict the form of the function to be

$$g(t) = \exp(h_\eta(t)) \tag{6.4}$$

where $h_\eta(t)$ is a neural network with parameter η. This definition of g is strictly positive for all t, satisfying the assumption of Proposition 2. By doing so, the f-divergence is now learnable through Algorithm 10, by computing the gradient $\nabla_{\phi_i} D_\eta = \nabla_{\phi_i} D_{f_\eta}$ with Eq. (6.3).

With dataset $\mathcal{D}$, the density ratio in Eq. (6.3) becomes $\frac{p(\theta|D)}{q_\phi(\theta)} = \frac{p(\mathcal{D}|\theta)p(\theta)}{q_\phi(\theta)p(\mathcal{D})}$. We estimate $p(\mathcal{D})$ through importance sampling and MC approximation. After doing this, $\frac{p(\theta_k|\mathcal{D})}{q_\phi(\theta_k)} = \frac{p(\mathcal{D}|\theta_k)p(\theta_k)}{q_\phi(\theta_k)} \Big/ \frac{1}{K}\sum_{k=1}^{K} \frac{p(\mathcal{D}|\theta_k)p(\theta_k)}{q_\phi(\theta_k)}$ which can be regarded as a self-normalized estimator (see Appendix E.1 for details).

Our method is different from [144] in the way that we use deep neural networks parameterization and enable learning the f-divergence through standard optimization. We denote meta-learning a divergence within f-divergences family as *meta-f*.

6.2.2 Meta-Learning Divergences and Variational Parameters (meta-D&ϕ)

In addition to learning the divergence objective, we also consider the few-shot setting where fast adaptation of the variational parameters to new tasks is desirable. Similar to MAML, the probabilistic models $\{p_{T_i}(\theta_i, \mathcal{D}_{T_i})\}$ share the same architecture, and the goal is to learn an initialization of variational parameters $\phi_i \leftarrow \phi$. On a specific

task, ϕ is adapted to be ϕ_i according to the learnable divergence D_η (which can be $-\mathcal{L}_\alpha$ or D_{f_η}):

$$\phi_i \leftarrow \phi - \beta \nabla_\phi D_\eta(q_\phi, T_i). \tag{6.5}$$

The updated ϕ_i is a function of both η and ϕ. For meta-update, besides updating divergence parameter η with Eq.(6.2), we also use the same meta-loss to update

$$\phi \leftarrow \phi - \tau \nabla_\phi \frac{1}{M} \sum_i \mathcal{J}(q_{\phi_i}, T_i). \tag{6.6}$$

We call this algorithm *meta-D&ϕ* which meta-learns both the divergence objective and variational parameters' initialization. It is summarized in Algorithm 11. Similar to the previous section, the divergence families in consideration are α- and f-divergence (denoted as meta-α&ϕ and meta-f&ϕ respectively).

6.3 Experiments

We evaluate the proposed approaches on a variety of tasks. For the mixture of Gaussians task, we perform distribution approximation (no data) and use different meta-losses to directly demonstrate the ability of meta-D (meta-learning divergences) and meta-D&ϕ (meta-learning divergences and variational parameters) to learn the optimal divergence. For all other experiments, we use negative log-likelihood as the meta-loss. For meta-D, we use standard VI (KL divergence) and VI with $\alpha = 0.5$ divergence which is a comonly used α-divergence [89, 144] as baselines. For meta-D&ϕ, we test it in few-shot setup (i.e. few training data), and compare it to learning ϕ only which is obtained by Algorithm 11 without updating η. During meta-testing, we test this learned ϕ with KL divergence (denoted by VI&ϕ). We also include results of VI without learning initialization in the few-shot setup

Table 6.1: Meta-D on MoG: learned value of α. BO (8 iters) has similar runtime as meta-α.

Methods	$\alpha = 0.5$	TV
meta-α	0.52±0.01	0.31±0.01
BO (8 iters)	0.81±0.03	0.69±0.08
BO (16 iters)	0.54±0.07	0.32±0.03

Table 6.2: Meta-D on MoG: rank of meta-loss over 10 test tasks.

Methods	$\alpha = 0.5$	TV
meta-α	**2.10**±0.70	2.10±0.30
meta-f	**2.10**±1.37	**1.00**±0.00
BO (8 iters)	3.50±0.67	4.00±0.00
BO (16 iters)	2.30±0.90	2.90±0.30

as a reference to show the gain of meta-learning initialization. Unless otherwise specified, we set $B = 1$. We discussed the effect of this hyperparameter in Appendix E.2 and put details of experimental setting in Appendix E.3.

Approximate Mixture of Gaussians (MoG)

We first verify the ability of our methods on learning good divergences using a 1-d distribution approximation problem. Each task includes approximating a mixture of two Gaussians p by a Gaussian distribution q_{ϕ^*} attained from $\min_\phi D_\eta(p\|q_\phi)$. The mixture of Gaussian distribution $p(\theta) = 0.5\mathcal{N}(\theta; \mu_1, \sigma_1^2) + 0.5\mathcal{N}(\theta; \mu_2, \sigma_2^2)$ is generated by

$$\mu_1 \sim \text{Unif}[0, 3], \quad \sigma_1 \sim \text{Unif}[0.5, 1.0];$$
$$\mu_2 = \mu_1 + 3, \quad \sigma_2 = \sigma_1 * 2.$$

Therefore each task has a different target distribution but with similar properties (the same $\mu_2 - \mu_1$ and σ_2/σ_1). As shown in Figure 6.1, the divergence choice has significant impact on the approximation.

We test our methods with two types of meta-loss $\mathcal{J}$: $D_{0.5}(q\|p)$ and total variation (TV). If $D_{0.5}(q\|p)$ is the metric we care about when evaluating the quality of approximation q, then a good divergence will be $D_{0.5}(q\|p)$ itself. This case is to verify our method is able to learn the preferred divergence given a rich enough

family $\{D_\eta\}$. In practice, the desired evaluation metric for approximation quality (e.g. log-likelihood) typically does not belong to α- or f-divergence family; to test this scenario we use the total variation distance (TV) to evaluate the performance of our method when meta-loss is beyond the divergence family.

We first test meta-D (meta-learning the divergences, Algorithm 10). As a baseline, we treat α as a hyperprameter and use Bayesian optimization (BO) [134] to optimize it. Note that BO is not applicable when the divergence set is f-divergence which is parameterized by a neural network, therefore BO is only used as a baseline for meta-α.

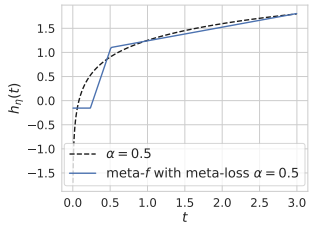
We report the learned values of α from meta-α and BO in Table 6.1. When the meta-loss is $D_{0.5}$, the learned α from meta-α is very close to 0.5, confirming that our method can pick up a desired divergence. Note that BO is less computationally efficient, as it needs to train a model from scratch every single time when evaluating a new value of α, while our method can update α based on the current model.[1] We test learning f-divergence and visualize the learned $h_\eta(t)$ (Eq.(6.4)) in Figure 6.2(a)&(b). When the meta-loss is $D_{0.5}(q\|p)$, the corresponding $h_{0.5}(t)$ for $D_{0.5}$ is analytical (Appendix E.3), and we see from Figure 6.2(a) that the learned $h_\eta(t) \approx h_{0.5}(t)+1.25$. This means meta-$D$ has learned the optimal divergence $D_{0.5}$, since $f(t)$ and $af(t)$ define the same divergence for $\forall a > 0$.

When the meta-loss is TV, the optimal divergence is not analytic. Therefore, we instead report the averaged rank of meta-losses on 10 test tasks in Table 6.2 (see Table E.1 in Appendix for averaged value of meta-losses). It clearly shows that meta-α and meta-f are superior over BO. Moreover, meta-f outperforms meta-α when

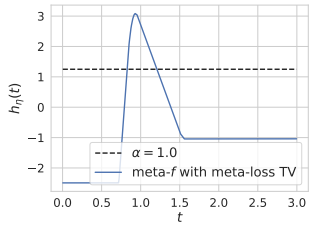
[1] We also considered BO in later sections but found it very inefficient (e.g. on the experiment in Section 6.3, BO can only conduct two searches given similar runtime as our methods) thus omitted the results.

Table 6.3: Meta-$D\&\phi$ on MoG: rank of meta-loss over 10 test tasks.

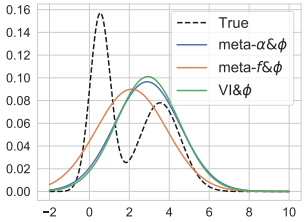
Method	$\alpha = 0.5$ (20 iters)	TV (20 iters)	$\alpha = 0.5$ (100 iters)	TV (100 iters)
VI&ϕ	2.70±0.46	2.70±0.46	2.40±0.49	2.50±0.50
meta-$\alpha\&\phi$	2.10±0.54	1.80±0.60	2.20±0.75	**1.40**±0.66
meta-$f\&\phi$	**1.20**±0.60	**1.50**±0.81	**1.40**±0.80	2.10±0.83



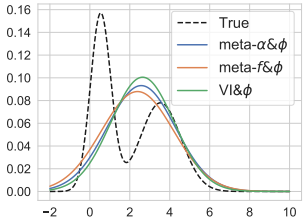
(a) Meta-loss: $\alpha = 0.5$



(b) Meta-loss: TV



(c) Meta-loss: $\alpha = 0.5$



(d) Meta-loss: TV

Figure 6.2: Visualization of (a)-(b) learned h_η and (c)-(d) approximate distribution after 20 updates. Meta-f refers to meta-learning divergences (Algorithm 10) within f-divergences. Meta-$\alpha\&\phi$ and Meta-$f\&\phi$ refer to meta-learning divergences and variational parameters (Algorithm 11) within α- and f-divergences respectively. VI&ϕ refers to meta-learning varitional parameters only.

the meta-loss is TV. From Figure 6.2(b), we can see that the learned f-divergence is not inside α-divergence, showing the benefit of using a larger divergence family. It also indicates that our f-divergence parameterization using a neural network is flexible and can lead to new f-divergences that are not used before.

Next we test meta-$D\&\phi$ (meta-learning divergences and variational parameters, Algorithm 11). During training, we perform $B = 20$ inner loop gradient updates. The learned α is 0.88 and 0.77 for meta-loss $D_{0.5}$ and TV respectively, which is

different from those reported in Table 6.1. We conjecture that this is related to the learned ϕ and B (the horizon length). During meta-testing, we start from the learned ϕ and train the variational parameters with the learned divergence for 20 and 100 iterations, corresponding to short and long horizons respectively. Table 6.3 summarizes the rankings. Our methods are better than VI&ϕ (which uses KL and only meta-learns ϕ) in all cases, demonstrating the benefit of learning a task-specific divergence instead of using the conventional VI for all. To further elaborate, we visualize in Figure 6.2(c)&(d) the approximate distributions after 20 steps. The q distributions obtained by meta-D&ϕ tend to fit the MoG more globally (mass-covering), resulting in better meta-losses when compared with VI&ϕ. Compared to Algorithm 10, Algorithm 11 helps shorten the training time on new tasks (100 v.s. 2000 iterations). Notably, meta-D&ϕ is able to provide this initialization along with divergence learning without extra cost.

Regression Tasks with Bayesian Neural Networks

The second test considers Bayesian neural network regression. The distribution of ground truth regression function is defined by a sinusoid function with heteroskedastic noise (which is a function of x, see Figure 6.3(a)): $y = A\sin(x+b) + A/2\,|\cos((x+b)/2)|\,\epsilon$, where the amplitude $A \in [5, 10]$, the phase $b \in [0, 1]$ and $\epsilon \sim \mathcal{N}(0, 1)$. The heteroskedastic noise makes the uncertainty estimate more crucial comparing with the sinusoid function fitting task in prior work [44, 78].

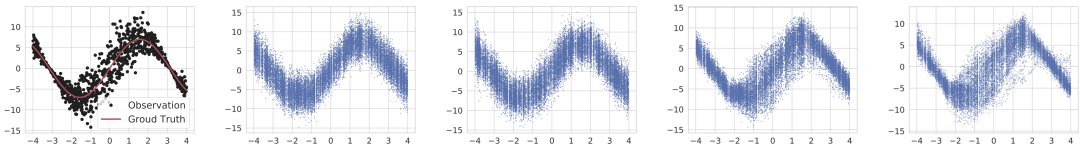
For Meta-D (meta-learning divergences, Algorithm 10), the quantitative results are summarized in Table 6.4. We can see that the test log-likelihood (LL) of both meta-α and meta-f are significantly better than VI and VI ($\alpha = 0.5$), while the root mean square error (RMSE) are similar for all methods. We visualize the

Table 6.4: Meta-D on sin: 10 test tasks and each task has 1000 training data (1000 epochs).

	Test LL	RMSE
VI	-0.59±0.01	0.44±0.01
VI ($\alpha = 0.5$)	-0.57±0.02	0.43±0.01
meta-α	**-0.39**±0.04	0.43±0.00
meta-f	-0.40±0.04	0.42±0.02

Table 6.5: Meta-$D\&\phi$ on sin: 10 test tasks and each task has 40 training data (300 epochs).

	Test LL	RMSE
VI	-3.94±0.18	0.51±0.02
VI$\&\phi$	-0.69±0.04	0.44±0.02
meta-$\alpha\&\phi$	**-0.43**±0.05	0.42±0.03
meta-$f\&\phi$	-0.46±0.04	0.43±0.02



(a) Ground Truth (b) VI (c) VI ($\alpha = 0.5$) (d) meta-α (e) meta-f

Figure 6.3: Meta-D for BNN regression: visualizing the predictive distributions on sinusoid data. With our proposed method to meta-learn the divergence (panels (d) and (e)), the learned distribution can accurately capture the uncertainty in different regions while with vanilla VI (panel (b)) or VI with typical $\alpha = 0.5$ fails to capture the varying uncertainty in different regions.

predictive distribution on an example sinusoid function in Figure 6.3. All methods fit the mean well which is consistent with the RMSE results. Meta-α and meta-f can reason about the heteroskedastic noise whereas VI and VI ($\alpha = 0.5$) used homoskedastic noise to fit the data resulting in bad test LL.

For Meta-$D\&\phi$ (meta-learning divergences and variational parameters, Algorithm 11), during meta-testing, we fine-tune the learned ϕ with learned divergence on 40 datapoints for 300 epochs. Again meta-$\alpha\&\phi$ and meta-$f\&\phi$ are able to model heteroskedastic predictive distribution while VI$\&\phi$ cannot. The quantitative results are reported in Table 6.5, and an example of predictive distribution is visualised in Figure E.3 (see Appendix). Meta-$D\&\phi$ achieves similar results as meta-D with only 40 training data and 300 epochs. Methods without learning initialization for this setup significantly under-perform, indicating that learning model initialization is essential when data is scarce.

Table 6.6: Meta-D (meta-learning divergences) on MNIST: marginal log-likelihood on 5 test tasks. Each task has 6000 training data. We train the model for 1000 epochs during meta-testing.

Digit	5	6	7	8	9
VI	-133.69± 0.23	-121.80±0.15	-92.25±0.40	-145.14±0.19	-119.64±0.23
VI ($\alpha = 0.5$)	-133.24±0.16	-121.90±0.71	-91.52±0.72	-144.90±0.31	-119.59±0.90
meta-α	**-132.74**±0.33	**-120.67**±0.36	**-90.62** ±0.45	-145.13±0.96	**-119.42**±0.36
meta-f	-133.21±0.44	-121.10± 0.20	-91.80±0.28	**-144.85**±0.31	**-119.42**±0.15

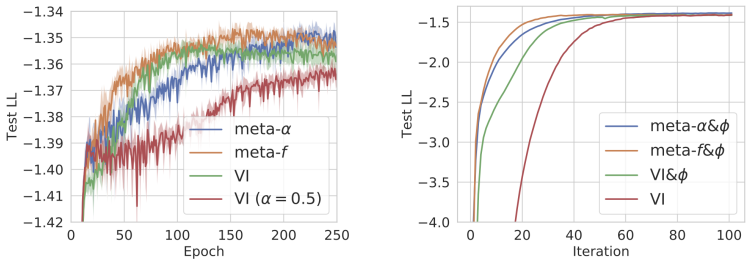
Table 6.7: Meta-$D\&\phi$ (meta-learning divergences and variational parameters) on MNIST: marginal log-likelihood on 5 test tasks. Each task has 100 training data. We train the model for 200 epochs during meta-testing.

Digit	5	6	7	8	9
VI	-177.92±0.46	-182.93±0.06	-125.57±0.41	-182.63±0.55	-161.68±0.27
VI$\&\phi$	-174.32±0.18	-176.17±0.26	-123.20±0.12	-177.96±0.23	-147.25±0.32
meta-$\alpha\&\phi$	-163.31±0.61	-163.19±0.36	**-115.52**±0.16	-173.35±0.38	-142.76±0.33
meta-$f\&\phi$	**-160.16**±0.16	**-154.16**±0.67	-122.61±0.43	**-165.83**±0.48	**-138.90**±0.10

Image Generation with Variational Auto-encoders

We also evaluate the image generation task with variational auto-encoders (VAEs). Specifically, we train VAEs to generate MNIST digits with different divergences. Generating each digit is regarded as a task and we use the first 5 digits (0-4) as the training tasks and the last 5 digits (5-9) as the test tasks.

We report the test marginal log-likelihood for each test digit in Table 6.6 and 6.7. Overall, these results align with other experiments that the meta-D and meta-$D\&\phi$ are both better than their counterparts. Meta-D and meta-$D\&\phi$ are better than VAE with common divergences on all 5 test tasks, indicating our methods have learned a suitable divergence.



(a) Meta-D (b) Meta-D&ϕ

Figure 6.4: Test log-likelihood on MovieLens.Panel (a) shows the results of meta-learning divergences only (Meta-D), and panel (b) shows the results of meta-learning both divergences and variational parameters (Meta-D&ϕ).

Recommender System with a Partial Variational Autoencoder

We test our method on recommender systems with a Partial Variational Auto-encoder (p-VAE) [93]. P-VAE is proposed to deal with partially observed data and has been shown to achieve state-of-the-art level performance on user rating prediction in recommender system [92]. We consider MovieLens 1M dataset [64] which contains 1,000,206 ratings of 3,952 movies from 6,040 users. We select four age groups as training tasks, and use the remaining three groups as test tasks. During meta-testing, we use 90%/10% and 60%/40% training-test split for Meta-D and Meta-D&ϕ, respectively. From Figure 6.4(a), we see that when applied to learning p-VAEs, meta-D outperforms standard VI (KL divergence) and VI with $\alpha = 0.5$ divergence in terms of test LL, showing that meta-D has learned a suitable divergence that leads to better test performance. Figure 6.4(b) implies that all methods with learned ϕ can converge quickly on the new task with only 100 iterations. Both meta-α&ϕ and meta-f&ϕ learn faster than VI&ϕ in meta-test time, indicating that the learned divergence can help fast adaptation.

CHAPTER 7

CONCLUSION

This thesis studied and improved the reliability-scalability trade-off of inference in probabilistic modeling. Specifically, we explored two themes, theoretically-guaranteed inference and efficient inference for reliable deep learning.

In Chapter 3, we proposed the Poisson-minibatching framework to generate unbiased samples with theoretical guarantees on the convergence rate and scalability, with applications to Gibbs sampling and Metropolis-Hastings. Our algorithms provably converge to the desired stationary distribution at a rate that is at most a constant factor slower than the full batch method, as measured by the spectral gap. Additionally, we demonstrate that inexact MH methods can lead to arbitrarily incorrect inference and argue for the use of exact methods. We also prove a lower bound on the batch size that any minibatch MH method must use to maintain exactness and convergence rate, and show our MH method is asymptotically optimal.

In Chapter 4, we introduced AMAGOLD, which achieves unbiased, efficient second-order SG-MCMC by infrequently applying the computationally-expensive Metropolis- Hasting adjustment step, amortizing the cost across multiple algorithm steps. We prove this is sufficient for convergence to the target distribution, and provide reversible and non-reversible versions for practical use. AMAGOLD's convergence rate is theoretically guaranteed: the bound captures the trade-off between the speed-up from minibatching and the convergence rate. Lastly, our work is complementary to, rather than exclusive with, other research in stochastic MCMC. In future work, it would be interesting to explore combining AMA with other SG-MCMC variants and minibatch MH methods.

In Chapter 5, we presented cyclical SG-MCMC methods to automatically explore complex multimodal distributions. Our approach is particularly compelling for Bayesian deep learning, which involves rich multimodal parameter posteriors corresponding to meaningfully different representations. We have also shown that our cyclical methods explore unimodal distributions more efficiently. These results are in accordance with theory we developed to show that cyclical SG-MCMC will converge faster to samples from a stationary distribution in general settings. Moreover, we show cyclical SG-MCMC methods provide more accurate uncertainty estimation, by capturing more diversity in the hypothesis space corresponding to settings of model parameters.

In Chapter 6, we proposed meta-learning divergences of VI which automates the selection of divergence objective in VI via meta-learning. It further allows meta-learning of variational parameter initialization for fast adaptation on new tasks. Within our meta-learning divergences framework, we consider two divergence families, α- and f-divergence, and design parameterizations of divergences to enable learning via gradient descent. Experimental results on Gaussian mixture approximation, regression with Bayesian neural networks, generative modeling and recommender systems demonstrate the superior performance of meta-learned divergences over standard divergences.

While probabilistic inference was once popular for many machine learning tasks, it is now often replaced by optimization methods in modern machine learning due to scalability issue. I hope that this thesis will help renew interest in probabilistic methods for machine learning and shed light on the future direction to continue pushing the frontier of probabilistic modeling.

APPENDIX A

SECTION 3.1 GIBBS SAMPLING WITH POISSON-MINIBATCHING

A.1 Fast Sampling of the Auxiliary Variables

In this section, we describe in detail the method used to sample the auxiliary variables s_ϕ and prove Statement 1. The method for doing so is described here in Algorithm 12.

Algorithm 12 Sample auxiliary variables s_ϕ

▷ pre-computation step; happens once
for $i = 1$ **to** n **do**
  $\Lambda_i \leftarrow \sum_{\phi \in A[i]} \frac{\lambda M_\phi}{L} + M_\phi$
  compute distribution ρ_i over $A[i]$ where

$$\rho_i(\phi) \propto \frac{\lambda M_\phi}{L} + M_\phi.$$

  process distribution ρ_i so that in future, it can be sampled from in constant time
end for

▷ to actually re-sample the auxiliary variables
given: current state $x \in \Omega$, variable i to resample
initialize sparse vector $s : A[i] \to \mathbb{Z}$
sample $B \sim \text{Poisson}(\Lambda_i)$
for $b = 1$ **to** B **do**
  sample $\phi \sim \rho_i$
  compute $\phi(x)$
  with probability $\frac{\frac{\lambda M_\phi}{L} + \phi(x)}{\frac{\lambda M_\phi}{L} + M_\phi}$ **update** sparse vector $s_\phi \leftarrow s_\phi + 1$
end for

To see that this is valid, let $B = \sum_i^n s_i$ where s_i are Poisson variables with parameters λ_i. We know that B is also Poisson distributed with parameter $\Lambda =$

$\sum_i^n \lambda_i$. Conditioned on the value of B, it is known that s_i follows a multinomial distribution with event probabilities λ_i/Λ and trial count B. Therefore, we can first sample $B \sim \text{Poisson}(\Lambda)$ and then sample

$$(s_1, \ldots s_n) \sim \text{Multinomial}\left(B, \left(\frac{\lambda_1}{\Lambda}, \ldots, \frac{\lambda_n}{\Lambda}\right)\right).$$

Our Algorithm 12 is only slightly more complicated than this process, in order to minimize the number of times that $\phi(x)$ is evaluated, but it can be seen to produce the valid distribution by the same reasoning.

The computational cost of Algorithm 12 is clearly proportional to B, and since

$$\mathbf{E}[B] = \Lambda_i = \sum_{\phi \in A[i]} \frac{\lambda M_\phi}{L} + M_\phi \leq \lambda + L,$$

it follows that the overall average computational cost will also be $\lambda + L$. This proves Statement 1.

A.2 Proofs and Derivations

A.2.1 Derivation of the joint distribution

In this subsection, we derive the joint distribution (3.1) by substituting the distributions of x and s into the conditional distribution of s given x. By the expression

of Poisson distribution for s_ϕ and the independence of s_ϕ, we have

$$\begin{aligned}
\pi(x,s) &= \pi(x)\pi(s|x) \\
&\propto \exp\left(\sum_{\phi\in\Phi}\phi(x)\right)\prod_{\phi\in\Phi}\pi(s_\phi|x) \\
&= \exp\left(\sum_{\phi\in\Phi}\left(\phi(x)+\log\pi(s_\phi|x)\right)\right) \\
&= \exp\left(\sum_{\phi\in\Phi}\left(\phi(x)+s_\phi\log\left(\frac{\lambda M_\phi}{L}+\phi(x)\right)-\log(s_\phi!)-\frac{\lambda M_\phi}{L}-\phi(x)\right)\right) \\
&= \exp\left(\sum_{\phi\in\Phi}\left(s_\phi\log\left(\frac{\lambda M_\phi}{L}+\phi(x)\right)-\log(s_\phi!)-\frac{\lambda M_\phi}{L}\right)\right) \\
&\propto \exp\left(\sum_{\phi\in\Phi}\left(s_\phi\log\left(\frac{\lambda M_\phi}{L}+\phi(x)\right)-\log(s_\phi!)\right)\right) \\
&= \exp\left(\sum_{\phi\in\Phi}\left(s_\phi\log\left(1+\frac{L}{\lambda M_\phi}\phi(x)\right)+s_\phi\log\left(\frac{\lambda M_\phi}{L}\right)-\log\left(s_\phi!\right)\right)\right).
\end{aligned}$$

A.2.2 Proof of Theorem 1

In this section, we prove that Poisson-Gibbs converges, and derive a bound on its convergence rate.

Proof. First, we will derive an expression for the transition operator of Poisson-Gibbs chain, and show it is reversible. Then we will bound the spectral gap.

If x and y are states which differ in only one variable i, the probability of transitioning from x to y will be the probability of choosing to sample variable i times the expected value over the random choice of s of the probability of sampling

$y(i)$ from ρ. That is,

$$
\begin{aligned}
&T(x,y)\\
&\quad= \frac{1}{n}\cdot \mathbf{E}\left[\rho(y(i))\right]\\
&\quad= \frac{1}{n}\cdot \mathbf{E}\left[\frac{\exp(U_{y(i)})}{\int \exp(U_u)\,du}\right]\\
&\quad= \frac{1}{n}\cdot \sum_s \frac{\exp(U_{y(i)})}{\int \exp(U_u)\,du}\cdot \prod_{\phi\in A[i]}\frac{1}{s_\phi!}\left(\frac{\lambda M_\phi}{L}+\phi(x)\right)^{s_\phi}\exp\left(-\left(\frac{\lambda M_\phi}{L}+\phi(x)\right)\right)\\
&\quad= \frac{1}{n}\cdot \sum_s \frac{\exp\left(\sum_{\phi\in A[i]} s_\phi \log\left(\frac{\lambda M_\phi}{L}+\phi(y)\right)\right)}{\int \exp\left(\sum_{\phi\in A[i]} s_\phi \log\left(\frac{\lambda M_\phi}{L}+\phi(z_u)\right)\right)\,du}\\
&\qquad\cdot \prod_{\phi\in A[i]}\left(1+\frac{L}{\lambda M_\phi}\phi(x)\right)^{s_\phi}\cdot\exp\left(-\phi(x)\right)\\
&\qquad\cdot \prod_{\phi\in A[i]}\frac{1}{s_\phi!}\left(\frac{\lambda M_\phi}{L}\right)^{s_\phi}\cdot\exp\left(-\frac{\lambda M_\phi}{L}\right)
\end{aligned}
$$

where z_u denotes x where $x(i)$ has been set equal to u. Note that s_ϕ here are non-negative integers that a Poisson variable can take, not variables. So if we let $r_\phi \sim \text{Poisson}\left(\frac{\lambda M_\phi}{L}\right)$ and r_ϕ to be all independent, we can write this as

$$
\begin{aligned}
T(x,y) &= \frac{1}{n}\cdot \mathbf{E}_r\left[\frac{\exp\left(\sum_{\phi\in A[i]} r_\phi \log\left(\frac{\lambda M_\phi}{L}+\phi(y)\right)\right)}{\int \exp\left(\sum_{\phi\in A[i]} r_\phi \log\left(\frac{\lambda M_\phi}{L}+\phi(z_u)\right)\right)\,du}\right.\\
&\qquad\left.\cdot \prod_{\phi\in A[i]}\left(1+\frac{L}{\lambda M_\phi}\phi(x)\right)^{r_\phi}\cdot\exp\left(-\phi(x)\right)\right]\\
&= \frac{1}{n}\cdot \mathbf{E}_r\left[\frac{\exp\left(\sum_{\phi\in A[i]} r_\phi\left(\log\left(1+\frac{L}{\lambda M_\phi}\phi(y)\right)+\log\left(1+\frac{L}{\lambda M_\phi}\phi(x)\right)\right)\right)}{\int \exp\left(\sum_{\phi\in A[i]} r_\phi \log\left(1+\frac{L}{\lambda M_\phi}\phi(z_u)\right)\right)\,du}\right.\\
&\qquad\left.\cdot\exp\left(-\sum_{\phi\in A[i]}\phi(x)\right)\right]
\end{aligned}
$$

Therefore, since

$$\pi(x) = \frac{1}{Z}\cdot\exp\left(\sum_{\phi\in\Phi}\phi(x)\right),$$

it follows that

$$\begin{aligned}
&\pi(x)T(x,y) \\
&\quad = \frac{1}{nZ} \cdot \mathbf{E}_r \left[\frac{\exp\left(\sum_{\phi \in A[i]} r_\phi \left(\log\left(1 + \frac{L}{\lambda M_\phi}\phi(y)\right) + \log\left(1 + \frac{L}{\lambda M_\phi}\phi(x)\right)\right)\right)}{\int \exp\left(\sum_{\phi \in A[i]} r_\phi \log\left(1 + \frac{L}{\lambda M_\phi}\phi(z_u)\right)\right)\, du} \right. \\
&\qquad \left. \cdot \exp\left(\sum_{\phi \in \Phi} \phi(x) - \sum_{\phi \in A[i]} \phi(x)\right) \right] \\
&\quad = \frac{\exp(U_{\neg i}(x))}{nZ} \\
&\qquad \cdot \mathbf{E}_r \left[\frac{\exp\left(\sum_{\phi \in A[i]} r_\phi \left(\log\left(1 + \frac{L}{\lambda M_\phi}\phi(y)\right) + \log\left(1 + \frac{L}{\lambda M_\phi}\phi(x)\right)\right)\right)}{\int \exp\left(\sum_{\phi \in A[i]} r_\phi \log\left(1 + \frac{L}{\lambda M_\phi}\phi(z_u)\right)\right)\, du} \right].
\end{aligned}$$

where we define $U_{\neg i}(x) = \sum_{\phi \notin A[i]} \phi(x)$. This expression is symmetric in x and y (note that $U_{\neg i}(x)$ does not depend on variable i), so it follows that the Markov chain is reversible, and its stationary distribution is indeed π.

We can proceed to try to bound its spectral gap, using the technique of Dirichlet forms. We start by simplifying our expression by defining

$$\bar{\phi}(x) = \frac{L\phi(x)}{\lambda M_\phi}.$$

Using this, we get

$$\begin{aligned}
&\pi(x)T(x,y) \\
&\quad = \frac{\exp(U_{\neg i}(x))}{nZ} \cdot \mathbf{E}_r \left[\frac{\exp\left(\sum_{\phi \in A[i]} r_\phi \left(\log\left(1 + \bar{\phi}(y)\right) + \log\left(1 + \bar{\phi}(x)\right)\right)\right)}{\int \exp\left(\sum_{\phi \in A[i]} r_\phi \log\left(1 + \bar{\phi}(z_u)\right)\right)\, du} \right].
\end{aligned}$$

We proceed by bringing the exponential on the top of this sum down to the bottom

and inside the integral, which produces

$$\begin{aligned}\pi(x)T(x,y) &= \frac{\exp\left(U_{\neg i}(x)\right)}{nZ} \cdot \mathbf{E}_r\left[\left(\int \exp\left(\sum_{\phi\in A[i]} r_\phi\left(\log\left(1+\bar{\phi}(z_u)\right)\right.\right.\right.\right. \\ &\qquad \left.\left.\left.\left. -\log\left(1+\bar{\phi}(x)\right)-\log\left(1+\bar{\phi}(y)\right)\right)\right)du\right)^{-1}\right] \\ &\geq \frac{\exp\left(U_{\neg i}(x)\right)}{nZ} \cdot \left(\mathbf{E}_r\left[\int \exp\left(\sum_{\phi\in A[i]} r_\phi\left(\log\left(1+\bar{\phi}(z_u)\right)\right.\right.\right.\right. \\ &\qquad \left.\left.\left.\left. -\log\left(1+\bar{\phi}(x)\right)-\log\left(1+\bar{\phi}(y)\right)\right)\right)du\right]\right)^{-1}\end{aligned}$$

where this inequality follows from Jensen's inequality and the fact that $1/x$ is convex. By converting the exp-of-sum to a product-of-exp, and recalling that the r_ϕ are independent, we can further reduce this to

$$\begin{aligned}\pi(x)T(x,y) &\geq \frac{\exp\left(U_{\neg i}(x)\right)}{nZ}\left(\int \mathbf{E}_r\left[\prod_{\phi\in A[i]} \exp\left(r_\phi\left(\log\left(1+\bar{\phi}(z_u)\right)\right.\right.\right.\right. \\ &\qquad \left.\left.\left.\left. -\log\left(1+\bar{\phi}(x)\right)-\log\left(1+\bar{\phi}(y)\right)\right)\right)\right]du\right)^{-1} \\ &= \frac{\exp\left(U_{\neg i}(x)\right)}{nZ}\left(\int \prod_{\phi\in A[i]} \mathbf{E}_r\left[\exp\left(r_\phi\left(\log\left(1+\bar{\phi}(z_u)\right)\right.\right.\right.\right. \\ &\qquad \left.\left.\left.\left. -\log\left(1+\bar{\phi}(x)\right)-\log\left(1+\bar{\phi}(y)\right)\right)\right)\right]du\right)^{-1}.\end{aligned}$$

This final expectation expression is just the moment generating function of the Poisson random variable r_ϕ evaluated at

$$t = \log\left(1+\bar{\phi}(z_u)\right) - \log\left(1+\bar{\phi}(x)\right) - \log\left(1+\bar{\phi}(y)\right).$$

Here, from the standard formula for that MGF, we get

$$\mathbf{E}_r[\exp(r_\phi t)] = \exp\left(\frac{\lambda M_\phi}{L}\left(\exp(t)-1\right)\right)$$

So

$$\begin{aligned}
&\exp(t) - 1 \\
&= \frac{1 + \bar{\phi}(z_u)}{(1 + \bar{\phi}(x))(1 + \bar{\phi}(y))} - 1 \\
&= \frac{\bar{\phi}(z_u) - \bar{\phi}(x) - \bar{\phi}(y) - \bar{\phi}(x)\bar{\phi}(y)}{(1 + \bar{\phi}(x))(1 + \bar{\phi}(y))} \\
&= \bar{\phi}(z_u) - \bar{\phi}(x) - \bar{\phi}(y) \\
&\qquad - \frac{\left(\bar{\phi}(z_u) - \bar{\phi}(x) - \bar{\phi}(y)\right)\left(\bar{\phi}(x) + \bar{\phi}(y) + \bar{\phi}(x)\bar{\phi}(y)\right) + \bar{\phi}(x)\bar{\phi}(y)}{(1 + \bar{\phi}(x))(1 + \bar{\phi}(y))}.
\end{aligned}$$

Since

$$0 \leq \bar{\phi}(x) = \frac{L\phi(x)}{\lambda M_\phi} \leq \frac{L}{\lambda} \leq \frac{1}{2}$$

(where here we're using the condition in the theorem statement that $2L \leq \lambda$) we can bound this with

$$\begin{aligned}
&\exp(t) - 1 \\
&\leq \bar{\phi}(z_u) - \bar{\phi}(x) - \bar{\phi}(y) \\
&\qquad - \frac{\left(-\bar{\phi}(x) - \bar{\phi}(y)\right)\left(\bar{\phi}(x) + \bar{\phi}(y)\right) + \left(1 - \bar{\phi}(x) - \bar{\phi}(y)\right)\bar{\phi}(x)\bar{\phi}(y)}{(1 + \bar{\phi}(x))(1 + \bar{\phi}(y))} \\
&\leq \bar{\phi}(z_u) - \bar{\phi}(x) - \bar{\phi}(y) + \frac{\left(\bar{\phi}(x) + \bar{\phi}(y)\right)\left(\bar{\phi}(x) + \bar{\phi}(y)\right)}{(1 + \bar{\phi}(x))(1 + \bar{\phi}(y))} \\
&\leq \bar{\phi}(z_u) - \bar{\phi}(x) - \bar{\phi}(y) + \left(\bar{\phi}(x) + \bar{\phi}(y)\right)^2 \\
&\leq \bar{\phi}(z_u) - \bar{\phi}(x) - \bar{\phi}(y) + \frac{4L^2}{\lambda^2}.
\end{aligned}$$

So,

$$\begin{aligned}
\mathbf{E}\exp(r_\phi t) &= \exp\left(\frac{\lambda M_\phi}{L}\left(\exp(t) - 1\right)\right) \\
&\leq \exp\left(\frac{\lambda M_\phi}{L}\left(\bar{\phi}(z_u) - \bar{\phi}(x) - \bar{\phi}(y) + \frac{4L^2}{\lambda^2}\right)\right) \\
&= \exp\left(\phi(z_u) - \phi(x) - \phi(y) + \frac{4LM_\phi}{\lambda}\right).
\end{aligned}$$

Substituting this into the original expression produces

$$
\begin{aligned}
&\pi(x)T(x,y) \\
&\geq \frac{\exp\left(U_{\neg i}(x)\right)}{nZ}\left(\int \prod_{\phi\in A[i]} \exp\left(\phi(z_u)-\phi(x)-\phi(y)+\frac{4LM_\phi}{\lambda}\right)du\right)^{-1} \\
&= \frac{\exp\left(U_{\neg i}(x)\right)}{nZ} \\
&\quad \cdot \left(\int \exp\left(\sum_{\phi\in A[i]}\phi(z_u)-\sum_{\phi\in A[i]}\phi(x)-\sum_{\phi\in A[i]}\phi(y)+\sum_{\phi\in A[i]}\frac{4LM_\phi}{\lambda}\right)du\right)^{-1} \\
&\geq \frac{\exp\left(U_{\neg i}(x)\right)}{nZ}\left(\int \exp\left(\sum_{\phi\in A[i]}\phi(z_u)-\sum_{\phi\in A[i]}\phi(x)-\sum_{\phi\in A[i]}\phi(y)+\frac{4L^2}{\lambda}\right)du\right)^{-1} \\
&= \exp\left(-\frac{4L^2}{\lambda}\right)\frac{\exp\left(U_{\neg i}(x)\right)}{nZ}\left(\int \exp\left(\bar{U}_u-\bar{U}_{x(i)}-\bar{U}_{y(i)}\right)du\right)^{-1} \\
&= \exp\left(-\frac{4L^2}{\lambda}\right)\frac{\exp\left(U_{\neg i}(x)\right)}{nZ}\frac{\exp(\bar{U}_{x(i)})\cdot\exp(\bar{U}_{y(i)})}{\int\exp(\bar{U}_u)du} \\
&= \exp\left(-\frac{4L^2}{\lambda}\right)\frac{1}{nZ}\frac{\exp(U(x))\cdot\exp(\bar{U}_{y(i)})}{\int\exp(\bar{U}_u)du}
\end{aligned}
$$

where $\bar{U}_v$ denotes the assignment of U_v in the plain Gibbs sampling algorithm (Algorithm 1),

$$\bar{U}_v = \sum_{\phi\in A[i]}\phi(z_v),$$

Finally, if we let G denote the transition probability operator of plain Gibbs sampling, we notice right away that

$$
\begin{aligned}
\pi(x)T(x,y) &\geq \exp\left(-\frac{4L^2}{\lambda}\right)\frac{1}{nZ}\frac{\exp(U(x))\cdot\exp(\bar{U}_{y(i)})}{\int\exp(\bar{U}_u)du} \\
&= \exp\left(-\frac{4L^2}{\lambda}\right)\pi(x)G(x,y).
\end{aligned}
$$

We will use the Dirichlet form argument to finish the proof. A real function f is square integrable with respect to probability measure π, if it satisfies

$$\int f(x)^2\pi(dx) < \infty.$$

Define $L^2(\pi)$ to be the Hilbert space of all such functions.

Let $L_0^2(\pi) \subset L^2(\pi)$ to be the Hilbert space that uses the same inner product but only contains functions such that

$$\mathbf{E}_\pi[f] = \int f(x)\pi(dx) = 0.$$

We also define the notation

$$\langle f, g \rangle = \int f(x)g(x)\pi(dx).$$

A special example is $\mathrm{Var}_\pi[f] = \langle f, f \rangle$.

From here, the Dirichlet form of a Markov chain associated with transition operator T is given by [50]

$$\mathbf{E}(f) = \frac{1}{2} \int \int (f(x) - f(y))^2 T(x, y)\pi(x)dxdy.$$

And the spectral gap can be written as [3]

$$\gamma = \inf_{f \in L_0^2(\pi): \mathrm{Var}_\pi[f]=1} \mathbf{E}(f).$$

The spectral gap is related to other common measurement of the convergence of MCMC. For example, it has the following relationship with the mean squared error e_π on a Markov chain $\{X_n\}_{n\in\mathbb{N}}$ [129],

$$e_\pi^2 \leq \frac{2}{n\gamma} \|f\|_2^2 .$$

With the expression of the spectral gap, it follows that

$$\begin{aligned}
\bar{\gamma} &= \inf_{f \in L_0^2(\pi): Var_\pi[f]=1} \left[\frac{1}{2} \int \int (f(x) - f(y))^2 T(x, y)\pi(x)\, dx\, dy \right] \\
&\geq \exp\left(-\frac{4L^2}{\lambda}\right) \cdot \inf_{f \in L_0^2(\pi): Var_\pi[f]=1} \left[\frac{1}{2} \int \int (f(x) - f(y))^2 G(x, y)\pi(x)\, dx\, dy \right] \\
&= \exp\left(-\frac{4L^2}{\lambda}\right) \cdot \gamma.
\end{aligned}$$

This proves the theorem. □

A.2.3 Proof of Theorem 5

Proof. Similar to the previous analysis of Poisson-Gibbs, we will show the PGITS is reversible by using the expression of the transition operator. Then we will bound the spectral gap.

Let $T_{i,s}(x, y)$ denote the probability of transitioning from state x to y given that we have already chosen to sample variable i with minibatch coefficients s. Then, the overall transition operator will be

$$T(x, y) = \mathbf{E}\left[T_{i,s}(x, y)\right]$$

where the expectation is taken over i and s.

Let the polynomial interpolant for $\exp(U_v)$ be $\tilde{f}(v)$ which is given in (3.3). Note that this interpolant is a function of the index i and the minibatch coefficients s. Then,

$$\begin{aligned} T_{i,s}(x, y) &= \rho(y(i)) \cdot \min(1, a) \\ &= \frac{\tilde{f}(y(i))}{\int \tilde{f}(u)du} \cdot \min\left(1, \frac{\exp(U_{y(i)})\tilde{f}(x(i))}{\exp(U_{x(i)})\tilde{f}(y(i))}\right) \end{aligned}$$

Therefore,

$$\begin{aligned} &T(x, y) \\ &= \frac{1}{n}\mathbf{E}\frac{\tilde{f}(y(i))}{\int \tilde{f}(u)du} \cdot \min\left(1, \frac{\exp(U_{y(i)})\tilde{f}(x(i))}{\exp(U_{x(i)})\tilde{f}(y(i))}\right) \\ &= \frac{1}{n}\mathbf{E}\frac{1}{\int \tilde{f}(u)du} \cdot \min\left(\tilde{f}(y(i)), \exp(U_{y(i)} - U_{x(i)})\tilde{f}(x(i))\right) \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{n}\mathbf{E}\frac{1}{\int \tilde{f}(u)du} \cdot \min\left(\tilde{f}(y(i)), \exp\left(\sum_{\phi\in A[i]} s_\phi \log \frac{1+\frac{L}{\lambda M_\phi}\phi(y)}{1+\frac{L}{\lambda M_\phi}\phi(x)}\right)\tilde{f}(x(i))\right) \\
&= \frac{1}{n}\sum_s \frac{1}{\int \tilde{f}(u)du} \cdot \min\left(\tilde{f}(y(i)), \exp\left(\sum_{\phi\in A[i]} s_\phi \log \frac{1+\frac{L}{\lambda M_\phi}\phi(y)}{1+\frac{L}{\lambda M_\phi}\phi(x)}\right)\tilde{f}(x(i))\right) \\
&\quad \cdot \exp\left(\sum_{\phi\in A[i]} s_\phi \log\left(\frac{\lambda M_\phi}{L}+\phi(x)\right) - \log(s_\phi!) - \left(\frac{\lambda M_\phi}{L}+\phi(x)\right)\right) \\
&= \frac{1}{n}\sum_s \frac{1}{\int \tilde{f}(u)du} \cdot \min\left(\tilde{f}(y(i))\exp\left(\sum_{\phi\in A[i]} s_\phi \log\left(1+\frac{L}{\lambda M_\phi}\phi(x)\right)\right),\right. \\
&\quad \left.\exp\left(\sum_{\phi\in A[i]} s_\phi \log\left(1+\frac{L}{\lambda M_\phi}\phi(y)\right)\right)\tilde{f}(x(i))\right) \\
&\quad \cdot \exp\left(\sum_{\phi\in A[i]}\left(s_\phi \log\left(\frac{\lambda M_\phi}{L}\right) - \log(s_\phi!) - \left(\frac{\lambda M_\phi}{L}+\phi(x)\right)\right)\right) \\
&= \frac{1}{n}\sum_s \frac{1}{\int \tilde{f}(u)du} \cdot \min\left(\tilde{f}(y(i))\exp\left(\sum_{\phi\in A[i]} s_\phi \log\left(1+\frac{L}{\lambda M_\phi}\phi(x)\right)\right),\right. \\
&\quad \left.\exp\left(\sum_{\phi\in A[i]} s_\phi \log\left(1+\frac{L}{\lambda M_\phi}\phi(y)\right)\right)\tilde{f}(x(i))\right) \\
&\quad \cdot \exp\left(\sum_{\phi\in A[i]}\left(s_\phi \log\left(\frac{\lambda M_\phi}{L}\right) - \log(s_\phi!) - \left(\frac{\lambda M_\phi}{L}\right)\right)\right)\exp(-U_{x(i)})
\end{aligned}
$$

Multiplying $\pi(x)$ on both sides,

$$
\begin{aligned}
&\pi(x)T(x,y) \\
&= \frac{\exp(U_{\neg i}(x))}{nZ}\sum_s \frac{1}{\int \tilde{f}(u)du} \cdot \min\left(\tilde{f}(y(i))\exp\left(\sum_{\phi\in A[i]} s_\phi \log\left(1+\frac{L}{\lambda M_\phi}\phi(x)\right)\right),\right. \\
&\quad \left.\tilde{f}(x(i))\exp\left(\sum_{\phi\in A[i]} s_\phi \log\left(1+\frac{L}{\lambda M_\phi}\phi(y)\right)\right)\right) \\
&\quad \cdot \exp\left(\sum_{\phi\in A[i]}\left(s_\phi \log\left(\frac{\lambda M_\phi}{L}\right) - \log(s_\phi!) - \left(\frac{\lambda M_\phi}{L}\right)\right)\right)
\end{aligned}
$$

This expression is symmetric in x and y, so it follows that

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

Thus the Markov chain is reversible, and its stationary distribution is π.

We now bound its spectral gap, using the technique of Dirichlet forms. First, as before, we start by re-writing the chain in terms of an expectation of a new random variable r_ϕ where $r_\phi \sim \text{Poisson}\left(\frac{\lambda M_\phi}{L}\right)$ and the r_ϕ are all independent. We also define $\bar{\phi}(x) = \frac{L\phi(x)}{\lambda M_\phi}$ as before. This gives us

$$\begin{aligned}
&\pi(x)T(x,y) \\
&= \frac{\exp(U_{\neg i}(x))}{nZ}\ \mathbf{E}_r\left[\frac{1}{\int \tilde{f}(u)du} \cdot \min\left(\tilde{f}(y(i))\exp\left(\sum_{\phi\in A[i]} r_\phi \log\left(1+\bar{\phi}(x)\right)\right),\right.\right. \\
&\qquad\left.\left.\tilde{f}(x(i))\exp\left(\sum_{\phi\in A[i]} r_\phi \log\left(1+\bar{\phi}(y)\right)\right)\right)\right] \\
&= \frac{\exp(U_{\neg i}(x))}{nZ}\ \mathbf{E}_r\left[\frac{1}{\int \tilde{f}(u)du} \cdot \min\left(\tilde{f}(y(i))\exp\left(U_{x(i)}\right), \tilde{f}(x(i))\exp\left(U_{y(i)}\right)\right)\right]
\end{aligned}$$

where now the $\tilde{f}$ are considered to be a function of r_ϕ rather than s_ϕ as before.

To proceed further we will need to use the fact that $\tilde{f}$ is a Chebyshev interpolant to bound its error compared with U. Recall that, here,

$$U_v = \sum_{\phi\in A[i]} r_\phi \log\left(1+\frac{L}{\lambda M_\phi}\phi(z_v)\right) = \sum_{\phi\in A[i]} r_\phi \log\left(1+\bar{\phi}(z_v)\right),$$

and $\tilde{f}(v) \approx \exp(U_v)$ in the sense of being a degree-m Chebyshev polynomial interpolant. Recall that we assumed that the each function ϕ, treated as a function in any single variable, must be analytic on a (shifted) Bernstein ellipse on the interval $[a,b]$ with parameter ρ (i.e. a standard Bernstein ellipse on $[-1,1]$ with parameter ρ shifted and scaled to have its foci at a and b), and that its magnitude

must be bounded by

$$|\phi(z)| \leq M_\phi$$

for any z in this ellipse (keeping all the other parameters as usual within $[a, b]$. It follows that the magnitude of the function U_v is bounded by

$$\begin{aligned}
|\exp(U_v)| &= \left| \exp\left(\sum_{\phi \in A[i]} r_\phi \log\left(1 + \frac{L}{\lambda M_\phi}\phi(z_v)\right)\right)\right| \\
&= \prod_{\phi \in A[i]} \left|1 + \frac{L}{\lambda M_\phi}\phi(z_v)\right|^{r_\phi} \\
&\leq \prod_{\phi \in A[i]} \left(1 + \frac{L}{\lambda}\right)^{r_\phi}.
\end{aligned}$$

Therefore, from Theorem 3, we know that

$$\left|\tilde{f}(v) - \exp(U_v)\right| \leq \frac{4\rho^{-m}}{\rho - 1} \cdot \prod_{\phi \in A[i]} \left(1 + \frac{L}{\lambda}\right)^{r_\phi} = \frac{4\rho^{-m}}{\rho - 1} \cdot \left(1 + \frac{L}{\lambda}\right)^{\sum_{\phi \in A[i]} r_\phi}.$$

Since we also assumed that $\phi(z)$ is always non-negative, U_v must also be non-negative, and so in particular $\exp(-U_v) \leq 1$, so

$$\left|\frac{\tilde{f}(v)}{\exp(U_v)} - 1\right| \leq \frac{4\rho^{-m}}{\rho - 1} \cdot \left(1 + \frac{L}{\lambda}\right)^{\sum_{\phi \in A[i]} r_\phi} \leq \frac{4\rho^{-m}}{\rho - 1} \cdot \exp\left(\frac{L}{\lambda} \sum_{\phi \in A[i]} r_\phi\right).$$

If we now define

$$C = \frac{4\rho^{-m}}{\rho - 1} \cdot \exp\left(\frac{L}{\lambda} \sum_{\phi \in A[i]} r_\phi\right),$$

then

$$(1 - C) \cdot \exp(U_v) \leq \tilde{f}(v) \leq (1 + C) \cdot \exp(U_v).$$

In particular, this means that

$$\min\left(\tilde{f}(y(i)) \exp\left(U_{x(i)}\right), \tilde{f}(x(i)) \exp\left(U_{y(i)}\right)\right) \geq (1 - C) \cdot \exp\left(U_{x(i)} + U_{y(i)}\right),$$

and

$$\frac{1}{\int \tilde{f}(u)\, du} \geq \frac{1}{1 + C} \cdot \frac{1}{\int \exp(U_u)\, du}.$$

Substituting this into our bound above gives

$$\pi(x)T(x,y) \geq \frac{\exp(U_{\neg i}(x))}{nZ}\ \mathbf{E}_r\left[\frac{1-C}{1+C}\cdot\frac{\exp\left(U_{x(i)}+U_{y(i)}\right)}{\int\exp(U_u)\ du}\right].$$

Now, recall that we set this up by sampling r_ϕ independently from a Poisson random variable $r_\phi \sim \text{Poisson}\left(\frac{\lambda M_\phi}{L}\right)$. This distribution is equivalent to assigning

$$\Lambda = \sum_{\phi\in A[i]}\frac{\lambda M_\phi}{L},$$

sampling the random variable $B \sim \text{Poisson}\,(\Lambda)$, and then sampling $r_\phi \sim \text{Multinomial}\left(B, \frac{\lambda M_\phi}{\Lambda L}\right)$. If we re-think our distribution as coming from this process, then by the Law of Total Expectation,

$$\pi(x)T(x,y) \geq \frac{\exp(U_{\neg i}(x))}{nZ}\ \mathbf{E}_B\left[\frac{1-C}{1+C}\cdot\mathbf{E}_r\left[\frac{\exp\left(U_{x(i)}+U_{y(i)}\right)}{\int\exp(U_u)\ du}\middle| B\right]\right],$$

where we can pull out the terms in C because we can write C to depend only on B as

$$C = \frac{4\rho^{-m}}{\rho-1}\cdot\exp\left(\frac{L}{\lambda}\sum_{\phi\in A[i]}r_\phi\right) = \frac{4\rho^{-m}}{\rho-1}\cdot\exp\left(\frac{LB}{\lambda}\right).$$

Next, we can bound this inner expectation with

$$\begin{aligned}
&\mathbf{E}_r\left[\frac{\exp\left(U_{x(i)}+U_{y(i)}\right)}{\int \exp(U_u)\,du}\middle| B\right] \\
&= \mathbf{E}_r\left[\frac{1}{\int \exp(U_u - U_{x(i)} - U_{y(i)})\,du}\middle| B\right] \\
&\geq \mathbf{E}_r\left[\int \exp(U_u - U_{x(i)} - U_{y(i)})\,du \middle| B\right]^{-1} \\
&= \mathbf{E}_r\Bigg[\int \exp\Bigg(\sum_{\phi\in A[i]} r_\phi\Bigg(\log\left(1+\bar{\phi}(z_u)\right) - \log\left(1+\bar{\phi}(x)\right) \\
&\qquad - \log\left(1+\bar{\phi}(y)\right)\Bigg)\Bigg)\,du\Bigg| B\Bigg]^{-1} \\
&= \mathbf{E}_r\left[\int \exp\left(\sum_{\phi\in A[i]} r_\phi t_\phi\right)\,du\middle| B\right]^{-1} \\
&= \left(\int \mathbf{E}_r\left[\exp\left(\sum_{\phi\in A[i]} r_\phi t_\phi\right)\middle| B\right]\,du\right)^{-1},
\end{aligned}$$

where we define

$$t_\phi = \log\left(1+\bar{\phi}(z_u)\right) - \log\left(1+\bar{\phi}(x)\right) - \log\left(1+\bar{\phi}(y)\right).$$

This inner expectation is now just the moment-generating function of the multinomial distribution. Applying the standard formula for that MGF gives us

$$\mathbf{E}_r\left[\exp\left(\sum_{\phi\in A[i]} r_\phi t_\phi\right)\middle| B\right] = \left(\sum_{\phi\in A[i]} \frac{\lambda M_\phi}{\Lambda L}\cdot\exp(t_\phi)\right)^B.$$

Substituting this back into our original expression gives

$$\pi(x)T(x,y) \geq \frac{\exp(U_{\neg i}(x))}{nZ}\,\mathbf{E}_B\left[\frac{1-C}{1+C}\cdot\left(\int\left(\sum_{\phi\in A[i]}\frac{\lambda M_\phi}{\Lambda L}\cdot\exp(t_\phi)\right)^B du\right)^{-1}\right].$$

Next, let $\delta > 0$ be a small constant, to be assigned later. Recall that for any non-negative random variable X and any event A, by the Law of Total Probability,

$$\mathbf{E}[X] = \mathbf{E}[X|A]\cdot\mathbf{P}(A) + \mathbf{E}[X|\neg A]\cdot\mathbf{P}(\neg A) \geq \mathbf{E}[X|A]\cdot\mathbf{P}(A).$$

So, since the interior of this expectation is a non-negative number, it follows that

$$\begin{aligned}
&\pi(x)T(x,y) \\
&\quad \geq \frac{\exp(U_{\neg i}(x))}{nZ}\, \mathbf{E}_B \left[\frac{1-C}{1+C} \cdot \left(\int \left(\sum_{\phi \in A[i]} \frac{\lambda M_\phi}{\Lambda L} \cdot \exp(t_\phi) \right)^B du \right)^{-1} \middle| C \leq \delta \right] \\
&\qquad \cdot \mathbf{P}_B(C \leq \delta) \\
&\quad \geq \frac{\exp(U_{\neg i}(x))}{nZ} \cdot \frac{1-\delta}{1+\delta} \cdot \mathbf{E}_B \left[\left(\int \left(\sum_{\phi \in A[i]} \frac{\lambda M_\phi}{\Lambda L} \cdot \exp(t_\phi) \right)^B du \right)^{-1} \middle| C \leq \delta \right] \\
&\qquad \cdot \mathbf{P}_B(C \leq \delta).
\end{aligned}$$

By Jensen's inequality again, we get

$$\begin{aligned}
&\pi(x)T(x,y) \\
&\quad \geq \frac{\exp(U_{\neg i}(x))}{nZ}\, \mathbf{E}_B \left[\frac{1-C}{1+C} \cdot \left(\int \left(\sum_{\phi \in A[i]} \frac{\lambda M_\phi}{\Lambda L} \cdot \exp(t_\phi) \right)^B du \right)^{-1} \middle| C \leq \delta \right] \\
&\qquad \cdot \mathbf{P}_B(C \leq \delta) \\
&\quad \geq \frac{\exp(U_{\neg i}(x))}{nZ} \cdot \frac{1-\delta}{1+\delta} \cdot \left(\int \mathbf{E}_B \left[\left(\sum_{\phi \in A[i]} \frac{\lambda M_\phi}{\Lambda L} \cdot \exp(t_\phi) \right)^B \middle| C \leq \delta \right] du \right)^{-1} \\
&\qquad \cdot \mathbf{P}_B(C \leq \delta).
\end{aligned}$$

Since this inner expectation is again non-negative, we can again apply our above inequality, but in the opposite direction, giving

$$\mathbf{E}\left[X|A\right] \leq \frac{\mathbf{E}\left[X\right]}{\mathbf{P}(A)}.$$

This produces

$$\begin{aligned}
\pi(x)T(x,y) \geq \frac{\exp(U_{\neg i}(x))}{nZ} \cdot \frac{1-\delta}{1+\delta} \cdot \left(\int \mathbf{E}_B \left[\left(\sum_{\phi \in A[i]} \frac{\lambda M_\phi}{\Lambda L} \cdot \exp(t_\phi) \right)^B \right] du \right)^{-1} \\
\cdot \mathbf{P}_B(C \leq \delta)^2.
\end{aligned}$$

Now, we are just left with the MGF of a Poisson-distributed random variable. This we already know to be

$$\begin{aligned}\mathbf{E}_B\left[\left(\sum_{\phi\in A[i]}\frac{\lambda M_\phi}{\Lambda L}\cdot\exp(t_\phi)\right)^B\right] &= \mathbf{E}_B\left[\exp\left(B\log\left(\sum_{\phi\in A[i]}\frac{\lambda M_\phi}{\Lambda L}\cdot\exp(t_\phi)\right)\right)\right]\\ &= \exp\left(\Lambda\left(\left(\sum_{\phi\in A[i]}\frac{\lambda M_\phi}{\Lambda L}\cdot\exp(t_\phi)\right)-1\right)\right)\\ &= \exp\left(\sum_{\phi\in A[i]}\frac{\lambda M_\phi}{L}\cdot(\exp(t_\phi)-1)\right),\end{aligned}$$

where in the last line we can leverage the fact that

$$\sum_{\phi\in A[i]}\frac{\lambda M_\phi}{\Lambda L}=1$$

to justify pulling the -1 inside the sum. From the analysis of Poisson-Gibbs, we had that

$$\exp(t_\phi)-1\leq\bar{\phi}(z_u)-\bar{\phi}(x)-\bar{\phi}(y)+\frac{4L^2}{\lambda^2}.$$

So,

$$\begin{aligned}&\mathbf{E}_B\left[\left(\sum_{\phi\in A[i]}\frac{\lambda M_\phi}{\Lambda L}\cdot\exp(t_\phi)\right)^B\right]\\ &\quad\leq\exp\left(\sum_{\phi\in A[i]}\frac{\lambda M_\phi}{L}\cdot\left(\bar{\phi}(z_u)-\bar{\phi}(x)-\bar{\phi}(y)+\frac{4L^2}{\lambda^2}\right)\right)\\ &\quad=\exp\left(\sum_{\phi\in A[i]}\left(\phi(z_u)-\phi(x)-\phi(y)+\frac{4LM_\phi}{\lambda}\right)\right)\\ &\quad\leq\exp\left(\bar{U}_u-\bar{U}_{x(i)}-\bar{U}_{y(i)}+\frac{4L^2}{\lambda}\right),\end{aligned}$$

where as in the analysis of Poisson-Gibbs, $\bar{U}_v$ denotes the assignment of U_v in the plain Gibbs sampling algorithm (Algorithm 1),

$$\bar{U}_v=\sum_{\phi\in A[i]}\phi(z_v).$$

Substituting this expression in to our overall bound, we get

$$\begin{aligned}\pi(x)T(x,y) &\geq \frac{\exp(U_{\neg i}(x))}{nZ} \cdot \frac{1-\delta}{1+\delta} \cdot \left(\int \exp\left(\bar{U}_u - \bar{U}_{x(i)} - \bar{U}_{y(i)} + \frac{4L^2}{\lambda} \right) \, du \right)^{-1} \\ &\qquad \cdot \mathbf{P}_B(C \leq \delta)^2 \\ &= \frac{\exp(U(x))}{nZ} \cdot \frac{1-\delta}{1+\delta} \cdot \frac{\exp(\bar{U}_{y(i)})}{\int \exp\left(\bar{U}_u\right) \, du} \\ &\qquad \cdot \exp\left(-\frac{4L^2}{\lambda}\right) \cdot \mathbf{P}_B(C \leq \delta)^2.\end{aligned}$$

Finally, if we let G denote the transition probability operator of plain Gibbs sampling, we notice right away that

$$\begin{aligned}\pi(x)T(x,y) &\geq \frac{1-\delta}{1+\delta} \cdot \exp\left(-\frac{4L^2}{\lambda}\right) \cdot \mathbf{P}_B(C \leq \delta)^2 \cdot \pi(x)G(x,y) \\ &\geq (1-2\delta) \cdot \exp\left(-\frac{4L^2}{\lambda}\right) \cdot \mathbf{P}_B(C \leq \delta)^2 \cdot \pi(x)G(x,y).\end{aligned}$$

To get a final bound, all we need to do is bound $\mathbf{P}_B(C \leq \delta)$. This is straightforward, since

$$\begin{aligned}\mathbf{P}_B(C \leq \delta) &= \mathbf{P}_B\left(\frac{4\rho^{-m}}{\rho-1} \cdot \exp\left(\frac{LB}{\lambda}\right) \leq \delta\right) \\ &= \mathbf{P}_B\left(\exp\left(\frac{LB}{\lambda}\right) \leq \frac{\rho-1}{4\rho^{-m}} \cdot \delta\right).\end{aligned}$$

Notice that by the MGF formula for B,

$$\mathbf{E}_B\left[\exp\left(\frac{LB}{\lambda}\right)\right] \leq \exp\left(\Lambda\left(\exp\left(\frac{L}{\lambda}\right) - 1\right)\right).$$

Since we chose a minibatch size parameter $\lambda \geq 2L$, it follows that $L/\lambda \leq 1/2$, and so

$$\exp\left(\frac{L}{\lambda}\right) - 1 \leq \frac{2L}{\lambda},$$

and so since also

$$\Lambda = \sum_{\phi \in A[i]} \frac{\lambda M_\phi}{L} \leq \lambda.$$

it follows that

$$\mathbf{E}_B\left[\exp\left(\frac{LB}{\lambda}\right)\right] \leq \exp\left(\lambda \cdot \frac{2L}{\lambda}\right) = \exp(2L).$$

Therefore, by Markov's inequality,

$$\begin{aligned}\mathbf{P}_B(C \geq \delta) &= \mathbf{P}_B\left(\exp\left(\frac{LB}{\lambda}\right) \geq \frac{\rho-1}{4\rho^{-m}} \cdot \delta\right) \\ &\leq \frac{\exp(2L)}{\frac{\rho-1}{4\rho^{-m}} \cdot \delta} \\ &\leq \frac{4\rho^{-m}}{\rho-1} \cdot \frac{\exp(2L)}{\delta}.\end{aligned}$$

Thus,

$$\begin{aligned}\mathbf{P}_B(C \leq \delta) &= 1 - \mathbf{P}_B(C \geq \delta) \\ &\geq 1 - \frac{4\rho^{-m}}{\rho-1} \cdot \frac{\exp(2L)}{\delta},\end{aligned}$$

and in particular

$$\begin{aligned}\mathbf{P}_B(C \leq \delta)^2 &= (1 - \mathbf{P}_B(C \geq \delta))^2 \\ &\geq 1 - 2\mathbf{P}_B(C \geq \delta) \\ &\geq 1 - \frac{8\rho^{-m}}{\rho-1} \cdot \frac{\exp(2L)}{\delta}.\end{aligned}$$

Substituting this back into our overall bound gives us

$$\begin{aligned}\pi(x)T(x,y) &\geq \frac{1-\delta}{1+\delta} \cdot \exp\left(-\frac{4L^2}{\lambda}\right) \cdot \mathbf{P}_B(C \leq \delta)^2 \cdot \pi(x)G(x,y) \\ &\geq (1-2\delta) \cdot \left(1 - \frac{8\rho^{-m}}{\rho-1} \cdot \frac{\exp(2L)}{\delta}\right) \cdot \exp\left(-\frac{4L^2}{\lambda}\right) \cdot \pi(x)G(x,y) \\ &\geq \left(1 - 2\delta - \frac{8\rho^{-m}}{\rho-1} \cdot \frac{\exp(2L)}{\delta}\right) \cdot \exp\left(-\frac{4L^2}{\lambda}\right) \cdot \pi(x)G(x,y).\end{aligned}$$

Finally, choosing the value of δ as

$$\delta = \frac{2\exp(L)}{\rho^{m/2} \cdot \sqrt{\rho-1}},$$

we get

$$\pi(x)T(x,y) \geq \left(1 - \frac{8\exp(L)\rho^{-m/2}}{\sqrt{\rho-1}}\right) \cdot \exp\left(-\frac{4L^2}{\lambda}\right) \cdot \pi(x)G(x,y).$$

Now applying the standard Dirichlet form argument, we get

$$\bar{\gamma} \geq \left(1 - \frac{8\exp(L)\rho^{-m/2}}{\sqrt{\rho-1}}\right) \cdot \exp\left(-\frac{4L^2}{\lambda}\right) \cdot \gamma,$$

which was the desired expression. □

A.2.4 Proof of Theorem 4

Proof. The reversibility can be proved by the same procedure as in Section A.2.3. By applying that same analysis, which did not depend on the manner in which the approximation $\tilde{f}$ was constructed, we can arrive at the expression

$$\begin{aligned}
&\pi(x)T(x,y) \\
&\quad = \frac{\exp(U_{\neg i}(x))}{nZ}\, \mathbf{E}_r\left[\frac{1}{\int \tilde{f}(u)du} \cdot \min\left(\tilde{f}(y(i))\exp\left(U_{x(i)}\right), \tilde{f}(x(i))\exp\left(U_{y(i)}\right)\right)\right].
\end{aligned}$$

By the assumption of $\phi(z)$, we have

$$\begin{aligned}
|U_v| &= \left|\sum_{\phi\in A[i]} r_\phi \log\left(1+\bar{\phi}(z_v)\right)\right| \\
&\leq \sum_{\phi\in A[i]} r_\phi \left|\log\left(1+\frac{L}{\lambda M_\phi}\phi(x)\right)\right| \\
&\leq \sum_{\phi\in A[i]} r_\phi \left|\frac{2L}{\lambda M_\phi}\phi(x)\right| \\
&\leq \frac{2L}{\lambda}\sum_{\phi\in A[i]} r_\phi.
\end{aligned}$$

where the second inequality holds because

$$|z| \leq \frac{1}{2} \quad \Rightarrow \quad |\log(1+z)| \leq 2\,|z|\,,$$

using the assumptions $\lambda \geq 2L$ and $|\phi(x)| \leq M_\phi$. Now applying Lemma 1 in Section A.4, assigning $\sigma = \sqrt{\rho}$ gives us,

$$\left|\tilde{U}_v - U_v\right| \leq \frac{8\rho^{-\frac{m}{2}}}{\sqrt{\rho}-1} \cdot \frac{L}{\lambda} \sum_{\phi \in A[i]} r_\phi,$$

for any v in the shifted-and-scaled Bernstein ellipse with parameter $\sqrt{\rho}$.

Next, since $\tilde{U}_v$ is a polynomial in v, $\exp(\tilde{U}_v)$ must be analytic everywhere in $\mathbb{C}$. In particular it must be analytic on the Bernstein ellipse on the interval $[a, b]$ with parameter $\sqrt{\rho}$. On that interval, it is bounded by

$$\begin{aligned}
\left|\exp(\tilde{U}_v)\right| &\leq \exp\left(\left|\tilde{U}_v\right|\right) \\
&\leq \exp\left(|U_v| + \left|\tilde{U}_v - U_v\right|\right) \\
&\leq \exp\left(\frac{2L}{\lambda} \sum_{\phi \in A[i]} r_\phi\right) \cdot \exp\left(\frac{8\rho^{-\frac{m}{2}}}{\sqrt{\rho}-1} \cdot \frac{L}{\lambda} \sum_{\phi \in A[i]} r_\phi\right) \\
&\leq \exp\left(\frac{4\rho^{-\frac{m}{2}} + \sqrt{\rho} - 1}{\sqrt{\rho}-1} \cdot \frac{2L}{\lambda} \sum_{\phi \in A[i]} r_\phi\right).
\end{aligned}$$

Now applying Theorem 3 using the Bernstein ellipse with parameter $\sqrt{\rho}$, we have, for any v on the interval $[a, b]$,

$$\left|\tilde{f}(v) - \exp(\tilde{U}_v)\right| \leq \frac{4\rho^{-\frac{k}{2}}}{\sqrt{\rho}-1} \cdot \exp\left(\frac{4\rho^{-\frac{m}{2}} + \sqrt{\rho} - 1}{\sqrt{\rho}-1} \cdot \frac{2L}{\lambda} \sum_{\phi \in A[i]} r_\phi\right)$$

Therefore, it follows that

$$\begin{aligned}
\left|\frac{\tilde{f}(v)}{\exp(U_v)} - 1\right| &\leq \left|\frac{\tilde{f}(v) - \exp(\tilde{U}_v) + \exp(\tilde{U}_v)}{\exp(U_v)} - 1\right| \\
&\leq \frac{\left|\tilde{f}(v) - \exp(\tilde{U}_v)\right|}{\exp(U_v)} + \left|\exp(\tilde{U}_v - U_v) - 1\right| \\
&\leq \left|\tilde{f}(v) - \exp(\tilde{U}_v)\right| + \exp\left(\left|\tilde{U}_v - U_v\right|\right) - 1,
\end{aligned}$$

where the last inequality is justified by the fact that U_v is non-negative and for any x, $|\exp(x) - 1| \leq \exp(|x|) - 1$. Now substituting in our bounds from above gives us

$$\left|\frac{\tilde{f}(v)}{\exp(U_v)} - 1\right| \leq \exp\left(\frac{8\rho^{-\frac{m}{2}}}{\sqrt{\rho}-1} \cdot \frac{L}{\lambda} \sum_{\phi \in A[i]} r_\phi\right) + \frac{4\rho^{-\frac{k}{2}}}{\sqrt{\rho}-1} \cdot \exp\left(\frac{4\rho^{-\frac{m}{2}} + \sqrt{\rho} - 1}{\sqrt{\rho}-1} \cdot \frac{2L}{\lambda} \sum_{\phi \in A[i]} r_\phi\right) - 1.$$

As before, we let $B = \sum_{\phi \in A[i]} r_\phi$ where $B \sim \text{Poisson}(\Lambda)$. Then

$$\left|\frac{\tilde{f}(v)}{\exp(U_v)} - 1\right| \leq \exp\left(\frac{8\rho^{-\frac{m}{2}}}{\sqrt{\rho}-1} \cdot \frac{LB}{\lambda}\right) + \frac{4\rho^{-\frac{k}{2}}}{\sqrt{\rho}-1} \cdot \exp\left(\frac{4\rho^{-\frac{m}{2}} + \sqrt{\rho} - 1}{\sqrt{\rho}-1} \cdot \frac{2LB}{\lambda}\right) - 1$$

We define

$$E = \exp\left(\frac{8\rho^{-\frac{m}{2}}}{\sqrt{\rho}-1} \cdot \frac{LB}{\lambda}\right) + \frac{4\rho^{-\frac{k}{2}}}{\sqrt{\rho}-1} \cdot \exp\left(\frac{4\rho^{-\frac{m}{2}} + \sqrt{\rho} - 1}{\sqrt{\rho}-1} \cdot \frac{2LB}{\lambda}\right) - 1,$$

and by following the same steps as used in Section A.2.3, with E in place of the C of that proof, we can get, for any constant $\delta > 0$,

$$\pi(x)T(x,y) \geq (1 - 2\delta) \cdot \exp\left(-\frac{4L^2}{\lambda}\right) \cdot \mathbf{P}_B(E \leq \delta)^2 \cdot \pi(x)G(x,y).$$

All that remains is to bound $\mathbf{P}_B(E \leq \delta)$. Using the MGF formula for B twice, we get that

$$\mathbf{E}_B(E) = \frac{4\rho^{-\frac{k}{2}}}{\sqrt{\rho}-1} \cdot \exp\left(\Lambda\left(\exp\left(\frac{4\rho^{-\frac{m}{2}} + \sqrt{\rho} - 1}{\sqrt{\rho}-1} \cdot \frac{2L}{\lambda}\right) - 1\right)\right) + \exp\left(\Lambda\left(\exp\left(\frac{8\rho^{-\frac{m}{2}}}{\sqrt{\rho}-1} \cdot \frac{L}{\lambda}\right) - 1\right)\right) - 1.$$

If we require that m is large enough that

$$4\rho^{-\frac{m}{2}} \leq \sqrt{\rho} - 1,$$

then

$$\mathbf{E}_B(E) \leq \frac{4\rho^{-\frac{k}{2}}}{\sqrt{\rho}-1} \cdot \exp\left(\Lambda\left(\exp\left(\frac{4L}{\lambda}\right)-1\right)\right) + \exp\left(\Lambda\left(\exp\left(\frac{8\rho^{-\frac{m}{2}}}{\sqrt{\rho}-1}\cdot\frac{L}{\lambda}\right)-1\right)\right)-1.$$

By Taylor's theorem, for $x > 0$,

$$\exp(x) - 1 = \exp(x) - \exp(0) \leq x \cdot \exp(x).$$

So, since $\Lambda \leq \lambda$, we can bound our expectation with

$$\begin{aligned}
\mathbf{E}_B(E) &\leq \frac{4\rho^{-\frac{k}{2}}}{\sqrt{\rho}-1} \cdot \exp\left(\Lambda \cdot \frac{4L}{\lambda} \cdot \exp\left(\frac{4L}{\lambda}\right)\right) \\
&\quad + \exp\left(\Lambda \cdot \frac{8\rho^{-\frac{m}{2}}}{\sqrt{\rho}-1} \cdot \frac{L}{\lambda} \cdot \exp\left(\frac{8\rho^{-\frac{m}{2}}}{\sqrt{\rho}-1}\cdot\frac{L}{\lambda}\right)\right) - 1 \\
&\leq \frac{4\rho^{-\frac{k}{2}}}{\sqrt{\rho}-1} \cdot \exp\left(4L \cdot \exp\left(\frac{4L}{\lambda}\right)\right) \\
&\quad + \exp\left(\frac{8\rho^{-\frac{m}{2}}}{\sqrt{\rho}-1} \cdot L \cdot \exp\left(\frac{8\rho^{-\frac{m}{2}}}{\sqrt{\rho}-1}\cdot\frac{L}{\lambda}\right)\right) - 1 \\
&\leq \frac{4\rho^{-\frac{k}{2}}}{\sqrt{\rho}-1} \cdot \exp\left(4L \cdot \exp\left(\frac{4L}{\lambda}\right)\right) \\
&\quad + \exp\left(\frac{8\rho^{-\frac{m}{2}}}{\sqrt{\rho}-1} \cdot L \cdot \exp\left(\frac{4L}{\lambda}\right)\right) - 1.
\end{aligned}$$

Since $\lambda \log(2) \geq 4L$, we can bound $\exp(4L/\lambda) \leq 2$, and so

$$\mathbf{E}_B(E) \leq \frac{4\rho^{-\frac{k}{2}}}{\sqrt{\rho}-1} \cdot \exp(8L) + \exp\left(\frac{16L\rho^{-\frac{m}{2}}}{\sqrt{\rho}-1}\right) - 1.$$

We now define

$$F = \frac{4 \cdot \exp(8L) \cdot \rho^{-\frac{k}{2}}}{\sqrt{\rho}-1} + \exp\left(\frac{16L\rho^{-\frac{m}{2}}}{\sqrt{\rho}-1}\right) - 1.$$

By Markov's inequality,

$$\mathbf{P}_B(E \geq \delta) \geq \frac{\mathbf{E}_B(E)}{\delta} \geq F/\delta.$$

It follows

$$\mathbf{P}_B(E \leq \delta)^2 = (1 - \mathbf{P}_B(E \geq \delta))^2 \geq 1 - 2\mathbf{P}_B(E \geq \delta) \geq 1 - 2F/\delta.$$

Substituting it back into the overall bound,

$$\begin{aligned}\pi(x)T(x,y) &\geq (1-2\delta) \cdot \exp\left(-\frac{4L^2}{\lambda}\right) \cdot \mathbf{P}_B(E \leq \delta)^2 \cdot \pi(x)G(x,y) \\ &\geq \left(1 - 2\delta - \frac{2F}{\delta}\right) \cdot \exp\left(-\frac{4L^2}{\lambda}\right) \cdot \pi(x)G(x,y)\end{aligned}$$

Let

$$\delta = \sqrt{F},$$

it becomes

$$\pi(x)T(x,y) \geq \left(1 - 4\sqrt{F}\right) \cdot \exp\left(-\frac{4L^2}{\lambda}\right) \cdot \pi(x)G(x,y)$$

Again, using the Dirichlet form we bound the spectral gap,

$$\bar{\gamma} \geq \left(1 - 4\sqrt{F}\right) \exp\left(\frac{-4L^2}{\lambda}\right) \cdot \gamma$$

□

A.3 PoissonMH

We apply our Poisson-minibatching method to Metropolis-Hastings sampling. In Poisson-minibatching MH (PoissonMH), we first generate a candidate x^* from the proposal distribution $q(x^*|x)$. Then the MH ratio will be calculated as following

$$p = \frac{\exp\left(\sum_{\phi \in S} s_\phi \log\left(1 + \frac{L}{\lambda M_\phi}\phi(x^*)\right)\right) q(x^*|x)}{\exp\left(\sum_{\phi \in S} s_\phi \log\left(1 + \frac{L}{\lambda M_\phi}\phi(x)\right)\right) q(x|x^*)}$$

We accept x^* with the probability $\min(1, p)$. After applying Poisson-minibatching, the MH ratio no longer needs to use the whole dataset which will reduce the computational cost significantly.

Theorem 2 is similar to the bounds of Poisson-Gibbs. As long as we set $\lambda = \Theta(L^2)$, the convergence is slowed down by at most a constant factor which is unrelated to the size of the problem.

A.3.1 Proof of Theorem 2

Proof. We begin with the transition probability from x to x^*

$$
\begin{aligned}
&T(x^*, x) \\
&= \mathbf{E}\left\{ q(x^*|x) \min\left(1, \frac{q(x|x^*)\pi(x^*, s)}{q(x^*|x)\pi(x, s)}\right)\right\} \\
&= \mathbf{E}\left\{ q(x^*|x) \min\left(1, \frac{q(x|x^*)\exp\left(\sum_{\phi\in\Phi}\left[s_\phi \log\left(\frac{\lambda M_\phi}{L} + \phi(x^*)\right) - \log s_\phi!\right]\right)}{q(x^*|x)\exp\left(\sum_{\phi\in\Phi}\left[s_\phi \log\left(\frac{\lambda M_\phi}{L} + \phi(x)\right) - \log s_\phi!\right]\right)}\right)\right\} \\
&= \mathbf{E}\left\{ q(x^*|x) \min\left(1, \frac{q(x|x^*)\exp\left(\sum_{\phi\in\Phi}\left[s_\phi \log\left(\frac{\lambda M_\phi}{L} + \phi(x^*)\right)\right]\right)}{q(x^*|x)\exp\left(\sum_{\phi\in\Phi}\left[s_\phi \log\left(\frac{\lambda M_\phi}{L} + \phi(x)\right)\right]\right)}\right)\right\} \\
&= \sum_s \left\{ q(x^*|x) \min\left(1, \frac{q(x|x^*)\exp\left(\sum_{\phi\in\Phi}\left[s_\phi \log\left(\frac{\lambda M_\phi}{L} + \phi(x^*)\right)\right]\right)}{q(x^*|x)\exp\left(\sum_{\phi\in\Phi}\left[s_\phi \log\left(\frac{\lambda M_\phi}{L} + \phi(x)\right)\right]\right)}\right)\right\} \prod_{\phi\in\Phi} p(s_\phi|x) \\
&= \sum_s \left\{ q(x^*|x) \min\left(\exp\left(\sum_{\phi\in\Phi}\left[s_\phi \log\left(\frac{\lambda M_\phi}{L} + \phi(x)\right) - \phi(x) - \frac{\lambda M_\phi}{L} - \log s_\phi!\right]\right),\right.\right. \\
&\qquad \left.\left. \frac{q(x|x^*)\exp\left(\sum_{\phi\in\Phi}\left[s_\phi \log\left(\frac{\lambda M_\phi}{L} + \phi(x^*)\right)\right]\right)}{q(x^*|x)\exp\left(\sum_{\phi\in\Phi}\phi(x) + \frac{\lambda M_\phi}{L} + \log s_\phi!\right)}\right)\right\} \\
&= \sum_s \left\{ q(x^*|x) \min\left(\exp\left(\sum_{\phi\in\Phi}\left[s_\phi \log\left(\frac{\lambda M_\phi}{L} + \phi(x)\right) - \phi(x) - \frac{\lambda M_\phi}{L} - \log s_\phi!\right]\right),\right.\right. \\
&\qquad \left.\left. \frac{q(x|x^*)}{q(x^*|x)}\exp\left(\sum_{\phi\in\Phi}\left[s_\phi \log\left(\frac{\lambda M_\phi}{L} + \phi(x^*)\right) - \phi(x) - \frac{\lambda M_\phi}{L} - \log s_\phi!\right]\right)\right)\right\}
\end{aligned}
$$

Multiplying $\pi(x)$ to both sides,

$$
\begin{aligned}
&\pi(x)T(x^*, x) \\
&= \frac{1}{Z} \exp\left(\sum_{\phi\in\Phi} \phi(x)\right) T(x^*, x) \\
&= \frac{1}{Z} \sum_s \min\left(q(x^*|x) \left(\exp\left(\sum_{\phi\in\Phi} \left[s_\phi \log\left(\frac{\lambda M_\phi}{L} + \phi(x)\right) - \frac{\lambda M_\phi}{L} - \log s_\phi!\right]\right),\right.\right. \\
&\qquad \left.\left. q(x|x^*) \exp\left(\sum_{\phi\in\Phi} \left[s_\phi \log\left(\frac{\lambda M_\phi}{L} + \phi(x^*)\right) - \frac{\lambda M_\phi}{L} - \log s_\phi!\right]\right)\right)\right)
\end{aligned}
$$

This implies the Markov chain is reversible.

We can continue to reduce this to

$$
\begin{aligned}
&\pi(x)T(x^*, x) \\
&= \frac{1}{Z} \sum_s \min\left(q(x^*|x) \exp\left(\sum_{\phi\in\Phi} s_\phi \left[\log\left(\frac{\lambda M_\phi}{L} + \phi(x)\right) - \log\frac{\lambda M_\phi}{L}\right]\right),\right. \\
&\qquad \left. q(x|x^*) \exp\left(\sum_{\phi\in\Phi} s_\phi \left[\log\left(\frac{\lambda M_\phi}{L} + \phi(x^*)\right) - \log\frac{\lambda M_\phi}{L}\right]\right)\right) \\
&\qquad \cdot \prod_{\phi\in\Phi} \frac{1}{s_\phi!} \exp\left(-\frac{\lambda M_\phi}{L}\right) \left(\frac{\lambda M_\phi}{L}\right)^{s_\phi} \\
&= \frac{1}{Z} \sum_s \min\left(q(x^*|x) \exp\left(\sum_{\phi\in\Phi} s_\phi \log\left(1 + \frac{L}{\lambda M_\phi}\phi(x)\right)\right),\right. \\
&\qquad \left. q(x|x^*) \exp\left(\sum_{\phi\in\Phi} s_\phi \log\left(1 + \frac{L}{\lambda M_\phi}\phi(x^*)\right)\right)\right) \\
&\qquad \cdot \prod_{\phi\in\Phi} \frac{1}{s_\phi!} \exp\left(-\frac{\lambda M_\phi}{L}\right) \left(\frac{\lambda M_\phi}{L}\right)^{s_\phi}
\end{aligned}
$$

Similar to the previous proof, s_ϕ here are non-negative integers that a Poisson variable can take, not variables. So if we let $r_\phi \sim \text{Poisson}\left(\frac{\lambda M_\phi}{L}\right)$ and r_ϕ to be all

independent, we can write this as

$$\pi(x)T(x^*,x) = \frac{1}{Z}\mathbf{E}\min\left(q(x^*|x)\exp\left(\sum_{\phi\in\Phi} r_\phi \log\left(1+\frac{L}{\lambda M_\phi}\phi(x)\right)\right),\right.$$
$$\left. q(x|x^*)\exp\left(\sum_{\phi\in\Phi} r_\phi \log\left(1+\frac{L}{\lambda M_\phi}\phi(x^*)\right)\right)\right)$$

Assume $G(x^*,x)$ is the transition operator of a plain MCMC. Consider the ratio,

$$\begin{aligned}
&\frac{\pi(x)T(x^*,x)}{\pi(x)G(x^*,x)} \\
&= \frac{1}{Z}\mathbf{E}\min\left(q(x^*|x)\exp\left(\sum_{\phi\in\Phi} r_\phi \log\left(1+\frac{L}{\lambda M_\phi}\phi(x)\right)\right),\right. \\
&\quad \left. q(x|x^*)\exp\left(\sum_{\phi\in\Phi} r_\phi \log\left(1+\frac{L}{\lambda M_\phi}\phi(x^*)\right)\right)\right) \\
&\quad \cdot\left[1\Big/\left(\frac{1}{Z}\min\left(q(x^*|x)\exp\left(\sum_{\phi\in\Phi}\phi(x)\right), q(x|x^*)\exp\left(\sum_{\phi\in\Phi}\phi(x^*)\right)\right)\right)\right]
\end{aligned}$$

We know that $\frac{\min(A,B)}{\min(C,D)} = \min\left(\frac{A}{\min(C,D)}, \frac{B}{\min(C,D)}\right) \geq \min\left(\frac{A}{C}, \frac{B}{D}\right)$. The last inequality is due to the fact that $\frac{1}{\min(C,D)} \geq \frac{1}{C}$ and $\frac{1}{\min(C,D)} \geq \frac{1}{D}$.

With this inequality, we can continue simplifying the ratio,

$$\begin{aligned}\frac{\pi(x)T(x^*,x)}{\pi(x)G(x^*,x)} &\geq \mathbf{E}\Bigg[\min\Bigg(\frac{\exp\left(\sum_{\phi\in\Phi} r_\phi \log\left(1+\frac{L}{\lambda M_\phi}\phi(x)\right)\right)}{\exp\left(\sum_{\phi\in\Phi}\phi(x)\right)},\\ &\qquad \frac{\exp\left(\sum_{\phi\in\Phi} r_\phi \log\left(1+\frac{L}{\lambda M_\phi}\phi(x^*)\right)\right)}{\exp\left(\sum_{\phi\in\Phi}\phi(x^*)\right)}\Bigg)\Bigg]\\ &= \mathbf{E}\Bigg[\min\Bigg(\exp\left(\sum_{\phi\in\Phi}\left(r_\phi\log\left(1+\frac{L}{\lambda M_\phi}\phi(x)\right)-\phi(x)\right)\right),\\ &\qquad \exp\left(\sum_{\phi\in\Phi}\left(r_\phi\log\left(1+\frac{L}{\lambda M_\phi}\phi(x^*)\right)-\phi(x^*)\right)\right)\Bigg)\Bigg]\\ &= \mathbf{E}\Bigg[\max\Bigg(\exp\left(\sum_{\phi\in\Phi}\left(\phi(x)-r_\phi\log\left(1+\frac{L}{\lambda M_\phi}\phi(x)\right)\right)\right),\\ &\qquad \exp\left(\sum_{\phi\in\Phi}\left(\phi(x^*)-r_\phi\log\left(1+\frac{L}{\lambda M_\phi}\phi(x^*)\right)\right)\right)\Bigg)^{-1}\Bigg]\end{aligned}$$

Because $f(x)=\frac{1}{x}$ is a convex function, by Jensen's inequality it follows

$$\begin{aligned}\frac{\pi(x)T(x^*,x)}{\pi(x)G(x^*,x)} &\geq \mathbf{E}\Bigg[\max\Bigg(\exp\left(\sum_{\phi\in\Phi}\left(\phi(x)-r_\phi\log\left(1+\frac{L}{\lambda M_\phi}\phi(x)\right)\right)\right),\\ &\qquad \exp\left(\sum_{\phi\in\Phi}\left(\phi(x^*)-r_\phi\log\left(1+\frac{L}{\lambda M_\phi}\phi(x^*)\right)\right)\right)\Bigg)\Bigg]^{-1}\end{aligned}$$

We have that the maximum of the product is less than the product of maximum, therefore

$$\begin{aligned}\frac{\pi(x)T(x^*,x)}{\pi(x)G(x^*,x)} &\geq \prod_{\phi\in\Phi}\mathbf{E}\Bigg[\max\Bigg(\exp\left(\phi(x)-r_\phi\log\left(1+\frac{L}{\lambda M_\phi}\phi(x)\right)\right),\\ &\qquad \exp\left(\phi(x^*)-r_\phi\log\left(1+\frac{L}{\lambda M_\phi}\phi(x^*)\right)\right)\Bigg)\Bigg]^{-1}\end{aligned}$$

Since $\max(A, B) \leq A + B$ when A and B are positive, it follows

$$\frac{\pi(x)T(x^*, x)}{\pi(x)G(x^*, x)} \geq \prod_{\phi \in \Phi} \mathbf{E}\left[\exp\left(\phi(x) - r_\phi \log\left(1 + \frac{L}{\lambda M_\phi}\phi(x)\right)\right) + \exp\left(\phi(x^*) - r_\phi \log\left(1 + \frac{L}{\lambda M_\phi}\phi(x^*)\right)\right)\right]^{-1}$$

$\mathbf{E}\left[\exp\left(- r_\phi \log\left(1 + \frac{L}{\lambda M_\phi}\phi(x)\right)\right)\right]$ is the moment generating function of the Poisson random variable r_ϕ evaluated at

$$t = -\log\left(1 + \frac{L}{\lambda M_\phi}\phi(x)\right)$$

We know that

$$\mathbf{E}\exp(r_\phi t) = \exp\left(\frac{\lambda M_\phi}{L}\left(\exp(t) - 1\right)\right)$$

Therefore,

$$\mathbf{E}\left[\exp\left(- r_\phi \log\left(1 + \frac{L}{\lambda M_\phi}\phi(x)\right)\right)\right] = \exp\left(-\frac{\phi(x)}{1 + \frac{L}{\lambda M_\phi}\phi(x)}\right)$$

Substituting this into the original expression produces

$$\begin{aligned}
\frac{\pi(x)T(x^*, x)}{\pi(x)G(x^*, x)} &\geq \left[2\prod_{\phi \in \Phi} \exp\left(- \frac{\phi(x)}{1 + \frac{L}{\lambda M_\phi}\phi(x)} + \phi(x)\right)\right]^{-1} \\
&\geq \left[2\prod_{\phi \in \Phi} \exp(M_\phi)\exp\left(- \frac{1}{1 + \frac{L}{\lambda}} + 1\right)\right]^{-1} \\
&= \left[2\prod_{\phi \in \Phi} \exp(M_\phi)\exp\left(\frac{L}{\lambda + L}\right)\right]^{-1} \\
&= \left[2\exp\left(\frac{L^2}{\lambda + L}\right)\right]^{-1} \\
&= \frac{1}{2}\exp\left(- \frac{L^2}{\lambda + L}\right)
\end{aligned}$$

From Dirichlet form argument, we get

$$\bar{\gamma} \geq \frac{1}{2} \exp\left(-\frac{L^2}{\lambda + L}\right) \cdot \gamma.$$

□

A.3.2 Additional Experiment: PoissonMH on Truncated Gaussian Mixture

We test PoissonMH on the truncated Gaussian mixture as in Section 3.1.4. The proposal is $q(x^*|x) = \mathcal{N}(x, 0.45^2 I)$. We set $\lambda = 500$. The estimated density is in Figure A.1 which is very close to the true density. This demonstrates the effectiveness of PoissonMH and the general applicability of Poisson-minibatching method.

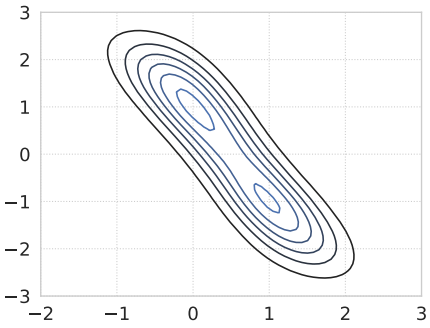


Figure A.1: The estimated density of PoissonMH on a truncated Gaussian mixture model.

A.4 Extended Results about Chebyshev Interpolants

In [137], Theorem 8.2 proves bounds on the error of a Chebyshev interpolant on the interval $[-1, 1]$. However, in order to apply this theorem to a second Chebyshev

interpolant that is a function of the first, we would need to bound the magnitude of that function *on a Bernstein ellipse*. To do this, we need the following extended version of Theorem 8.2, which bounds the error not only on the interval $[-1, 1]$ but more generally on a Bernstein ellipse.

Lemma 1. *Assume $U : \mathbb{C} \to \mathbb{C}$ is analytic in the open Bernstein ellipse $B([-1, 1], \rho)$, where the Bernstein ellipse is a region in the complex plane bounded by an ellipse with foci at ± 1 and semimajor-plus-semiminor axis length $\rho > 1$. If for all $x \in B([-1, 1], \rho)$, $|U(x)| \leq V$ for some constant $V > 0$, then for any constant $1 < \sigma < \rho$, the error of the Chebyshev interpolant on the smaller Bernstein ellipse $B([-1, 1], \sigma)$ is bounded by*

$$|\tilde{U}(x) - U(x)| \leq \frac{4V}{\rho/\sigma - 1} \cdot \left(\frac{\rho}{\sigma}\right)^{-m}.$$

Proof. This proof is essentially identical to that of Theorem 8.2 in [137], except that the error is bounded in a Bernstein ellipse rather than over only the real interval $[-1, 1]$.

First, note that one parameterization of the boundary of the Bernstein ellipse with parameter ρ is

$$\left\{ \frac{z + z^{-1}}{2} \middle| z \in \mathbb{C},\ |z| = \rho \right\},$$

and the open ellipse itself can be written as

$$B([-1, 1], \rho) = \left\{ \frac{z + z^{-1}}{2} \middle| z \in \mathbb{C},\ \rho^{-1} \leq |z| \leq \rho \right\}.$$

Now, Theorem 8.1 from [137] states that the Chebyshev coefficients of a function that satisfies the conditions of this theorem (boundedness and analyticity in a Bernstein ellipse) are bounded by $|a_0| \leq V$ and

$$|a_k| \leq 2V\rho^{-k},\ k \geq 1.$$

That is, for a_k bounded in this way,

$$U(x) = \sum_{k=0}^{\infty} a_k T_k(x)$$

at least for all x in the ρ-Bernstein ellipse on which f is analytic. (While [137] only states explicitly that this holds for $x \in [-1, 1]$, the fact that it also holds on the rest of the Bernstein ellipse follows directly from the fact that both sides of the equation are analytic over that region, using the identity theory for holomorphic functions.) Formula (4.9) from [137] states that

$$U(x) - \tilde{U}_m(x) = \sum_{k=m+1}^{\infty} a_k \left(T_k(x) - T_{l(k,m)}(x)\right)$$

where $\tilde{U}_m$ denotes the degree-m Chebyshev interpolant, and

$$l(k, m) = \left|((k + m - 1) \bmod 2m) - (m - 1)\right|.$$

Notice in particular that it always holds that $l(k, m) \leq m + 1$. Now, for x inside the Bernstein ellipse $B([-1, 1], \sigma)$, there will always exist a $z \in \mathbb{C}$ such that $\sigma^{-1} \leq |z| \leq \sigma$ and

$$x = \frac{z + z^{-1}}{2}.$$

For such an x, and for any k,

$$|T_k(x)| = \left|T_k\left(\frac{z + z^{-1}}{2}\right)\right| = \left|\frac{z^k + z^{-k}}{2}\right| = \frac{|z|^k + |z|^{-k}}{2} \leq \sigma^k,$$

where the second equality is a well-known property of the Chebyshev polynomials.

It follows that, for any x in this Bernstein ellipse,

$$\begin{aligned}
\left|U(x)-\tilde{U}_m(x)\right| &= \left|\sum_{k=m+1}^{\infty} a_k\left(T_k(x)-T_{l(k,m)}(x)\right)\right| \\
&\le \sum_{k=m+1}^{\infty} |a_k|\cdot\left|T_k(x)-T_{l(k,m)}(x)\right| \\
&\le \sum_{k=m+1}^{\infty} 2V\rho^{-k}\cdot\left(\sigma^k+\sigma^{l(k,m)}\right) \\
&\le 4V\sum_{k=m+1}^{\infty}\rho^{-k}\sigma^k \\
&\le 4V\left(\frac{\sigma}{\rho}\right)^{m+1}\sum_{k=0}^{\infty}\left(\frac{\sigma}{\rho}\right)^k \\
&\le 4V\left(\frac{\sigma}{\rho}\right)^{m+1}\frac{1}{1-\frac{\sigma}{\rho}} \\
&\le 4V\left(\frac{\sigma}{\rho}\right)^{m}\frac{1}{\rho/\sigma-1}.
\end{aligned}$$

This is the desired result. □

APPENDIX B

SECTION 3.2 METROPOLIS-HASTINGS WITH POISSON-MINIBATCHING

B.1 Proofs and Derivations

B.1.1 Proof of Theorem 6

In this section, we prove Theorem 6, which asserts that any inexact stateless MH algorithm can produce arbitrarily large bias between its target distribution (the distribution we are trying to sample from) and its stationary distribution (the distribution that the chain actually produces samples from asymptotically).

Proof. Let $\mathcal{A}$ denote the `SubsMH` in Algorithm 5 of the minibatch MH method in question. Since $\mathcal{A}$ is inexact, there must exist a state space Θ, proposal distribution q, and target distribution μ, satisfying Assumption 1 with parameters $c_1, \ldots, c_N, C, M$, where

$$\mu(\theta) \propto \exp\left(-\sum_{i=1}^{N} V_i(\theta)\right)$$

for some N and energy functions $V_1, \ldots, V_N$, such that $\mathcal{A}$ run on μ with proposal distribution q does not have stationary distribution μ.

Next, let $a_\mu(\theta, \theta')$ denote the acceptance probability of algorithm $\mathcal{A}$ on the above task for a proposed transition from θ to θ'. Assume by way of contradiction that on this problem, it is always true that

$$\frac{a_\mu(\theta, \theta')}{a_\mu(\theta', \theta)} = \frac{\mu(\theta')q(\theta|\theta')}{\mu(\theta)q(\theta'|\theta)}.$$

If this were true, then the overall transition probability of this chain, for $\theta \neq \theta'$, would be

$$T_\mu(\theta, \theta') = q(\theta'|\theta) \cdot a_\mu(\theta, \theta')$$

and it would hold that

$$\mu(\theta) T_\mu(\theta, \theta') = \mu(\theta') T_\mu(\theta', \theta).$$

That is, the chain would be reversible, also known as satisfying detailed balance. But it is a standard result that for any reversible chain, μ must be a stationary distribution of that chain. We have now derived a contradiction, which establishes that our assumption is false. That is, there exists a $\theta, \theta' \in \Theta$ such that

$$\frac{a_\mu(\theta, \theta')}{a_\mu(\theta', \theta)} \neq \frac{\mu(\theta') \cdot q(\theta|\theta')}{\mu(\theta) \cdot q(\theta'|\theta)}.$$

Explicitly, this means that if we define the function ΔV such that

$$\Delta V(i) = V_i(\theta) - V_i(\theta'),$$

then for this subsampling problem,

$$\frac{\mathbf{E}\left[\mathcal{A}(\Delta V, N, q(\theta|\theta')/q(\theta'|\theta), c_1, \ldots, c_N, C, M(\theta, \theta'))\right]}{\mathbf{E}\left[\mathcal{A}(-\Delta V, N, q(\theta'|\theta)/q(\theta|\theta'), c_1, \ldots, c_N, C, M(\theta, \theta'))\right]} \neq \frac{\mu(\theta') \cdot q(\theta|\theta')}{\mu(\theta) \cdot q(\theta'|\theta)}. \quad \text{(B.1)}$$

Without loss of generality, assume that

$$q(\theta|\theta')/q(\theta'|\theta) \leq 1.$$

(This is without loss of generality since we can ensure it is the case by swapping θ and θ'.) We fixed θ and θ' to be the pair satisfying Equation B.1 throughout this section.

Constructing an example. We use this to prove the theorem by a constructive example. Let $x_1, \ldots, x_N$ be defined by

$$x_i = \Delta V(i) = V_i(\theta) - V_i(\theta').$$

Define X as the sum

$$X = \sum_{i=1}^{N} x_i.$$

For some parameter $K \in \mathbb{N}$ (to be defined later), consider the state space Ω defined as

$$\Omega = \{(k, z) \mid k \in \{0, \ldots, K-1\},\ 0 \leq z \leq \exp(kX)\},$$

using the natural measure for a finite disjoint union of measure spaces. Define a target distribution over Ω given by the density

$$\pi(k, z) \propto \exp\left(-\sum_{i=1}^{N} k \cdot x_i\right),$$

or equivalently

$$\pi(k, z) \propto \exp\left(-\sum_{i=1}^{N} U_i(k, z)\right) \text{ where } U_i(k, z) = kx_i.$$

Define a proposal distribution $\hat{q}$, such that, starting from (k, z):

- With probability $1/4$, we sample z' uniformly from $[0, \exp(kX)]$ and propose a transition to (k, z').
- With probability $1/4$, we propose a transition to $(k-1, z)$, if it is in Ω.
- With probability $\frac{1}{4} \cdot \frac{q(\theta|\theta')}{q(\theta'|\theta)}$, we propose a transition to $(k+1, z)$, if it is in Ω.
- With the remaining probability, we just propose to stay at (k, z).

This is effectively acting as a random walk over k, and our goal will be to show that while the true target distribution π has a marginal in k that is the uniform distribution, the minibatch MH method causes the chain's transition to be biased to step more in one direction than another, resulting in a highly biased stationary distribution (where we can make the bias arbitrarily large by setting K).

We use the same c_i and C as before, and define a new function $\hat{M}$ such that

$$\hat{M}((k,z),(k+1,z)) = \hat{M}((k,z),(k-1,z)) = M(\theta,\theta')$$

and $\hat{M}(\cdots) = 0$ for other proposed transitions (we can set $\hat{M}$ however we want for pairs of states that are never proposed in a transition, since this will not affect the algorithm). Clearly, this setup satisfies Assumption 1, since the original distribution did.

Now, consider what our minibatch MH method will do when run on this task. There are three cases to consider.

Proposed changes in z. When a proposed change in z is made, the resulting ΔU will be uniformly 0, and the probability of the reverse transition will be equal (1/4 in both directions), so the algorithm will be passed the arguments

$$\mathcal{A}(0, N, 1, c_1, \ldots, c_N, C, 0).$$

Since this does not depend at all on z or k, this means that the acceptance probability of these transitions will be the same regardless of the state. Call this probability α_0.

A proposal to decrease k. When a proposal is made to decrease k, the probability of the forward and reverse transitions will be

$$\hat{q}((k-1,z)|(k,z)) = \frac{1}{4} \text{ and } \hat{q}((k,z)|(k-1,z)) = \frac{1}{4} \cdot \frac{q(\theta|\theta')}{q(\theta'|\theta)}.$$

It follows that

$$\frac{\hat{q}((k,z)|(k-1,z))}{\hat{q}((k-1,z)|(k,z))} = \frac{q(\theta|\theta')}{q(\theta'|\theta)}.$$

The energy function difference for this proposal will be

$$\Delta U(i) = U_i((k,z)) - U_i((k-1,z)) = kx_i - (k-1)x_i = x_i,$$

so in particular $\Delta U = \Delta V$. And, of course for this transition $\hat{M}$ will take on the value $M(\theta, \theta')$. So, the minibatch MH algorithm will be passed the arguments

$$\mathcal{A}(\Delta V, N, q(\theta|\theta')/q(\theta'|\theta), c_1, \ldots, c_N, C, M(\theta, \theta')),$$

and so it will accept with probability

$$\mathbf{E}\left[\mathcal{A}(\Delta V, N, q(\theta|\theta')/q(\theta'|\theta), c_1, \ldots, c_N, C, M(\theta, \theta'))\right].$$

Call this probability α_-.

A proposal to increase k. When a proposal is made to increase k, the probability of the forward and reverse transitions will be

$$\hat{q}((k+1,z)|(k,z)) = \frac{1}{4} \cdot \frac{q(\theta|\theta')}{q(\theta'|\theta)}. \text{ and } \hat{q}((k,z)|(k+1,z)) = \frac{1}{4}.$$

It follows that

$$\frac{\hat{q}((k,z)|(k+1,z))}{\hat{q}((k+1,z)|(k,z))} = \frac{q(\theta'|\theta)}{q(\theta|\theta')}.$$

The energy function difference for this proposal will be

$$\Delta U(i) = U_i((k,z)) - U_i((k+1,z)) = kx_i - (k+1)x_i = -x_i,$$

so in particular $\Delta U = -\Delta V$. And, as before for this transition $\hat{M}$ will take on the value $M(\theta, \theta')$. So, the minibatch MH algorithm will be passed the arguments

$$\mathcal{A}(-\Delta V, N, q(\theta'|\theta)/q(\theta|\theta'), c_1, \ldots, c_N, C, M(\theta, \theta')),$$

and so it will accept with probability

$$\mathbf{E}\left[\mathcal{A}(-\Delta V, N, q(\theta'|\theta)/q(\theta|\theta'), c_1, \ldots, c_N, C, M(\theta, \theta'))\right].$$

Define the probability α_+ as

$$\alpha_+ = \mathbf{E}\left[\mathcal{A}(-\Delta V, N, q(\theta'|\theta)/q(\theta|\theta'), c_1, \ldots, c_N, C, M(\theta, \theta'))\right] \cdot \frac{q(\theta|\theta')}{q(\theta'|\theta)}.$$

The resulting Markov chain. From the above analysis, we can conclude that the Markov chain that results from subsampling algorithm $\mathcal{A}$ applied to this method is as follows. Starting from (k, z), if we let $\hat{T}$ denote the transition operator of this Markov chain,

- With probability $\frac{1}{4} \cdot \alpha_0$, we sample z' uniformly from $[0, \exp(kX)]$ and transition to (k, z').
- With probability $\frac{1}{4} \cdot \alpha_-$, we transition to $(k-1, z)$, if it is in Ω.
- With probability $\frac{1}{4} \cdot \alpha_+$, we transition to $(k+1, z)$, if it is in Ω.
- With the remaining probability, we just stay at (k, z).

Consider the distribution

$$\nu(k, z) \propto \left(\frac{\alpha_+}{\alpha_-}\right)^k.$$

It is easy to see that this Markov chain satisfies detailed balance with ν as its stationary distribution. In particular,

$$\begin{aligned}
\nu(k, z) \cdot T((k-1, z)|(k, z)) &= \left(\frac{\alpha_+}{\alpha_-}\right)^k \cdot \frac{1}{4} \cdot \alpha_- \\
&= \left(\frac{\alpha_+}{\alpha_-}\right)^{k-1} \cdot \frac{1}{4} \cdot \alpha_+ \\
&= \nu(k-1, z) \cdot T((k, z)|(k-1, z)).
\end{aligned}$$

So ν will be a stationary distribution of the minibatch MH chain $\hat{T}$.

Observe that the marginal distribution of k in π is

$$\pi(k) = \int_0^{\exp(kX)} \pi(k, z)\, dz \propto \exp\left(-\sum_{i=1}^{N} k \cdot x_i\right) \cdot \exp(kX) = 1,$$

so the marginal distribution of k in the target distribution is actually the uniform distribution. On the other hand, using the same derivation, the marginal distribution

of k in ν is

$$\nu(k) \propto \left(\frac{\alpha_+}{\alpha_-}\right)^k \cdot \exp(kX) = \left(\frac{\alpha_+}{\alpha_-} \cdot \exp(X)\right)^k.$$

We know immediately by substituting our definitions of α_+ and α_- into (B.1) that

$$\frac{\alpha_-}{\alpha_+} \neq \frac{\mu(\theta')}{\mu(\theta)} = \exp\left(\sum_{i=1}^{N}(V_i(\theta) - V_i(\theta')\right) = \exp\left(\sum_{i=1}^{N} x_i\right) = \exp(X).$$

As a consequence, we know that

$$\frac{\alpha_+}{\alpha_-} \cdot \exp(X) \neq 1.$$

Call this constant

$$A = \frac{\alpha_+}{\alpha_-} \cdot \exp(X),$$

and observe that $A \neq 1$ and that A is independent of our choice of K (which still remains unset). This gives

$$\nu(k) \propto A^k.$$

Explicitly, this distribution will be

$$\nu(k) = \frac{1}{\sum_{k=0}^{K-1} A^k} \cdot A^k = \frac{1 - A}{1 - A^K} \cdot A^k.$$

Since the total variation distance between two probability measures is lower bounded by the TV-distance between their marginal distributions in any one variable, and similarly the KL divergence is *also* lower bounded by the KL divergence between its marginal distributions in any one variable (both these facts follow directly from the monotonicity property of the f-divergence, of which the KL-divergence and TV-distance are both instances), to prove this theorem it suffices to show both TV-distance and KL-divergence bounds on the marginal distributions in k. We do this now.

Bounding the total variation distance. Now, we compute the total variation distance between π and ν. For this bit of the proof, we will just consider the marginal distribution in k, as this provides a lower bound on the TV distance between the joint distribution. For simplicity, for the rest of the proof, we let $\tilde{\pi}$ denote this marginal distribution of k in ν, and also let π denote the marginal distribution of k in π. By the definition of total variation distance,

$$\begin{aligned}\mathrm{TV}(\pi, \tilde{\pi}) &= \frac{1}{2} \sum_{k=0}^{K-1} |\tilde{\pi}(k) - \pi(k)| \\ &= \frac{1}{2} \sum_{k=0}^{K-1} \left| \frac{1-A}{1-A^K} \cdot A^k - \frac{1}{K} \right|.\end{aligned}$$

If $A < 1$,

$$\begin{aligned}\mathrm{TV}(\pi, \tilde{\pi}) &= \sum_{k=0}^{K_0} \left(\frac{1-A}{1-A^K} \cdot A^k - \frac{1}{K} \right) \\ &= \frac{1-A^{K_0}}{1-A^K} - \frac{K_0}{K} \end{aligned} \tag{B.2}$$

where K_0 is the largest k such that

$$\frac{1-A}{1-A^K} \cdot A^k > \frac{1}{K}.$$

By solving the above equation, we have

$$K_0 = \left\lfloor \frac{\log(1-A^K) - \log(1-A) - \log(K)}{\log(A)} \right\rfloor.$$

We can lower bound K_0 by

$$\begin{aligned}K_0 &\geq \frac{\log(1-A^K) - \log(1-A) - \log(K)}{\log(A)} - 1 \\ &\geq \frac{-\log(1-A) - \log(K)}{\log(A)} - 1.\end{aligned}$$

It follows that the first term in (B.2) becomes

$$\frac{1-A^{K_0}}{1-A^K} \geq \frac{1 - \frac{1}{KA(1-A)}}{1-A^K} \geq 1 - \frac{1}{KA(1-A)}.$$

We can also upper bound K_0 and then the second term can be bounded as the following

$$\frac{K_0}{K} \leq \frac{\log(1-A^K) - \log(K)}{K\log(A)}.$$

When $K \geq \frac{\log\left(1-\exp\left(-\frac{1}{2}\right)\right)}{\log(A)}$, we have $\log(1-A^K) \geq -\frac{1}{2}$. Since $\log(K) \leq K^{\frac{1}{2}}$ and $K^{-1} \leq K^{-\frac{1}{2}}$, we have

$$\frac{K_0}{K} \leq \frac{-\frac{1}{2}K^{-1} - K^{-\frac{1}{2}}}{\log(A)} \leq -\left(\frac{3}{2\log(A)}\right)K^{-\frac{1}{2}}.$$

Therefore, the TV distance is bounded by

$$\begin{aligned}\mathrm{TV}(\pi,\tilde{\pi}) &\geq 1 - \frac{1}{KA(1-A)} + \left(\frac{3}{2\log(A)}\right)K^{-\frac{1}{2}} \\ &\geq 1 + \left(\frac{3}{2\log(A)} - \frac{1}{A(1-A)}\right)K^{-\frac{1}{2}}.\end{aligned}$$

To make $\mathrm{TV}(\pi,\tilde{\pi}) \geq \delta$, we just need to set

$$K \geq \frac{\left(\frac{3}{2\log(A)} - \frac{1}{A(1-A)}\right)^2}{(1-\delta)^2}.$$

Similarly, if $A > 1$,

$$\begin{aligned}\mathrm{TV}(\pi,\tilde{\pi}) &= \sum_{k=K_0}^{K-1}\left(\frac{1-A}{1-A^K}\cdot A^k - \frac{1}{K}\right) \\ &= \frac{A^K - A^{K_0}}{A^K - 1} - \frac{K-K_0}{K} \\ &= \frac{K_0}{K} - \frac{A^{K_0}-1}{A^K-1}\end{aligned}$$

where

$$K_0 = \left\lceil \frac{\log(A^K-1) - \log(A-1) - \log(K)}{\log(A)} \right\rceil$$

which is the smallest k such that

$$\frac{1-A}{1-A^K}\cdot A^k > \frac{1}{K}.$$

We can get an upper bound of K_0 by

$$\begin{aligned} K_0 &\leq \frac{\log(A^K - 1) - \log(A-1) - \log(K)}{\log(A)} + 1 \\ &= \log_A \left(\frac{A^K - 1}{K(A-1)} \right) + 1. \end{aligned}$$

Therefore,

$$\begin{aligned} \frac{A^{K_0} - 1}{A^K - 1} &\leq \frac{A \cdot \left(\frac{A^K - 1}{K(A-1)} \right) - 1}{A^K - 1} \\ &= \frac{A}{K(A-1)} - \frac{1}{A^K - 1}. \end{aligned}$$

We can lower bound K_0 by

$$K_0 \geq \log_A \left(A^K - 1 \right) - \log_A(A-1) - \log_A(K).$$

When $K \geq 1 - \log_A(A-1)$, $A^K - 1 \geq A^{K-1}$. Then we have

$$\begin{aligned} K_0 &\geq \log_A \left(A^{K-1} \right) - \log_A(A-1) - \log_A(K) \\ &= K - 1 - \log_A(A-1) - \log_A(K). \end{aligned}$$

It follows that

$$\frac{K_0}{K} \geq 1 - \frac{1}{K} - \frac{\log_A(A-1)}{K} - \frac{\log_A(K)}{K}.$$

Since $\log(K) \leq K^{\frac{1}{2}}$ and $K^{-1} \leq K^{-\frac{1}{2}}$, the TV distance can be bounded by

$$\begin{aligned} \mathrm{TV}(\pi, \tilde{\pi}) &\geq 1 - \frac{1}{K} - \frac{\log_A(A-1)}{K} - \frac{\log_A(K)}{K} - \frac{A}{K(A-1)} + \frac{1}{A^K - 1} \\ &\geq 1 - \left(1 + \log_A(A-1) + \frac{1}{\log(A)} + \frac{A}{A-1} \right) K^{-\frac{1}{2}}. \end{aligned}$$

To make $\mathrm{TV}(\pi, \tilde{\pi}) \geq \delta$, we just need

$$K \geq \left(\frac{1 + \log_A(A-1) + \frac{1}{\log(A)} + \frac{A}{A-1}}{1 - \delta} \right)^2.$$

Since we could set K arbitrarily, it is clear that we can do this.

Bounding the KL divergence. We can compute KL divergence between π and $\tilde{\pi}$ as follows

$$\begin{aligned}
\mathrm{KL}(\pi, \tilde{\pi}) &= \sum_{k=0}^{K-1} \frac{1}{K} \cdot \log\left(\frac{1}{K} \cdot \frac{1-A^K}{(1-A)A^k}\right) \\
&= \frac{1}{K} \cdot \sum_{k=0}^{K-1} \left[\log\left(\frac{1}{K} \cdot \frac{1-A^K}{(1-A)}\right) - k\log(A)\right] \\
&= \log\left(\frac{1-A^K}{K(1-A)}\right) - \frac{\log(A)}{K}\sum_{k=0}^{K-1} k \\
&= \log\left(\frac{1-A^K}{K(1-A)}\right) - \frac{(K-1)\log(A)}{2}
\end{aligned}$$

If $A < 1$, we have

$$\begin{aligned}
\mathrm{KL}(\pi, \tilde{\pi}) &= \log\left(1-A^K\right) - \log((1-A)K) - \frac{K\log(A)}{2} + \frac{\log(A)}{2} \\
&\geq \log\left(1-A^K\right) - \left(\frac{1-A+\log(A)}{2}\right)K + \frac{\log(A)}{2}.
\end{aligned}$$

The last equation is because $\log(x) \leq \frac{x}{2}$.

To further simplify the above equation, we first note that $1 - A + \log(A) < 0$ when $A \neq 1$. And then when $K \geq \log_A\left(1 - A^{\frac{1}{2}}\right)$, we have $1 - A^K \geq A^{\frac{1}{2}}$. It follows that we can simplify it to be

$$\mathrm{KL}(\pi, \tilde{\pi}) \geq \log(A) - \left(\frac{1-A+\log(A)}{2}\right)K.$$

To make $\mathrm{KL}(\pi, \tilde{\pi}) \geq \rho$, it is clear that we just need to set

$$K \geq \frac{2(\rho - \log(A))}{A - 1 - \log(A)}.$$

Consider when $A > 1$,

$$\mathrm{KL}(\pi, \tilde{\pi}) = \log\left(\frac{A^K - 1}{K(A-1)}\right) - \frac{(K-1)\log(A)}{2}.$$

If $K \geq \frac{\log(2)}{\log(A)}$, we have that $A^K - 1 \geq \frac{A^K}{2}$. It follows that

$$\begin{aligned}\mathrm{KL}(\pi, \tilde{\pi}) &\geq K\log(A) - \log(K) - \log(2A-2) - \frac{K\log(A)}{2} \\ &= \frac{K\log(A)}{2} - \log(K) - \log(2A-2).\end{aligned}$$

To make $\mathrm{KL}(\pi, \tilde{\pi}) \geq \rho$, we need

$$\frac{K\log(A)}{2} - \log(K) \geq \rho + \log(2A-2).$$

Let $K = \exp(y)$. By Taylor series, we know $\exp(y) \geq \frac{y^2}{2}$. Then it follows that

$$\frac{y^2\log(A)}{4} - y \geq \rho + \log(2A-2).$$

Solve the above inequality, we can get

$$y \geq \frac{1 + 2 \cdot \frac{\log(A)}{4} \cdot \Big(\rho + \log(2A-2)\Big)}{2 \cdot \frac{\log(A)}{4}} = \frac{2 + \log(A)\Big(\rho + \log(2A-2)\Big)}{\log(A)}.$$

It follows that it suffices to set

$$K \geq \exp\left(\frac{2 + \log(A)\Big(\rho + \log(2A-2)\Big)}{\log(A)}\right).$$

Concluding the proof. The theorem now follows from choosing a K large enough that both the TV distance inequality we derived and the KL divergence inequality we derived are satisfied. □

Connection between Theorem 6 and TV Bound of Inexact MH Methods

Some inexact methods such as MHSubLhd [11] have bounded TV distance between the target distribution and the approximate distribution (see Proposition 3.2 in [11]). We would like to emphasize that Theorem 6 is compatible with these results.

Specifically, Proposition 3.2 assumes P_{MH} has a bounded mixing time. It is well known that this produces a TV bound for any kernel by coupling [83]. Our theorem does not have this assumption; it suggests that for MHSubLhd, with a given user-specified error, there exists a target distribution and proposal satisfying Theorem 6, on which P_{MH} either does not have bounded mixing time or the mixing time is large enough such that the TV bound is greater than δ.

B.1.2 Proof of Statement 2

Proof. We prove this by construction. Consider a dataset $\{x_i\}_{i=1}^N$. The data instances can take two values $\{-\frac{M}{N}, \frac{M}{N}\}$ where M is a positive constant. Assume that half of the data instances take value $\frac{M}{N}$ and the remaining take $-\frac{M}{N}$. Let the target distribution be $\pi(\theta) = \frac{1}{Z}\exp\left(\theta \cdot \sum_{i=1}^N x_i\right)$ and the domain for θ be $\{0, 1, \ldots, K-1\}$. We define the proposal distribution to be the following

$$p(\theta,\theta) = \frac{1}{2}, \quad \text{for all } \theta; \quad p(\theta,\theta-1) = \frac{1}{4}, \quad p(\theta,\theta+1) = \frac{1}{4} \quad \text{for } \theta \in \{1,\ldots,K-2\};$$

and $p(0,1) = p(K-1,K-2) = \frac{1}{2}$.

Recall that FMH factorizes the target distribution $\pi(\theta)$ and the proposal distribution $p(\theta)$ as follows

$$\pi(\theta) \propto \prod_{i=1}^m \pi_i(\theta), \qquad p(\theta) \propto \prod_{i=1}^m p_i(\theta)$$

where $m \geq 1$ and π_i and p_i are some non-negative functions. Then the acceptance rate is given by

$$a_{\text{FMH}}(\theta,\theta') = \prod_{i=1}^m \min\left(1, \frac{\pi(\theta')p_i(\theta',\theta)}{\pi(\theta)p_i(\theta,\theta')}\right).$$

A common choice is to set $m = N$. On this example, we can write the acceptance

rate of transitioning from θ to $\theta' = \theta + 1$ in FMH as follows

$$a_{\text{FMH}}(\theta, \theta') = \prod_{i=1}^{N} \min\left(1, \exp(x_i)\right) = \left(\exp\left(-\frac{M}{N}\right)\right)^{\frac{N}{2}} = \exp\left(-\frac{M}{2}\right).$$

It is easy to show that the acceptance rate of transitioning from θ to $\theta' = \theta - 1$ in FMH is the same.

When $M > -2\log(p)$, it is clear that the acceptance rate of FMH is less than p. By contrast, the acceptance rate of standard MH is

$$a_{\text{MH}}(\theta, \theta') = \min\left(1, \exp\left(\pm \sum_{i=1}^{N} x_i\right)\right) = 1.$$

In order to preserve geometric ergodicity, [32] introduces *truncated FMH* (TFMH) which forces FMH degrade to standard MH when the energy exceeds a threshold R. If we set hyperparameter $R > M/2$, then in each step, the value of a_{TFMH} will be the same as a_{FMH}. Therefore, if setting $M > -2\log(p)$, we have

$$\frac{a_{\text{TFMH}}}{a_{\text{MH}}} \leq \frac{p}{1} = p.$$

If we set $R \leq M/2$, TFMH falls back to standard, full-batch MH — using the whole dataset at each step. This proves the statement. □

B.1.3 Proof of Theorem 7

In this section, we prove Theorem 7, which asserts that TunaMH is reversible and has stationary distribution π, and gives bounds on its spectral gap relative to the spectral gap of the original Metropolis-Hastings algorithm.

Proof. For convenience, we prove Theorem 7 using Algorithm 14 statement which is statistically equivalent to Algorithm 6. The transition operator can be written

as the following

$$
\begin{aligned}
&T(\theta,\theta')\\
&=\mathbf{E}\left\{q(\theta'|\theta)\min\left(1,\frac{q(\theta|\theta')\exp\left(\sum_i\left[s_i\log\left(\frac{\lambda c_i}{C}+\phi_i(\theta')\right)-\log s_i!\right]\right)}{q(\theta'|\theta)\exp\left(\sum_i\left[s_i\log\left(\frac{\lambda c_i}{C}+\phi_i(\theta)\right)-\log s_i!\right]\right)}\right)\right\}\\
&=\mathbf{E}\left\{q(\theta'|\theta)\min\left(1,\frac{q(\theta|\theta')\exp\left(\sum_i\left[s_i\log\left(\frac{\lambda c_i}{C}+\phi_i(\theta')\right)\right]\right)}{q(\theta'|\theta)\exp\left(\sum_i\left[s_i\log\left(\frac{\lambda c_i}{C}+\phi_i(\theta)\right)\right]\right)}\right)\right\}\\
&=\sum_s\left\{q(\theta'|\theta)\min\left(1,\frac{q(\theta|\theta')\exp\left(\sum_i\left[s_i\log\left(\frac{\lambda c_i}{C}+\phi_i(\theta')\right)\right]\right)}{q(\theta'|\theta)\exp\left(\sum_i\left[s_i\log\left(\frac{\lambda c_i}{C}+\phi_i(\theta)\right)\right]\right)}\right)\right\}\prod_i p(s_i|\theta,\theta')\\
&=\sum_s\left\{q(\theta'|\theta)\min\left(\exp\left(\sum_i\left[s_i\log\left(\frac{\lambda c_i}{C}+\phi_i(\theta)\right)-\phi_i(\theta)-\frac{\lambda c_i}{C}-\log s_i!\right]\right),\right.\right.\\
&\qquad\left.\left.\frac{q(\theta|\theta')\exp\left(\sum_i\left[s_i\log\left(\frac{\lambda c_i}{C}+\phi_i(\theta')\right)\right]\right)}{q(\theta'|\theta)\exp\left(\sum_i\phi_i(\theta)+\frac{\lambda c_i}{C}+\log s_i!\right)}\right)\right\}\\
&=\sum_s\left\{q(\theta'|\theta)\min\left(\exp\left(\sum_i\left[s_i\log\left(\frac{\lambda c_i}{C}+\phi_i(\theta)\right)-\phi_i(\theta)-\frac{\lambda c_i}{C}-\log s_i!\right]\right),\right.\right.\\
&\qquad\left.\left.\frac{q(\theta|\theta')}{q(\theta'|\theta)}\exp\left(\sum_i\left[s_i\log\left(\frac{\lambda c_i}{C}+\phi_i(\theta')\right)-\phi_i(\theta)-\frac{\lambda c_i}{C}-\log s_i!\right]\right)\right)\right\}
\end{aligned}
$$

Multiplying $\pi(\theta)$ to both sides produces

$$
\begin{aligned}
&\pi(\theta)T(\theta,\theta')\\
&\quad=\frac{1}{Z}\exp\left(-\sum_i U_i(\theta)\right)T(\theta,\theta')\\
&\quad=\frac{1}{Z}\sum_s\min\left(q(\theta'|\theta)\exp\left(\sum_i\left[s_i\log\left(\frac{\lambda c_i}{C}+\phi_i(\theta)\right)\right.\right.\right.\\
&\qquad\left.\left.\left.-\frac{U_i(\theta)+U_i(\theta')}{2}-\frac{c_i}{2}M(\theta,\theta')-\frac{\lambda c_i}{C}-\log s_i!\right]\right),\\
&\qquad q(\theta|\theta')\exp\left(\sum_i\left[s_i\log\left(\frac{\lambda c_i}{C}+\phi_i(\theta')\right)\right.\right.\\
&\qquad\left.\left.\left.-\frac{U_i(\theta)+U_i(\theta')}{2}-\frac{c_i}{2}M(\theta,\theta')-\frac{\lambda c_i}{C}-\log s_i!\right]\right)\right)\Bigg).
\end{aligned}
$$

It is clear that the expression is symmetric in θ and θ'. Therefore the chain is reversible and its stationary distribution is $\pi(\theta)$. This proves the first part of the

theorem.

To prove the second part of the theorem, the bound on the spectral gap, we continue to reduce the transition probability in the previous proof to

$$
\begin{aligned}
&\pi(\theta)T(\theta,\theta') \\
&\quad= \frac{1}{Z}\sum_s \min\Bigg(q(\theta'|\theta)\exp\Bigg(\sum_i\Bigg[s_i\log\left(\frac{\lambda c_i}{C}+\phi_i(\theta)\right) \\
&\qquad -\frac{U_i(\theta)+U_i(\theta')}{2}-\frac{c_i}{2}M(\theta,\theta')-s_i\log\frac{\lambda c_i}{C}\Bigg]\Bigg), \\
&\qquad q(\theta|\theta')\exp\Bigg(\sum_i\Bigg[s_i\log\left(\frac{\lambda c_i}{C}+\phi_i(\theta')\right) \\
&\qquad -\frac{U_i(\theta)+U_i(\theta')}{2}-\frac{c_i}{2}M(\theta,\theta')-s_i\log\frac{\lambda c_i}{C}\Bigg]\Bigg)\Bigg) \\
&\qquad \cdot\prod_i\frac{1}{s_i!}\exp\left(-\frac{\lambda c_i}{C}\right)\left(\frac{\lambda c_i}{C}\right)^{s_i} \\
&\quad= \frac{1}{Z}\sum_s \min\Bigg(q(\theta'|\theta)\exp\Bigg(\sum_i\Bigg[s_i\log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta)\right) \\
&\qquad -\frac{U_i(\theta)+U_i(\theta')}{2}-\frac{c_i}{2}M(\theta,\theta')\Bigg]\Bigg), \\
&\qquad q(\theta|\theta')\exp\left(\sum_i\left[s_i\log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta')\right)-\frac{U_i(\theta)+U_i(\theta')}{2}-\frac{c_i}{2}M(\theta,\theta')\right]\right)\Bigg) \\
&\qquad \cdot\prod_i\frac{1}{s_i!}\exp\left(-\frac{\lambda c_i}{C}\right)\left(\frac{\lambda c_i}{C}\right)^{s_i}.
\end{aligned}
$$

Note that s_i here are non-negative integers that a Poisson variable can take, not variables. So if we let $r_i \sim \text{Poisson}\left(\frac{\lambda c_i}{C}\right)$ and r_i to be all independent, we can

write this as

$$\pi(\theta)T(\theta,\theta') = \frac{1}{Z}\mathbf{E}\min\left(q(\theta'|\theta)\exp\left(\sum_i r_i \log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta)\right)\right),\right.$$
$$\left. q(\theta|\theta')\exp\left(\sum_i r_i \log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta')\right)\right)\right)$$
$$\cdot \exp\left[-\frac{1}{2}\left(\sum_i U_i(\theta)+\sum_i U_i(\theta')+CM(\theta,\theta')\right)\right].$$

Assume $G(\theta,\theta')$ is the transition operator of standard MH. Consider the ratio

$$\frac{\pi(\theta)T(\theta,\theta')}{\pi(\theta)G(\theta,\theta')}$$
$$= \frac{1}{Z}\mathbf{E}\min\left(q(\theta'|\theta)\exp\left(\sum_i r_i \log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta)\right)\right),\right.$$
$$\left. q(\theta|\theta')\exp\left(\sum_i r_i \log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta')\right)\right)\right)$$
$$\cdot \exp\left[-\frac{1}{2}\left(\sum_i U_i(\theta)+\sum_i U_i(\theta')+CM(\theta,\theta')\right)\right]$$
$$\cdot \left[1\Big/\left(\frac{1}{Z}\min\left(q(\theta'|\theta)\exp\left(-\sum_i U_i(\theta)\right), q(\theta|\theta')\exp\left(-\sum_i U_i(\theta')\right)\right)\right)\right].$$

We know that $\frac{\min(A,B)}{\min(C,D)} = \min\left(\frac{A}{\min(C,D)}, \frac{B}{\min(C,D)}\right) \geq \min\left(\frac{A}{C}, \frac{B}{D}\right)$. The last inequality is due to the fact that $\frac{1}{\min(C,D)} \geq \frac{1}{C}$ and $\frac{1}{\min(C,D)} \geq \frac{1}{D}$.

With this inequality, we can continue simplifying the ratio,

$$
\begin{aligned}
&\frac{\pi(\theta)T(\theta,\theta')}{\pi(\theta)G(\theta,\theta')} \\
&\quad \geq \mathbf{E}\left[\min\left(\frac{\exp\left(\sum_i r_i \log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta)\right)\right)}{\exp\left(-\sum_i U_i(\theta)\right)}, \frac{\exp\left(\sum_i r_i \log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta')\right)\right)}{\exp\left(-\sum_i U_i(\theta')\right)}\right)\right] \\
&\qquad \cdot \exp\left[-\frac{1}{2}\left(\sum_i U_i(\theta)+\sum_i U_i(\theta')+CM(\theta,\theta')\right)\right] \\
&\quad = \mathbf{E}\left[\min\left(\exp\left(\sum_i\left(r_i\log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta)\right)-\phi_i(\theta)\right)\right),\right.\right. \\
&\qquad \left.\left.\exp\left(\sum_i\left(r_i\log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta')\right)-\phi_i(\theta')\right)\right)\right)\right] \\
&\quad = \mathbf{E}\left[\max\left(\exp\left(\sum_i\left(\phi_i(\theta)-r_i\log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta)\right)\right)\right),\right.\right. \\
&\qquad \left.\left.\exp\left(\sum_i\left(\phi_i(\theta')-r_i\log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta')\right)\right)\right)\right)^{-1}\right].
\end{aligned}
$$

Because $f(x)=\frac{1}{x}$ is a convex function, by Jensen's inequality it follows

$$
\begin{aligned}
\frac{\pi(\theta)T(\theta,\theta')}{\pi(\theta)G(\theta,\theta')} \geq \mathbf{E}\Bigg[&\max\left(\exp\left(\sum_i\left(\phi_i(\theta)-r_i\log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta)\right)\right)\right),\right. \\
&\left.\exp\left(\sum_i\left(\phi_i(\theta')-r_i\log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta')\right)\right)\right)\right]^{-1}.
\end{aligned}
$$

We use $\max(A,B)\leq(A^p+B^p)^{\frac{1}{p}}$ to remove the max function.

$$
\begin{aligned}
\frac{\pi(\theta)T(\theta,\theta')}{\pi(\theta)G(\theta,\theta')} \geq \mathbf{E}\Bigg[&\left(\exp\left(p\sum_i\left(\phi_i(\theta)-r_i\log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta)\right)\right)\right)+\right. \\
&\left.\exp\left(p\sum_i\left(\phi_i(\theta')-r_i\log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta')\right)\right)\right)^{\frac{1}{p}}\right]^{-1}.
\end{aligned}
$$

Since $x^{\frac{1}{p}}$ is concave, by Jensen's inequality

$$\begin{aligned}\frac{\pi(\theta)T(\theta,\theta')}{\pi(\theta)G(\theta,\theta')} \geq \mathbf{E}\Bigg[\exp\left(p\sum_i\left(\phi_i(\theta) - r_i\log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta)\right)\right)\right)+ \\ \exp\left(p\sum_i\left(\phi_i(\theta') - r_i\log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta')\right)\right)\right)\Bigg]^{-\frac{1}{p}} \\ = \Bigg[\prod_i \mathbf{E}\exp\left(p\phi_i(\theta) - pr_i\log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta)\right)\right)+ \\ \prod_i \mathbf{E}\exp\left(p\phi_i(\theta') - pr_i\log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta')\right)\right)\Bigg]^{-\frac{1}{p}}.\end{aligned}$$

$\mathbf{E}\left[\exp\left(-pr_i\log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta)\right)\right)\right]$ is the moment generating function of the Poisson random variable r_i evaluated at

$$t = -p\log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta)\right).$$

We know that

$$\mathbf{E}\exp(r_i t) = \exp\left(\frac{\lambda c_i}{C}\left(\exp(t)-1\right)\right),$$

therefore,

$$\mathbf{E}\left[\exp\left(-pr_i\log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta)\right)\right)\right] = \exp\left(\frac{\lambda c_i}{C}\left(1+\frac{C}{\lambda c_i}\phi_i(\theta)\right)^{-p} - \frac{\lambda c_i}{C}\right).$$

Substituting this into the original expression produces

$$\begin{aligned}\frac{\pi(\theta)T(\theta,\theta')}{\pi(\theta)G(\theta,\theta')} \geq \Bigg[\prod_i \exp\left(\frac{\lambda c_i}{C}\left(1+\frac{C}{\lambda c_i}\phi_i(\theta)\right)^{-p} - \frac{\lambda c_i}{C} + p\phi_i(\theta)\right) \\ + \prod_i \exp\left(\frac{\lambda c_i}{C}\left(1+\frac{C}{\lambda c_i}\phi_i(\theta')\right)^{-p} - \frac{\lambda c_i}{C} + p\phi_i(\theta')\right)\Bigg]^{-\frac{1}{p}}.\end{aligned}$$

Considering the term inside exp. Define a function $f(y) = \frac{\lambda c_i}{C}\left(1+\frac{C}{\lambda c_i}y\right)^{-p} - \frac{\lambda c_i}{C} + py$ for $y \geq 0$. It is clear that $f(0) = 0$. The first derivative is

$$f'(y) = p + (-p)\left(1+\frac{C}{\lambda c_i}y\right)^{-p-1}$$

which is also 0 at $y = 0$. The second and third derivatives are

$$f''(y) = (-p)(-p-1)\frac{C}{\lambda c_i}\left(1 + \frac{C}{\lambda c_i}y\right)^{-p-2}, \tag{B.3}$$

$$f'''(y) = (-p)(-p-1)(-p-2)\left(\frac{C}{\lambda c_i}\right)^2\left(1 + \frac{C}{\lambda c_i}y\right)^{-p-3}. \tag{B.4}$$

By Taylor series, we have

$$f(y) = f(0) + f'(0)y + \frac{f''(0)}{2!}y^2 + \frac{f'''(v)}{3!}y^3$$

where v is between 0 and y. By (B.4), we know that $f'''(v) \leq 0$, therefore since $y \geq 0$, we have

$$\begin{aligned} f(y) &\leq f(0) + f'(0)y + \frac{f''(0)}{2!}y^2 \\ &= \frac{f''(0)}{2!}y^2. \end{aligned}$$

Substituting $y = \phi_i(\theta)$ produces

$$\begin{aligned} f(\phi_i(\theta)) &\leq (-p)(-p-1)\frac{C}{\lambda c_i}\phi_i^2(\theta) \\ &\leq (-p)(-p-1)\frac{C}{\lambda c_i}c_i^2 M^2(\theta, \theta'). \end{aligned}$$

Similarly, we can get

$$f(\phi_i(\theta')) \leq p(p+1)\frac{C}{\lambda c_i}c_i^2 M^2(\theta, \theta').$$

Substituting these to the spectral ratio, we get

$$\begin{aligned} \frac{\pi(\theta)T(\theta, \theta')}{\pi(\theta)G(\theta, \theta')} &\geq \left[2\prod_i \exp\left(p(p+1)\frac{C}{\lambda c_i}c_i^2 M^2(\theta, \theta')\right)\right]^{-\frac{1}{p}} \\ &= \left[2\exp\left(\sum_i p(p+1)\frac{C}{\lambda}c_i M^2(\theta, \theta')\right)\right]^{-\frac{1}{p}} \\ &= \left[2\exp\left(p(p+1)\frac{C^2}{\lambda}M^2(\theta, \theta')\right)\right]^{-\frac{1}{p}} \\ &= 2^{-\frac{1}{p}}\exp\left(-(p+1)\frac{C^2}{\lambda}M^2(\theta, \theta')\right). \end{aligned}$$

Now, we maximize the R.H.S. with respect to p. Let $E = \frac{C^2}{\lambda} M^2(\theta, \theta')$, then it becomes

$$2^{-\frac{1}{p}} \exp\left(-(p+1)E\right) = \exp\left(-E - pE - \frac{1}{p}\log 2\right).$$

The maximum is attained at $p = \sqrt{\frac{\log 2}{E}}$ and the value is

$$\exp\left(-E - 2\sqrt{E \log 2}\right).$$

It follows that

$$\frac{\pi(\theta)T(\theta,\theta')}{\pi(\theta)G(\theta,\theta')} \geq \exp\left(-\frac{C^2}{\lambda}M^2(\theta,\theta') - 2\sqrt{\frac{C^2}{\lambda}M^2(\theta,\theta')\log 2}\right).$$

We set $\lambda = \chi C^2 M^2(\theta, \theta')$, it becomes

$$\frac{\pi(\theta)T(\theta,\theta')}{\pi(\theta)G(\theta,\theta')} \geq \exp\left(-\frac{1}{\chi} - 2\sqrt{\frac{\log 2}{\chi}}\right).$$

We complete the theorem by a Dirichlet form argument. We can write the Dirichlet form $\mathcal{E}(f)$ of a Markov chain with transition operator G as [50]:

$$\mathcal{E}(f) = \frac{1}{2}\int\int \left[\left(f(\theta) - f(\theta')\right)^2\right] G(\theta,\theta')\pi(\theta)d\theta d\theta'.$$

If we let $L_0^2(\pi)$ to be the Hilbert space of functions f such that f has mean zero and is square integrable with respect to probability measure π. It follows that the spectral gap γ of a Markov chain is [3]

$$\gamma = \inf_{f \in L_0^2(\pi): Var_\pi[f]=1} \mathcal{E}(f).$$

From this, it is easy to get that

$$\begin{aligned}
\bar{\gamma} &= \inf_{f \in L_0^2(\pi): Var_\pi[f]=1} \left[\frac{1}{2}\int\int \left[\left(f(\theta) - f(\theta')\right)^2\right] T(\theta,\theta')\pi(\theta)d\theta d\theta'\right] \\
&\geq \exp\left(-\frac{1}{\chi} - 2\sqrt{\frac{\log 2}{\chi}}\right) \\
&\quad \cdot \inf_{f \in L_0^2(\pi): Var_\pi[f]=1} \left[\frac{1}{2}\int\int \left[\left(f(\theta) - f(\theta')\right)^2\right] G(\theta,\theta')\pi(\theta)d\theta d\theta'\right] \\
&= \exp\left(-\frac{1}{\chi} - 2\sqrt{\frac{\log 2}{\chi}}\right) \cdot \gamma.
\end{aligned}$$

□

B.1.4 Proof of Theorem 8

First, we will show the following lemma, which gives half of what we want to have in the theorem.

Lemma 2. *Considering the same setting as the theorem, the average batch size B of any exact, stateless minibatch MH algorithm at any iteration follows*

$$\mathbf{E}[B] \geq 2^{-18} \cdot \kappa C^2 M^2(\theta, \theta') - 2^{-4} \cdot \kappa.$$

Proof. We prove the lemma by construction. First, observe that since the state space Θ has at least two states, we can restrict our attention to just two of those states, by choosing a π that has zero mass on any other state in the space and a q that never proposes transitioning out to any of those other states (at which π has zero mass). Such a proposal will still be ergodic, so it still satisfies our general assumption that we consider only ergodic chains in this section. Without loss of generality, suppose that those two states are $\{-\frac{M}{2}, \frac{M}{2}\}$ (this is without loss of generality because we can always just rename the states), and let C denote the constant in the theorem statement and define (with a bit of abuse of notation) the constant $M := M(-\frac{M}{2}, \frac{M}{2})$. By doing this, we can (again without loss of generality) restrict our attention to the case where $\Theta = \{-\frac{M}{2}, \frac{M}{2}\}$.

Next, we construct our counterexample. Let the dataset be $\{x_i\}_{i=1}^N$ where $x_i \in \{-1, 1\}$. We let the domain for parameter θ to be $\{-\frac{M}{2}, \frac{M}{2}\}$, and the target distribution to be

$$\pi(\theta) = \frac{1}{Z} \exp\left(-\sum_{i=1}^{N} U_i(\theta)\right) = \frac{1}{Z} \exp\left(-\frac{C\theta}{N} \sum_{i=1}^{N} x_i\right)$$

where $U_i(\theta) = \frac{C}{N} \cdot \theta x_i$. Note that by letting N become large, any minibatch MH algorithm that queries the energy difference oracle some number of times will observe a distribution of energy differences that is arbitrarily close to a sequence of independent identically distributed random variables supported on $\{\pm\frac{CM}{N}\}$.

We define $c_i = \frac{C}{N}$, and the proposal distribution to be

$$p(\theta, \theta) = \frac{1}{2}, \qquad p(\theta, -\theta) = \frac{1}{2} \qquad \text{for } \theta \in \left\{ -\frac{M}{2}, \frac{M}{2} \right\}.$$

Now, let $0 < q < 1$ be some constant, and consider two cases: (1) $\frac{1}{N}\sum_i x_i = q$ and (2) $\frac{1}{N}\sum_i x_i = -q < 0$. Suppose that in both cases the x_i are shuffled at random. These two cases will have different stationary distributions,

$$\pi_1(\theta) = \frac{1}{Z}\exp\left(-Cq\theta\right) \qquad \text{and} \qquad \pi_2(\theta) = \frac{1}{Z}\exp\left(Cq\theta\right),$$

and an exact algorithm must be able to distinguish between them. Therefore by using these cases, we can get a bound on the required batch size needed for the exact MH algorithm to distinguish between them. First, we observe that the two cases are symmetric, such that if T_1 is the transition matrix of the chain in case (1) and T_2 is the transition matrix of the chain in case (2), then $T_1(\theta, \theta') = T_2(\theta', \theta)$. Let $0 < \psi < \frac{1}{2}$ denote the probability that T_1 transitions from $\frac{M}{2}$ to $-\frac{M}{2}$. Then because the MH method is exact and the chain is reversible, the probability of the reverse transition is $\psi\exp(-CMq)$. So, explicitly, the transition operators will look like

$$T_1 = \begin{bmatrix} 1-\psi & \psi e^{-CMq} \\ \psi & 1-\psi e^{-CMq} \end{bmatrix} \qquad \text{and} \qquad T_2 = \begin{bmatrix} 1-\psi e^{-CMq} & \psi \\ \psi e^{-CMq} & 1-\psi \end{bmatrix}.$$

The eigenvectors and eigenvalues of this are

$$T_1\pi_1 = \pi_1 \qquad \text{and} \qquad T_1 \begin{bmatrix} -1 \\ 1 \end{bmatrix} = (1 - \psi - \psi\exp(-CMq)) \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

Suppose that we initialize both chains uniformly on $\{-\frac{M}{2}, \frac{M}{2}\}$. Observe that

$$\begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix} = \begin{bmatrix} \frac{\exp(-CMq)}{1+\exp(-CMq)} \\ \frac{1}{1+\exp(-CMq)} \end{bmatrix} + \frac{1-\exp(-CMq)}{2(1+\exp(-CMq))} \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix},$$

the first vector being π_1 and the second being a multiple of the other eigenvector. Equivalently,

$$\begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix} = \pi_1 + \frac{1}{2}\tanh\left(\frac{CMq}{2}\right) \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix},$$

and so for any t, after t steps of the Markov chain, the distribution will be

$$T_1^t \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix} = \pi_1 + \frac{1}{2}\tanh\left(\frac{CMq}{2}\right) \cdot (1-\psi-\psi\exp(-CMq))^t \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

Similarly,

$$T_2^t \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix} = \pi_2 + \frac{1}{2}\tanh\left(\frac{CMq}{2}\right) \cdot (1-\psi-\psi\exp(-CMq))^t \cdot \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

So, the total variation distance between the state of the chains at time t will be bounded by

$$\begin{aligned} &\mathrm{TV}\left(T_1^t \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}, T_2^t \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}\right) \\ &\quad \geq \mathrm{TV}(\pi_1, \pi_2) - \tanh\left(\frac{CMq}{2}\right) \cdot (1-\psi-\psi\exp(-CMq))^t. \end{aligned}$$

Also observe that

$$\begin{aligned} \mathrm{TV}(\pi_1, \pi_2) &= \frac{1}{2}\left\| \begin{bmatrix} \frac{\exp(-CMq)}{1+\exp(-CMq)} \\ \frac{1}{1+\exp(-CMq)} \end{bmatrix} - \begin{bmatrix} \frac{1}{1+\exp(-CMq)} \\ \frac{\exp(-CMq)}{1+\exp(-CMq)} \end{bmatrix} \right\|_1 \\ &= \frac{1-\exp(-CMq)}{1+\exp(-CMq)} = \tanh\left(\frac{CMq}{2}\right), \end{aligned}$$

so

$$\mathrm{TV}\left(T_1^t\begin{bmatrix}1/2\\1/2\end{bmatrix}, T_2^t\begin{bmatrix}1/2\\1/2\end{bmatrix}\right) \geq \tanh\left(\frac{CMq}{2}\right)\cdot\left(1-(1-\psi-\psi\exp(-CMq))^t\right).$$

Also, since we know that our algorithm is guaranteed to have spectral gap ratio at least κ with the original chain, it follows that $\psi \geq \kappa/2$, and so

$$\mathrm{TV}\left(T_1^t\begin{bmatrix}1/2\\1/2\end{bmatrix}, T_2^t\begin{bmatrix}1/2\\1/2\end{bmatrix}\right) \geq \tanh\left(\frac{CMq}{2}\right)\cdot\left(1-\left(1-\frac{\kappa}{2}-\frac{\kappa}{2}\exp(-CMq)\right)^t\right).$$

Now, denote the exact minibatch algorithm to be $\mathcal{A}$. As it runs, the algorithm $\mathcal{A}$ will request data examples by querying the energy difference oracle. Under case (1), we let y_i denote the ith sample that $\mathcal{A}$ *would have observed* if it requested i or more samples, and similarly we let z_i denote the analogous sample in case (2). Fix some constant $t \in \mathbf{N}$ (which we will set later). We let K_1 denote the total number of samples observed by $\mathcal{A}$ across the first t iterations in case (1), and set

$$\mu = \{y_1, y_2, \ldots, y_{K_1}\}.$$

Similarly, we let K_2 denote the number of samples observed by $\mathcal{A}$ across the first t iterations in case (2), and set

$$\nu = \{z_1, z_2, \ldots, z_{K_2}\}.$$

Now, we fix some constant K (to be set later), and consider the following coupling between the behavior of $\mathcal{A}$ across its first t iterations in case (1) and in case (2). First, let all internal randomness of $\mathcal{A}$ and the proposal process under case (1) and (2) be the same, which means that for a given observation of data examples, the algorithm $\mathcal{A}$ will make the same decision, such as whether to require more data examples or not and whether to accept or not. Second, choose a coupling that

minimizes the probability that

$$(y_1, y_2, \ldots, y_{K1}) \neq (z_1, z_2, \ldots, z_{K2}).$$

Such a coupling is guaranteed to exist by the Coupling Lemma, and the probability that these two are not equal will be equal to the total variation distance between their distributions. Third, assign all the other y_i and z_i, for $i > K$, independently according to their distribution.

We are interested in the quantity $p(\mu \neq \nu)$, which bounds the probability that the algorithm may make a different decision in cases (1) and (2). We can decompose this probability into two terms,

$$\begin{aligned} p(\mu \neq \nu) = p(&\mu \neq \nu \text{ and } y_j = z_j \text{ for all } j \leq K) \\ &+ p(\mu \neq \nu \text{ and } y_j \neq z_j \text{ for some } j \leq K). \end{aligned}$$

If $\mu \neq \nu$ but $y_j = z_j$ for all $j \leq K$, the only way that this is possible is for $K_1 > K$ (and, symmetrically, also $K_2 > K$), since otherwise the algorithms would behave identically. So,

$$p(\mu \neq \nu) \leq p(K_1 > K) + p(y_j \neq z_j \text{ for some } j \leq K). \tag{B.5}$$

By Markov's inequality,

$$p(\mu \neq \nu) \leq \frac{\mathbf{E}[K_1]}{K} + p(y_j \neq z_j \text{ for some } j \leq K).$$

For the second term of (B.5), we can reduce the case to only considering K samples. Let S_y be the total number of samples y_i that are -1 and let S_z be the total number of samples z_i that are -1. Since $\mathcal{A}$ is effectively sampling a shuffled dataset at some arbitrary indices without replacement, both of these random variables S_y and S_z are—properly speaking—hypergeometric random variables. However, since our dataset size N is arbitrary here, we can by setting N very large work in

the limit (as $N \to \infty$) in which these variables become binomial (since sampling with replacement and without replacement can be made to have arbitrarily close to the same distribution by making the dataset large). Observe that (in this limit) S_y follows a binomial distribution $B(K, \frac{1-q}{2})$ and S_z follows a binomial distribution $B(K, \frac{1+q}{2})$. Clearly, if $S_y = S_z$, then we can arrange the coupling so that $(y_1, \ldots, y_K) = (z_1, \ldots, z_K)$. So, by the Coupling Lemma,

$$p(y_j \neq z_j \text{ for some } j \leq K) = p(S_y \neq S_z) = \text{TV}(S_y, S_z).$$

From the analysis in [1], we can bound the total variance distance between these two binomial variables with

$$\text{TV}(S_y, S_z) \leq \sqrt{e} \cdot \frac{\tau}{(1-\tau)^2}$$

where $\tau = \sqrt{\frac{K+2}{2}} \cdot q < 1$. Substituting these bounds, we get

$$p(\mu \neq \nu) \leq \frac{\mathbf{E}[K_1]}{K} + \sqrt{e} \cdot \frac{\tau}{(1-\tau)^2}.$$

But the probability that $\mu \neq \nu$ must be an upper bound on the probability that the distributions of the chains in case (1) and (2) after t steps are not equal, since if $\mu = \nu$ in the coupling then the two chains are in the same state. So, using our bound from earlier, we get

$$\tanh\left(\frac{CMq}{2}\right) \cdot \left(1 - \left(1 - \frac{1}{2}\kappa - \frac{1}{2}\kappa \exp(-CMq)\right)^t\right) \leq \frac{\mathbf{E}[K_1]}{K} + \sqrt{e} \cdot \frac{\tau}{(1-\tau)^2}.$$

Now isolating $\mathbf{E}[K_1]$ gives

$$K \cdot \tanh\left(\frac{CMq}{2}\right) \cdot \left(1 - \left(1 - \frac{1}{2}\kappa - \frac{1}{2}\kappa \exp(-CMq)\right)^t\right) - K \cdot \sqrt{e} \cdot \frac{\tau}{(1-\tau)^2} \leq \mathbf{E}[K_1].$$

Also, observe that

$$\left(1 - \frac{1}{2}\kappa - \frac{1}{2}\kappa \exp(-CMq)\right)^t \leq \left(1 - \frac{1}{2}\kappa\right)^t \leq \exp\left(-\frac{\kappa t}{2}\right),$$

so

$$K \cdot \tanh\left(\frac{CMq}{2}\right) \cdot \left(1 - \exp\left(-\frac{\kappa t}{2}\right)\right) - K \cdot \sqrt{e} \cdot \frac{\tau}{(1-\tau)^2} \leq \mathbf{E}[K_1].$$

This gives us the lower bound on $\mathbf{E}[K_1]$ that we are interested in. Now, it remains to assign q, K, and t. We start by assigning t such that

$$t = \left\lceil 2\kappa^{-1} \log(2) \right\rceil,$$

in which case

$$\exp\left(-\frac{\kappa t}{2}\right) \leq \frac{1}{2}$$

and so

$$K \cdot \frac{1}{2} \cdot \tanh\left(\frac{CMq}{2}\right) - K \cdot \sqrt{e} \cdot \frac{\tau}{(1-\tau)^2} \leq \mathbf{E}[K_1].$$

Now, we add some simplifying assumptions, which we will validate are true later. We assume that

$$\tau = \sqrt{\frac{K+2}{2}} \cdot q \leq \frac{1}{2};$$

in this case

$$\sqrt{e} \cdot \frac{\tau}{(1-\tau)^2} \cdot K \leq 4\sqrt{e} \cdot \tau \leq 5\sqrt{K+2} \cdot q.$$

We set q such that

$$CMq = 1,$$

and we assume that CM is large enough that this assignment of q is within range (i.e. $0 < q < 1$). This gives us

$$K \cdot \frac{1}{2} \cdot \tanh\left(\frac{1}{2}\right) - 5K\sqrt{K+2} \cdot \frac{1}{CM} \leq \mathbf{E}[K_1].$$

Since $\tanh(1/2) > 5/16$, we can simplify this to

$$K \cdot \frac{5}{32} - 5K\sqrt{K+2} \cdot \frac{1}{CM} \leq \mathbf{E}[K_1].$$

All that remains is to assign K. We assign K such that

$$\sqrt{K+2} \cdot \frac{1}{CM} = \frac{1}{64}.$$

In this case, we get

$$K = \frac{C^2M^2}{4096} - 2,$$

and our bound reduces to

$$\left(\frac{C^2M^2}{4096} - 2\right) \cdot \frac{5}{64} \leq \mathbf{E}[K_1].$$

We can simplify this further to

$$2^{-16} \cdot C^2M^2 - \frac{5}{32} \leq \mathbf{E}[K_1].$$

Now, this is a bound on the expected number of samples taken across t iterations. This means that the number of samples taken in any given iteration will be bounded by

$$\frac{\mathbf{E}[K_1]}{t} \geq \frac{2^{-16} \cdot C^2M^2 - \frac{5}{32}}{2\kappa^{-1}\log(2) + 1} = \frac{2^{-16} \cdot \kappa C^2M^2 - \frac{5\kappa}{32}}{2\log(2) + \kappa}.$$

A few more loose bounds, leveraging $\kappa < 1$, gives us

$$\frac{\mathbf{E}[K_1]}{t} \geq 2^{-18} \cdot \kappa C^2M^2 - \frac{\kappa}{16}.$$

This proves the lemma. □

Next, we will show the following lemma, which characterizes what happens when CM is small.

Lemma 3. *Considering minibatch MH algorithms in the same setting as the theorem, the expected batch size at any iteration must be lower bounded by*

$$\mathbf{E}[B] \geq \frac{\kappa}{2} \min\left(CM(\theta, \theta'), 1\right).$$

Proof. Here, we will prove a lower bound that characterizes the limits of exact stateless minibatch MH algorithms when they use very few examples. Again, without loss of generality we consider a reduction to the two-state case as we did

in the proof of the previous lemma. Suppose that a exact stateless minibatch MH algorithm with the same forward and backward proposal probabilities (given some $c_1, \ldots, c_N$, C, and M) requests any energy function examples at all only with probability p. Consider two cases, which have the same $c_1, \ldots, c_N$, C and M. In the first case,

$$\sum_{i=1}^{n}(U_i(\theta) - U_i(\theta')) = CM(\theta, \theta'),$$

while in the second case,

$$\sum_{i=1}^{n}(U_i(\theta) - U_i(\theta')) = -CM(\theta, \theta').$$

These are clearly possible by setting U_i to the limits of what is covered by the bounds. In the first case, the baseline MH method would accept with probability 1. In the second case, it will accept with probability $\exp(-CM(\theta, \theta'))$. Since the stateless MH algorithm is reversible, it must accept in the first case with some probability a and in the second case with probability $a \cdot \exp(-CM(\theta, \theta'))$. But, the algorithm can only distinguish the two cases if it requests samples, which only happens with probability at most p. So,

$$a - a \cdot \exp(-CM(\theta, \theta') \leq p.$$

Since we know that it must be the case that $a \geq \kappa$ (from a straightforward analysis of a two-state case), it follows that

$$\frac{p}{\kappa} \geq \frac{p}{a} \geq 1 - \exp(-CM(\theta, \theta')) \geq \frac{1}{2}\min\left(CM(\theta, \theta'), 1\right).$$

Since p is an obvious lower bound on the expected value of the batch size, it follows that

$$\mathbf{E}[B] \geq \frac{\kappa}{2}\min\left(CM(\theta, \theta'), 1\right).$$

□

To prove Theorem 8 we now combine the results of these two lemmas. We have

$$\mathbf{E}[B] \geq 2^{-18} \cdot \kappa C^2 M^2(\theta, \theta') - 2^{-4} \cdot \kappa.$$

and

$$\mathbf{E}[B] \geq \frac{\kappa}{2} \min\left(CM(\theta, \theta'), 1\right).$$

Since these are both lower bounds, we can combine them to get

$$\begin{aligned}\mathbf{E}[B] &\geq \max\left(2^{-18} \cdot \kappa C^2 M^2(\theta, \theta') - 2^{-4} \cdot \kappa, \frac{\kappa}{2} \min\left(CM(\theta, \theta'), 1\right)\right) \\ &= \kappa \cdot \max\left(2^{-18} \cdot C^2 M^2(\theta, \theta') - 2^{-4}, \frac{1}{2} \min\left(CM(\theta, \theta'), 1\right)\right).\end{aligned}$$

It is obvious from a simple big-$\mathcal{O}$ analysis here that there exists a global constant $\zeta > 0$ such that

$$\mathbf{E}[B] \geq \zeta \cdot \kappa \left(C^2 M^2(\theta, \theta') + CM(\theta, \theta')\right).$$

This proves the theorem.

B.1.5 Proof of Corollary 1

Proof. Recall that the lower bound on the batch size in each iteration is

$$\mathbf{E}[B] \geq \zeta \cdot \kappa \left(C^2 M^2(\theta, \theta') + CM(\theta, \theta')\right).$$

Since $C = \mathcal{O}(N)$ and $M(\theta, \theta') = \mathcal{O}(N^{-(h+1)/2})$, the expectation of the batch size follows

$$\mathbf{E}[B] = \mathcal{O}(C^2 M^2(\theta, \theta') + CM(\theta, \theta')) = \mathcal{O}(CM(\theta, \theta')) = \mathcal{O}(N^{1-h}/2).$$

When $h = 1$, $\mathbf{E}[B] = \mathcal{O}(1)$ and when $h = 2$, $\mathbf{E}[B] = \mathcal{O}(1/\sqrt{N})$. □

B.1.6 Derivation of Equation (3.4)

Based on the bound in Theorem 7, to make sure that the spectral ratio $\bar{\gamma}/\gamma \geq \kappa$, we can set χ such that

$$\exp\left(-\frac{1}{\chi} - 2\sqrt{\frac{\log 2}{\chi}}\right) = \kappa.$$

Solving the above equation gives us

$$\chi = \frac{(2\log 2 - \log\kappa + 2\sqrt{\log 2(\log 2 - \log\kappa)})}{\log^2\kappa} \leq \frac{4}{(1-\kappa)\log(1/\kappa)}.$$

Since the spectral gap ratio is monotonically increasing w.r.t. χ, we can instead set χ to the upper bound

$$\chi = \frac{4}{(1-\kappa)\log(1/\kappa)}$$

which guarantees that $\bar{\gamma}/\gamma \geq \kappa$.

B.2 Construction of Algorithm 6

Algorithm 6 can be derived by carefully replacing the global bounds on the energy in PoissonMH [156] with local bounds on the energy differences (Assumption 1). PoissonMH is a variant of Poisson Gibbs and therefore inherits the same assumptions for Gibbs sampling on graphical models, which are often violated in the applications of MH. In particular, PoissonMH works on *factor graphs* which define a distribution $\pi(\theta)$ over a set of factors $\{\phi_i(\theta)\}_{i=1}^N$ as follows

$$\pi(\theta) \propto \exp\left(\sum_{i=1}^{N} \phi_i(\theta)\right).$$

PoissonMH assumes that each factor ϕ_i is non-negative without the loss of generality (we can add a positive constant to ϕ_i to make it non-negative without

Algorithm 13 PoissonMH

given: initial state $\theta \in \Theta$; proposal dist. q; hyperparameter λ; Global bounds M_i, L
loop
  propose $\theta' \sim q(\cdot|\theta)$
  for $i \in \{1, \ldots, N\}$ **do**
    sample $s_i \sim \text{Poisson}\left(\frac{\lambda M_i}{L} + \phi_i(\theta)\right)$
  end for
  form minibatch $\mathcal{S} \leftarrow \{i | s_i > 0\}$
  compute MH ratio $r \leftarrow \frac{\exp\left(\sum_{i\in\mathcal{S}} s_i \log\left(1+\frac{L}{\lambda M_i}\phi_i(\theta')\right)\right) q(\theta'|\theta)}{\exp\left(\sum_{i\in\mathcal{S}} s_i \log\left(1+\frac{L}{\lambda M_i}\phi_i(\theta)\right)\right) q(\theta|\theta')}$
  with probability $\min(1, r)$, set $\theta \leftarrow \theta'$
end loop

changing the distribution) and is bounded globally by a constant M_i. That is

$$0 \leq \phi_i(\theta) \leq M_i \text{ for all } \theta.$$

This assumption does not hold for most applications of MH, such as the linear and logistic regression experiments in Section 3.2.4.

Let $L = \sum_i M_i$ and define Poisson auxiliary variable s_i as the following

$$s_i|\theta \sim \text{Poisson}\left(\frac{\lambda M_i}{L} + \phi_i(\theta)\right),$$

where $\lambda > 0$ is a hyperparameter. Running standard MH on the joint distribution of θ and s_i results in the following acceptance ratio

$$r_{\text{PoissonMH}}(\theta, \theta') = \frac{\exp\left(\sum_i s_i \log\left(1 + \frac{L}{\lambda M_i}\phi_i(\theta')\right)\right) q(\theta'|\theta)}{\exp\left(\sum_i s_i \log\left(1 + \frac{L}{\lambda M_i}\phi_i(\theta)\right)\right) q(\theta|\theta')}.$$

Here, the sum is essentially performed over the set of index i whose s_i is greater than zero. When $s_i = 0$, it is clear that the factor ϕ_i will not appear in the acceptance ratio $r_{\text{PoissonMH}}$. Thus PoissonMH enables using a subset of factors for the MH decision step (Algorithm 13).

To construct our method from this, we can define the factor ϕ_i in the factor graph to be

$$\phi_i(x) = \frac{U_i(\theta) + U_i(\theta')}{2} - U_i(x) + \frac{c_i}{2} M(\theta, \theta') \tag{B.6}$$

where $x \in \{\theta, \theta'\}$. It is easy to see that ϕ_i satisfy $0 \leq \phi_i(x) \leq c_i M(\theta, \theta')$. And then we define the Poisson variables s_i as the follows

$$\begin{aligned} s_i|(\theta, \theta') &\sim \text{Poisson}\left(\frac{\lambda c_i}{C} + \phi_i(\theta)\right) \\ &= \text{Poisson}\left(\frac{\lambda c_i}{C} + \frac{U_i(\theta') - U_i(\theta) + c_i M(\theta, \theta')}{2}\right). \end{aligned}$$

These Poisson auxiliary variables $\{s_i\}_{i=1}^N$ are called *local*, because their distributions change each iteration depending on the current pair (θ, θ') and only rely on local bounds in Assumption 1. This is in contrast to the *global* auxiliary variables used in PoissonMH and FlyMC which are used to form a joint distribution with θ and both require global bounds in their conditional distributions.

The acceptance ratio r_{TunaMH} is the same as $r_{\text{PoissonMH}}$ but with the new definitions of s_i and ϕ_i. We outline TunaMH using the notation of ϕ_i and s_i in Algorithm 14.

We now show that Algorithm 14 is statistically equivalent to Algorithm 6. To see this, we first use *thinning*, a commonly used technique [84, 13, 18, 32, 156], to quickly resample all s_i from their new distributions in each iteration in Algorithm 14. This is achieved by replacing the global bounds with the local bounds in Algorithm 4 in the Appendix of [156]. Specifically, we first sample B from a Poisson distribution

$$B \sim \text{Poisson}(\lambda + CM(\theta, \theta')).$$

Here $\lambda + CM(\theta, \theta')$ is an upper bound on $\mathbf{E}[\sum_i s_i]$. We then form the minibatch by running

for $b \in \{1, \ldots, B\}$ **do**
  sample i_b such that $\mathbf{P}(i_b = i) = c_i/C$, for $i = 1 \ldots N$
  with probability $\frac{\lambda c_{i_b} + C\phi_{i_b}(\theta)}{\lambda c_{i_b} + C c_{i_b} M(\theta,\theta')}$ **add** i_b to $\mathcal{I}$
end for

Algorithm 14 TunaMH

given: initial state $\theta \in \Theta$; proposal dist. q; λ; Asm. 1 parameters c_i, C, M; function ϕ_i defined in (B.6)
loop
  propose $\theta' \sim q(\cdot|\theta)$ and **compute** $M(\theta, \theta')$
  for $i \in \{1, \ldots, N\}$ **do**
    sample $s_i \sim \text{Poisson}\left(\frac{\lambda c_i}{C} + \phi_i(\theta)\right)$
  end for
  form minibatch $\mathcal{S} \leftarrow \{i | s_i > 0\}$
  compute MH ratio $r \leftarrow \frac{\exp\left(\sum_{i\in\mathcal{S}} s_i \log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta')\right)\right) q(\theta'|\theta)}{\exp\left(\sum_{i\in\mathcal{S}} s_i \log\left(1+\frac{C}{\lambda c_i}\phi_i(\theta)\right)\right) q(\theta|\theta')}$
  with probability $\min(1, r)$, set $\theta \leftarrow \theta'$
end loop

By substituting $\lambda = \chi C^2 M^2(\theta, \theta')$ and the expression of ϕ_i, we can get the part of "form minibatch $\mathcal{I}$" in Algorithm 6.

To see that the MH ratio in Algorithm 6 and 14 are equivalent, we can write out r in Algorithm 14 using the above fast way of resampling s_i

$$r_{\text{TunaMH}} = \frac{\exp\left(\sum_{i\in\mathcal{I}} \log\left(1 + \frac{C}{\lambda c_i}\phi_i(\theta')\right)\right) q(\theta'|\theta)}{\exp\left(\sum_{i\in\mathcal{I}} \log\left(1 + \frac{C}{\lambda c_i}\phi_i(\theta)\right)\right) q(\theta|\theta')}.$$

We then substitute the definition of ϕ_i in (B.6) and it follows that

$$r_{\text{TunaMH}} = \exp\left(\sum_{i\in\mathcal{I}} \left(\log\left(\frac{2\lambda c_i + C\left(U_i(\theta) - U_i(\theta') + c_i M(\theta,\theta')\right)}{2\lambda c_i + C\left(U_i(\theta') - U_i(\theta) + c_i M(\theta,\theta')\right)}\right)\right)\right) \cdot \frac{q(\theta'|\theta)}{q(\theta|\theta')}.$$

We can rearrange the log term inside r_{TunaMH} as

$$\begin{aligned}
&\log\left(\frac{2\lambda c_i + C\left(U_i(\theta) - U_i(\theta') + c_i M(\theta,\theta')\right)}{2\lambda c_i + C\left(U_i(\theta') - U_i(\theta) + c_i M(\theta,\theta')\right)}\right) \\
&\quad= \log\left(\frac{2\lambda c_i + C\left(U_i(\theta) - U_i(\theta')\right) + c_i C M(\theta,\theta')}{2\lambda c_i + C\left(U_i(\theta') - U_i(\theta)\right) + c_i C M(\theta,\theta')}\right) \\
&\quad= \log\left(\frac{1 + \frac{C}{2\lambda c_i + c_i C M(\theta,\theta')}\left(U_i(\theta) - U_i(\theta')\right)}{1 + \frac{C}{2\lambda c_i + c_i C M(\theta,\theta')}\left(U_i(\theta') - U_i(\theta)\right)}\right) \\
&\quad= 2\,\text{artanh}\left(\frac{C\left(U_i(\theta) - U_i(\theta')\right)}{c_i(2\lambda + C M(\theta,\theta'))}\right).
\end{aligned}$$

So r_{TunaMH} can be written as

$$r_{\text{TunaMH}} = \exp\left(2\sum_{i\in\mathcal{I}} \text{artanh}\left(\frac{C\left(U_i(\theta) - U_i(\theta')\right)}{c_i(2\lambda + C M(\theta,\theta'))}\right)\right) \cdot \frac{q(\theta'|\theta)}{q(\theta|\theta')}.$$

Finally setting λ to be $\chi C^2 M^2(\theta,\theta')$ produces the MH ratio in Algorithm 6.

By proving the equivalence of the minibatch and the MH ratio, we show that Algorithm 6 and 14 are statistically equivalent.

B.3 Theoretically Optimal Value of χ

The overall wall-clock time L for a chain to converge can be represented as the number of steps times the wall-clock time l of each step. We then minimize an upper bound of this overall wall-clock time to get the optimal value of χ.

Consider a lazy Markov chain on a finite state Θ. The *relaxation time* t_{rel} of a Markov chain is defined to be the inverse of the spectral gap γ: $t_{\text{rel}} = 1/\gamma$. The *mixing time* t_{mix}, i.e. the number of steps required for a chain to converge to within TV distance δ to the target distribution π, is bounded by [83]

$$t_{\text{mix}} \leq t_{\text{rel}} \log\left(\frac{1}{\delta \cdot \min_{\theta\in\Theta} \pi(\theta)}\right).$$

It follows that the overall wall-clock time L is upper bouned by

$$L = l \cdot t_{\text{mix}} \leq l \cdot t_{\text{rel}} \log\left(\frac{1}{\delta \cdot \min_{\theta\in\Theta} \pi(\theta)}\right).$$

We assume that the expected wall clock time to run a step is proportional to the batch size plus some constant, which measures the cost of computing the proposal. Specifically, We use η and ξ to denote the time to get a proposal θ' and compute a U_i in a step. Then we can write the time of a step l as

$$l = B\xi + \eta.$$

In order to minimize L, we can instead minimize its upper bound, which is equivalent to minimize

$$l \cdot t_{\text{rel}} = (B\xi + \eta) \cdot \frac{1}{\gamma}. \tag{B.7}$$

Recall that for TunaMH, the average batch size over all steps is

$$\mathbf{E}_{(\theta,\theta')\sim\pi(\theta)q(\theta'|\theta)}[\chi C^2 M^2(\theta,\theta') + CM(\theta,\theta')],$$

and the spectral gap $\bar{\gamma}$ is lower bounded by the spectral gap of standar MH γ such that

$$\bar{\gamma} \geq \exp\left(-\frac{1}{\chi} - 2\sqrt{\frac{\log 2}{\chi}}\right) \cdot \gamma.$$

Substituting the expression of batch size and spectral gap to (B.7) gives

$$l \cdot t_{\text{rel}} \leq \left(\mathbf{E}_{(\theta,\theta')\sim\pi(\theta)q(\theta'|\theta)}[\chi C^2 M^2(\theta,\theta') + CM(\theta,\theta')]\xi + \eta\right) \cdot \exp\left(\frac{1}{\chi} + 2\sqrt{\frac{\log 2}{\chi}}\right) \cdot \frac{1}{\gamma}.$$

To minimize the RHS of the above equation over χ, we let the derivative w.r.t.

χ to be zero and get,

$$\begin{aligned}
&\xi C^2 \mathbf{E}_{(\theta,\theta')\sim\pi(\theta)q(\theta'|\theta)}[M^2(\theta,\theta')]\chi^{-1} + (\xi C \mathbf{E}_{(\theta,\theta')\sim\pi(\theta)q(\theta'|\theta)}[M(\theta,\theta')] + \eta)\chi^{-2} \\
&\quad + \sqrt{\log 2}\xi C^2 \mathbf{E}_{(\theta,\theta')\sim\pi(\theta)q(\theta'|\theta)}[M^2(\theta,\theta')]\chi^{-\frac{1}{2}} \\
&\quad + \sqrt{\log 2}(\xi C \mathbf{E}_{(\theta,\theta')\sim\pi(\theta)q(\theta'|\theta)}[M(\theta,\theta')] + \eta)\chi^{-\frac{3}{2}} \\
&\quad = \xi C^2 \mathbf{E}_{(\theta,\theta')\sim\pi(\theta)q(\theta'|\theta)}[M^2(\theta,\theta')].
\end{aligned}$$

When χ is small, the LHS is approximately $(\xi C \mathbf{E}_{(\theta,\theta')\sim\pi(\theta)q(\theta'|\theta)}[M(\theta,\theta')] + \eta)\chi^{-2}$ which gives us

$$\chi = \sqrt{\frac{\xi C \mathbf{E}_{(\theta,\theta')\sim\pi(\theta)q(\theta'|\theta)}[M(\theta,\theta')] + \eta}{\xi C^2 \mathbf{E}_{(\theta,\theta')\sim\pi(\theta)q(\theta'|\theta)}[M^2(\theta,\theta')]}}.$$

When it is quick to get a proposal ($\eta \approx 0$) and the variance of M is small, we can further simplify it to

$$\chi = \frac{1}{\sqrt{C \mathbf{E}_{(\theta,\theta')\sim\pi(\theta)q(\theta'|\theta)}[M(\theta,\theta')]}}.$$

In practice, we can get the above theoretically optimal value of χ by empirically estimating the mean and variance of $M(\theta,\theta')$. Note that even if these empirical estimates are accurate, there may exist better χ, since the upper bounds (the mixing time bound and the spectral gap bound) we use to get the optimal value may be loose. We give a simpler heuristic to tune χ in practice in Section 3.2.4.

B.4 Experimental Details and Additional Results

Experiment in Section 3.2.1

To verify Theorem 6, we empirically construct a distribution in the form of Section B.1.1 such that AustereMH and MHminibatch are biased on. Note that the proof in

Section B.1.1 shows there must exist such a distribution for any inexact minibatch method but does not tell us how to find one for a specific method. Therefore, in order to find such a distribution, we construct an example and empirically test whether AustereMH and MHminibatch are biased on it.

We let data x_i take one of two values $\{-1, 5\}$. Consider a dataset of size 6000. We let 5000 data take value -1 and the remaining 1000 data take value 5. Define the target distribution $\pi(\theta)$ to be

$$\pi(\theta) \propto \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\theta \cdot x_i\right)$$

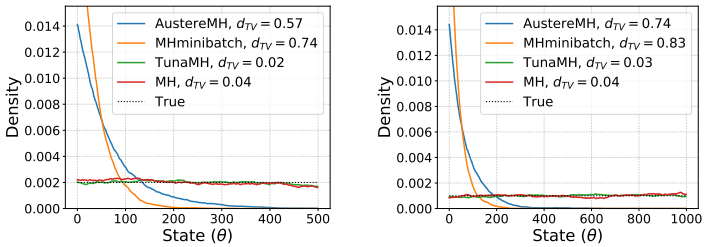
where the domain of θ is $\{0, 1, \ldots, K-1\}$. Therefore the number of state is K. Since $\sum_i x_i = 0$, it is clear to see that the stationary distribution of θ is a uniform distribution. We define the proposal distribution to be the following

$$p(\theta,\theta) = \frac{1}{2}, \quad \text{for all } \theta; \quad p(\theta,\theta-1) = \frac{1}{4}, \quad p(\theta,\theta+1) = \frac{1}{4} \quad \text{for } \theta \in \{1,\ldots,K-2\};$$

and $p(0,1) = p(K-1,K-2) = \frac{1}{2}$.

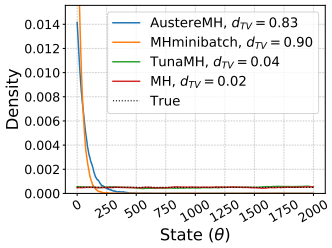
We set the hyperparameter error ϵ in AustereMH to be 0.01 and δ in MHminibatch to be 5, following the setting in their original papers [81, 132]. We set batch size m in both methods to be 30. We find that AustereMH and MHminibatch are both inexact on this example and the error increases as we increase K. Thus we empirically verify the statement in Theorem 6.

Besides the density estimate comparison on $K = 200$ shown in Figure 3.4b, we additionally report the estimate results on other values of K in Figure B.1. We see that the results are similar, all showing that TunaMH and standard MH can give accurate estimate whereas inexact methods are seriously wrong.

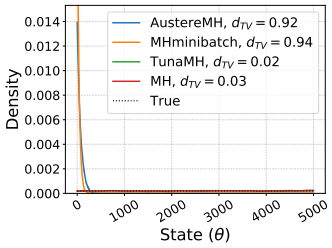


(a) $K = 500$

(b) $K = 1000$



(e) $K = 2000$



(e) $K = 5000$

Figure B.1: Density estimate comparison on $K = 500, 1000, 2000, 5000$.

On Robust Linear Regression We further tested AustereMH on robust linear regression in Section 3.2.4 with $N = 5000$. We computed the MSE between estimated and true parameters. MH, TunaMH and AustereMH obtained MSE 0.149, 0.15 and 1.19 respectively, indicating inexact method error can be large on typical problems.

Robust Linear Regression

We follow the experimental setup of robust linear regression (RLR) in [32]. Specifically, we have data $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. The likelihood is modeled by a student's t-distribution with degrees of freedom v:

$$p(y_i|\theta, x_i) = \text{Student}(y_i - \theta^\intercal x_i|v).$$

It follows that

$$U_i(\theta) = \frac{v+1}{2} \log\left(1 + \frac{(y_i - \theta^\intercal x_i)^2}{v}\right),$$

and the first derivative

$$\partial_j U_i(\theta) = -(v+1)\frac{x_{ij}(y_i - \theta^\intercal x_i)}{v + (y_i - \theta^\intercal x_i)^2}.$$

Since the function U_i is Lipschitz continuous, we can easily get the bound used in TunaMH, TFMH and SMH. We set $M(\theta, \theta') = \|\theta - \theta'\|_2$ and then it follows

$$c_i = \sup_{\theta \in \mathbb{R}} \|\nabla U_i(\theta)\|_2 = \frac{v+1}{2\sqrt{v}} \|x_i\|_2 .$$

The data x_i and y_i is generated as follows

$$y_i = \sum_j x_{ij} + \epsilon_i$$

where $\epsilon_i \sim \mathcal{N}(0, 1)$.

In Section 3.2.4, we set $v = 4$, $d = 100$ and use a flat prior $p(\theta) = 1$. Note that our problem dimension d is much larger than that in the SMH paper [32] ($d = 10$). This makes the control variates in SMH problematic since the bounds they require appear to scale badly in high dimensions.

To reach the target acceptance rate, we set the stepsize in each method as in Table B.1 and B.2. For TunaMH and TunaMH-MAP, we set $\chi = 1e-5$ for $N = 5000, 20000$ and $\chi = 1e-4$ for $N = 50000, 100000$. For FlyMC and FlyMC-MAP, we set the probability for a data going from dark to bright $q_{d\to b}$ to be 0.01. Without the MAP, we collect 80000 samples after 200000 step burnin. With the MAP, we collect 80000 samples without burnin.

Table B.1: Stepsize of methods without the MAP.

	MH	TFMH	FlyMC	TunaMH
RLR $N = 5000$	4e-3	1e-4	2.7e-3	8e-4, $\chi = 1e-5$
RLR $N = 20000$	2e-3	3e-5	1.5e-3	3e-4, $\chi = 1e-5$
RLR $N = 50000$	1.3e-3	1.2e-5	9e-4	2e-4, $\chi = 1e-4$
RLR $N = 100000$	9e-4	6e-6	7e-4	1.7e-4, $\chi = 1e-4$
TGM	3e-1	2.2e-2	1e-2	1e-1
LR	5e-3	1e-4	2e-3	1e-3

Table B.2: Stepsize of methods with the MAP.

	MH-MAP	SMH-1	SMH-2	FlyMC-MAP	TunaMH-MAP
RLR $N = 5000$	4e-3	4e-3	4e-3	6e-3	8e-4, $\chi = 1e-5$
RLR $N = 20000$	2e-3	2e-3	2e-3	3.5e-3	3e-4, $\chi = 1e-5$
RLR $N = 50000$	1.2e-3	1.2e-3	1.2e-3	2.5e-3	1.2e-4, $\chi = 1e-4$
RLR $N = 100000$	9e-4	5.9e-4	8e-4	1.7e-3	7e-5. $\chi = 1e-4$
TGM	-	1e-1	-	1e-2	-

Additional Experimental Results with $d = 10$ We ran RLR experiment with $d = 10$ and $N = 10^5$ to compare the performance in low dimensions. The ESS/S for TFMH, FlyMC, TunaMH are 0.02, 0.75, & 1.7, respectively; SMH-1, SMH-2, FlyMC-MAP and TunaMH-MAP are 174.7, 5969.5, 730.8, & 730.1 respectively. This suggests TunaMH is significantly better without MAP/control variates. With MAP/control variates, TunaMH is better than SMH-1, similar to FlyMC and worse than SMH-2.

Truncated Gaussian Mixture

The data in this truncated Gaussian mixture (TGM) task is generated as follows

$$x_i \sim \frac{1}{2}\mathcal{N}(\theta_1, \sigma_x^2) + \frac{1}{2}\mathcal{N}(\theta_1 + \theta_2, \sigma_x^2)$$

where $\theta_1 = 0, \theta_2 = 1$ and $\sigma^2 = 2$. The posterior θ has two modes at $(\theta_1, \theta_2) = (0, 1)$ and $(\theta_1, \theta_2) = (1, -1)$. In order to get the bounds required by all methods, we truncate the Gaussian by setting $\theta_1, \theta_2 \in [-3, 3]$.

For simplicity we assume a flat prior $p(\theta) = 1$. Then the energy is given by

$$\begin{aligned} U_i(\theta) &= -\log p(x_i|\theta) \\ &= \log(2\sqrt{2\pi}\sigma_x) - \log\left[\exp\left(-\frac{(x_i-\theta_1)^2}{2\sigma_x^2}\right) + \exp\left(-\frac{(x_i-\theta_1-\theta_2)^2}{2\sigma_x^2}\right)\right]. \end{aligned}$$

Denote $E_1 = \exp\left(-\frac{(x_i-\theta_1)^2}{2\sigma_x^2}\right)$ and $E_2 = \exp\left(-\frac{(x_i-\theta_1-\theta_2)^2}{2\sigma_x^2}\right)$. To get the upper bound in TunaMH, TFMH and SMH, we compute the gradient

$$\begin{aligned} \frac{\partial U_i(\theta)}{\partial \theta_1} &= -\frac{1}{E_1+E_2}\left(E_1 \cdot \frac{x_i-\theta_1}{\sigma_x^2} + E_2 \cdot \frac{x_i-\theta_1-\theta_2}{\sigma_x^2}\right), \\ \frac{\partial U_i(\theta)}{\partial \theta_2} &= -\frac{1}{E_1+E_2}\left(E_2 \cdot \frac{x_i-\theta_1-\theta_2}{\sigma_x^2}\right). \end{aligned}$$

Since $\theta_i \in [-3, 3]$, it follows that

$$\begin{aligned} \left|\frac{\partial U_i(\theta)}{\partial \theta_1}\right| &\le \frac{|x_i|+3}{\sigma_x^2} + \frac{|x_i|+3+3}{\sigma_x^2} \le \frac{2\,|x_i|+9}{\sigma_x^2}, \\ \left|\frac{\partial U_i(\theta)}{\partial \theta_2}\right| &\le \frac{|x_i|+3+3}{\sigma_x^2} \le \frac{|x_i|+6}{\sigma_x^2}. \end{aligned}$$

Therefore we can set $M(\theta, \theta') = \|\theta - \theta'\|_2$ and

$$c_i = \sqrt{\left(\frac{2\,|x_i|+9}{\sigma_x^2}\right)^2 + \left(\frac{|x_i|+6}{\sigma_x^2}\right)^2}.$$

To use the control variate in SMH, we need to compute the second derivatives

$$
\begin{aligned}
&\frac{\partial^2 U_i(\theta)}{\partial^2 \theta_1} \\
&\quad = \frac{1}{(E_1+E_2)^2} \cdot \left(E_1 \cdot \frac{x_i - \theta_1}{\sigma_x^2} + E_2 \cdot \frac{x_i - \theta_1 - \theta_2}{\sigma_x^2} \right)^2 \\
&\quad - \left[E_1 \cdot \left(\left(\frac{x_i - \theta_1}{\sigma_x^2} \right)^2 - \frac{1}{\sigma_x^2} \right) + E_2 \cdot \left(\left(\frac{x_i - \theta_1 - \theta_2}{\sigma_x^2} \right)^2 - \frac{1}{\sigma_x^2} \right) \right] \cdot \frac{1}{E_1 + E_2} \\
&\frac{\partial^2 U_i(\theta)}{\partial \theta_1 \partial \theta_2} \\
&\quad = \frac{1}{(E_1+E_2)^2} \cdot \left(E_2 \cdot \left(\frac{x_i - \theta_1 - \theta_2}{\sigma_x^2} \right) \right) \cdot \left(E_1 \cdot \frac{x_i - \theta_1}{\sigma_x^2} + E_2 \cdot \frac{x_i - \theta_1 - \theta_2}{\sigma_x^2} \right) \\
&\quad - \left[E_2 \left(\left(\frac{x_i - \theta_1 - \theta_2}{\sigma_x^2} \right)^2 - \frac{1}{\sigma_x^2} \right) \right] \cdot \frac{1}{E_1 + E_2} \\
&\frac{\partial^2 U_i(\theta)}{\partial^2 \theta_2} \\
&\quad = \frac{1}{(E_1+E_2)^2} \cdot \left(E_1 \cdot \frac{x_i - \theta_1}{\sigma_x^2} + E_2 \cdot \frac{x_i - \theta_1 - \theta_2}{\sigma_x^2} \right)^2 \\
&\quad - \left[E_2 \cdot \left(\left(\frac{x_i - \theta_1 - \theta_2}{\sigma_x^2} \right)^2 - \frac{1}{\sigma_x^2} \right) \right] \cdot \frac{1}{E_1 + E_2}.
\end{aligned}
$$

Given the parameter space, we have the upper bounds

$$
\begin{aligned}
\left| \frac{\partial^2 U_i(\theta)}{\partial^2 \theta_1} \right| &\leq \left(\frac{2\,|x_i| + 9}{\sigma_x^2} \right)^2 + \left(\frac{|x_i| + 3}{\sigma_x^2} \right)^2 + \left(\frac{|x_i| + 6}{\sigma_x^2} \right)^2 + \frac{2}{\sigma_x^2} \\
\left| \frac{\partial^2 U_i(\theta)}{\partial \theta_1 \partial \theta_2} \right| &\leq \frac{2\,|x_i| + 9}{\sigma_x^2} \cdot \frac{|x_i| + 6}{\sigma_x^2} + \left(\frac{|x_i| + 6}{\sigma_x^2} \right)^2 + \frac{1}{\sigma_x^2} \\
\left| \frac{\partial^2 U_i(\theta)}{\partial^2 \theta_2} \right| &\leq \left(\frac{2\,|x_i| + 9}{\sigma_x^2} \right)^2 + \left(\frac{|x_i| + 6}{\sigma_x^2} \right)^2 + \frac{1}{\sigma_x^2}.
\end{aligned}
$$

It follows

$$
\bar{U}_{2,i} = \left(\frac{2\,|x_i| + 9}{\sigma_x^2} \right)^2 + \left(\frac{|x_i| + 3}{\sigma_x^2} \right)^2 + \left(\frac{|x_i| + 6}{\sigma_x^2} \right)^2 + \frac{2}{\sigma_x^2}.
$$

which is required in SMH-1.

To get the lower bounds in FlyMC, we use the first-order Taylor expansion for $U_i(\theta)$. Higher order approximation is possible but would require heavier computa-

tion. By Taylor expansion,

$$U_i(\theta) = U_i(\theta^0) + \nabla U_i(\theta^0)^\intercal(\theta - \theta^0) + \frac{1}{2}(\theta - \theta^0)^\intercal \nabla^2 U_i(c)(\theta - \theta^0)$$

where c is between θ and θ^0.

Then we can define $\log B_i(\theta)$ in FlyMC as the follows

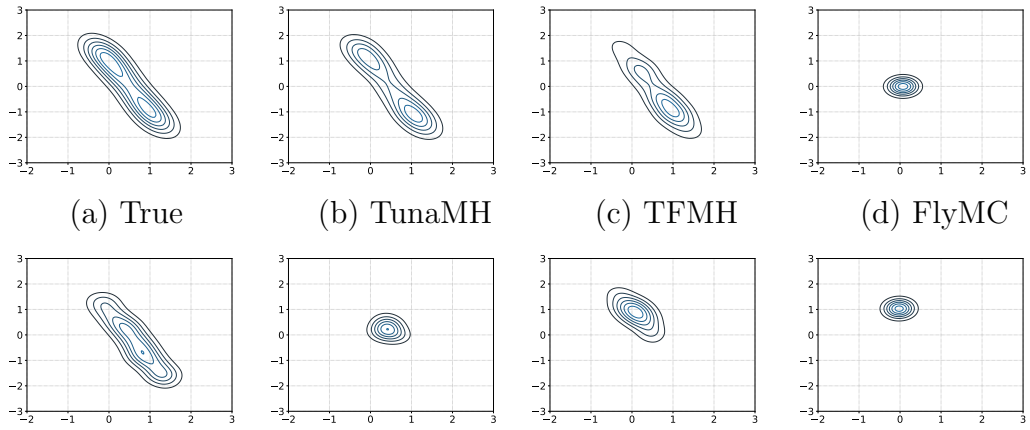
$$\begin{aligned}\log B_i(\theta) &= -U_i(\theta^0) - \nabla U_i(\theta^0)^\intercal(\theta - \theta^0) - \frac{1}{2} \cdot \max_c \left\|\nabla^2 U_i(c)\right\|_1 \cdot \left\|\theta - \theta^0\right\|_1^2 \\ &= -U_i(\theta^0) - \nabla U_i(\theta^0)^\intercal(\theta - \theta^0) - \frac{1}{2} \cdot \bar{U}_{2,i} \cdot \left\|\theta - \theta^0\right\|_1^2 .\end{aligned}$$

The sum of $\log B_i$ is

$$\sum_{i=1}^N \log B_i(\theta) = -N \cdot U_i(\theta^0) - \left(\sum_{i=1}^N \nabla U_i(\theta^0)\right)^\intercal (\theta - \theta^0) - \frac{1}{2} \cdot \sum_{i=1}^N \bar{U}_{2,i} \cdot \left\|\theta - \theta^0\right\|_1^2 .$$

We set θ^0 to be 0 and the MAP solution in standard and MAP-tuned FlyMC respectively.

We tune the stepsize of each method to reach the acceptance rate 60% and the value of stepsize is summarized in Table B.1 and B.2. We set $\chi = 10^{-4}$ in TunaMH and $q_{d\to b} = 0.01$ in FlyMC and FlyMC-MAP. We compute the symmetric KL between the run-average density estimate and the true distribution. Since this is a two-dimensional problem, we are able to visualize the density estimate. As shown in Figure B.2, we plot the density estimate after running the method for 1 second. It is clear to see that the density estimate of TunaMH is close to the truth whereas all other methods are unable to provide accurate density estimate given the time budget.



(a) True (b) TunaMH (c) TFMH (d) FlyMC

(e) PoissonMH (f) MH (g) SMH-1 (h) FlyMC-MAP

Figure B.2: Visualization of the density estimate after 1 second.

Logistic Regression on MNIST

MNIST with only 7s and 9s images contains 12214 training data and 2037 test data. Let h be the sigmoid function. Let the label $y_i \in \{0, 1\}$, then the model in logistic regression (LR) is

$$p(y_i = 1) = h(\theta^\intercal x_i) = \frac{1}{1 + \exp\left(-\theta^\intercal x_i\right)}.$$

It follows that

$$U_i(\theta) = -y_i \log h\left(\theta^\intercal x_i\right) - (1 - y_i) \log h\left(-\theta^\intercal x_i\right).$$

It is easy to see that

$$|\partial_j U_i| = |(h(\theta^\intercal x_i) - y_i) x_{ij}| \leq 1 \cdot |x_{ij}|.$$

Thus we can set $M(\theta, \theta')$ to be $\|\theta - \theta'\|_2$ and c_i to be $\|x_i\|_2$. We use this bound for TunaMH, TFMH and SMH. For FlyMC, we use the same bound on logistic regression as in the FlyMC paper [98].

We set the target acceptance rate to be 60% and the resulted stepsize is reported in Table B.1. We set $q_{d \to b}$ to be 0.1 following [98].

APPENDIX C

SECTION 4 AMORTIZED METROPOLIS ADJUSTMENT

C.1 Proofs

C.1.1 Proof of Results in Section 4.2

In this section, we will provide proofs of the results that we asserted in Section 4.2 about reversibility and skew-reversibility of our algorithms. First, for completeness we re-prove the fact that a skew-reversible chain has stationary distribution π, which is known, but not as well-known as the corresponding result for reversible chains.

Lemma 4. *If G is a skew-reversible chain, that is one that satisfies (4.4), then π is its stationary distribution.*

Proof. Since G is skew-reversibile, by definition it satisfies for any states x and y the conditions that $\pi(x) = \pi(x^{\perp})$ and

$$\pi(x)G(x,y) = \pi(y^{\perp})G(y^{\perp},x^{\perp}).$$

By combining these two we can easily get

$$\pi(x)G(x,y) = \pi(y)G(y^{\perp},x^{\perp}).$$

Next, summing up over all x in the while state space Ω,

$$\sum_{x\in\Omega}\pi(x)G(x,y) = \sum_{x\in\Omega}\pi(y)G(y^{\perp},x^{\perp}) = \pi(y)\sum_{x\in\Omega}G(y^{\perp},x^{\perp}).$$

Since $\perp$ denotes an involution, it follows that summing up over x for all x in the state space is equal to summing up over all $x^{\perp}$, so

$$\sum_{x \in \Omega} \pi(x) G(x, y) = \pi(y) \sum_{x \in \Omega} G(y^{\perp}, x) = \pi(y),$$

where the last equality follows from the fact that for any Markov chain, the sum of the probabilities of transitioning into all states is always 1. So, we've shown that

$$\sum_{x \in \Omega} \pi(x) G(x, y) = \pi(y)$$

which can be written in matrix form as $\pi G = \pi$; this proves the lemma. □

Lemma 5. *The amortized Metropolis-Hastings procedure described in Section 4.2 using acceptance probability (4.2) results in a chain that is reversible with stationary distribution π.*

Proof. According to the algorithm described in Section 4.2, the probability density of transitioning from state x to state $y \neq x$ via intermediate states $x = x_0, x_1, x_2, \ldots, x_{T-1}, x_T = y$ is

$$\mathbf{E}\left[\tau \cdot \prod_{t=0}^{T-1} P(x_t, x_{t+1}; \zeta_t)\right],$$

where the expected value here is taken over the randomness used to select the stochastic samples ζ_t. This follows from the law of total expectation. This means that the total probability of transitioning from x to $y \neq x$ is just the integral of this over the intermediate states

$$G(x, y) = \int \mathbf{E}\left[\tau \cdot \prod_{t=0}^{T-1} P(x_t, x_{t+1}; \zeta_t)\right] \, dx_1 \cdot dx_2 \cdots dx_{T-1},$$

Now substituting in the value of τ from (4.2) gives us

$$\begin{aligned}
&G(x,y) \\
&= \int \mathbf{E}\left[\min\left(1, \frac{\pi(y)}{\pi(x)}\prod_{t=0}^{T-1}\frac{P(x_{t+1},x_t;\zeta_t)}{P(x_t,x_{t+1};\zeta_t)}\right)\cdot\prod_{t=0}^{T-1}P(x_t,x_{t+1};\zeta_t)\right]\, dx_1\cdot dx_2\cdots dx_{T-1} \\
&= \int \mathbf{E}\left[\min\left(\prod_{t=0}^{T-1}P(x_t,x_{t+1};\zeta_t), \frac{\pi(y)}{\pi(x)}\prod_{t=0}^{T-1}P(x_{t+1},x_t;\zeta_t)\right)\right]\, dx_1\cdot dx_2\cdots dx_{T-1}.
\end{aligned}$$

Multiplying both sides by $\pi(x)$,

$$\begin{aligned}
&\pi(x)G(x,y) \\
&= \int \mathbf{E}\left[\min\left(\pi(x)\prod_{t=0}^{T-1}P(x_t,x_{t+1};\zeta_t), \pi(y)\prod_{t=0}^{T-1}P(x_{t+1},x_t;\zeta_t)\right)\right] \\
&\quad dx_1\cdot dx_2\cdots dx_{T-1}.
\end{aligned}$$

From here, the fact that G is reversible follows directly from a substitution of $x_t \mapsto x_{T-t}$ in the integral, combined with the observation that the ζ_t are i.i.d. and so exchangeable. □

Lemma 6. *The amortized Metropolis-Hastings procedure for skew-reversible chains described in Section 4.2 using acceptance probability (4.5) results in a chain that is skew-reversible with stationary distribution π, as long as π satisfies $\pi(x) = \pi(x^\perp)$.*

Proof. As above, the probability density of transitioning from state x to state $y \neq x$ via intermediate states $x = x_0, x_1, x_2, \ldots, x_{T-1}, x_T = y$ is

$$\mathbf{E}\left[\tau\cdot\prod_{t=0}^{T-1}P(x_t,x_{t+1};\zeta_t)\right],$$

where the expected value here is taken over the randomness used to select the stochastic samples ζ_t. This follows from the law of total expectation. This means that the total probability of transitioning from x to $y \neq x$ is just the integral of

this over the intermediate states

$$G(x,y) = \int \mathbf{E}\left[\tau \cdot \prod_{t=0}^{T-1} P(x_t, x_{t+1}; \zeta_t)\right] \, dx_1 \cdot dx_2 \cdots dx_{T-1},$$

Now substituting in the value of τ from (4.2) gives us

$$\begin{aligned}
&G(x,y) \\
&= \int \mathbf{E}\left[\min\left(1, \frac{\pi(y)}{\pi(x)} \prod_{t=0}^{T-1} \frac{P(x_{t+1}^{\perp}, x_t^{\perp}; \zeta_t)}{P(x_t, x_{t+1}; \zeta_t)}\right) \cdot \prod_{t=0}^{T-1} P(x_t, x_{t+1}; \zeta_t)\right] \\
&\quad dx_1 \cdot dx_2 \cdots dx_{T-1} \\
&= \int \mathbf{E}\left[\min\left(\prod_{t=0}^{T-1} P(x_t, x_{t+1}; \zeta_t), \frac{\pi(y)}{\pi(x)} \prod_{t=0}^{T-1} P(x_{t+1}^{\perp}, x_t^{\perp}; \zeta_t)\right)\right] \\
&\quad dx_1 \cdot dx_2 \cdots dx_{T-1}.
\end{aligned}$$

Multiplying both sides by $\pi(x)$, and leveraging the fact that $\pi(x) = \pi(x^{\perp})$,

$$\begin{aligned}
&\pi(x)G(x,y) \\
&= \int \mathbf{E}\left[\min\left(\pi(x) \prod_{t=0}^{T-1} P(x_t, x_{t+1}; \zeta_t), \pi(y^{\perp}) \prod_{t=0}^{T-1} P(x_{t+1}^{\perp}, x_t^{\perp}; \zeta_t)\right)\right] \\
&\quad dx_1 \cdot dx_2 \cdots dx_{T-1}.
\end{aligned}$$

From here, the fact that G is skew-reversible follows directly from a substitution of $x_t \mapsto x_{T-t}^{\perp}$ in the integral (which is a valid substitution without introducing an extra constant term because the involution $\perp$ is measure-preserving by assumption), combined with the observation that the ζ_t are i.i.d. and so exchangeable. □

Lemma 7. *A skew-reversible chain will become reversible by resampling the momentum at the beginning of outer loop.*

Proof. Assume the chain starts at (θ, r) and ends at (θ^*, r^*). By the skew-detailed balance, we have

$$\pi(\theta, r)G((\theta, r), (\theta^*, r^*)) = \pi(\theta^*, -r^*)G((\theta^*, -r^*), (\theta, -r))$$

Since the momentum is resampled and is independent of θ, we can integrate it and describe the chain in terms of θ

$$\pi(\theta)H(\theta,\theta^*) := \int \pi(\theta)\pi(r)G((\theta,r),(\theta^*,r^*))drdr^*$$

Similarly, we have

$$\pi(\theta^*)H(\theta^*,\theta) := \int \pi(\theta^*)\pi(-r^*)G((\theta^*,-r^*),(\theta,-r))dr^*dr$$

By the skew-detailed balance we know

$$\pi(\theta)H(\theta,\theta^*) = \pi(\theta^*)H(\theta^*,\theta)$$

This proves the lemma. □

C.1.2 Proof of Theorem 9

Proof. First, we consider resampling momentum in Algorithm 8 and will show that the chain is reversible. We consider the probability of starting from θ and going through a particular sequence of $r_{t+\frac{1}{2}}$ and θ_t and arriving at (θ^*, r^*). We have $G(\theta,\theta^*)$, which is the transition probability from θ to θ^*, as the following

$$G(\theta,\theta^*) = \mathbf{E}\int \mathbf{P}\left(\theta_0,\theta_1,\ldots,\theta_{T-1},\theta^*\middle|\theta,\tilde{U}_0,\ldots,\tilde{U}_{T-1}\right)\min(1,a(\boldsymbol{\theta}))d\theta_0\cdots d\theta_{T-1}$$

where $\boldsymbol{\theta} = \{\theta_0,\ldots,\theta_{T-1},\theta^*\}$ and the expectation is taken over the stochastic energy function samples $\tilde{U}_t$.

Next, we want to derive the probability density in terms of r and $\boldsymbol{\eta} = \{\eta_0,\ldots,\eta_{T-1}\}$. This involves a change of variables in the PDF formula. We notice that $\boldsymbol{\theta}$ is a bijective function of r and $\boldsymbol{\eta}$. By the rule of change of variables,

we know that

$$\begin{aligned}&\mathbf{P}\left(\theta_0, \theta_1, \ldots, \theta_{T-1}, \theta^* \middle| \theta, \tilde{U}_0, \ldots, \tilde{U}_{T-1}\right) \min(1, a(\boldsymbol{\theta}))\\ =\ &\mathbf{P}\left(r, \eta_0, \ldots, \eta_{T-1} \middle| \theta, \tilde{U}_0, \ldots, \tilde{U}_{T-1}\right) \min(1, a(\boldsymbol{\eta}, r)))det^{-1}(D_{(\boldsymbol{\eta},r)}(\boldsymbol{\theta}))\end{aligned}$$

where $D_{(\boldsymbol{\eta},r)}(\boldsymbol{\theta}, \theta^*)$ is the Jacobian matrix.

To get this Jacobian matrix, we first apply the chain rule,

$$D_{(\boldsymbol{\eta},r)}(\boldsymbol{\theta}) = D_{\boldsymbol{r}}(\boldsymbol{\theta}) \cdot D_{(\boldsymbol{\eta},r)}\boldsymbol{r}$$

where $\boldsymbol{r} = \{r, r, \ldots, r_{T-\frac{1}{2}}\}$.

Since the derivative of θ_t with respect to any $r_{s-\frac{1}{2}}$ for $s > t$ is zero, it follows that $D_{\boldsymbol{r}}(\boldsymbol{\theta})$ will be triangular, and so the determinant is just the product of the diagonal entries. From our formula for the update rule,

$$\begin{aligned}\theta_t &= \theta + \frac{1}{2}\epsilon\sigma^{-2}r_{t-\frac{1}{2}}, \text{ for } t = 0, T\\ \theta_t &= \theta_{t-1} + \epsilon\sigma^{-2}r_{-\frac{1}{2}}, \text{ for } t = 1, \ldots, T-1.\end{aligned}$$

Therefore,

$$\begin{aligned}\frac{\partial\theta_0}{\partial r_{t-\frac{1}{2}}} &= \frac{1}{2}\epsilon\sigma^{-2}I_d, \text{ for } t = 0, T\\ \frac{\partial\theta_t}{\partial r_{t-\frac{1}{2}}} &= \epsilon\sigma^{-2}I_d, \text{ for } t = 1, \ldots, T-1.\end{aligned}$$

It follows that

$$\det\left(D_{\boldsymbol{r}}(\boldsymbol{\theta})\right) = \frac{1}{4^d}\left(\epsilon\sigma^{-2}\right)^{(T+1)d}.$$

Similarly, the derivative of η_t with respect to any $r_{s-\frac{1}{2}}$ for $s > t$ is zero, it follows that $D_{(\boldsymbol{\eta},r)}\boldsymbol{r}$ will be triangular, and so the determinant is just the product of the diagonal entries. That is,

$$D_{\boldsymbol{r}}(\boldsymbol{\eta}, r)) = \prod_{t=0}^{T-1}\frac{\partial\eta_t}{\partial r_{t+\frac{1}{2}}}.$$

From our original formula for the update rule,

$$r_{t+\frac{1}{2}} = r_{t-\frac{1}{2}} - \epsilon\nabla\tilde{U}_t(\theta_t) - \epsilon\beta\left(r_{t-\frac{1}{2}} + r_{t+\frac{1}{2}}\right) + \eta_t$$

we have

$$(1+\epsilon\beta)r_{t+\frac{1}{2}} = r_{t-\frac{1}{2}} - \epsilon\nabla\tilde{U}_t(\theta_t) - \epsilon\beta r_{t-\frac{1}{2}} + \eta_t,$$

and so

$$\frac{\partial\eta_t}{\partial r_{t+\frac{1}{2}}} = (1+\epsilon\beta)I_d.$$

It follows that

$$\det\left(D_{(\boldsymbol{\eta},r)}\boldsymbol{r}\right) = (1+\epsilon\beta)^{-Td}$$

Now we can get that

$$\begin{aligned}\det\left(D_{(\boldsymbol{\eta},r)}(\boldsymbol{\theta})\right) &= \det\left(D_{\boldsymbol{r}}(\boldsymbol{\theta})\right)\cdot\det\left(D_{(\boldsymbol{\eta},r)}\boldsymbol{r}\right)\\ &= (1+\epsilon\beta)^{-Td}\cdot\frac{1}{4^d}\left(\epsilon\sigma^{-2}\right)^{(T+1)d}\end{aligned}$$

Thus,

$$\begin{aligned}G(\theta,\theta^*) &= \mathbf{E}\int\mathbf{P}\left(\theta_0,\theta_1,\ldots,\theta_{T-1},\theta^*\middle|\theta,\tilde{U}_0,\ldots,\tilde{U}_{T-1}\right)\min(1,a(\boldsymbol{\theta}))d\theta_0\cdots d\theta_{T-1}\\ &= (1+\epsilon\beta)^{Td}\cdot 4^d\left(\epsilon\sigma^{-2}\right)^{-(T+1)d}\mathbf{E}\int\mathbf{P}\left(r,\eta_0,\ldots,\eta_{T-1}\middle|\theta,\tilde{U}_0,\ldots,\tilde{U}_{T-1}\right)\\ &\qquad\min(1,a(\boldsymbol{\eta},r)))d\theta_0\cdots d\theta_{T-1}\end{aligned}$$

By the distribution of r and η_t, we know that

$$\begin{aligned}&\mathbf{P}\left(r,\eta_0,\eta_2,\ldots,\eta_{T-2}\middle|\theta,\tilde{U}_0,\ldots,\tilde{U}_{T-1}\right)\\ &= \left(2\pi\sigma^2\right)^{\frac{-d}{2}}\cdot\exp\left(-\frac{\|r\|^2}{2\sigma^2}\right)\cdot\prod_{t=0}^{T-1}\left(8\pi\epsilon\beta\sigma^2\right)^{\frac{-d}{2}}\cdot\exp\left(-\frac{\|\eta_t\|^2}{8\epsilon\beta\sigma^2}\right)\\ &= \left(2\pi\sigma^2\right)^{\frac{-d}{2}}\cdot\left(8\pi\epsilon\beta\sigma^2\right)^{\frac{-Td}{2}}\cdot\exp\left(-\frac{\|r\|^2}{2\sigma^2}\right)\cdot\exp\left(-\frac{1}{8\epsilon\beta\sigma^2}\sum_{t=0}^{T-1}\|\eta_t\|^2\right).\end{aligned}$$

Notice that

$$
\begin{aligned}
&\sum_{t=0}^{T-1} \|\eta_t\|^2 \\
&= \sum_{t=0}^{T-1} \left\| r_{t+\frac{1}{2}} - r_{t-\frac{1}{2}} + \epsilon \nabla \tilde{U}_t(\theta_t) + \epsilon\beta \left(r_{t-\frac{1}{2}} + r_{t+\frac{1}{2}} \right) \right\|^2 \\
&= \sum_{t=0}^{T-1} \left\| r_{t+\frac{1}{2}} - r_{t-\frac{1}{2}} + \epsilon \nabla \tilde{U}_t(\theta_t) \right\|^2 \\
&\quad + 2\epsilon\beta \left(r_{t+\frac{1}{2}} - r_{t-\frac{1}{2}} + \epsilon \nabla \tilde{U}_t(\theta_t) \right)^T \left(r_{t-\frac{1}{2}} + r_{t+\frac{1}{2}} \right) + \epsilon^2\beta^2 \left\| r_{t-\frac{1}{2}} + r_{t+\frac{1}{2}} \right\|^2 \\
&= \sum_{t=0}^{T-1} \left\| r_{t+\frac{1}{2}} - r_{t-\frac{1}{2}} + \epsilon \nabla \tilde{U}_t(\theta_t) \right\|^2 + 2\epsilon\beta \left(\left\| r_{t+\frac{1}{2}} \right\|^2 - \left\| r_{t-\frac{1}{2}} \right\|^2 \right) \\
&\quad + 2\epsilon^2\beta \nabla \tilde{U}_t(\theta_t)^T \left(r_{t-\frac{1}{2}} + r_{t+\frac{1}{2}} \right) + \epsilon^2\beta^2 \left\| r_{t-\frac{1}{2}} + r_{t+\frac{1}{2}} \right\|^2 \\
&= 2\epsilon\beta \left(\left\| r_{T-\frac{1}{2}} \right\|^2 - \|r\|^2 \right) + \sum_{t=0}^{T-1} \left\| r_{t+\frac{1}{2}} - r_{t-\frac{1}{2}} + \epsilon \nabla \tilde{U}_t(\theta_t) \right\|^2 \\
&\quad + 4\epsilon\beta\sigma^2 \left(\rho_{t+\frac{1}{2}} - \rho_{t-\frac{1}{2}} \right) + \epsilon^2\beta^2 \left\| r_{t-\frac{1}{2}} + r_{t+\frac{1}{2}} \right\|^2 \\
&= 2\epsilon\beta \left(\|r^*\|^2 - \|r\|^2 \right) + 4\epsilon\beta\sigma^2 \left(\rho_{T-\frac{1}{2}} - \rho_{-\frac{1}{2}} \right) \\
&\quad + \sum_{t=0}^{T-1} \left\| r_{t+\frac{1}{2}} - r_{t-\frac{1}{2}} + \epsilon \nabla \tilde{U}_t(\theta_t) \right\|^2 + \epsilon^2\beta^2 \left\| r_{t-\frac{1}{2}} + r_{t+\frac{1}{2}} \right\|^2 .
\end{aligned}
$$

By substituting this above and recalling that $\rho_{-\frac{1}{2}} = 0$,

$$
\begin{aligned}
& G(\theta, \theta^*) \\
&= (1+\epsilon\beta)^{Td} \cdot 4^d \left(\epsilon\sigma^{-2}\right)^{-(T+1)d} \mathbf{E} \int \mathbf{P}\left(r, \eta_0, \eta_2, \ldots, \eta_{T-2}, \theta^* \middle| \theta, \tilde{U}_0, \ldots, \tilde{U}_{T-1}\right) \\
& \quad \min(1, a(\boldsymbol{\eta}, r)) d\theta_0 \cdots d\theta_{T-1} \\
&= (1+\epsilon\beta)^{Td} \cdot 4^d \left(\epsilon\sigma^{-2}\right)^{-(T+1)d} \mathbf{E} \int \left(2\pi\sigma^2\right)^{\frac{-d}{2}} \cdot \left(8\pi\epsilon\beta\sigma^2\right)^{\frac{-Td}{2}} \cdot \exp\left(-\frac{\|r\|^2}{2\sigma^2}\right) \\
& \quad \cdot \exp\left(-\frac{1}{8\epsilon\beta\sigma^2} \cdot 2\epsilon\beta\left(\|r^*\|^2 - \|r\|^2\right)\right) \\
& \quad \cdot \exp\left(-\frac{1}{8\epsilon\beta\sigma^2} \cdot 4\epsilon\beta\sigma^2\left(\rho_{T-\frac{1}{2}} - \rho_{-\frac{1}{2}}\right)\right) \\
& \quad \cdot \exp\left(-\frac{1}{8\epsilon\beta\sigma^2} \cdot \sum_{t=0}^{T-1} \left\| r_{t+\frac{1}{2}} - r_{t-\frac{1}{2}} + \epsilon\nabla\tilde{U}_t(\theta_t)\right\|^2 + \epsilon^2\beta^2 \left\| r_{t-\frac{1}{2}} + r_{t+\frac{1}{2}}\right\|^2\right) \\
& \quad \cdot \min(1, a) d\theta_0 \cdots d\theta_{T-1} \\
&= (1+\epsilon\beta)^{Td} \cdot 4^d \left(\epsilon\sigma^{-2}\right)^{-(T+1)d} \mathbf{E} \int \left(2\pi\sigma^2\right)^{\frac{-d}{2}} \cdot \left(8\pi\epsilon\beta\sigma^2\right)^{\frac{-Td}{2}} \\
& \quad \cdot \exp\left(-\frac{1}{4\sigma^2}\left(\|r^*\|^2 + \|r\|^2\right)\right) \cdot \exp\left(-\frac{1}{2}\rho_{T-\frac{1}{2}}\right) \\
& \quad \cdot \exp\left(-\frac{1}{8\epsilon\beta\sigma^2} \cdot \sum_{t=0}^{T-1} \left\| r_{t+\frac{1}{2}} - r_{t-\frac{1}{2}} + \epsilon\nabla\tilde{U}_t(\theta_t)\right\|^2 + \epsilon^2\beta^2 \left\| r_{t-\frac{1}{2}} + r_{t+\frac{1}{2}}\right\|^2\right) \\
& \quad \cdot \min(1, a) d\theta_0 \cdots d\theta_{T-1}
\end{aligned}
$$

where $\boldsymbol{r}$ are to be understood as functions of the θ_t, and the integral is taken over θ_t.

Substituting the expression of a, then the term inside the integral is

$$
\begin{aligned}
&\left(2\pi\sigma^2\right)^{\frac{-d}{2}} \cdot \left(8\pi\epsilon\beta\sigma^2\right)^{\frac{-Td}{2}} \cdot \exp\left(-\frac{1}{4\sigma^2}\left(\|r^*\|^2 + \|r\|^2\right)\right) \\
&\quad \cdot \exp\left(-\frac{1}{2}\rho_{T-\frac{1}{2}}\right) \\
&\quad \cdot \exp\left(-\frac{1}{8\epsilon\beta\sigma^2} \cdot \sum_{t=0}^{T-1} \left\| r_{t+\frac{1}{2}} - r_{t-\frac{1}{2}} + \epsilon\nabla\tilde{U}_t(\theta_t)\right\|^2 + \epsilon^2\beta^2 \left\| r_{t-\frac{1}{2}} + r_{t+\frac{1}{2}}\right\|^2\right) \\
&\quad \cdot \min\left(1, \exp\left(U(\theta) - U(\theta^*) + \rho_{T-\frac{1}{2}}\right)\right) \\
&= \left(2\pi\sigma^2\right)^{\frac{-d}{2}} \cdot \left(8\pi\epsilon\beta\sigma^2\right)^{\frac{-Td}{2}} \cdot \exp\left(-\frac{1}{4\sigma^2}\left(\|r^*\|^2 + \|r\|^2\right)\right) \\
&\quad \cdot \exp\left(-\frac{1}{8\epsilon\beta\sigma^2} \cdot \sum_{t=0}^{T-1} \left\| r_{t+\frac{1}{2}} - r_{t-\frac{1}{2}} + \epsilon\nabla\tilde{U}_t(\theta_t)\right\|^2 + \epsilon^2\beta^2 \left\| r_{t-\frac{1}{2}} + r_{t+\frac{1}{2}}\right\|^2\right) \\
&\quad \cdot \exp\left(U(\theta)\right) \cdot \min\left(\exp\left(-U(\theta) - \frac{1}{2}\rho_{T-\frac{1}{2}}\right), \exp\left(-U(\theta^*) + \frac{1}{2}\rho_{T-\frac{1}{2}}\right)\right).
\end{aligned}
$$

Finally, this probability multiplied by the probability of θ_0, which is $\frac{1}{Z}\exp\left(-U(\theta)\right)$,

is

$$
\begin{aligned}
&\pi(\theta)G(\theta,\theta^*)\\
&= \frac{1}{Z}\cdot(1+\epsilon\beta)^{Td}\cdot 4^d\left(\epsilon\sigma^{-2}\right)^{-(T+1)d}\\
&\quad \mathbf{E}\int \left(2\pi\sigma^2\right)^{\frac{-d}{2}}\cdot\left(8\pi\epsilon\beta\sigma^2\right)^{\frac{-Td}{2}}\cdot\exp\left(-\frac{1}{4\sigma^2}\left(\|r^*\|^2+\|r\|^2\right)\right)\\
&\quad \cdot\exp\left(-\frac{1}{8\epsilon\beta\sigma^2}\cdot\sum_{t=0}^{T-1}\left\|r_{t+\frac{1}{2}}-r_{t-\frac{1}{2}}+\epsilon\nabla\tilde{U}_t(\theta_t)\right\|^2+\epsilon^2\beta^2\left\|r_{t-\frac{1}{2}}+r_{t+\frac{1}{2}}\right\|^2\right)\\
&\quad \cdot\exp\left(U(\theta)\right)\cdot\min\left(\exp\left(-U(\theta)-\frac{1}{2}\rho_{T-\frac{1}{2}}\right),\exp\left(-U(\theta^*)+\frac{1}{2}\rho_{T-\frac{1}{2}}\right)\right)\\
&\quad d\theta_0\cdots d\theta_{T-1}\\
&= \frac{1}{Z}\cdot(1+\epsilon\beta)^{Td}\cdot 4^d\left(\epsilon\sigma^{-2}\right)^{-(T+1)d}\cdot\left(2\pi\sigma^2\right)^{\frac{-d}{2}}\cdot\left(8\pi\epsilon\beta\sigma^2\right)^{\frac{-Td}{2}}\cdot\\
&\quad \mathbf{E}\int\exp\left(-\frac{1}{4\sigma^2}\left(\|r^*\|^2+\|r\|^2\right)\right)\\
&\quad \cdot\exp\left(-\frac{1}{8\epsilon\beta\sigma^2}\cdot\sum_{t=0}^{T-1}\left(\left\|r_{t+\frac{1}{2}}-r_{t-\frac{1}{2}}+\epsilon\nabla\tilde{U}_t(\theta_t)\right\|^2+\epsilon^2\beta^2\left\|r_{t-\frac{1}{2}}+r_{t+\frac{1}{2}}\right\|^2\right)\right)\\
&\quad \cdot\min\left(\exp\left(-U(\theta)-\frac{1}{2}\rho_{T-\frac{1}{2}}\right),\exp\left(-U(\theta^*)+\frac{1}{2}\rho_{T-\frac{1}{2}}\right)\right)d\theta_0\cdots d\theta_{T-1}\\
&= \frac{1}{Z}\cdot(1+\epsilon\beta)^{Td}\cdot\beta^{-\frac{Td}{2}}\cdot 2^{-\frac{3(T-1)d}{2}}\cdot\pi^{-\frac{(T+1)d}{2}}\cdot\epsilon^{-\frac{(3T+2)d}{2}}\cdot\sigma^{(T+1)d}\\
&\quad \cdot\mathbf{E}\int\exp\left(-\frac{1}{4\sigma^2}\left(\|r^*\|^2+\|r\|^2\right)\right)\\
&\quad \cdot\exp\left(-\frac{1}{8\epsilon\beta\sigma^2}\cdot\sum_{t=0}^{T-1}\left(\left\|r_{t+\frac{1}{2}}-r_{t-\frac{1}{2}}+\epsilon\nabla\tilde{U}_t(\theta_t)\right\|^2+\epsilon^2\beta^2\left\|r_{t-\frac{1}{2}}+r_{t+\frac{1}{2}}\right\|^2\right)\right)\\
&\quad \cdot\exp\left(-\frac{U(\theta)+U(\theta^*)}{2}\right)\cdot\exp\left(-\frac{1}{2}\left|U(\theta)-U(\theta^*)+\rho_{T-\frac{1}{2}}\right|\right)d\theta_0\cdots d\theta_{T-1}.
\end{aligned}
$$

And writing this out explicitly in terms of

$$\rho_{T-\frac{1}{2}}=\frac{1}{2}\epsilon\sigma^{-2}\sum_{t=0}^{T-1}\nabla\tilde{U}_t(\theta_t)^T\left(r_{t-\frac{1}{2}}+r_{t+\frac{1}{2}}\right),$$

we get

$$
\begin{aligned}
&\pi(\theta)G(\theta,\theta^*)\\
&=\frac{1}{Z}\cdot(1+\epsilon\beta)^{Td}\cdot\beta^{-\frac{Td}{2}}\cdot 2^{-\frac{3(T-1)d}{2}}\cdot\pi^{-\frac{(T+1)d}{2}}\cdot\epsilon^{-\frac{(3T+2)d}{2}}\cdot\sigma^{(T+1)d}\\
&\quad\cdot\mathbf{E}\int\exp\left(-\frac{1}{4\sigma^2}\left(\|r^*\|^2+\|r\|^2\right)\right)\\
&\quad\cdot\exp\left(-\frac{1}{8\epsilon\beta\sigma^2}\cdot\sum_{t=0}^{T-1}\left(\left\|r_{t+\frac{1}{2}}-r_{t-\frac{1}{2}}+\epsilon\nabla\tilde{U}_t(\theta_t)\right\|^2+\epsilon^2\beta^2\left\|r_{t-\frac{1}{2}}+r_{t+\frac{1}{2}}\right\|^2\right)\right)\\
&\quad\cdot\exp\left(-\frac{U(\theta)+U(\theta^*)}{2}\right)\\
&\quad\cdot\exp\left(-\frac{1}{2}\left|U(\theta)-U(\theta^*)+\frac{1}{2}\epsilon\sigma^{-2}\sum_{t=0}^{T-1}\nabla\tilde{U}_t(\theta_t)^T\left(r_{t-\frac{1}{2}}+r_{t+\frac{1}{2}}\right)\right|\right)\\
&\quad d\theta_0\cdots d\theta_{T-1}.
\end{aligned}
$$

Now, for this forward path from θ to θ^*, consider the reverse leapfrog trajectory from θ^* to θ. This trajectory will have the same values for $\theta,\theta_0,\ldots,\theta^*$ in the reversed order and will have negated values for $r_{-\frac{1}{2}},r_{1-\frac{1}{2}},\ldots,r_{T-\frac{1}{2}}$ in the reversed order again. Because of this negation, the values of $\rho_{\frac{1}{2}},\rho_{1+\frac{1}{2}},\ldots,\rho_{T-\frac{1}{2}}$ will also be negated. It follows that $\pi(\theta^*)G(\theta^*,\theta)$ will have the same expression.

Therefore,

$$\pi(\theta)G(\theta,\theta^*)=\pi(\theta^*)G(\theta^*,\theta).$$

This shows that Algorithm 8 with resampling momentum is reversible.

Now we show that the chain satisfies *skew detailed balance* and the stationary distribution of θ is $\pi(\theta)$ if not resampling momentum. Skew detailed balance means that the chain satisfies the following condition [139]

$$\pi(x)G(x,y)=\pi\left(y^\perp\right)G\left(y^\perp,x^\perp\right)$$

where G is the transition probability.

By Section C.1.1, we know that a chain that satisfies the above condition will have invariant distribution $\pi(x)$.

In our setting, $x = (\theta, r)$ and $x^{\perp} = (\theta, -r)$. Given this, the skew detailed balance is

$$\pi(\theta, r)G((\theta, r), (\theta^*, r^*)) = \pi(\theta^*, -r^*)G((\theta^*, -r^*), (\theta, -r)).$$

Next we will show that Algorithm 8 without resampling momentum satisfies the above condition and it naturally follows that Algorithm 8 without resampling converges to the desired distribution.

We consider the joint distribution of (θ, r). By a similar analysis of resampling case, we can get that

$$\begin{aligned}
&\pi(\theta, r) \cdot G((\theta, r), (\theta^*, r^*)) \\
&= \frac{1}{Z} \cdot (1 + \epsilon\beta)^{Td} \cdot \beta^{-\frac{Td}{2}} \cdot 2^{-\frac{3(T-1)d}{2}} \cdot \pi^{-\frac{(T+1)d}{2}} \cdot \epsilon^{-\frac{(3T+2)d}{2}} \cdot \sigma^{(T+1)d} \\
&\quad \cdot \mathbf{E} \int \exp\left(-\frac{1}{4\sigma^2}\left(\|r^*\|^2 + \|r\|^2\right)\right) \\
&\quad \cdot \exp\left(-\frac{1}{8\epsilon\beta\sigma^2} \cdot \sum_{t=0}^{T-1}\left(\left\|r_{t+\frac{1}{2}} - r_{t-\frac{1}{2}} + \epsilon\nabla\tilde{U}_t(\theta_t)\right\|^2 + \epsilon^2\beta^2\left\|r_{t-\frac{1}{2}} + r_{t+\frac{1}{2}}\right\|^2\right)\right) \\
&\quad \cdot \exp\left(-\frac{U(\theta) + U(\theta^*)}{2}\right) \\
&\quad \cdot \exp\left(-\frac{1}{2}\left|U(\theta) - U(\theta^*) + \frac{1}{2}\epsilon\sigma^{-2}\sum_{t=0}^{T-1}\nabla\tilde{U}_t(\theta_t)^T\left(r_{t-\frac{1}{2}} + r_{t+\frac{1}{2}}\right)\right|\right) \\
&\quad d\theta_0 \cdots d\theta_{T-1}.
\end{aligned}$$

Again, the reverse trajectory will have the same values for $\theta, \theta_0, \ldots, \theta^*$ and will have negated values for $r_{-\frac{1}{2}}, r_{1-\frac{1}{2}}, \ldots, r_{T-\frac{1}{2}}$ in the reversed order. Therefore, $\pi(\theta^*, -r^*)G((\theta^*, -r^*), (\theta, -r))$ will have the same expression.

It follows that

$$\pi(\theta, r)G((\theta, r), (\theta^*, r^*)) = \pi(\theta^*, -r^*)G((\theta^*, -r^*), (\theta, -r))$$

which is what we want. □

C.1.3 Proof of Theorem 10

In this section we prove a bound on the convergence rate of AMAGOLD as compared with second-order Langevin dynamics (L2MC).

Proof. We start with the expression we derived for the transition probability in the proof of reversibility.

$$\begin{aligned}
&\pi(\theta)G(\theta, \theta^*) \\
&= \frac{1}{Z} \cdot (1+\epsilon\beta)^{Td} \cdot \beta^{-\frac{Td}{2}} \cdot 2^{-\frac{(3T-1)d}{2}} \cdot \pi^{-\frac{(T+1)d}{2}} \cdot \epsilon^{-\frac{(3T+2)d}{2}} \cdot \sigma^{(T+1)d} \\
&\quad \cdot \mathbf{E} \int \exp\left(-\frac{1}{4\sigma^2}\left(\|r^*\|^2 + \|r\|^2\right)\right) \\
&\quad \cdot \exp\left(-\frac{1}{8\epsilon\beta\sigma^2} \cdot \sum_{t=0}^{T-1}\left(\left\|r_{t+\frac{1}{2}} - r_{t-\frac{1}{2}} + \epsilon\nabla\tilde{U}_t(\theta_t)\right\|^2 + c^2\epsilon^2\beta^2\left\|r_{t-\frac{1}{2}} + r_{t+\frac{1}{2}}\right\|^2\right)\right) \\
&\quad \cdot \exp\left(-\frac{U(\theta) + U(\theta^*)}{2}\right) \\
&\quad \cdot \exp\left(-\frac{1}{2}\left|U(\theta) - U(\theta^*) + \frac{1}{2}\epsilon\sigma^{-2}\sum_{t=0}^{T-1}\nabla\tilde{U}_t(\theta_t)^T\left(r_{t-\frac{1}{2}} + r_{t+\frac{1}{2}}\right)\right|\right) \\
&\quad d\theta_0 \cdots d\theta_{T-1}.
\end{aligned}$$

Since

$$\theta_t = \theta_{t-1} + \epsilon\sigma^{-2}r_{t-\frac{1}{2}},$$

if we define θ_{-1} and θ_T by convention such that

$$\frac{\theta_{-1} + \theta_0}{2} = \theta \qquad \text{and} \qquad \frac{\theta_{T-1} + \theta_T}{2} = \theta^*,$$

it follows that for all $t \in \{0, \ldots, T-1\}$

$$r_{t+\frac{1}{2}} - r_{t-\frac{1}{2}} = \epsilon^{-1}\sigma^2 \left(\theta_{t+1} - 2\theta_t + \theta_{t-1}\right)$$

and

$$r_{t+\frac{1}{2}} + r_{t-\frac{1}{2}}\epsilon^{-1}\sigma^2 \left(\theta_{t+1} - \theta_{t-1}\right)$$

so we can write the above transition probability explicitly in terms of the θ_t as

$$\begin{aligned}
&\pi(\theta)G(\theta, \theta^*) \\
&= \frac{1}{Z} \cdot (1+\epsilon\beta)^{Td} \cdot \beta^{-\frac{Td}{2}} \cdot 2^{-\frac{(3T-1)d}{2}} \cdot \pi^{-\frac{(T+1)d}{2}} \cdot \epsilon^{-\frac{(3T+2)d}{2}} \cdot \sigma^{(T+1)d} \\
&\quad \cdot \mathbf{E} \int \exp\left(-\frac{c}{4\sigma^2}\left(\|r^*\|^2 + \|r\|^2\right)\right) \\
&\quad \cdot \exp\left(-\frac{1}{8\epsilon\beta\sigma^2} \cdot \sum_{t=0}^{T-1} \left(\left\| \epsilon^{-1}\sigma^2\left(\theta_{t+1} - 2\theta_t + \theta_{t-1}\right) + \epsilon\nabla\tilde{U}_t(\theta_t) \right\|^2 \right.\right. \\
&\quad \left.\left. + c^2\epsilon^2\beta^2 \left\| \epsilon^{-1}\sigma^2\left(\theta_{t+1} - \theta_{t-1}\right) \right\|^2 \right) \right) \cdot \exp\left(-\frac{U(\theta) + U(\theta^*)}{2} \right) \\
&\quad \cdot \exp\left(-\frac{1}{2} \left| U(\theta) - U(\theta^*) + \frac{1}{2}\epsilon\sigma^{-2} \sum_{t=0}^{T-1} \nabla\tilde{U}(\theta_t)^T \left(\epsilon^{-1}\sigma^2\left(\theta_{t+1} - \theta_{t-1}\right)\right) \right| \right) \\
&\quad \cdot d\theta_0 \cdots d\theta_{T-1}.
\end{aligned}$$

Simplifying this a bit, we get

$$
\begin{aligned}
&\pi(\theta)G(\theta,\theta^*)\\
&= \frac{1}{Z}\cdot(1+\epsilon\beta)^{Td}\cdot\beta^{-\frac{Td}{2}}\cdot 2^{-\frac{(3T-1)d}{2}}\cdot\pi^{-\frac{(T+1)d}{2}}\cdot\epsilon^{-\frac{(3T+2)d}{2}}\cdot\sigma^{(T+1)d}\\
&\quad\cdot\mathbf{E}\int\exp\left(-\frac{\sigma^2}{\epsilon^2}\left(\|\theta^*-\theta_{T-1}\|^2+\|\theta_0-\theta\|^2\right)\right)\\
&\quad\cdot\exp\left(-\frac{\sigma^2}{8\epsilon^3\beta}\cdot\sum_{t=0}^{T-1}\left\|\theta_{t+1}-2\theta_t+\theta_{t-1}+\epsilon^2\sigma^{-2}\nabla\tilde{U}_t(\theta_t)\right\|^2\right)\\
&\quad\cdot\exp\left(-\frac{\beta\sigma^2}{8\epsilon}\cdot\sum_{t=0}^{T-1}\|\theta_{t+1}-\theta_{t-1}\|^2\right)\\
&\quad\cdot\exp\left(-\frac{U(\theta)+U(\theta^*)}{2}\right)\\
&\quad\cdot\exp\left(-\frac{1}{2}\left|U(\theta)-U(\theta^*)+\frac{1}{2}\sum_{t=0}^{T-1}\nabla\tilde{U}_t(\theta_t)^T\left(\theta_{t+1}-\theta_{t-1}\right)\right|\right)\,d\theta_0\cdots d\theta_{T-1}.
\end{aligned}
$$

Next, let

$$
\begin{aligned}
N_t &= \nabla\tilde{U}_t(\theta_t)-\nabla U_t(\theta_t),\\
A_t &= \theta_{t+1}-2\theta_t+\theta_{t-1}+\epsilon^2\sigma^{-2}\nabla U_t(\theta_t),\\
B_t &= \theta_{t+1}-\theta_{t-1},\\
C_t &= U(\theta)-U(\theta^*)+\frac{1}{2}\sum_{t=0}^{T-1}\nabla U_t(\theta_t)^T\left(\theta_{t+1}-\theta_{t-1}\right).
\end{aligned}
$$

Notice that only N_t depends on the randomness of the stochastic gradient samples.

Then,

$$
\begin{aligned}
&\pi(\theta)G(\theta,\theta^*)\\
&= \frac{1}{Z}\cdot(1+\epsilon\beta)^{Td}\cdot\beta^{-\frac{Td}{2}}\cdot 2^{-\frac{(3T-1)d}{2}}\cdot\pi^{-\frac{(T+1)d}{2}}\cdot\epsilon^{-\frac{(3T+2)d}{2}}\cdot\sigma^{(T+1)d}\\
&\quad\cdot\int\mathbf{E}\Bigg[\exp\left(-\frac{\sigma^2}{\epsilon^2}\left(\|\theta^*-\theta_{T-1}\|^2+\|\theta_0-\theta\|^2\right)\right)\\
&\quad\cdot\exp\left(-\frac{\sigma^2}{8\epsilon^3\beta}\cdot\sum_{t=0}^{T-1}\left(\|A_t\|^2+2\epsilon^2\sigma^{-2}A_t^TN_t+\epsilon^4\sigma^{-4}\|N_t\|^2\right)\right)\\
&\quad\cdot\exp\left(-\frac{\beta\sigma^2}{8\epsilon}\cdot\sum_{t=0}^{T-1}\|B_t\|^2\right)\\
&\quad\cdot\exp\left(-\frac{U(\theta)+U(\theta^*)}{2}\right)\\
&\quad\cdot\exp\left(-\frac{1}{2}\left|C_t+\frac{1}{2}\sum_{t=0}^{T-1}N_t^TB_t\right|\right)\Bigg]\,d\theta_0\cdots d\theta_{T-1}.
\end{aligned}
$$

Now, for any constant $c>1$, we can bound

$$
\begin{aligned}
\mathbf{E}\left[\left|\sum_{t=0}^{T-1}N_t^TB_t\right|\right] &\le \sqrt{\mathbf{E}\left[\left(\sum_{t=0}^{T-1}N_t^TB_t\right)^2\right]}\\
&= \sqrt{\sum_{t=0}^{T-1}B_t^T\mathbf{E}\left[N_tN_t^T\right]B_t}\\
&\le \sqrt{\sum_{t=0}^{T-1}\frac{V^2}{d}\|B_t\|^2}\\
&\le \frac{V^2}{d}\frac{\epsilon}{2(c-1)\beta\sigma^2}+(c-1)\frac{\beta\sigma^2}{2\epsilon}\sum_{t=0}^{T-1}\|B_t\|^2.
\end{aligned}
$$

Additionally, we know that $\mathbf{E}\left[N_t\right]=0$ and

$$
\mathbf{E}\left[\|N_t\|^2\right]=\mathbf{E}\left[\mathbf{tr}\left(N_tN_t^T\right)\right]\le\mathbf{tr}\left(\frac{V^2}{d}I\right)=V^2.
$$

So, since by Jensen's inequality, $\mathbf{E}\left[\exp(X)\right] \geq \exp(\mathbf{E}\left[X\right])$, we can bound this with

$$\begin{aligned}
&\pi(\theta)G(\theta,\theta^*) \\
&\geq \frac{1}{Z}\cdot(1+\epsilon\beta)^{Td}\cdot\beta^{-\frac{Td}{2}}\cdot 2^{-\frac{(3T-1)d}{2}}\cdot\pi^{-\frac{(T+1)d}{2}}\cdot\epsilon^{-\frac{(3T+2)d}{2}}\cdot\sigma^{(T+1)d} \\
&\quad\cdot\int\exp\left(-\frac{\sigma^2}{\epsilon^2}\left(\|\theta^*-\theta_{T-1}\|^2+\|\theta_0-\theta\|^2\right)\right) \\
&\quad\cdot\exp\left(-\frac{\sigma^2}{8\epsilon^3\beta}\cdot\sum_{t=0}^{T-1}\|A_t\|^2\right)\cdot\exp\left(-\frac{\epsilon TV^2}{8\sigma^2\beta}\right) \\
&\quad\cdot\exp\left(-\frac{c\sigma^2\beta}{8\epsilon}\cdot\sum_{t=0}^{T-1}\|B_t\|^2\right) \\
&\quad\cdot\exp\left(-\frac{U(\theta)+U(\theta^*)}{2}\right) \\
&\quad\cdot\exp\left(-\frac{1}{2}\left|C_t\right|\right)\cdot\exp\left(-\frac{1}{(c-1)Td}\cdot\frac{\epsilon TV^2}{8\sigma^2\beta}\right)\,d\theta_0\cdots d\theta_{T-1}.
\end{aligned}$$

Now, this is a lower bound on the AMAGOLD chain with parameters (ϵ,σ,β). Next, we consider the transition probability of a *rescaled* chain, with slightly different parameters, that will be set as a function of c. Specifically, consider the chain with parameters $(\epsilon,\sigma\cdot c^{-1/4},\beta\cdot c^{-1/2})$. (We will set the parameter c later; at this point in the proof it is just an arbitrary constant $c>1$.) If we call this rescaled chain

G_r, then by substitution of the parameters into the above expression, we get

$$
\begin{aligned}
&\pi(\theta)G_r(\theta,\theta^*) \\
&\geq \frac{1}{Z}\cdot\left(1+c^{-1/2}\epsilon\beta\right)^{Td}\cdot\beta^{-\frac{Td}{2}}\cdot 2^{-\frac{3(T-1)d}{2}}\cdot\pi^{-\frac{(T+1)d}{2}}\cdot c^{-\frac{d}{4}}\cdot\epsilon^{-\frac{(3T+2)d}{2}}\cdot\sigma^{(T+1)d} \\
&\quad\cdot\int\mathbf{E}\exp\left(-\frac{\sigma^2}{c^{1/2}\epsilon^2}\left(\|\theta^*-\theta_{T-1}\|^2+\|\theta_0-\theta\|^2\right)\right) \\
&\quad\cdot\exp\left(-\frac{\sigma^2}{8\epsilon^3\beta}\cdot\sum_{t=0}^{T-1}\|A_t\|^2\right)\cdot\exp\left(-\frac{c\epsilon TV^2}{8\sigma^2\beta}\right) \\
&\quad\cdot\exp\left(-\frac{\sigma^2\beta}{8\epsilon}\cdot\sum_{t=0}^{T-1}\|B_t\|^2\right) \\
&\quad\cdot\exp\left(-\frac{U(\theta)+U(\theta^*)}{2}\right) \\
&\quad\cdot\exp\left(-\frac{1}{2}\left|C_t\right|\right)\cdot\exp\left(-\frac{1}{(c-1)Td}\cdot\frac{c\epsilon TV^2}{8\sigma^2\beta}\right)\,d\theta_0\cdots d\theta_{T-1} \\
&\geq \frac{1}{Z}\cdot\left(1+c^{-1/2}\epsilon\beta\right)^{Td}\cdot\beta^{-\frac{Td}{2}}\cdot 2^{-\frac{3(T-1)d}{2}}\cdot\pi^{-\frac{(T+1)d}{2}}\cdot c^{-\frac{d}{4}}\cdot\epsilon^{-\frac{(3T+2)d}{2}}\cdot\sigma^{(T+1)d} \\
&\quad\cdot\exp\left(-\frac{c\epsilon TV^2}{8\sigma^2\beta}\right)\cdot\exp\left(-\frac{1}{(c-1)Td}\cdot\frac{c\epsilon TV^2}{8\sigma^2\beta}\right) \\
&\quad\cdot\int\mathbf{E}\exp\left(-\frac{\sigma^2}{\epsilon^2}\left(\|\theta^*-\theta_{T-1}\|^2+\|\theta_0-\theta\|^2\right)\right) \\
&\quad\cdot\exp\left(-\frac{\sigma^2}{8\epsilon^3\beta}\cdot\sum_{t=0}^{T-1}\|A_t\|^2\right) \\
&\quad\cdot\exp\left(-\frac{\sigma^2\beta}{8\epsilon}\cdot\sum_{t=0}^{T-1}\|B_t\|^2\right) \\
&\quad\cdot\exp\left(-\frac{U(\theta)+U(\theta^*)}{2}\right) \\
&\quad\cdot\exp\left(-\frac{1}{2}\left|C_t\right|\right)\,d\theta_0\cdots d\theta_{T-1}.
\end{aligned}
$$

On the other hand, consider the transition probability of the full-gradient L2MC chain with parameters (ϵ,σ,β). This chain will be the same as the AMAGOLD chain, except that $N_t=0$ always. So, if we call this chain's transition probability

$\bar{G}$, we will have

$$\begin{aligned}
&\pi(\theta)\bar{G}(\theta,\theta^*) \\
&= \frac{1}{Z}\cdot(1+\epsilon\beta)^{Td}\cdot\beta^{-\frac{Td}{2}}\cdot 2^{-\frac{(3T-1)d}{2}}\cdot\pi^{-\frac{(T+1)d}{2}}\cdot\epsilon^{-\frac{(3T+2)d}{2}}\cdot\sigma^{(T+1)d} \\
&\quad\cdot\int\exp\left(-\frac{\sigma^2}{\epsilon^2}\left(\|\theta^*-\theta_{T-1}\|^2+\|\theta_0-\theta\|^2\right)\right) \\
&\quad\cdot\exp\left(-\frac{\sigma^2}{8\epsilon^3\beta}\cdot\sum_{t=0}^{T-1}\|A_t\|^2\right) \\
&\quad\cdot\exp\left(-\frac{\beta\sigma^2}{8\epsilon}\cdot\sum_{t=0}^{T-1}\|B_t\|^2\right) \\
&\quad\cdot\exp\left(-\frac{U(\theta)+U(\theta^*)}{2}\right) \\
&\quad\cdot\exp\left(-\frac{1}{2}|C_t|\right)\,d\theta_0\cdots d\theta_{T-1}.
\end{aligned}$$

Using this, we can simplify our bound on the transition probability of the AM-AGOLD chain to

$$\begin{aligned}
\pi(\theta)G_r(\theta,\theta^*) \geq &\left(\frac{1+c^{-1/2}\epsilon\beta}{1+\epsilon\beta}\right)^{Td}\cdot c^{-\frac{d}{4}}\cdot\exp\left(-\frac{c\epsilon TV^2}{8\sigma^2\beta}\right) \\
&\cdot\exp\left(-\frac{1}{(c-1)Td}\cdot\frac{c\epsilon TV^2}{8\sigma^2\beta}\right)\cdot\pi(\theta)\bar{G}(\theta,\theta^*).
\end{aligned}$$

Thus,

$$\begin{aligned}
\frac{\pi(\theta)G_r(\theta,\theta^*)}{\pi(\theta)\bar{G}(\theta,\theta^*)} \geq &\left(\frac{1+c^{-1/2}\epsilon\beta}{1+\epsilon\beta}\right)^{Td}\cdot c^{-\frac{d}{4}}\cdot\exp\left(-\frac{c\epsilon TV^2}{8\sigma^2\beta}\right) \\
&\cdot\exp\left(-\frac{1}{(c-1)Td}\cdot\frac{c\epsilon TV^2}{8\sigma^2\beta}\right).
\end{aligned}$$

All that remains to get a bound is to set c appropriately. Since

$$c^{-\frac{d}{4}} = \exp\left(-\frac{d}{4}\log(c)\right) \geq \exp\left(-\frac{d}{4}(c-1)\right),$$

and

$$c^{-1/2} \geq 1-\frac{c-1}{2},$$

we can bound this with

$$\begin{aligned}
&\frac{\pi(\theta)G_r(\theta,\theta^*)}{\pi(\theta)\bar{G}(\theta,\theta^*)} \\
&\geq \left(\frac{1+\left(1-\frac{c-1}{2}\right)\epsilon\beta}{1+\epsilon\beta}\right)^{Td} \cdot \exp\left(-\frac{d}{4}(c-1) - \frac{c\epsilon TV^2}{8\sigma^2\beta} - \frac{1}{(c-1)Td}\cdot\frac{c\epsilon TV^2}{8\sigma^2\beta}\right) \\
&\geq \left(1-\frac{(c-1)\epsilon\beta}{2(1+\epsilon\beta)}\right)^{Td} \cdot \exp\left(-\frac{d}{4}(c-1) - \frac{c\epsilon TV^2}{8\sigma^2\beta} - \frac{1}{(c-1)Td}\cdot\frac{c\epsilon TV^2}{8\sigma^2\beta}\right).
\end{aligned}$$

Since for any $0 \leq x < 1/2$, it holds that $1 - x \geq \exp(-2x)$, as long as

$$\frac{(c-1)\epsilon\beta}{1+\epsilon\beta} \leq 1,$$

it holds that

$$1 - \frac{(c-1)\epsilon\beta}{2(1+\epsilon\beta)} \geq \exp\left(-\frac{(c-1)\epsilon\beta}{1+\epsilon\beta}\right).$$

So, under this assumption,

$$\begin{aligned}
&\frac{\pi(\theta)G_r(\theta,\theta^*)}{\pi(\theta)\bar{G}(\theta,\theta^*)} \\
&\geq \exp\left(-\frac{(c-1)\epsilon\beta Td}{1+\epsilon\beta} - \frac{d}{4}(c-1) - \frac{c\epsilon TV^2}{8\sigma^2\beta} - \frac{1}{(c-1)Td}\cdot\frac{c\epsilon TV^2}{8\sigma^2\beta}\right) \\
&= \exp\left(-\frac{\epsilon TV^2}{8\sigma^2\beta} - \frac{\epsilon V^2}{8\sigma^2\beta d}\right) \\
&\qquad \cdot \exp\left(-(c-1)\left(\frac{(1+\epsilon\beta(1+4T))d}{4(1+\epsilon\beta)} + \frac{\epsilon TV^2}{8\sigma^2\beta}\right) - \frac{\epsilon V^2}{8(c-1)\sigma^2\beta d}\right) \\
&= \exp\left(-\frac{\epsilon TV^2}{8\sigma^2\beta} - \frac{\epsilon V^2}{8\sigma^2\beta d}\right) \\
&\qquad \cdot \exp\left(-(c-1)\left(\frac{3}{2}Td + \frac{\epsilon TV^2}{8\sigma^2\beta}\right) - \frac{\epsilon V^2}{8(c-1)\sigma^2\beta d}\right).
\end{aligned}$$

If we also assume that

$$\frac{\epsilon V^2}{4\sigma^2\beta d} \leq 1,$$

then

$$\frac{\epsilon TV^2}{8\sigma^2\beta} \leq \frac{Td}{2},$$

and so

$$\frac{\pi(\theta)G_r(\theta,\theta^*)}{\pi(\theta)\bar{G}(\theta,\theta^*)} \geq \exp\left(-\frac{\epsilon TV^2}{8\sigma^2\beta} - \frac{\epsilon V^2}{8\sigma^2\beta d}\right) \cdot \exp\left(-(c-1)2Td - \frac{\epsilon V^2}{8(c-1)\sigma^2\beta d}\right).$$

Next, set

$$c - 1 = \sqrt{\frac{\epsilon V^2}{16\sigma^2\beta T d^2}}.$$

From this, we will get

$$\begin{aligned}\frac{\pi(\theta)G_r(\theta,\theta^*)}{\pi(\theta)\bar{G}(\theta,\theta^*)} &\geq \exp\left(-\frac{\epsilon TV^2}{8\sigma^2\beta} - \frac{\epsilon V^2}{8\sigma^2\beta d}\right) \cdot \exp\left(-\sqrt{\frac{\epsilon TV^2}{\sigma^2\beta}}\right) \\ &\geq \exp\left(-\frac{\epsilon TV^2}{4\sigma^2\beta} - \sqrt{\frac{\epsilon TV^2}{\sigma^2\beta}}\right).\end{aligned}$$

Now, in order for this to hold, we needed

$$\frac{(c-1)\epsilon\beta}{1+\epsilon\beta} \leq 1.$$

With our setting of c, and our other assumption,

$$c - 1 = \sqrt{\frac{\epsilon V^2}{16\sigma^2\beta T d^2}} = \sqrt{\frac{\epsilon V^2}{4\sigma^2\beta d} \cdot \frac{1}{4Td}} \leq \sqrt{\frac{1}{4Td}} \leq 1,$$

so the bound will trivially hold. Thus the only added assumption we needed is the one stated in the Theorem statement, that

$$\frac{\epsilon V^2}{4\sigma^2\beta d} \leq 1.$$

Now we apply the standard Dirichlet form argument. The spectral gap of a Markov chain can be written as [3]

$$\gamma = \inf_{f \in L_0^2(\pi): Var_\pi[f]=1} \mathcal{E}(f)$$

where $L_0^2(\pi)$ denotes the Hilbert space of all functions that are square integrable with respect to probability measure π and have mean zero. $\mathcal{E}(f)$ is the Dirichlet

form of a Markov chain associated with transition operator T [50]:

$$\mathcal{E}(f) = \frac{1}{2}\int\int \left[(f(\theta) - f(\theta^*))^2\right] G(\theta, \theta^*)\pi(\theta)d\theta d\theta^*$$

By the expression of the spectral gap, it follows that

$$\begin{aligned}
\gamma &= \inf_{f\in L_0^2(\pi):Var_\pi[f]=1} \left[\frac{1}{2}\int\int \left[(f(\theta) - f(\theta^*))^2\right] G(\theta, \theta^*)\pi(\theta)d\theta d\theta^*\right] \\
&\geq \exp\left(-\frac{\epsilon T V^2}{4\sigma^2\beta} - \sqrt{\frac{\epsilon T V^2}{\sigma^2\beta}}\right) \\
&\quad \cdot \inf_{f\in L_0^2(\pi):Var_\pi[f]=1} \left[\frac{1}{2}\int\int \left[(f(\theta) - f(\theta^*))^2\right] \bar{G}(\theta, \theta)\pi(\theta)d\theta d\theta^*\right] \\
&= \exp\left(-\frac{\epsilon T V^2}{4\sigma^2\beta} - \sqrt{\frac{\epsilon T V^2}{\sigma^2\beta}}\right) \cdot \bar{\gamma}
\end{aligned}$$

This finishes the proof. □

C.2 Reformulation of AMAGOLD Algorithm

We reformulate our algorithm by setting $v = \epsilon\sigma^{-2}r$,$b = \epsilon\beta$, $h = \epsilon^2\sigma^{-2}$ and outline the algorithm after reformulation in Algorithm 15.

Algorithm 15 Reformulated AMAGOLD

given: Energy U, initial state $\theta \in \Theta$
loop
  optionally, resample momentum: $v \sim \mathcal{N}(0, h\mathbf{I})$
  initialize momentum and energy acc: $v_{-\frac{1}{2}} \leftarrow v$, $\rho_{-\frac{1}{2}} \leftarrow 0$
  half position update: $\theta_0 \leftarrow \theta + \frac{1}{2} v_{-\frac{1}{2}}$
  for $t = 0$ **to** $T - 1$ **do**
    if $t \neq 0$ **then**
      position update: $\theta_t \leftarrow \theta_{t-1} + v_{t-\frac{1}{2}}$
    end if
    sample noise $\eta_t \sim \mathcal{N}(0, 4hb)$
    sample random energy component $\tilde{U}_t$
    update momentum: $v_{t+\frac{1}{2}} \leftarrow \left((1-b) v_{t-\frac{1}{2}} - h \nabla \tilde{U}_t(\theta_t) + \eta_t\right) / (1+b)$
    update energy acc: $\rho_{t+\frac{1}{2}} \leftarrow \rho_{t-\frac{1}{2}} + \frac{1}{2} \nabla \tilde{U}_t(\theta_t)^T \left(v_{t-\frac{1}{2}} + v_{t+\frac{1}{2}}\right)$
  end for
  half position update: $\theta_T \leftarrow \theta_{T-1} + \frac{1}{2} v_{T-\frac{1}{2}}$
  new values: $\theta^* \leftarrow \theta_T$, $v^* \leftarrow v_{T-\frac{1}{2}}$
  $a \leftarrow \exp\left(U(\theta) - U(\theta^*) + \rho_{T-\frac{1}{2}}\right)$
  with probability $\min(1, a)$ **update** $\theta \leftarrow \theta^*$, $v \leftarrow v^*$ (as long as $\theta^* \in \Theta$)
  otherwise update $v \leftarrow -v_{-\frac{1}{2}}$
end loop

C.3 Additional Experiments Results and Setting Details

Double Well Potential

We visualize the estimated density on additional step size settings. Consistent with Figure 4.1d, it is clear here that SGHMC is very sensitive to step size. A small change in step size will cause a big difference in the estimated density. In contrast, AMAGOLD is more robust and can work well with a large range of step sizes.

When the setup of step size is inappropriate, as in Figures C.1a and b where it is fixed to be too small, either SGHMC or AMAGOLD converges in the training time. This is because the chain moves too slowly toward the stationary distribution. However, AMAGOLD with step size tuning is able to automatically adjust the step size based on the information provided by MH step. As shown in Figure 4.1c, tuned AMAGOLD can determine a step size that causes convergence given the same training time budget. All results are obtained by collecting 10^5 samples with 1000 burn-in samples.

Two-Dimensional Synthetic Distributions

Analytical Expression

Dist1: $\mathcal{N}(z_1; z_2^2/4, 1)\mathcal{N}(z_2; 0, 4)$

Dist2: $0.5\mathcal{N}\left(\boldsymbol{z}; 0, \begin{bmatrix} 2 & 1.8 \\ 1.8 & 2 \end{bmatrix}\right) + 0.5\mathcal{N}\left(\boldsymbol{z}; 0, \begin{bmatrix} 2 & -1.8 \\ -1.8 & 2 \end{bmatrix}\right)$

Runtime Comparisons We report runtime comparisons between AMAGOLD and SGHMC on Dist1 and Dist2 with step size 0.15 (Figure C.2). This experiment

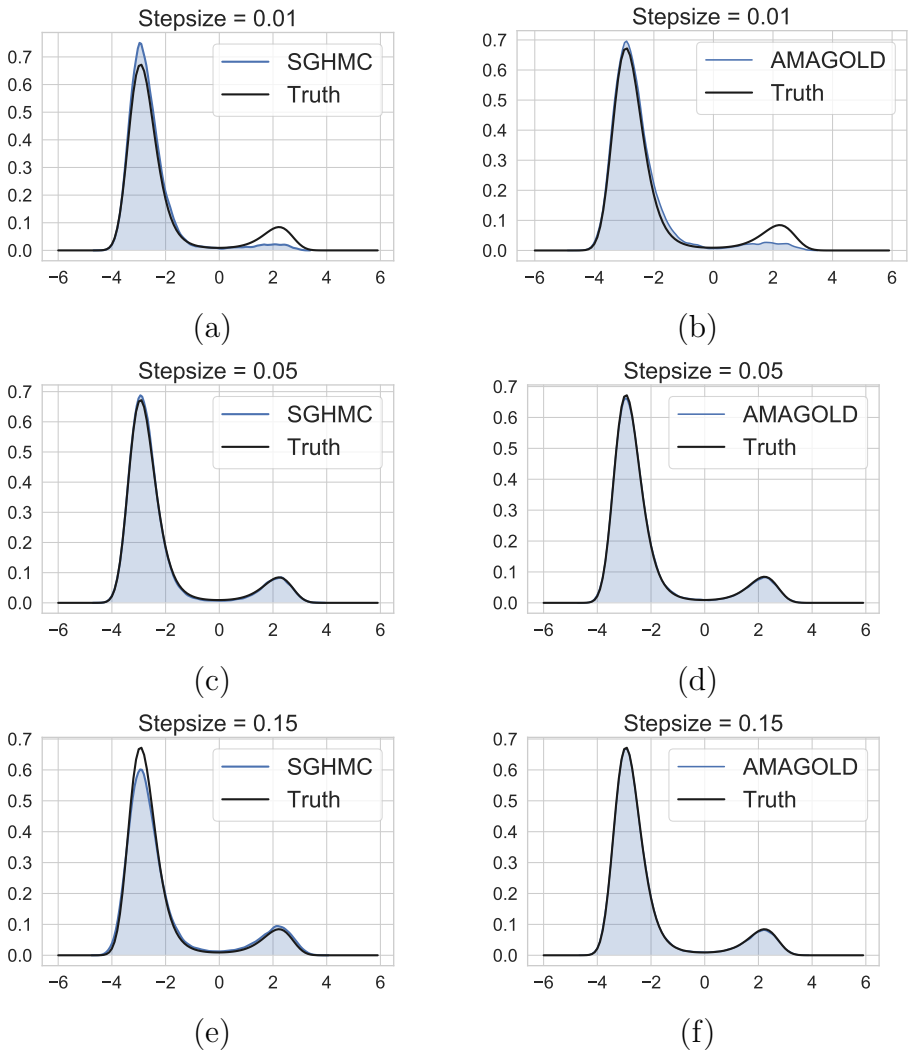


Figure C.1: Estimated densities of SGHMC (1st column) and AMAGOLD (2nd column) on varying step sizes.

uses the analytical energy expression (no data examples), so there is no speed-up of stochastic methods over full-batch methods. At the beginning, SGHMC converges faster due to the lack of MH step, but eventually it converges to a biased distribution. AMAGOLD is not much slower than SGHMC, which shows that AMA can reduce the amount of computation of adding MH step while keep the chain unbiased.

Additional Note on Figure 4.2 It is worth noting that, even though it is lower than SGHMC's, AMAGOLD's KL divergence grows when the step size is large

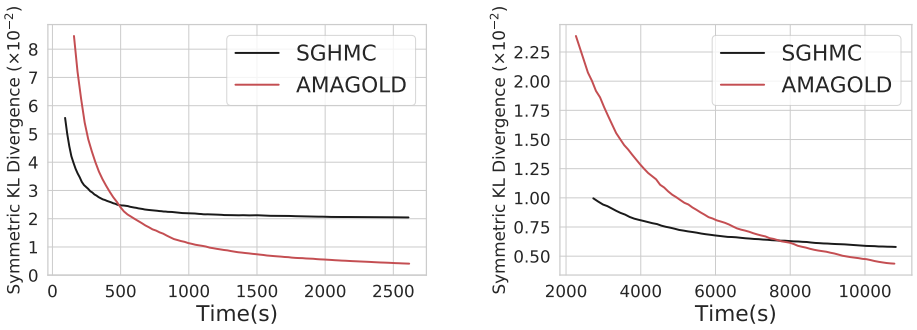


Figure C.2: Runtime comparisons between SGHMC and AMAGOLD on synthetic distributions (a) Dist1 and (b) Dist2.

compared to full-batch methods. This is because the MH acceptance probability decreases, causing the chain to converge more slowly. This is expected. It is well-known that stochastic methods are more sensitive to step sizes than full-batch methods [110]. However, since AMAGOLD's KL divergence grows much slower than SGHMC's, AMAGOLD is more robust to different step sizes

Bayesian Logistic Regression

We report the acceptance probability of AMAGOLD on *Heart* for varying step sizes in Figure C.3. For a large range of step sizes, the acceptance rate is sufficiently high to allow the chain converge fast, demonstrated in Figure 4.4. The acceptance rate may become very low with a large step size resulting in slow move. But this undesired acceptance probability can be easily detected and avoided in practice.

Bayesian Neural Networks

The architecture of Bayesian Neural Networks is a two-layer MLP with first hidden layer size 500 and the second hidden layer size 256.

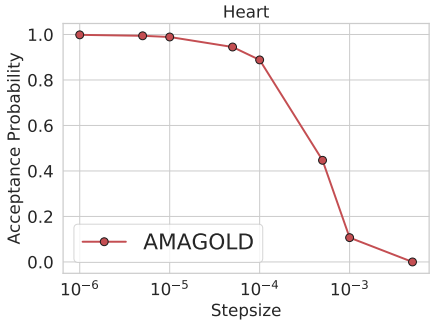


Figure C.3: The acceptance probability of the MH step in AMAGOLD for varying step sizes on the Heart dataset.

APPENDIX D

SECTION 5 CYCLICAL STOCHASTIC GRADIENT MCMC

D.1 Proofs

D.1.1 Assumptions

Assumptions in weak convergence analysis

In the analysis, we define a functional ψ that solves the following *Poisson Equation*:

$$\mathcal{L}\psi(\theta_k) = \phi(\theta_k) - \bar{\phi}, \text{ or equivalently, } \frac{1}{K}\sum_{k=1}^{K}\mathcal{L}\psi(\theta_k) = \hat{\phi} - \bar{\phi}. \tag{D.1}$$

The solution functional $\psi(\theta_k)$ characterizes the difference between $\phi(\theta_k)$ and the posterior average $\bar{\phi}$ for every θ_k, thus would typically possess a unique solution, which is at least as smooth as ϕ under the elliptic or hypoelliptic settings [101]. Following [25, 143], we make certain assumptions on the solution functional, ψ, of the Poisson equation (D.1).

Assumption 3. *ψ and its up to 3rd-order derivatives, $\mathcal{D}^k\psi$, are bounded by a function $\mathcal{V}$, i.e., $\|\mathcal{D}^k\psi\| \leq H_k\mathcal{V}^{p_k}$ for $k = (0, 1, 2, 3)$, $H_k, p_k > 0$. Furthermore, the expectation of $\mathcal{V}$ on $\{\theta_k\}$ is bounded: $\sup_l \mathbb{E}\mathcal{V}^p(\theta_k) < \infty$, and $\mathcal{V}$ is smooth such that $\sup_{s\in(0,1)} \mathcal{V}^p\left(s\theta + (1-s)\,\theta'\right) \leq C\left(\mathcal{V}^p\left(\theta\right) + \mathcal{V}^p\left(\theta'\right)\right)$, $\forall\theta, \theta', p \leq \max\{2p_k\}$ for some $C > 0$.*

Assumptions in convergence under the Wasserstein distance

Following existing work in [119], we adopt the following standard assumptions summarized in Assumption 4.

Assumption 4.
- *There exists some constants $A \geq 0$ and $B \geq 0$, such that $U(0) \leq A$ and $\nabla U(0) \leq B$.*
- *The function U is L_U-smooth : $\|\nabla U(w) - \nabla U(v)\| \leq L_U \|w - v\|$.*
- *The function U is $(m_u, b) - dissipative$, which means for some $m_U > 0$ and $b > 0$ $\langle w, \nabla U(w) \rangle \geq m_U \|w\|^2 - b$.*
- *There exists some constant $\delta \in [0, 1)$, such that $\mathbb{E}[\|\nabla \tilde{U}_k(w) - \nabla U(w)\|^2] \leq 2\sigma(M_U^2 \|w\|^2 + B^2)$.*
- *We can choose μ_0 which satisfies the requirement: $\kappa_0 := log \int e^{\|w\|^2} \mu_0(w) dw < \infty$.*

D.1.2 Proof of Theorem 11

To prove the theorem, we borrow tools developed by [25, 141]. We first rephrase the stepsize assumptions in general SG-MCMC in Assumption 5.

Assumption 5. *The algorithm adopts an N-th order integrator. The step sizes $\{h_k\}$ are such that $0 < h_{k+1} < h_k$, and satisfy 1) $\sum_{k=1}^{\infty} h_k = \infty$; and 2) $\lim_{K \to \infty} \frac{\sum_{k=1}^{K} h_k^{N+1}}{\sum_{k=1}^{K} h_k} = 0$.*

Our prove can be derived by the following results from [25].

Lemma 8 ([25])**.** *11 Let $S_K \triangleq \sum_{k=1}^{K} h_k$. Under Assumptions 3 and 5, for a smooth test function ϕ, the bias and MSE of a decreasing-step-size SG-MCMC with*

a Nth-order integrator at time S_L are bounded as:

$$\textit{BIAS: } \left|\mathbb{E}\tilde{\phi} - \bar{\phi}\right| = O\left(\frac{1}{S_K} + \frac{\sum_{k=1}^{K} h_k^{N+1}}{S_K}\right) \tag{D.2}$$

$$\textit{MSE: } \mathbb{E}\left(\tilde{\phi} - \bar{\phi}\right)^2 \leq C\left(\sum_l \frac{h_k^2}{S_K^2}\mathbb{E}\left\|\Delta V_l\right\|^2 + \frac{1}{S_K} + \frac{(\sum_{k=1}^{K} h_k^{N+1})^2}{S_K^2}\right) . \tag{D.3}$$

Note that Assumption 5 is only required if one wants to prove the asymptotically unbias of an algorithm. Lemma 11 still applies even if Assumption 5 is not satisfied. In this case one would obtain a biased algorithm, which is the case of cSGLD.

Proof of Theorem 11. Our results is actually a special case of Lemma 11. To see that, first note that our cSGLD adopts a first order integrator, thus $N = 1$. To proceed, note that $S_K = \sum_{k=1}^{K} \alpha_k = O(\alpha_0 K)$, and

$$\begin{aligned}
\sum_{j=0}^{K-1} \alpha_{j+1}^2 &= \frac{\alpha_0^2}{4}\sum_{j=0}^{K-1}[\cos(\frac{\pi mod(j-1,[K/M])}{[K/M]}) + 1]^2 \\
&= \frac{\alpha_0^2}{4}\sum_{j=0}^{K-1}[\cos^2(\frac{\pi mod(j-1,K/M)}{K/M}) + 1]^2 \\
&= \frac{\alpha_0^2}{4}\frac{K}{M}(\frac{M}{2} + M) = \frac{3\alpha_0^2 K}{8} .
\end{aligned} \tag{D.4}$$

As a result, for the bias, we have

$$\begin{aligned}
\left|\mathbb{E}\tilde{\phi} - \bar{\phi}\right| &= O\left(\frac{1}{S_K} + \frac{\sum_{k=1}^{K} h_k^{N+1}}{S_K}\right) = O\left(\frac{1}{\alpha_0 K} + \frac{3\alpha_0^2 K/8}{\alpha_0 K}\right) \\
&= O\left(\frac{1}{\alpha_0 K} + \alpha_0\right) .
\end{aligned}$$

For the MSE, note the first term $\sum_l \frac{h_k^2}{S_K^2}\mathbb{E}\left\|\Delta V_l\right\|^2$ has a higher order than other terms, thus it is omitted in the big-O notation, *i.e.*,

$$\begin{aligned}
\mathbb{E}\left(\tilde{\phi} - \bar{\phi}\right)^2 &= O\left(\frac{1}{\alpha_0 K} + (\frac{3\alpha_0^2 K/8}{\alpha_0 K})^2\right) \\
&= O\left(\frac{1}{\alpha_0 K} + \alpha_0^2\right) .
\end{aligned}$$

This completes the proof. □

D.1.3 Proof of Theorem 12

Proof of the bound for $W_2(\mu_K, \nu_\infty)$ in cSGLD. Firstly, we introduce the following SDE

$$\mathrm{d}\theta_t = -\nabla U(\theta_t)\mathrm{d}t + \sqrt{2}\mathrm{d}\mathcal{W}_t \ , \tag{D.5}$$

Let ν_t denote the distribution of θ_t, and the stationary distribution of (D.5) be $p(\theta|\mathcal{D})$, which means $\nu_\infty = p(\theta|\mathcal{D})$.

$$\theta_{k+1} = \theta_k - \nabla\tilde{U}_k(\theta_k)\alpha_{k+1} + \sqrt{2\alpha_{k+1}}\xi_{k+1} \tag{D.6}$$

Further, let μ_k denote the distribution of θ_k.

Since

$$W_2(\mu_K, \nu_\infty) \leq W_2(\mu_K, \nu_{\sum_{k=1}^K \alpha_k}) + W_2(\nu_{\sum_{k=1}^K \alpha_k}, \nu_\infty) \tag{D.7}$$

, we need to give the bounds for these two parts respectively.

W$_2(\mu_K, \nu_{\sum_{k=1}^K \alpha_k})$ For the first part, $W_2(\mu_K, \nu_{\sum_{k=1}^K \alpha_k})$, our proof is based on the proof of Lemma 3.6 in [119] with some modifications. We first assume $\mathbb{E}(\nabla\tilde{U}(w)) = \nabla U(w), \ \ \forall w \in \mathbb{R}^d$, which is a general assumption according to the way we choose the minibatch. And we define $p(t)$ which will be used in the following proof:

$$p(t) = \{k \in \mathbb{Z} | \sum_{i=1}^{k} \alpha_i \leq t < \sum_{i=1}^{k+1} \alpha_i\}$$

Then we focus on the following continuous-time interpolation of θ_k:

$$\underline{\theta}(t) = \theta_0 - \int_0^t \nabla\tilde{U}\left(\underline{\theta}(\sum_{k=1}^{p(s)} \alpha_k)\right)\mathrm{d}s + \sqrt{2}\int_0^t d\mathcal{W}_s^{(d)}$$

where $\nabla\tilde{U} \equiv \nabla\tilde{U}_k$ for $t \in \left[\sum_{i=1}^{k}\alpha_i, \sum_{i=1}^{k+1}\alpha_i\right)$. And for each k , $\underline{\theta}(\sum_{i=1}^{k}\alpha_i)$ and θ_k have the same probability law μ_k.

Since $\underline{\theta}(t)$ is not a Markov process, we define the following process which has the same one-time marginals as $\underline{\theta}(t)$

$$V(t) = \theta_0 - \int_0^t G_s\left(V(s)\right)\mathrm{d}s + \sqrt{2}\int_0^t d\mathcal{W}_s^{(d)}$$

with

$$G_t(x) := \mathbb{E}\left[\nabla\tilde{U}\left(\underline{\theta}(\sum_{i=1}^{q(t)}\alpha_i)\right)|\underline{\theta}(t) = x\right]$$

Let $\mathbf{P}_V^t := \mathcal{L}\left(V(s) : 0 \le s \le t\right)$ and $\mathbf{P}_\theta^t := \mathcal{L}\left(\theta(s) : 0 \le s \le t\right)$ and according to the proof of Lemma 3.6 in [119], we can derive a similar result for the relative entropy of $\mathbf{P}_V^t$ and $\mathbf{P}_\theta^t$:

$$\begin{aligned} D_{KL}(\mathbf{P}_V^t \,\|\, \mathbf{P}_\theta^t) &= -\int \mathrm{d}\,\mathbf{P}_V^t \log\frac{\mathrm{d}\,\mathbf{P}_V^t}{\mathrm{d}\,\mathbf{P}_\theta^t} \\ &= \frac{1}{4}\int_0^t \mathbb{E}\|\nabla U(V(s)) - G_s(V(s))\|^2\mathrm{d}s \\ &= \frac{1}{4}\int_0^t \mathbb{E}\|\nabla U(\underline{\theta}(s)) - G_s(\underline{\theta}(s))\|^2\mathrm{d}s \end{aligned}$$

The last line follows the fact that $\mathcal{L}(\underline{\theta}(s)) = \mathcal{L}(V(s)),\ \forall s$.

Then we will let $t = \sum_{k=1}^{K}\alpha_k$ and we can use the martingale property of the integral

to derive:

$$
\begin{aligned}
&D_{KL}(\mathbf{P}_V^{\sum_{k=1}^K \alpha_k} \,\|\, \mathbf{P}_\theta^{\sum_{k=1}^K \alpha_k}) \\
&= \frac{1}{4}\sum_{j=0}^{K-1} \int_{\sum_{k=1}^j \alpha_k}^{\sum_{k=1}^{j+1} \alpha_k} \mathbb{E}\|\nabla U(\underline{\theta}(s)) - G_s(\underline{\theta}(s))\|^2 \mathrm{d}s \\
&\leq \frac{1}{2}\sum_{j=0}^{K-1} \int_{\sum_{k=1}^j \alpha_k}^{\sum_{k=1}^{j+1} \alpha_k} \mathbb{E}\|\nabla U(\underline{\theta}(s)) - \nabla U(\underline{\theta}(\sum_{k=1}^{q(s)} \alpha_i)\|^2 \mathrm{d}s \\
&\quad + \frac{1}{2}\sum_{j=0}^{K-1} \int_{\sum_{k=1}^j \alpha_k}^{\sum_{k=1}^{j+1} \alpha_k} \mathbb{E}\|\nabla U(\underline{\theta}(\sum_{k=1}^{q(s)} \alpha_i) - G_s(\underline{\theta}(\sum_{k=1}^{q(s)} \alpha_i)\|^2 \mathrm{d}s \\
&\leq \frac{L_U^2}{2}\sum_{j=0}^{K-1} \int_{\sum_{k=1}^j \alpha_k}^{\sum_{k=1}^{j+1} \alpha_k} \mathbb{E}\|\underline{\theta}(s) - \underline{\theta}(\sum_{k=1}^{q(s)} \alpha_i)\|^2 \mathrm{d}s \qquad \text{(D.8)} \\
&\quad + \frac{1}{2}\sum_{j=0}^{K-1} \int_{\sum_{k=1}^j \alpha_k}^{\sum_{k=1}^{j+1} \alpha_k} \mathbb{E}\|\nabla U(\underline{\theta}(\sum_{k=1}^{q(s)} \alpha_i) - G_s(\underline{\theta}(\sum_{k=1}^{q(s)} \alpha_i)\|^2 \mathrm{d}s \qquad \text{(D.9)}
\end{aligned}
$$

For the first part (D.8), we consider some $s \in [\sum_{k=1}^j \alpha_k, \sum_{k=1}^{j+1} \alpha_k)$, for which the following holds:

$$
\begin{aligned}
&\underline{\theta}(s) - \underline{\theta}(\sum_{k=1}^j \alpha_k) \\
&= -(s - \sum_{k=1}^j \alpha_k)\nabla \tilde{U}_k(\theta_k) + \sqrt{2}(\mathcal{W}_s^{(d)} - \mathcal{W}_{\sum_{k=1}^j \alpha_k}^{(d)}) \\
&= -(s - \sum_{k=1}^j \alpha_k)\nabla U(\theta_k) + (s - \sum_{k=1}^j \alpha_k)(\nabla U(\theta_k) - \nabla \tilde{U}_k(\theta_k)) + \sqrt{2}(\mathcal{W}_s^{(d)} - \mathcal{W}_{\sum_{k=1}^j \alpha_k}^{(d)})
\end{aligned} \qquad \text{(D.10)}
$$

Thus, we can use Lemma 3.1 and 3.2 in [119] for the following result:

$$
\begin{aligned}
&\mathbb{E}\|\underline{\theta}(s) - \underline{\theta}(\sum_{k=1}^j \alpha_k)\|^2 \\
&\leq 3\alpha_{j+1}^2 \mathbb{E}\|\nabla U(\theta_j)\|^2 + 3\alpha_{j+1}^2 \mathbb{E}\|\nabla U(\theta_j) - \nabla \tilde{U}_j(\theta_j)\|^2 + 6\alpha_{j+1} d \\
&\leq 12\alpha_{j+1}^2 (L_U^2 \mathbb{E}\|\theta_j\|^2 + B^2) + 6\alpha_{j+1} d
\end{aligned}
$$

Hence we can bound the first part, (choosing $\alpha_0 \leq 1$),

$$\begin{aligned}
&\frac{L_U^2}{2}\sum_{j=0}^{K-1}\int_{\sum_{k=1}^{j}\alpha_k}^{\sum_{k=1}^{j+1}\alpha_k}\mathbb{E}\|\underline{\theta}(s)-\underline{\theta}(\sum_{k=1}^{q(s)}\alpha_i)\|^2\mathrm{d}s\\
&\leq \frac{L_U^2}{2}\sum_{j=0}^{K-1}\left[12\alpha_{j+1}^3(L_U^2\mathbb{E}\|\theta_j\|^2+B^2)+6\alpha_{j+1}^2 d\right]\\
&\leq L_U^2\max_{0\leq j\leq K-1}\left[6(L_U^2\mathbb{E}\|\theta_j\|^2+B^2)+3d\right](\sum_{j=0}^{K-1}\alpha_{j+1}^2)\\
&\leq L_U^2\max_{0\leq j\leq K-1}\left[6(L_U^2\mathbb{E}\|\theta_j\|^2+B^2)+3d\right]\frac{3\alpha_0^2 K}{8} \qquad \text{(D.11)}
\end{aligned}$$

The last line (D.11) follows from[1] (D.4). The second part (D.9) can be bounded as follows:

$$\begin{aligned}
&\frac{1}{2}\sum_{j=0}^{K-1}\int_{\sum_{k=1}^{j}\alpha_k}^{\sum_{k=1}^{j+1}\alpha_k}\mathbb{E}\|\nabla U(\underline{\theta}(\sum_{k=1}^{q(s)}\alpha_i)-G_s(\underline{\theta}(\sum_{k=1}^{q(s)}\alpha_i)\|^2\mathrm{d}s\\
&=\frac{1}{2}\sum_{j=0}^{K-1}\alpha_{j+1}\mathbb{E}\|\nabla U(\theta_j)-\nabla\tilde{U}(\theta_j)\|^2\\
&\leq \sigma\max_{0\leq j\leq K-1}(L_U^2\mathbb{E}\|\theta_j\|^2+B^2)\sum_{j=0}^{K-1}\alpha_{j+1}\\
&\leq \sigma\max_{0\leq j\leq K-1}(L_U^2\mathbb{E}\|\theta_j\|^2+B^2)(\frac{\alpha_0}{2}\sum_{j=0}^{K-1}(\cos(\frac{\pi mod(j,K/M)}{K/M})+1))\\
&\leq \sigma\max_{0\leq j\leq K-1}(L_U^2\mathbb{E}\|\theta_j\|^2+B^2)(\frac{K\alpha_0}{2})
\end{aligned}$$

[1]Note: we only focus on the case when $K \mod M = 0$.

Due to the data-processing inequality for the relative entropy, we have

$$
\begin{aligned}
&D_{KL}(\mu_K \| \nu_{\sum_{k=1}^K \alpha_k}) \leq D_{KL}(\mathbf{P}_V^t \,\|\, \mathbf{P}_\theta^t) \\
&\leq \frac{L_U^2}{2} \sum_{j=0}^{K-1} \int_{\sum_{k=1}^j \alpha_k}^{\sum_{k=1}^{j+1} \alpha_k} \mathbb{E}\|\underline{\theta}(s) - \underline{\theta}(\sum_{k=1}^{q(s)} \alpha_i)\|^2 \mathrm{d}s \\
&\quad + \frac{1}{2} \sum_{j=0}^{K-1} \int_{\sum_{k=1}^j \alpha_k}^{\sum_{k=1}^{j+1} \alpha_k} \mathbb{E}\|\nabla U(\underline{\theta}(\sum_{k=1}^{q(s)} \alpha_i) - G_s(\underline{\theta}(\sum_{k=1}^{q(s)} \alpha_i)\|^2 \mathrm{d}s \\
&\leq L_U^2 \max_{0 \leq j \leq K-1} \left[6(L_U^2 \mathbb{E}\|\theta_j\|^2 + B^2) + 3d\right] \frac{3\alpha_0^2 K}{8} \\
&\quad + \sigma \max_{0 \leq j \leq K-1} (L_U^2 \mathbb{E}\|\theta_j\|^2 + B^2)(\frac{K\alpha_0}{2})
\end{aligned}
$$

According to the proof of Lemma 3.2 in [119], we can bound the term $\mathbb{E}\|\theta_k\|^2$

$$
\mathbb{E}\|\theta_{k+1}\|^2 \leq (1 - 2\alpha_{k+1} m_U + 4\alpha_{k+1}^2 M_U^2)\mathbb{E}\|\theta_k\|^2 + 2\alpha_{k+1} b + 4\alpha_{k+1}^2 B^2 + \frac{2\alpha_{k+1} d}{\beta}
$$

Similar to the statement of Lemma 3.2 in [119], we can fix $\alpha_0 \in (0, 1 \wedge \frac{m_U}{4M_U^2})$. Then, we can know that

$$
\mathbb{E}\|\theta_{k+1}\|^2 \leq (1 - 2\alpha_{min} m_U + 4\alpha_{min}^2 M_U^2)\mathbb{E}\|\theta_k\|^2 + 2\alpha_0 b + 4\alpha_0^2 B^2 + \frac{2\alpha_0 d}{\beta} \quad \text{(D.12)}
$$

, where α_{min} is defined as $\alpha_{min} \triangleq \frac{\alpha_0}{2}\left[\cos\left(\frac{\pi \bmod(\lceil K/M \rceil - 1, \lceil K/M \rceil)}{\lceil K/M \rceil}\right) + 1\right]$.

There are two cases to consider.

- If $1 - 2\alpha_{min} m_U + 4\alpha_{min}^2 M_U^2 \leq 0$, then from (D.12) it follows that

$$
\begin{aligned}
\mathbb{E}\|\theta_{k+1}\|^2 &\leq 2\alpha_0 b + 4\alpha_0^2 B^2 + \frac{2\alpha_0 d}{\beta} \\
&\leq \mathbb{E}\|\theta_0\|^2 + 2(b + 2B^2 + \frac{d}{\beta})
\end{aligned}
$$

- If $0 \leq 1 - 2\alpha_{min} m_U + 4\alpha_{min}^2 M_U^2 \leq 1$, then iterating (D.12) gives

$$
\begin{aligned}
\mathbb{E}\|\theta_k\|^2 &\leq (1 - 2\alpha_{min} m_U + 4\alpha_{min}^2 M_U^2)^k \mathbb{E}\|\theta_0\|^2 + \frac{\alpha_0 b + 2\alpha_0^2 B^2 + \frac{\alpha_0 d}{\beta}}{\alpha_{min} m_U - 2\alpha_{min}^2 M_U^2} \\
&\leq \mathbb{E}\|\theta_0\|^2 + \frac{2\alpha_0}{m_U \alpha_{min}}(b + 2B^2 + \frac{d}{\beta})
\end{aligned}
$$

Now, we have

$$\begin{aligned}&\max_{0\le j\le K-1}(L_U^2\mathbb{E}\|\theta_j\|^2+B^2)\\&\le(L_U^2(\kappa_0+2(1\wedge\frac{\alpha_0}{m_U\alpha_{min}})(b+2B^2+d))+B^2):=C_0\end{aligned}$$

Due to the expression of $\frac{\alpha_0}{\alpha_{min}}$, C_0 is independent of α_0. Then we denote the $6L_U^2(C_0+d)$ as C_1 and we can derive

$$D_{KL}(\mu_K\|\nu_{\sum_{k=1}^K\alpha_k})\le C_1(\frac{3\alpha_0^2K}{8})+\sigma C_0(\frac{K\alpha_0}{2})$$

Then according to Proposition 3.1 in [17] and Lemma 3.3 in [119], if we denote $\kappa_0+2b+2d$ as C_2, we can derive the following result:

$$\begin{aligned}&W_2(\mu_K,\nu_{\sum_{k=1}^K\alpha_k})\\&\le(12+C_2(\sum_{k=1}^K\alpha_k))^{\frac{1}{2}}\cdot[D_{KL}(\mu_K\|\nu_{\sum_{k=1}^K\alpha_k})^{\frac{1}{2}}+D_{KL}(\mu_K\|\nu_{\sum_{k=1}^K\alpha_k})^{\frac{1}{4}}]\\&\le(12+\frac{C_2K\alpha_0}{2})^{\frac{1}{2}}\cdot[(\frac{3C_1\alpha_0^2K}{8}+\frac{K\sigma C_0\alpha_0}{2})^{\frac{1}{2}}+(\frac{3C_1\alpha_0^2K}{16}+\frac{K\sigma C_0\alpha_0}{4})^{\frac{1}{4}}]\end{aligned}$$

$\boldsymbol{W}_2(\nu_{\sum_{k=1}^K\alpha_k},\nu_\infty)$ We can directly get the following results from (3.17) in [119] that there exist some positive constants (C_3,C_4),

$$W_2(\nu_{\sum_{k=1}^K\alpha_k},\nu_\infty)\le C_3\exp(-\sum_{k=1}^K\alpha_k/C_4)$$

Now combining the bounds for $W_2(\mu_K,\nu_{\sum_{k=1}^K\alpha_k})$ and $W_2(\nu_{\sum_{k=1}^K\alpha_k},\nu_\infty)$, substituting $\alpha_0=O(1/K^\beta)$, and noting $W_2(\nu_{\sum_{k=1}^K\alpha_k},\nu_\infty)$ decreases w.r.t. K, we arrive at the bound stated in the theorem. □

Relation with SGLD

For the standard polynomially-decay-stepsize SGLD, the convergence rate is bounded as

$$W_2(\tilde{\mu}_K, \nu_\infty) \leq W_2(\tilde{\mu}_K, \nu_{\sum_{k=1}^K h_k}) + W_2(\nu_{\sum_{k=1}^K h_k}, \nu_\infty) \tag{D.13}$$

where $W_2(\tilde{\mu}_K, \nu_{\sum_{k=1}^K h_k}) \leq (6 + h_0 \sum_{k=1}^K \frac{1}{k})^{\frac{1}{2}} \cdot$

$$[(D_1 h_0^2 \frac{\pi^2}{6} + \sigma D_0 h_0 \sum_{k=1}^K \frac{1}{k})^{\frac{1}{2}} + (D_1 h_0^2 \frac{\pi^2}{16} + \sigma D_0 \frac{h_0}{2} \sum_{k=1}^K \frac{1}{k})^{\frac{1}{4}}]$$

and $W_2(\nu_{\sum_{k=1}^K h_k}, \nu_\infty) \leq C_3 \exp(-\frac{\sum_{k=1}^K h_k}{C_4})$.

Proof of the bound of $W_2(\tilde{\mu}_K, \nu_\infty)$ in the standard SGLD. Similar to the proof of $W_2(\mu_K, \nu_\infty)$ in cSGLD, we get the following update rule for SGLD with the stepsize following a polynomial decay *i.e.*, $h_k = \frac{h_0}{k}$,

$$\theta_{k+1} = \theta_k - \nabla \tilde{U}_k(\theta_k) h_{k+1} + \sqrt{2h_{k+1}} \xi_{k+1}$$

Let $\tilde{\mu}_k$ denote the distribution of θ_k.

Since

$$W_2(\tilde{\mu}_K, \nu_\infty) \leq W_2(\tilde{\mu}_K, \nu_{\sum_{k=1}^K h_k}) + W_2(\nu_{\sum_{k=1}^K h_k}, \nu_\infty),$$

we need to give the bounds for these two parts respectively.

$W_2(\tilde{\mu}_K, \nu_{\sum_{k=1}^K h_k})$

We first assume $\mathbb{E}(\nabla \tilde{U}(w)) = \nabla U(w), \quad \forall w \in \mathbb{R}^d$, which is a general assumption according to the way we choose the minibatch. Following the proof in [119] and

the analysis of the SPOS method in [153], we define the following $p(t)$ which will be used in the following proof:

$$p(t) = \{k \in \mathbb{Z} | \sum_{i=1}^{k} h_i \leq t < \sum_{i=1}^{k+1} h_i\}$$

Then we focus on the following continuous-time interpolation of θ_k:

$$\underline{\theta}(t) = \theta_0 - \int_0^t \nabla \tilde{U} \left(\underline{\theta}(\sum_{k=1}^{p(s)} h_k) \right) \mathrm{d}s + \sqrt{2} \int_0^t d\mathcal{W}_s^{(d)},$$

where $\nabla \tilde{U} \equiv \nabla \tilde{U}_k$ for $t \in \left[\sum_{i=1}^{k} h_i, \sum_{i=1}^{k+1} h_i\right)$. And for each k , $\underline{\theta}(\sum_{i=1}^{k} h_i)$ and θ_k have the same probability law $\tilde{\mu}_k$.

Since $\underline{\theta}(t)$ is not a Markov process, we define the following process which has the same one-time marginals as $\underline{\theta}(t)$

$$V(t) = \theta_0 - \int_0^t G_s\left(V(s)\right) \mathrm{d}s + \sqrt{2} \int_0^t d\mathcal{W}_s^{(d)}$$

with

$$G_t(x) := \mathbb{E}\left[\nabla \tilde{U} \left(\underline{\theta}(\sum_{i=1}^{q(t)} h_i) \right) | \underline{\theta}(t) = x \right]$$

Let $\mathbf{P}_V^t := \mathcal{L}\left(V(s) : 0 \leq s \leq t\right)$ and $\mathbf{P}_\theta^t := \mathcal{L}\left(\theta(s) : 0 \leq s \leq t\right)$ and according to the proof of Lemma 3.6 in [119], we can derive the similar result for the relative entropy of $\mathbf{P}_V^t$ and $\mathbf{P}_\theta^t$:

$$\begin{aligned} D_{KL}(\mathbf{P}_V^t \,\|\, \mathbf{P}_\theta^t) &= -\int \mathrm{d}\,\mathbf{P}_V^t \log \frac{\mathrm{d}\,\mathbf{P}_V^t}{\mathrm{d}\,\mathbf{P}_\theta^t} \\ &= \frac{1}{4} \int_0^t \mathbb{E} \| \nabla U(V(s)) - G_s(V(s)) \|^2 \mathrm{d}s \\ &= \frac{1}{4} \int_0^t \mathbb{E} \| \nabla U(\underline{\theta}(s)) - G_s(\underline{\theta}(s)) \|^2 \mathrm{d}s \end{aligned}$$

The last line follows the fact that $\mathcal{L}(\underline{\theta}(s)) = \mathcal{L}(V(s)),\ \forall s$.

Then we will let $t = \sum_{k=1}^{K} h_k$ and we can use the martingale property of integral to derive:

$$
\begin{aligned}
&D_{KL}(\mathbf{P}_V^{\sum_{k=1}^{K} h_k} \,\|\, \mathbf{P}_\theta^{\sum_{k=1}^{K} h_k}) \\
&= \frac{1}{4} \sum_{j=0}^{K-1} \int_{\sum_{k=1}^{j} h_k}^{\sum_{k=1}^{j+1} h_k} \mathbb{E}\|\nabla U(\underline{\theta}(s)) - G_s(\underline{\theta}(s))\|^2 \mathrm{d}s \\
&\leq \frac{1}{2} \sum_{j=0}^{K-1} \int_{\sum_{k=1}^{j} h_k}^{\sum_{k=1}^{j+1} h_k} \mathbb{E}\|\nabla U(\underline{\theta}(s)) - \nabla U(\underline{\theta}(\sum_{k=1}^{q(s)} h_i)\|^2 \mathrm{d}s \\
&\quad + \frac{1}{2} \sum_{j=0}^{K-1} \int_{\sum_{k=1}^{j} h_k}^{\sum_{k=1}^{j+1} h_k} \mathbb{E}\|\nabla U(\underline{\theta}(\sum_{k=1}^{q(s)} h_i) - G_s(\underline{\theta}(\sum_{k=1}^{q(s)} h_i)\|^2 \mathrm{d}s \\
&\leq \frac{L_U^2}{2} \sum_{j=0}^{K-1} \int_{\sum_{k=1}^{j} h_k}^{\sum_{k=1}^{j+1} h_k} \mathbb{E}\|\underline{\theta}(s) - \underline{\theta}(\sum_{k=1}^{q(s)} h_i)\|^2 \mathrm{d}s \qquad \text{(D.14)} \\
&\quad + \frac{1}{2} \sum_{j=0}^{K-1} \int_{\sum_{k=1}^{j} h_k}^{\sum_{k=1}^{j+1} h_k} \mathbb{E}\|\nabla U(\underline{\theta}(\sum_{k=1}^{q(s)} h_i) - G_s(\underline{\theta}(\sum_{k=1}^{q(s)} h_i)\|^2 \mathrm{d}s \qquad \text{(D.15)}
\end{aligned}
$$

For the first part (D.14), we consider some $s \in [\sum_{k=1}^{j} h_k, \sum_{k=1}^{j+1} h_k)$, the following equation holds:

$$
\begin{aligned}
&\underline{\theta}(s) - \underline{\theta}(\sum_{k=1}^{j} h_k) \\
&= -(s - \sum_{k=1}^{j} h_k)\nabla \tilde{U}_k(\theta_k) + \sqrt{2}(\mathcal{W}_s^{(d)} - \mathcal{W}_{\sum_{k=1}^{j} h_k}^{(d)}) \\
&= -(s - \sum_{k=1}^{j} h_k)\nabla U(\theta_k) + (s - \sum_{k=1}^{j} h_k)(\nabla U(\theta_k) - \nabla \tilde{U}_k(\theta_k)) + \sqrt{2}(\mathcal{W}_s^{(d)} - \mathcal{W}_{\sum_{k=1}^{j} h_k}^{(d)})
\end{aligned} \qquad \text{(D.16)}
$$

Thus, we can use Lemma 3.1 and 3.2 in [119] for the following result:

$$
\begin{aligned}
&\mathbb{E}\|\underline{\theta}(s) - \underline{\theta}(\sum_{k=1}^{j} h_k)\|^2 \\
&\leq 3h_{j+1}^2 \mathbb{E}\|\nabla U(\theta_j)\|^2 + 3h_{j+1}^2 \mathbb{E}\|\nabla U(\theta_j) - \nabla \tilde{U}_j(\theta_j)\|^2 + 6h_{j+1}d \\
&\leq 12h_{j+1}^2 (L_U^2 \mathbb{E}\|\theta_j\|^2 + B^2) + 6h_{j+1}d
\end{aligned}
$$

Hence we can bound the first part, (choosing $h_0 \leq 1$),

$$\begin{aligned}
&\frac{L_U^2}{2} \sum_{j=0}^{K-1} \int_{\sum_{k=1}^{j} h_k}^{\sum_{k=1}^{j+1} h_k} \mathbb{E}\|\underline{\theta}(s) - \underline{\theta}(\sum_{k=1}^{q(s)} h_k)\|^2 \mathrm{d}s \\
&\leq \frac{L_U^2}{2} \sum_{j=0}^{K-1} \left[12 h_{j+1}^3 (L_U^2 \mathbb{E}\|\theta_j\|^2 + B^2) + 6 h_{j+1}^2 d\right] \\
&\leq L_U^2 \max_{0 \leq j \leq K-1} \left[6(L_U^2 \mathbb{E}\|\theta_j\|^2 + B^2) + 3d\right] (\sum_{j=0}^{K-1} h_{j+1}^2) \\
&\leq L_U^2 \max_{0 \leq j \leq K-1} \left[6(L_U^2 \mathbb{E}\|\theta_j\|^2 + B^2) + 3d\right] \frac{\pi^2}{6} h_0^2 \qquad \text{(D.17)}
\end{aligned}$$

where the last line follows from the fact that

$$\sum_{j=0}^{K-1} \frac{1}{(j+1)^3} \leq \sum_{j=0}^{K-1} \frac{1}{(j+1)^2} \leq \sum_{j=0}^{\infty} \frac{1}{(j+1)^2} = \frac{\pi^2}{6} \ .$$

The second part (D.15) can be bounded as follows:

$$\begin{aligned}
&\frac{1}{2} \sum_{j=0}^{K-1} \int_{\sum_{k=1}^{j} h_k}^{\sum_{k=1}^{j+1} h_k} \mathbb{E}\|\nabla U(\underline{\theta}(\sum_{k=1}^{q(s)} h_i) - G_s(\underline{\theta}(\sum_{k=1}^{q(s)} h_i)\|^2 \mathrm{d}s \\
&= \frac{1}{2} \sum_{j=0}^{K-1} h_{j+1} \mathbb{E}\|\nabla U(\theta_j) - \nabla \tilde{U}(\theta_j)\|^2 \\
&\leq \sigma \max_{0 \leq j \leq K-1} (L_U^2 \mathbb{E}\|\theta_j\|^2 + B^2) \sum_{j=0}^{K-1} h_{j+1} \\
&\leq \sigma \max_{0 \leq j \leq K-1} (L_U^2 \mathbb{E}\|\theta_j\|^2 + B^2)(h_0 \sum_{j=1}^{K} \frac{1}{j})
\end{aligned}$$

Due to the data-processing inequality for the relative entropy, we have

$$
\begin{aligned}
D_{KL}(\tilde{\mu}_K \| \nu_{\sum_{k=1}^K h_k}) &\leq D_{KL}(\mathbf{P}_V^t \parallel \mathbf{P}_\theta^t) \\
&\leq \frac{L_U^2}{2} \sum_{j=0}^{K-1} \int_{\sum_{k=1}^j h_k}^{\sum_{k=1}^{j+1} h_k} \mathbb{E}\|\underline{\theta}(s) - \underline{\theta}(\sum_{k=1}^{q(s)} h_i)\|^2 \mathrm{d}s \\
&\quad + \frac{1}{2} \sum_{j=0}^{K-1} \int_{\sum_{k=1}^j h_k}^{\sum_{k=1}^{j+1} h_k} \mathbb{E}\|\nabla U(\underline{\theta}(\sum_{k=1}^{q(s)} h_i) - G_s(\underline{\theta}(\sum_{k=1}^{q(s)} h_i)\|^2 \mathrm{d}s \\
&\leq L_U^2 \max_{0 \leq j \leq K-1} \left[6(L_U^2 \mathbb{E}\|\theta_j\|^2 + B^2) + 3d\right] \frac{\pi^2}{6} h_0^2 \\
&\quad + \sigma \max_{0 \leq j \leq K-1} (L_U^2 \mathbb{E}\|\theta_j\|^2 + B^2)(h_0 \sum_{j=1}^K \frac{1}{j})
\end{aligned}
$$

Similar to the proof of cSGLD , we have

$$
\max_{0 \leq j \leq K-1} (L_U^2 \mathbb{E}\|\theta_j\|^2 + B^2) \leq D_0
$$

Then we denote the $6L_U^2(D_0 + d)$ as D_1 and we can derive

$$
D_{KL}(\tilde{\mu}_K \| \nu_{\sum_{k=1}^K h_k}) \leq D_1 h_0^2 \frac{\pi^2}{6} + \sigma D_0 h_0 \sum_{j=1}^K \frac{1}{j}
$$

Then according to Proposition 3.1 in [17] and Lemma 3.3 in [119], if we denote $\kappa_0 + 2b + 2d$ as D_2, we can derive the following result,

$$
\begin{aligned}
&W_2(\tilde{\mu}_K, \nu_{\sum_{k=1}^K h_k}) \\
&\leq [12 + D_2(\sum_{k=1}^K h_k)]^{1/2} \cdot [(D_{KL}(\tilde{\mu}_K \| \nu_{\sum_{k=1}^K h_k}))^{1/2} + (D_{KL}(\tilde{\mu}_K \| \nu_{\sum_{k=1}^K h_k})/2)^{1/4}] \\
&= [12 + D_2(h_0 \sum_{j=1}^K \frac{1}{j})]^{1/2} \cdot [(D_1 h_0^2 \frac{\pi^2}{6} + \sigma D_0 h_0 \sum_{j=1}^K \frac{1}{j})^{1/2} + (D_1 h_0^2 \frac{\pi^2}{12} + \sigma D_0 h_0 \sum_{j=1}^K \frac{1}{2j})^{1/4}]
\end{aligned}
$$

Now we derive the bound for $W_2(\tilde{\mu}_K, \nu_{\sum_{k=1}^K h_k})$.

$W_2(\nu_{\sum_{k=1}^K h_k}, \nu_\infty)$

We can directly get the following results from (3.17) in [119] that there exist some positive constants (C_3, C_4),

$$W_2(\nu_{\sum_{k=1}^K h_k}, \nu_\infty) \leq C_3 \exp(-\sum_{k=1}^K h_k / C_4)$$

□

Based on the convergence error bounds, we discuss an informal comparison with standard SGLD. Consider the following two cases.We must emphasize that since the term $W_2(\mu_K, \nu_{\sum_{k=1}^K \alpha_k})$ in the (D.7) increases w.r.t. K, our α_0 must be set small enough in practice. Hence, in this informal comparison, we also set α_0 small enough to make $W_2(\mu_K, \nu_{\sum_{k=1}^K \alpha_k})$ less important.

i) If the initial stepsizes satisfy $\alpha_0 \geq h_0$, our algorithm cSGLD runs much faster than the standard SGLD in terms of the amount of "diffusion time" *i.e.*, the "t" indexing θ_t in the continuous-time SDE mentioned above. This result follows from $\sum_{k=1}^K \alpha_k = \frac{K\alpha_0}{2}$ and $\sum_{k=1}^K h_k = \sum_{k=1}^K \frac{h_0}{k} = \mathcal{O}(h_0 \log K) \ll \frac{K\alpha_0}{2}$. In standard SGLD, since the error described by $W_2(\tilde{\mu}_K, \nu_{\sum_{k=1}^K h_k})$ increases w.r.t. K, h_0 needs to be set small enough in practice to reduce the error. Following the general analysis of SGLD in [119, 149], the dominant term in the decomposition (D.13) will be $W_2(\nu_{\sum_{k=1}^K h_k}, \nu_\infty)$ since it decreases exponentially fast with the increase of t and $W_2(\tilde{\mu}_K, \nu_{\sum_{k=1}^K h_k})$ is small due to the setting of small h_0. Since $\sum_{k=1}^K \alpha_k$ increases much faster in our algorithm than the term $\sum_{k=1}^K h_k$ in standard SGLD, our algorithm thus endows less error for K iterations, *i.e.*, $W_2(\nu_{\sum_{k=1}^K \alpha_k}, \nu_\infty) \ll W_2(\nu_{\sum_{k=1}^K h_k}, \nu_\infty)$. Hence, our algorithm outperforms standard SGLD, as will be verified in our experiments.

ii) Instead of setting the h_0 small enough, one may consider increasing h_0 to make standard SGLD run as "fast" as our proposed algorithm, *i.e.*, $\sum_{k=1}^{K} h_k \approx \sum_{k=1}^{K} \alpha_k$. Now the $W_2(\nu_{\sum_{k=1}^{K} h_k}, \nu_\infty)$ in (D.13) is almost the same as the $W_2(\nu_{\sum_{k=1}^{K} h_k}, \nu_\infty)$ in (D.7). However, in this case, it is worth noting that h_0 scales as $\mathcal{O}(\alpha_0 K/\log K)$. We can notice that h_0 is much larger than the α_0 and thus the $W_2(\tilde{\mu}_K, \nu_{\sum_{k=1}^{K} h_k})$ cannot be ignored. Now the h_0^2 term in $W_2(\tilde{\mu}_K, \nu_{\sum_{k=1}^{K} h_k})$ would scale as $\mathcal{O}(\alpha_0^2 K^2/\log^2 K)$, which makes $W_2(\tilde{\mu}_K, \nu_{\sum_{k=1}^{K} \alpha_k})$ in (D.13) much larger than our $W_2(\mu_K, \nu_{\sum_{k=1}^{K} \alpha_k})$ defined in (D.7) since $\mathcal{O}(\alpha_0^2 K^2/\log^2 K) \gg \mathcal{O}(\alpha_0^2 K)$. Again, our algorithm cSGLD achieves a faster convergence rate than standard SGLD.

D.2 Combining Samples

In cyclical SG-MCMC, we obtain samples from multiple modes of a posterior distribution by running the cyclical step size schedule for many periods. We now show how to effectively utilize the collected samples. We consider each cycle exploring different part of the target distribution $p(\theta|\mathcal{D})$ on a metric space Θ. As we have M cycles in total, the mth cycle characterizes a local region $\Theta_m \subset \Theta$, defining the "sub-posterior" distribution: $p_m(\theta|\mathcal{D}) = \frac{p(\theta|\mathcal{D})\mathbf{1}_{\Theta_m}}{w_m}$, with $w_m = \int_{\Theta_m} p(\theta|\mathcal{D})d\theta$, where w_m is a normalizing constant. For a testing function $f(\theta)$, we are often interested in its true posterior expectation $\bar{f} = \int f(\theta)p(\theta|\mathcal{D})d\theta$. The sample-based estimation is

$$\hat{f} = \sum_{m=1}^{M} w_m \hat{f}_m \quad \text{with} \quad \hat{f}_m = \frac{1}{K_m} \sum_{j=1}^{K_m} f(\theta_j^{(m)}), \tag{D.18}$$

where K_m is the number of samples from the mth cycle, and $\theta^{(m)} \in \Theta_m$.

The weight for each cycle w_i is estimated using the *harmonic mean* method [60, 118]: $\hat{w}_m \approx [\frac{1}{K_m} \sum_{j=1}^{K_m} \frac{1}{p(\mathcal{D}|\theta_j^{(m)})}]^{-1}$. This approach provides a simple and consistent

estimator, where the only additional cost is to traverse the training dataset to evaluate the likelihood $p(\mathcal{D}|\theta_j^{(m)})$ for each sample $\theta_j^{(m)}$. We evaluate the likelihood once off-line and store the result for testing.

If Θ_m are not disjoint, we can assume new sub-regions $\tilde{\Theta}_m$ which are disjoint and compute the estimator as following

$$\hat{f}_m = \frac{1}{\overline{n}_m} \sum_{m=1}^{M} \sum_{j=1}^{K_m} f(\theta_m^{(j)}) \mathbf{1}_{\tilde{\Theta}_m}(\theta_j^{(m)})$$

where

$$\overline{n}_m = \sum_{m=1}^{M} \sum_{j=1}^{K_m} \mathbf{1}_{\tilde{\Theta}_m}(\theta_j^{(m)})$$

and $\mathbf{1}_{\tilde{\Theta}_m}(\theta_j^{(m)})$ equals 1 only when $\theta_j^{(m)} \in \tilde{\Theta}_m$. By doing so, our estimator still holds even if Θ_m are not disjoint.

D.3 Theoretical Analysis under Convex Assumption

Firstly, we introduce the following SDE

$$\mathrm{d}\theta_t = -\nabla U(\theta_t)\mathrm{d}t + \sqrt{2}\mathrm{d}\mathcal{W}_t \ , \tag{D.5}$$

Let ν_t denote the distribution of θ_t, and the stationary distribution of (D.5) be $p(\theta|\mathcal{D})$, which means $\nu_\infty = p(\theta|\mathcal{D})$.

However, the exact evaluation of the gradient ∇U is computationally expensive. Hence, we need to adopt noisy evaluations of ∇U. For simplicity, we assume that at any point θ_k, we can observe the value

$$\nabla \tilde{U}_k = \nabla U(\theta_k) + \zeta_k$$

where $\zeta_k : k = 0, 1, 2, ...$ is a sequence of random (noise) vectors. Then the algorithm is defined as:

$$\theta_{k+1} = \theta_k - \alpha_{k+1}\nabla\tilde{U}_k + \sqrt{2\alpha_{k+1}}\xi_{k+1} \tag{D.19}$$

Further, let μ_k denote the distribution of θ_k.

Following the existing work in [35], we adopt the following standard assumptions summarized in Assumption 6,

Assumption 6.

- *For some positive constants m and M, it holds*

$$U(\theta) - U(\theta') - \nabla U(\theta')^T(\theta - \theta') \geq (m/2)\|\theta - \theta'\|_2^2$$
$$\|\nabla U(\theta) - \nabla U(\theta')\|_2 \leq M\|\theta - \theta'\|_2$$

  for any $\theta, \theta' \in \mathbb{R}^d$

- *(bounded bias)* $\mathbb{E}[\|\mathbb{E}(\zeta_k|\theta_k)\|_2^2] \leq \delta^2 d$
- *(bounded variance)* $\mathbb{E}[\|\zeta_k - \mathbb{E}(\zeta_k|\theta_k)\|_2^2 \leq \sigma^2 d$
- *(independence of updates)* ξ_{k+1} *in* (D.19) *is independent of* $(\zeta_1, \zeta_2, ..., \zeta_k)$

D.3.1 Theorem

Under Assumption 6 in the appendix and $\alpha_0 \in (0, \frac{1}{m} \wedge \frac{2}{M})$, if we define the α_{min} as $\frac{\alpha_0}{2}\left[\cos\left(\frac{\pi \bmod(\lceil K/M\rceil - 1, \lceil K/M\rceil)}{\lceil K/M\rceil}\right) + 1\right]$, we can derive the the following bounds.

If $m\alpha_{min} + M\alpha_0 \leq 2$, then $W_2(\mu_{k+1}, \nu_\infty) \leq$

$$(1 - m\alpha_{min})^K W_2(\mu_0, \nu_\infty) + \frac{(1.65M\alpha_0^{3/2} + \alpha_0\delta)d^{1/2}}{m\alpha_{min}} + \frac{\delta^2\alpha_0 d^{1/2}}{1.65M\alpha_0^{1/2} + \delta + \sqrt{m\alpha_{min}}\delta}.$$

If $m\alpha_{min} + M\alpha_0 > 2$, then $W_2(\mu_{k+1}, \nu_\infty) \leq$

$$(1-(2-M\alpha_0))^K W_2(\mu_0,\nu_\infty) + \frac{(1.65M\alpha_0^{3/2} + \alpha_0\delta)d^{1/2}}{2-M\alpha_0} + \frac{\delta^2\alpha_0 d^{1/2}}{1.65M\alpha_0^{1/2} + \delta + \sqrt{2-M\alpha_0}\delta},$$

where the M, m, δ, σ are some positive constants defined in Assumption 6

D.3.2 Proof

Proof. According to the (5.1), we can find that the stepsize α_k varies from α_0 to α_{min}, where α_{min} is defined as $\alpha_{min} \triangleq \frac{\alpha_0}{2}\left[\cos\left(\frac{\pi \bmod(\lceil K/M\rceil - 1, \lceil K/M\rceil)}{\lceil K/M\rceil}\right) + 1\right]$. When $0 < \alpha_0 < \min(2/M, 1/m)$, it is easy for us to know that $0 < \alpha_k < \min(2/M, 1/m)$ for every $k > 0$. Then we can derive that all the $\rho_k \triangleq \max(1 - m\alpha_k, M\alpha_k - 1)$ will satisfy $0 < \rho_k < 1$. Now according to the Proposition 2 in [35], we can derive the result that

$$W_2(\mu_k + 1, \nu_\infty)^2 \leq \{\rho_{k+1}W_2(\mu_k,\nu_\infty) + 1.65M(\alpha_{k+1}^3 d)^{1/2} + \alpha_{k+1}\delta\sqrt{p}\}^2 + \delta^2\alpha_{k+1}^2 d \tag{D.20}$$

Then we will use another lemma derived from [35].

Lemma 9. *If A,B,C are non-negative numbers such that A ∈ (0,1) and the sequence of non-negative numbers y_k satisfies the following inequality*

$$y_{k+1}^2 \leq [(1-A)y_k + C]^2 + B^2$$

for every integer $k > 0$. Then,

$$y_k \leq (1-A)^k y_0 + \frac{C}{A} + \frac{B^2}{C + \sqrt{A}B}$$

Using Lemma 9, we can finish our proof now.

- If $m\alpha_{min} + M\alpha_0 \leq 2$, the ρ_k will satisfy $\rho_k \leq 1 - m\alpha_{min}$ for every $k > 0$. Then the (D.20) will turn into

$$W_2(\mu_{k+1}, \nu_\infty)^2$$
$$\leq \{(1 - m\alpha_{min})W_2(\mu_k, \nu_\infty) + 1.65M(\alpha_0^3 d)^{1/2} + \alpha_0 \delta d^{1/2}\}^2 + (\delta\alpha_0 d^{1/2})^2$$

  for every $k > 0$. Then we can set $A = m\alpha_{min}$, $C = 1.65M(\alpha_0^3 d)^{1/2} + \alpha_0\delta d^{1/2}$, $B = \delta\alpha_0 d^{1/2}$ and we can get the result.

- If $m\alpha_{min} + M\alpha_0 > 2$, the ρ_k will satisfy $\rho_k \leq M\alpha_0 - 1$ for every $k > 0$. Then the (D.20) will turn into

$$W_2(\mu_{k+1}, \nu_\infty)^2$$
$$\leq \{[1 - (2 - M\alpha_0)]W_2(\mu_k, \nu_\infty) + 1.65M(\alpha_0^3 d)^{1/2} + \alpha_0 \delta d^{1/2}\}^2 + (\delta\alpha_0 d^{1/2})^2$$

  for every $k > 0$. Then we can set $A = 2 - M\alpha_0$, $C = 1.65M(\alpha_0^3 d)^{1/2} + \alpha_0\delta d^{1/2}$, $B = \delta\alpha_0 d^{1/2}$ and we can get the result.

□

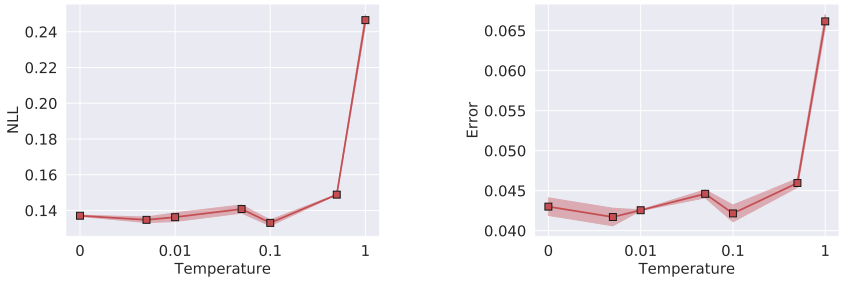
D.4 Tempering in Bayesian Neural Networks

Tempering is common in modern Bayesian deep learning, for both variational inference and MCMC approaches [85, 112, 48]. In general, tempering reflects the belief that the model capacity is misspecified. This combination of beliefs with data is what shapes the posterior we want to use to form a good predictive distribution.

Although we use the prescribed temperature in pSGLD [85] for all neural network experiments in the main text ($T \approx 0.0045$), we here investigate the effect of temperature T on performance. We show negative log-likelihood (NLL) and classification

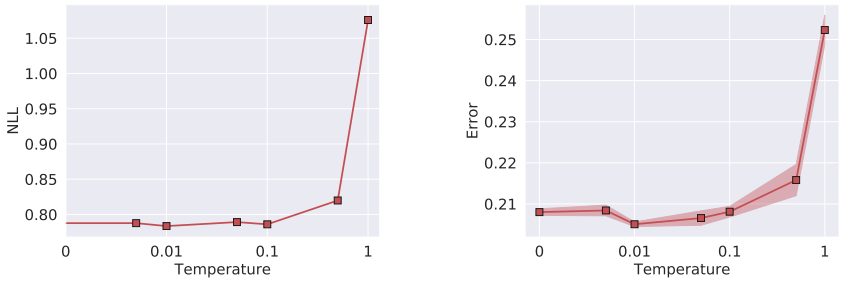
error as a function of temperature on CIFAR-10 and CIFAR-100 using cSGLD with the same setup as in Section 5.4. We consider $T \in [1, 0.5, 0.1, 0.05, 0.01, 0.005, 0]$. Figure D.1 and D.2 show the results on CIFAR-10 and CIFAR-100, respectively. On CIFAR-10, the best performance is achieved at $T = 0.1$ with NLL 0.1331 and error 4.22%. On CIFAR-100, the best performance is achieved at $T = 0.01$ with NLL 0.7835 and error 20.53%. We find that the optimal temperature is often less than 1. We hypothesize that this result is due to the model misspecification common to neural networks.

Indeed, modern neural networks are particularly overparametrized. Tempering enables one to use a model with similar inductive biases to a modern neural network, but with a more well calibrated capacity (which is especially important when we are doing Bayesian integration instead of optimization). Indeed, we show that by sampling from the tempered posterior, we outperform optimization. Learning the amount of tempering by cross-validation is a principled way of aligning the tempering procedure with representing a reasonable posterior. We have shown that sampling with cSGMCMC with tempering helps in terms of both NLL and accuracy, which indicates that we are finding a better predictive distribution.



(a) Test negative log-likelihood (b) Test error

Figure D.1: NLL and error (%) as a function of temeprature on CIFAR-10 using cSGLD. The best performance of both NLL and error is achieved at $T = 0.1$.



(a) Test negative log-likelihood (b) Test error

Figure D.2: NLL and error (%) as a function of temeprature on CIFAR-100 using cSGLD. The best performance of both NLL and error is achieved at $T = 0.01$.

D.5 Additional Experimental Results and Setting Details

Sensitivity of Hyperparameters

Compared to SG-MCMC, there are two additional hyperparameters in Algorithm 9: the number of cycles M and the proportion of exploration stage β. We now study how sensitive they are when comparing to the parallel MCMC. With the same setup as in Section 5.4, We compare our method with M cycles and L epochs per cycle with running M chains parallel MCMC for L epochs. The training budget is 200 epochs. In Table 5.2, $M = 4$ and $\beta = 0.8$ on CIFAR-10. We compare cSGLD and parallel SGLD with smaller and larger values of M and β. In Table D.1, we see that the conclusion that cSG-MCMC is better than parallel SG-MCMC holds with different values of M and β.

Hyperparameters Setting in Practice

Given the training budget, there is a trade-off between the number of cycles M and the cycle length. We find that it works well in practice by setting the cycle length such that the model with optimization methods will be close to a mode after

running for that length. (e.g. the cycle length for CIFAR-10 is 50 epochs. The model optimized by SGD can achieve about 5% error after 50 epochs which means the model is close but not fully converge to a mode after 50 epochs.) Once the cycle length is fixed, M is fixed. β needs tuning for different tasks by cross-validation. Generally, β needs to be tuned so that the sampler has enough time to reach a good region before starting sampling.

	$M=2, \beta=0.8$	$M=5, \beta=0.8$	$M=4, \beta=0.7$	$M=4, \beta=0.9$
cSGLD	4.27	4.33	4.08	4.34
Parallel SGLD	5.49	7.38	6.03	6.03

Table D.1: Comparison of test error (%) between cSG-MCMC and parallel algorithm with varying values of hyperparameters on CIFAR-10.

Synthetic Multimodal Distribution

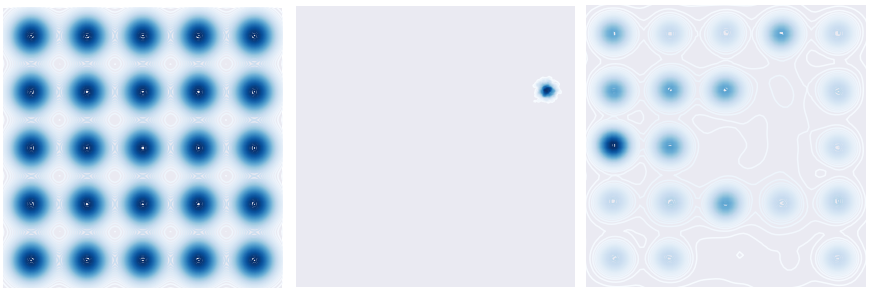
The density of the distribution is

$$F(x) = \sum_{i=1}^{25} \lambda \mathcal{N}(x|\mu_i, \Sigma),$$

where $\lambda = \frac{1}{25}$, $\mu = \{-4, -2, 0, 2, 4\}^\top \times \{-4, -2, 0, 2, 4\}$, $\Sigma = \begin{bmatrix} 0.03 & 0 \\ 0 & 0.03 \end{bmatrix}$.

In Figure D.3, we show the estimated density for SGLD and cSGLD in the non-parallel setting.

To quantitatively show the ability of different algorithms to explore multi-modal distributions, we define the *mode-coverage* metric: when the number of samples falling within the radius r of a mode center is larger than a threshold $\bar{n}$, we consider this mode covered. On this dataset, we choose $r = 0.25$ and $\bar{n} = 100$. Table D.2 shows the mode-coverage for several algorithms, based on 10 different runs.



(a) Target (b) SGLD (c) cSGLD

Figure D.3: Sampling from a mixture of 25 Gaussians in the non-parallel setting. With a budget of 50K samples, traditional SGLD has only discovered one of the 25 modes, while our proposed cSGLD has explored significantly more of the distribution.

Algorithm	**Mode coverage**
SGLD	1.8±0.13
cSGLD	**6.7**±0.52
Parallel SGLD	18±0.47
Parallel cSGLD	**24.4**±0.22

Table D.2: Mode coverage over 10 different runs, ± standard error.

Bayesian Logistic Regression

We consider Bayesian logistic regression (BLR) on three real-world datasets from the UCI repository: *Australian* (15 covariates, 690 data points), *German* (25 covariates, 1000 data points) and *Heart* (14 covariates, 270 data points). For all experiments, we collect 5000 samples with 5000 burn-in iterations. Following the settings in [85], we report median *effective sample size* (ESS) in Table D.3.

Note that BLR is *unimodal* in parameter space. We use this experiment as an adversarial situation for cSG-MCMC, which we primarily designed to explore multiple modes. We note that even in the unimodal setting, cSG-MCMC more effectively explores the parameter space than popular alternatives. We can also use these experiments to understand how samplers respond to varying parameter

dimensionality and training set sizes.

Overall, cSG-MCMC dramatically outperforms SG-MCMC, which demonstrates the fast mixing rate due to the warm restarts. On the small dataset *Heart*, SGHMC and cSGHMC achieve the same results, because the posterior of BLR on this dataset is simple. However, in higher dimensional spaces (*e.g., Australian* and *German*), cSG-MCMC shows significantly higher ESS; this result means that each cycle in cSG-MCMC can characterize a different region of the posteriors, combining multiple cycles yields more accurate overall approximation.

	Australian	**German**	**Heart**
SGLD	1676	492	2199
cSGLD	2138	978	2541
SGHMC	1317	2007	**5000**
cSGHMC	**4707**	**2436**	**5000**

Table D.3: Effective sample size for samples for the unimodal posteriors in Bayesian linear regression, obtained using cyclical and traditional SG-MCMC algorithms, respectively.

For both cSGLD and cSGHMC, $M = 100$, $\beta = 0.01$. For cSGLD, $\alpha_0 N = 1.2, 0.5, 1.5$ for Austrilian, German and Hear respectively. For cSGHMC $\alpha_0 N = 0.5, 0.3, 1.0$ for Austrilian, German and Hear respectively. For SG-MCMC, the stepsize is a for the first 5000 iterations and then switch to the decay schedule (5.1) with $b = 0$, $\gamma = 0.55$. $aN = 1.2, 0.5, 1.5$ for Austrilian, German and Hear respectively for SGLD and $aN = 0.5, 0.3, 1.0$ for Austrilian, German and Hear respectively for SGHMC. $\eta = 0.5$ in cSGHMC and SGHMC.

Assume that we collect $\{\theta_b\}_{b=1}^{B}$ samples. *Effective sample size* (ESS) is computed by

$$\text{ESS} = \frac{B}{1 + 2\sum_{s}^{B-1}(1 - \frac{s}{B})\rho_s}$$

where ρ_s is estimated by

$$\hat{\rho}_s = \frac{1}{\hat{\sigma}^2(B-s)} \sum_{b=s+1}^{B} (\theta_b - \hat{\mu})(\theta_{b-s} - \hat{\mu})$$

Similar to [70], $\hat{\sigma}^2$ and $\hat{\mu}$ are obtained by running an independent sampler. We use HMC in this section.

Bayesian Neural Networks

For SG-MCMC, the stepsize decays from 0.1 to 0.001 for the first 150 epochs and then switch to the decay schedule (5.1) with $a = 0.01, b = 0$ and $\gamma = 0.5005$. $\eta = 0.9$ in cSGHMC, Snapshot-SGDM and SGHMC.

Uncertainty Evaluation

For both cSG-MCMC and Snapshot, $M = 4$. $\beta = 0.8$ in cSG-MCMC. $\alpha_0 N = 0.01$ and 0.008 for cSGLD and cSGHMC respectively. For SG-MCMC, the stepsize is a for the first 50 iterations and then switch to the decay schedule (5.1) with $b = 0$, $\gamma = 0.5005$. $aN = 0.01$ for SGLD and $aN = 0.008$ for SGHMC. $\eta = 0.5$ in cSGHMC, Snapshot-SGDM and SGHMC.

APPENDIX E

SECTION 6 META-LEARNING DIVERGENCES OF VARIATIONAL INFERENCE

E.1 Computing Equation (6.3) in Practice

With dataset $\mathcal{D}$, the density ratio in f-divergence becomes $\frac{p(\theta|D)}{q_\phi(\theta)} = \frac{p(D|\theta)p(\theta)}{q_\phi(\theta)p(D)}$. We estimate $p(D)$ through importance sampling and MC approximation: $p(D) = E_{\theta\sim p(\theta)}[p(D|\theta)] = E_{\theta\sim q_\phi(\theta)}[\frac{p(D|\theta)p(\theta)}{q_\phi(\theta)}] \approx \frac{1}{K}\sum_{k=1}^{K}\frac{p(D|\theta_k)p(\theta_k)}{q_\phi(\theta_k)}$ where $\theta_k \sim q_\phi(\theta)$. After doing this, the density ratio becomes $\frac{p(\theta_k|D)}{q_\phi(\theta_k)} = \frac{p(D|\theta_k)p(\theta_k)}{q_\phi(\theta_k)} \Big/ \frac{1}{K}\sum_{k=1}^{K}\frac{p(D|\theta_k)p(\theta_k)}{q_\phi(\theta_k)}$ which can be regarded as a self-normalized estimator, similar to the normalization importance weight in [90]. A self-normalized estimator generally helps stabilize the training especially at the beginning. We use Eq.(6.3) with this estimator and stochastic approximation of gradients for all experiments except for the mixture of Gaussians task where we can directly compute $p(\theta)/q_\phi(\theta)$.

E.2 Effect of Hyperparameter B

Similar to other MAML-based algorithms [44, 45, 78], the cost of our method increases as hyperparameter B increases and the value of B could potentially affect the results. As in prior work, we treat B as a hyperparameter and tune it for each task. Empirically, we found setting $B = 1$ is enough for most tasks we considered in the experimental section. For example, we also tried $B = 2, 5$ for meta-α in Section 6.3 and found that they gave similar values of learned α as $B = 1$ ($\alpha = 0.10, 0.13, 0.16$ for $B = 1, 2, 5$ respectively). Setting B larger will be costly and even cause gradients to be problematic due to requiring higher-order derivatives.

We may combine our methods with recent techniques in meta-learning [46, 47, 121] to allow large B, which is an interesting future work.

E.3 Additional Experimental Results and Setting Details

Task Distribution $p(\mathcal{T})$

When the number of training tasks is finite (which is often the case in practice) such as image generation with MNIST (Section 6.3) and recommender system with MovieLens (Section 6.3), the task distribution $p(\mathcal{T})$ is defined as a uniform distribution over all training tasks for both meta-D (Algorithm 10) and meta-$D\&\phi$ (Algorithm 11). When the number of training tasks is infinite such as Gaussian mixture approximation (Section 6.3) and sinusoid regression (Section 6.3), we use a uniform distribution over all training tasks as $p(\mathcal{T})$ for meta-$D\&\phi$ but a uniform distribution over a subsampled set of training tasks as $p(\mathcal{T})$ for meta-D (the set size is 10 and 20 for Gaussian mixture approximation and sinusoid regression respectively). This is to avoid storing too many models since meta-D allows each task T_i has its own model ϕ_i.

Parameterization of f-Divergence in Practice

Based on the Proposition 1 and 2, we can parameterize f-divergence by parameterizing $g(t) = t^2 f''(t) = \exp(h_\eta(t))$ where h_η is a neural network with parameter η. However, this way of parameterization makes it hard to learn the divergences whose $g(t)$ is very small when t is small (because $h(t)$ has to output negative numbers with large absolute values), such as Renyi divergence with $\alpha \approx 0$. These

kinds of divergences behave like approximating the expectation in Eq.(6.3) only with θ whose $p(\theta)/q_\phi(\theta)$ is large, which is important for modeling bimodal and heteroscedastic distributions [39].

To alleviate this issue, we can instead parameterize $f''(t) = \exp(h_\eta(t))$, then $g(t)$ in Eq.(6.3) becomes $t^2 \exp(h_\eta(t))$. It is easy to see that this parameterization solves the above issue due to t^2 which becomes small when t is small. However, it is hard to learn the divergences that put similar weights to MC samples (e.g. standard $\mathrm{KL}(q\|p)$, which gives the equal weights). These two ways of parameterization are statistically equivalent but have different inductive bias. Parameterizing f'' tends to learn a divergence that puts different weights to MC samples according to $p(\theta)/q_\phi(\theta)$ (due to t^2). On the other hand, parameterizing $g(t)$ tends to give relatively similar weights to MC samples. In the experiments, we parameterize $g(t)$ when the learned α from meta-α is close to 1 (Section 6.3 and 6.3) and parameterize $f''(t)$ when the learned α is close to 0 (Section 6.3 and 6.3). We found this strategy works well in practice.

Model Architecture for f-divergence

On all experiments, we parameterize $h_\eta(t)$ in f-divergence by a neural network with 2 hidden layers with 100 hidden units and RELU nonlinearilities. In practice, we find that pretraining $h_\eta(t)$ to be the standard KL divergence can stabilize the training at the beginning. We initialize $h_\eta(t)$ in this way for all experiments.

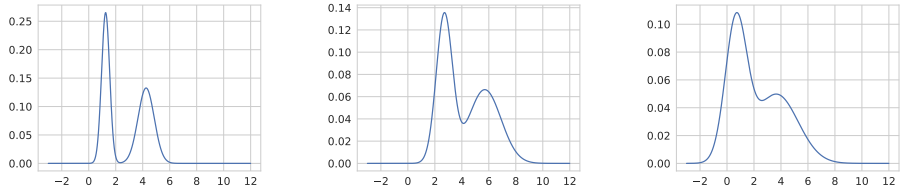


Figure E.1: Three examples of mixture of Gaussians. Each task includes approximating a mixture of Gaussians by a Gaussian distribution.

Approximate Mixture of Gaussians

In this experiment, each task is to approximate a mixture of Gaussians by a Gaussian distribution. We give examples of the mixture of Gaussians in Figure E.1. The expectation in Eq.(2.5) and (6.3) is computed by MC approximation with 1000 particles. Note that $p(\theta)$ is computable, since we know the parameters of p.

TV Distance TV is a common distance measure for probability distributions. It is defined as

$$\mathrm{TV}(p, q) = \sup_x |p(x) - q(x)| = \frac{1}{2} \int |p(x) - q(x)| \, dx.$$

For $\alpha \in (0, 1]$, TV is related to α-divergence by $\alpha \, \mathrm{TV}^2 \leq 2 \cdot D_\alpha(p \| q)$ [56].

Note that although TV belongs to a more general f-divergence family by setting $f_{\mathrm{tv}}(t) = |t - 1| / 2$, it does not belong to the f-divergence we defined in the section. Since $f_{\mathrm{TV}}(t)$ is not twice-differentiable. Therefore, we can use TV as an example to test the performance of our methods when meta-loss is beyond α- and f-divergence.

Bayesian Optimization We used a standard setup of BO, following [134]. To ensure fair comparisons, we implemented BO through a public and stable library [1].

[1] `https://github.com/fmfn/BayesianOptimization`.

We set the search region for BO to be $\alpha \in [0, 3]$. The acquisition function is the upper confidence bound with kappa 0.1. We used the same data for training meta-D for BO. Specifically, the objective function that BO minimizes is the meta-loss ($D_{0.5}$ or TV). Every time BO selects an α, we train 10 models with that α-divergence on the training sets of 10 training tasks respectively and get the mean of log-likelihood on the test sets of the 10 training tasks. Each time the model is trained for 2000 iterations. It is possible to choose the best α for each test task by BO, i.e. every time we have a new test task we run BO to select an α for this task. However, by doing this, we are not able to extract any common knowledge from the previous tasks and running BO for each task could be very expensive. We did not include this baseline because it does not satisfy the meta-learning setting and the cost will be much higher than our methods.

Analytical Expression of $h_f(t)$ When f-divergence is $D_{0.5}$, the f function is $f(t) = \frac{t^{0.5}}{-0.5^2}$. Then we can write out the corresponding $h_f(t)$ as

$$h_f(t) = \log g_f(t) = \log f''(t) + 2 \log t = 0.5 \log t.$$

Because the definition of f-divergence is invariant to constant scaling of the function f, i.e. f and af define the same divergence for $\forall a > 0$, we consider the corresponding $h_f(t)$ for af which is

$$h_f(t) = 0.5 \log t + \log a.$$

In Figure 2, we compare the learned $h_\eta(t)$ and the ground truth $h_f(t)$. We found that the learned $h_\eta(t)$ is very close to $0.5 \log t + 1.25$, which means that our method has learned the optimal divergence $D_{0.5}$. We conjecture that the constant a for the learned f is related to the learning rate. In fact, this gives f-divergence the ability to automatically adjust the learning rate through the scaling constant. For example, the constant a is $\exp(1.25) > 1$ which may suggest that the learning rate

β is a bit small. Note that α-divergence does not have this ability. We plot f for $t \in [0, 3]$ in Figure 6.2 since we find most t lie in this range.

Additional Experimental Results For meta-D, we report in Table E.1 the meta-losses on 10 test tasks, which are obtained by executing the learned divergence minimization algorithm for 2000 iterations. The error bar is large due to the large variance among different tasks, so we report the ranking in Table 6.2, similar to [93], to clearly show the adavantages of meta-D over BO. Similarly, we also report the meta-losses for meta-$D\&\phi$ over 10 tasks in Table E.2.

Table E.1: Meta-D on MoG: value of meta-loss over 10 test tasks.

Methods	$\alpha = 0.5$	TV
ground truth	0.0811±0.0277	-
meta-α	0.0811±0.0277	0.2143±0.0936
meta-f	**0.0795**±0.0301	**0.2020**±0.1024
BO (8 iters)	0.0833±0.0289	0.2203±0.0898
BO (16 iters)	0.0811±0.0277	0.2143±0.0936

Table E.2: Meta-$D\&\phi$ on MoG: value of meta-loss over 10 test tasks.

Methods\Meta-loss	$\alpha = 0.5$ (20 iters)	TV (20 iters)	$\alpha = 0.5$ (100 iters)	TV (100 iters)
VI&ϕ	0.1237±0.0539	0.2572±0.1137	0.0905±0.0332	0.2321±0.0961
meta-$\alpha\&\phi$	0.1207±0.0500	0.2462±0.1043	0.0879±0.0305	**0.2263**±0.0936
meta-$f\&\phi$	**0.0793**±0.0237	**0.2344**±0.0955	**0.0784**±0.0332	0.2301±0.0949

Regression Tasks with Bayesian Neural Networks

Each task includes a regression problem on a sinusoid wave; see Figure E.2 for examples of the sinusoid waves. The BNN model is a two-layer neural network with hidden layer size 20 and RELU nonlinearities. For meta-learning divergence

only, the training set size is 1000 and is obtained by sampling $x \in [-4, 4]$ uniformly. We use $K = 100$ and batch size 40 of which 20 data points are for updating ϕ_i and 20 points are for updating η. We train meta-D for 1000 epochs. To evaluate the performance, we train the model with the learned divergence and VI respectively on new tasks for 1000 epochs. For learning both the divergence objective and initial variational parameters, we sample 20 tasks each time where each task has 40 data points. We use 20 points for updating ϕ_i and the other 20 points for updating divergence η and the shared initialization ϕ. $B = 4$ for meta-$\alpha\&\phi$. To evaluate, we start with the learned initialization and train the variational parameters with the learned divergence for 300 epochs.

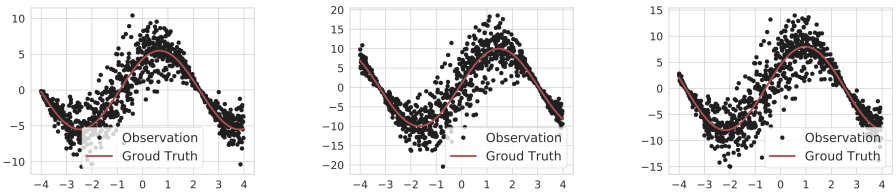


Figure E.2: Three examples of sinusoid waves. Each task includes a regression on a sinusoid wave.

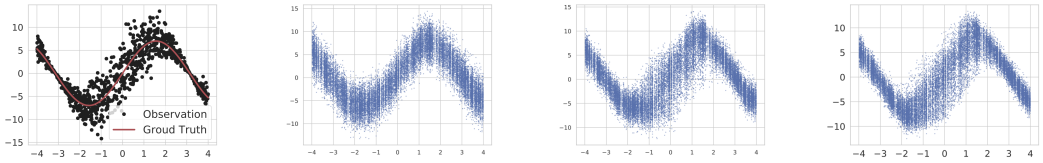
Additional Experimental Results We provide the learned value of α from meta-α and meta-$\alpha\&\phi$ in Table E.3. The predictive distributions on an example test task are given in Figure E.3. Similar to the results of meta-D, meta-$D\&\phi$ is also able to model heteroskedastic noise while VI$\&\phi$ cannot.

Table E.3: Learned value of α of meta-α and meta-$\alpha\&\phi$ on sinusoid regression.

	meta-α	meta-$\alpha\&\phi$
α	0.10	0.12

Table E.4: Learned value of α of meta-α and meta-$\alpha\&\phi$ on MNIST.

	meta-α	meta-$\alpha\&\phi$
α	0.14	0.80



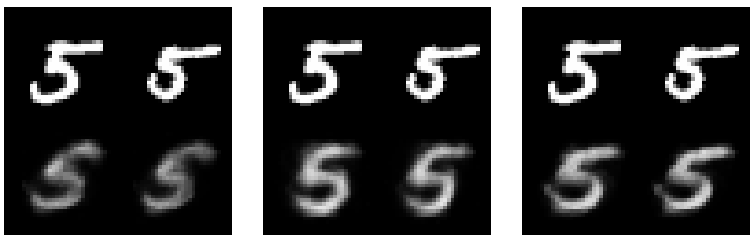
(a) Ground Truth (b) VI&ϕ (c) meta-α&ϕ (d) meta-f&ϕ

Figure E.3: Meta-D&ϕ on sin: the predictive distribution on a sinusoid wave.

Image Generation with Variational Auto-Encoders

During meta-training, we sample 128 images of each task/digit and use half of the images to compute Eq.(6.1) and the other half for computing the meta-loss. The number of training epochs is 600. We set $K = 10$. For all methods, we use the same architecture (100 hidden units and 3 latent variables) and the marginal log-likelihood estimator as in [79]. We report the learned value of α from meta-α and meta-α&ϕ in Table E.4. The meta-loss is the negative marginal log-likelihood.

Examples of Reconstructed Images Besides comparing marginal likelihood in Section 6.3, we visualize the reconstructed images on two examples of digit 5. As shown in Fig E.4, our methods produce higher quality of images because the reconstructed images are sharper and are able to capture different writing styles whereas the images from standard VI are blurry.



(a) VI&ϕ (b) meta-α&ϕ (c) meta-f&ϕ

Figure E.4: Reconstructed images of digit 5 by Meta-D&ϕ.

Recommender System

We split the users into seven age groups: under 18, 18-24, 25-34, 35-44, 45-49, 50-55 and above 56, and regard predicting the ratings of users within the same age group as a task since the users with similar age may have similar preferences. We select 4 age groups (under 18, 25-34, 45-49, above 56) as training tasks, and use the remaining as test tasks. The meta-loss is the negative log-likelihood as used in [92].

For meta-D (Algorithm 10), during meta-training, we sample 100 users per task (400 users in total) and use half of the observed ratings to compute Eq.(6.1) and the other half for computing the meta-loss. The number of training epochs is 400. During meta-testing, we use 90%/10% training-test split for the three test tasks and train p-VAE with the learned divergence.

For meta-$D\&\phi$ (Algorithm 11), we compare our method with getting a p-VAE model initialization only, which can be regarded as a combination of MAML and p-VAE (denoted VI$\&\phi$). During evaluation, we apply 60%/40% training-test split for the test tasks and train the learned p-VAE model with the learned divergence.

Additional Experimental Results We provide the learned value of α from meta-α and meta-$\alpha\&\phi$ in Table E.5. And we visualize the learned $h_\eta(t)$ from meta-f and meta-$f\&\phi$ in Figure E.5. Besides the test log-likelihood, there are other popular evaluation metric being used in recommender system and sometimes they are not consistent with each other. Therefore, we also evaluate the performance of our method in terms of other common metrics: test root mean square error (RMSE) and test mean absolute error (MAE). For both metrics, our methods performs better than the baseline in the setting of learning inference algorithm and the setting of learning inference algorithm and model parameters (see Figure E.6 and

E.7).

Table E.5: Learned value of α of meta-α and meta-$\alpha\&\phi$ on MovieLens.

	meta-α	meta-$\alpha\&\phi$
α	0.90	1.06

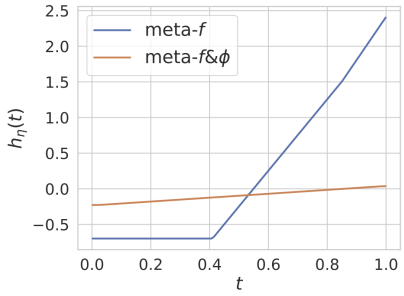


Figure E.5: Visualization of learned $h_\eta(t)$ from meta-f and meta-$f\&\phi$ on MovieLens.

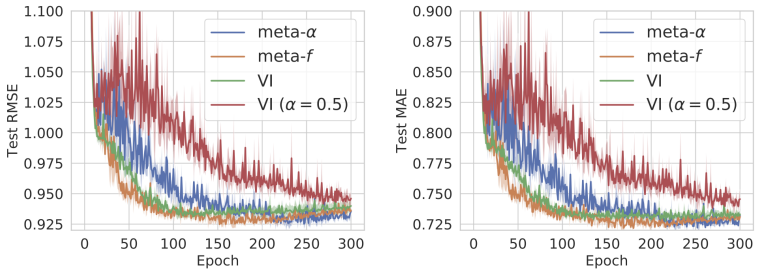


Figure E.6: Meta-D on ML: Comparison of meta-D and VI in terms of test RMSE and test MAE.

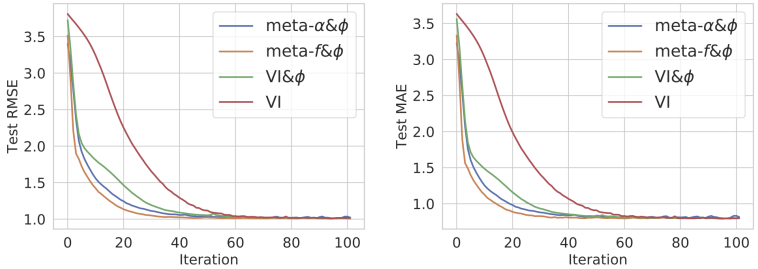


Figure E.7: Meta-$D\&\phi$ on ML: Comparison of meta-$D\&\phi$ and VI$\&\phi$ in terms of test RMSE and test MAE.

Comparison with MLAP [5]

Meta-Learning by Adjusting Priors (MLAP) is a method that meta-learns the Bayesian prior. We compared our method with it to show the importance of learning divergence. We tested on the permuted pixels experiment on MNIST, following the same experimental setup in [5]. As this is a few-shot learning setup, we run meta-$\alpha\&\phi$ (meta-learning divergences and variational parameter). Our method attained test error 2.97% which outperformed the best result 3.40% (attained by MLAP-M) in [5] significantly. This further demonstrates the importance of learning divergence and the effectiveness of our method on finding the suitable divergence.

BIBLIOGRAPHY

[1] José A Adell and Pedro Jodrá. Exact Kolmogorov and total variation distances between some familiar discrete distributions. *Journal of Inequalities and Applications*, 2006(1):64307, 2006.

[2] Sungjin Ahn, Babak Shahbaba, and Max Welling. Distributed stochastic gradient MCMC. In *ICML*, 2014.

[3] Shigeki Aida. Uniform positivity improving property, sobolev inequalities, and spectral gaps. *Journal of functional analysis*, 158(1):152–185, 1998.

[4] Shun-ichi Amari. *Differential-geometrical methods in statistics*, volume 28. Springer Science & Business Media, 2012.

[5] Ron Amit and Ron Meir. Meta-learning by adjusting priors based on extended PAC-Bayes theory. In *International Conference on Machine Learning*, pages 205–214. PMLR, 2018.

[6] Christophe Andrieu and Gareth O. Roberts. The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, 37(2):697–725, 2009.

[7] Peter Auer, Mark Herbster, and Manfred K Warmuth. Exponentially many local minima for single neurons. In *NIPS*, 1996.

[8] Robert Bamler, Cheng Zhang, Manfred Opper, and Stephan Mandt. Perturbative black box variational inference. In *Advances in Neural Information Processing Systems*, pages 5079–5088, 2017.

[9] Marco Banterle, Clara Grazian, Anthony Lee, and Christian P Robert. Accelerating Metropolis-Hastings algorithms by delayed acceptance. *Foundations of Data Science*, 1:103, 2019.

[10] Rémi Bardenet, Arnaud Doucet, and Chris Holmes. On Markov chain Monte Carlo methods for tall data. *The Journal of Machine Learning Research*, 18(1):1515–1557, 2017.

[11] Rémi Bardenet, Arnaud Doucet, and Chris Holmes. Towards scaling up Markov chain Monte Carlo: an adaptive subsampling approach. In *International Conference on Machine Learning*, 2014.

[12] Andrew R Barron and Thomas M Cover. Minimum complexity density estimation. *IEEE transactions on information theory*, 37(4):1034–1054, 1991.

[13] Joris Bierkens, Paul Fearnhead, Gareth Roberts, et al. The zig-zag process and super-efficient sampling for Bayesian analysis of big data. *The Annals of Statistics*, 47(3):1288–1320, 2019.

[14] Christopher M Bishop. *Pattern recognition and machine learning*. Springer Science+ Business Media, 2006.

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

[16] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *International Conference on Machine Learning*, 2015.

[17] François Bolley and Cédric Villani. Weighted csiszár-kullback-pinsker inequalities and applications to transportation inequalities. In *Annales de la Faculte des sciences de Toulouse*. Université Paul Sabatier, 2005.

[18] Alexandre Bouchard-Côté, Sebastian J Vollmer, and Arnaud Doucet. The bouncy particle sampler: A nonreversible rejection-free Markov chain Monte Carlo method. *Journal of the American Statistical Association*, 113(522):855–867, 2018.

[19] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of Markov Chain Monte Carlo*. CRC Press, 2011.

[20] AD Bruce. Universality in the two-dimensional continuous spin model. *Journal of Physics A: Mathematical and General*, 18(14):L873, 1985.

[21] Yaroslav Bulatov. Not MNIST Dataset. 2011. `http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html`.

[22] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

[23] David M Ceperley. Path integrals in the theory of condensed helium. *Reviews of Modern Physics*, 67(2):279, 1995.

[24] Changyou Chen, David Carlson, Zhe Gan, Chunyuan Li, and Lawrence Carin. Bridging the gap between stochastic gradient MCMC and stochastic optimization. In *Artificial Intelligence and Statistics*, 2016.

[25] Changyou Chen, Nan Ding, and Lawrence Carin. On the convergence of stochastic gradient MCMC algorithms with high-order integrators. In *Advances in Neural Information Processing Systems*, pages 2278–2286, 2015.

[26] Haoyu Chen, Daniel Seita, Xinlei Pan, and John Canny. An efficient minibatch acceptance test for Metropolis-Hastings. *arXiv preprint arXiv:1610.06848*, 2016.

[27] Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient Hamiltonian Monte Carlo. In *International Conference on Machine Learning*, pages 1683–1691, 2014.

[28] Tzuu-Shuh Chiang and Chii-Ruey Hwang. Diffusion for global optimization in rn. *SIAM J. Control Optim.*, pages 737–753, 1987.

[29] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, 2015.

[30] J Andrés Christen and Colin Fox. Markov chain monte carlo using an approximation. *Journal of Computational and Graphical statistics*, 14(4):795–810, 2005.

[31] Andrzej Cichocki and Shun-ichi Amari. Families of alpha-beta-and gamma-divergences: Flexible and robust measures of similarities. *Entropy*, 12(6):1532–1568, 2010.

[32] Robert Cornish, Paul Vanetti, Alexandre Bouchard-Côté, George Deligiannidis, and Arnaud Doucet. Scalable Metropolis-Hastings for exact Bayesian inference with large datasets. *International Conference on Machine Learning*, 2019.

[33] Imre Csiszár, Paul C Shields, et al. Information theory and statistics: A tutorial. *Foundations and Trends® in Communications and Information Theory*, 1(4):417–528, 2004.

[34] Arnak S Dalalyan and Avetik Karagulyan. User-friendly guarantees for the

langevin monte carlo with inaccurate gradient. *Stochastic Processes and their Applications*, 2019.

[35] Arnak S. Dalalyan and Avetik Karagulyan. User-friendly guarantees for the langevin monte carlo with inaccurate gradient. *Stochastic Processes and their Applications*, 2019.

[36] Rianne de Heide, Alisa Kirichenko, Nishant Mehta, and Peter Grünwald. Safe-bayesian generalized linear regression. *arXiv preprint arXiv:1910.09227*, 2019.

[37] Chris De Sa, Vincent Chen, and Wing Wong. Minibatch Gibbs Sampling on Large Graphical Models. In *International Conference on Machine Learning*, 2018.

[38] Guillaume Dehaene and Simon Barthelmé. Expectation propagation in the large data limit. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 80(1):199–217, 2018.

[39] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Learning and policy search in stochastic dynamical systems with Bayesian neural networks. *International Conference on Learning Representations*, 2017.

[40] Nan Ding, Youhan Fang, Ryan Babbush, Changyou Chen, Robert D Skeel, and Hartmut Neven. Bayesian sampling using stochastic gradient thermostats. In *Advances in neural information processing systems*, pages 3203–3211, 2014.

[41] Sander Dommers, Christof Kuelske, and Philipp Schriever. Continuous spin models on annealed generalized random graphs. *Stochastic Processes and their Applications*, 127(11):3719–3753, 2017.

[42] Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics letters B*, 195(2):216–222, 1987.

[43] Raaz Dwivedi, Yuansi Chen, Martin J Wainwright, and Bin Yu. Log-concave sampling: Metropolis-hastings algorithms are fast! *arXiv preprint arXiv:1801.02309*, 2018.

[44] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International*

Conference on Machine Learning-Volume 70, pages 1126–1135. JMLR. org, 2017.

[45] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pages 9516–9527, 2018.

[46] Sebastian Flennerhag, Pablo G Moreno, Neil D Lawrence, and Andreas Damianou. Transferring knowledge across learning processes. *International Conference on Learning Representations*, 2019.

[47] Sebastian Flennerhag, Andrei A Rusu, Razvan Pascanu, Hujun Yin, and Raia Hadsell. Meta-learning with warped gradient descent. *International Conference on Learning Representations*, 2020.

[48] Meire Fortunato, Charles Blundell, and Oriol Vinyals. Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798*, 2017.

[49] Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. Cyclical annealing schedule: A simple approach to mitigating KL vanishing. *NAACL*, 2019.

[50] Masatoshi Fukushima, Yoichi Oshima, and Masayoshi Takeda. *Dirichlet forms and symmetric Markov processes*, volume 19. Walter de Gruyter, 2010.

[51] Zhe Gan, Chunyuan Li, Changyou Chen, Yunchen Pu, Qinliang Su, and Lawrence Carin. Scalable bayesian learning of recurrent neural networks for language modeling. *arXiv preprint arXiv:1611.08034*, 2016.

[52] Xuefeng Gao, Mert Gürbüzbalaban, and Lingjiong Zhu. Global convergence of stochastic gradient Hamiltonian Monte Carlo for non-convex stochastic optimization: Non-asymptotic performance bounds and momentum-based acceleration. *arXiv preprint arXiv:1809.04618*, 2018.

[53] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of DNNs. In *Advances in Neural Information Processing Systems*, pages 8789–8798, 2018.

[54] Andrew Gelman, Alex Kiss, and Jeffrey Fagan. An Analysis of the New York City Police Department's Stop-and-Frisk Policy in the Context of Claims of

Racial Bias. *Journal of the American Statistical Association*, 102(479):813–823, 2007.

[55] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741, 1984.

[56] Gustavo L Gilardoni. On Pinsker's and Vajda's type inequalities for Csiszár's f-divergences. *IEEE Transactions on Information Theory*, 56(11):5377–5386, 2010.

[57] Wenbo Gong, Yingzhen Li, and José Miguel Hernández-Lobato. Meta-learning for stochastic gradient MCMC. *International Conference on Learning Representations*, 2019.

[58] Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*, 2014.

[59] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical Bayes. *International Conference on Learning Representations*, 2018.

[60] Peter J Green. Reversible jump MCMC computation and bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

[61] Ulf Grenander and Michael I Miller. Representations of knowledge in complex systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(4):549–581, 1994.

[62] Peter Grünwald, Thijs Van Ommen, et al. Inconsistency of bayesian inference for misspecified linear models, and a proposal for repairing it. *Bayesian Analysis*, 12(4):1069–1103, 2017.

[63] Martin Hairer, Andrew M Stuart, Sebastian J Vollmer, et al. Spectral gaps for a Metropolis–Hastings algorithm in infinite dimensions. *The Annals of Applied Probability*, 24(6):2455–2490, 2014.

[64] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):19, 2016.

[65] W. Keith Hastings. Monte Carlo sampling methods using Markov chains and their applications. 1970.

[66] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[67] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *ICML*, 2015.

[68] José Miguel Hernández-Lobato, Yingzhen Li, Mark Rowland, Daniel Hernández-Lobato, Thang Bui, and Richard Turner. Black-box α-divergence minimization. *International Conference on Machine Learning,*, 2016.

[69] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[70] Matthew D Hoffman and Andrew Gelman. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

[71] Alan M Horowitz. A generalized guided Monte Carlo algorithm. *Physics Letters B*, 268(2):247–252, 1991.

[72] Rein Houthooft, Yuhua Chen, Phillip Isola, Bradly Stadie, Filip Wolski, OpenAI Jonathan Ho, and Pieter Abbeel. Evolved policy gradients. In *Advances in Neural Information Processing Systems*, pages 5400–5409, 2018.

[73] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *ICLR*, 2017.

[74] K Hukushima and Y Sakai. An irreversible Markov-chain Monte Carlo method with skew detailed balance conditions. In *Journal of Physics: Conference Series*, volume 473, page 012012. IOP Publishing, 2013.

[75] Ernst Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik A Hadrons and Nuclei*, 31(1):253–258, 1925.

[76] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

[77] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[78] Taesup Kim, Jaesik Yoon, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. *Advances in Neural Information Processing Systems*, 2018.

[79] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *International Conference on Learning Representations4*, 2014.

[80] Daphne Koller, Nir Friedman, and Francis Bach. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[81] Anoop Korattikara, Yutian Chen, and Max Welling. Austerity in MCMC land: Cutting the Metropolis-Hastings budget. In *International Conference on Machine Learning*, pages 181–189, 2014.

[82] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.

[83] David A Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Society, 2017.

[84] PA W Lewis and Gerald S Shedler. Simulation of nonhomogeneous Poisson processes by thinning. *Naval research logistics quarterly*, 26(3):403–413, 1979.

[85] Chunyuan Li, Changyou Chen, David Carlson, and Lawrence Carin. Preconditioned stochastic gradient Langevin dynamics for deep neural networks. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[86] Chunyuan Li, Changyou Chen, Kai Fan, and Lawrence Carin. High-order stochastic gradient thermostats for Bayesian learning of deep models. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[87] Dangna Li and Wing H Wong. Mini-batch tempered MCMC. *arXiv preprint arXiv:1707.09705*, 2017.

[88] Ke Li and Jitendra Malik. Learning to optimize. *arXiv preprint arXiv:1606.01885*, 2016.

[89] Yingzhen Li, José Miguel Hernández-Lobato, and Richard E Turner. Stochastic expectation propagation. In *Advances in neural information processing systems*, pages 2323–2331, 2015.

[90] Yingzhen Li and Richard E Turner. Rényi divergence variational inference. In *Advances in Neural Information Processing Systems*, pages 1073–1081, 2016.

[91] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. 2016.

[92] Chao Ma, Wenbo Gong, José Miguel Hernández-Lobato, Noam Koenigstein, Sebastian Nowozin, and Cheng Zhang. Partial VAE for hybrid recommender system. In *NIPS Workshop on Bayesian Deep Learning*, 2018.

[93] Chao Ma, Sebastian Tschiatschek, Konstantina Palla, Jose Miguel Hernandez Lobato, Sebastian Nowozin, and Cheng Zhang. Eddi: Efficient dynamic discovery of high-value information with partial VAE. *International Conference on Machine Learning*, 2019.

[94] Yi-An Ma, Tianqi Chen, and Emily Fox. A complete recipe for stochastic gradient mcmc. In *Advances in Neural Information Processing Systems*, pages 2917–2925, 2015.

[95] Yi-An Ma, Tianqi Chen, Lei Wu, and Emily B Fox. A unifying framework for devising efficient and irreversible MCMC samplers. *arXiv preprint arXiv:1608.05973*, 2016.

[96] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 1992.

[97] Dougal Maclaurin and Ryan P Adams. Firefly Monte Carlo: Exact MCMC with subsets of data. In *UAI*, pages 543–552, 2014.

[98] Dougal Maclaurin and Ryan Prescott Adams. Firefly Monte Carlo: Exact MCMC with subsets of data. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[99] Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, pages 13132–13143, 2019.

[100] Jeremy R Manning, Rajesh Ranganath, Kenneth A Norman, and David M Blei. Topographic factor analysis: a Bayesian model for inferring brain networks from neural data. *PloS one*, 9(5):e94914, 2014.

[101] J. C. Mattingly, A. M. Stuart, and M. V. Tretyakov. Construction of numerical time-average and stationary measures via Poisson equations. *SIAM J. NUMER. ANAL.*, 48(2):552–577, 2010.

[102] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[103] Manon Michel, Johannes Mayer, and Werner Krauth. Event-chain Monte Carlo for classical continuous spin models. *EPL (Europhysics Letters)*, 112(2):20003, 2015.

[104] Thomas P Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 362–369. Morgan Kaufmann Publishers Inc., 2001.

[105] Tom Minka et al. Divergence measures and message passing. Technical report, Technical report, Microsoft Research, 2005.

[106] Radford M Neal. *Bayesian learning for neural networks*. New York: Springer-Verlag, 1996.

[107] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[108] Radford M Neal et al. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11):2, 2011.

[109] Arvind Neelakantan, Luke Vilnis, Quoc V Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. Adding gradient noise improves learning for very deep networks. *ICLR workship*, 2016.

[110] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM J. on Optimization*, 19(4):1574–1609, January 2009.

[111] Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

[112] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4467–4477, 2017.

[113] Sheehan Olver and Alex Townsend. Fast inverse transform sampling in one and two dimensions. *arXiv preprint arXiv:1307.1223*, 2013.

[114] Emma Pierson, Sam Corbett-Davies, and Sharad Goel. Fast threshold tests for detecting discrimination. volume 84 of *Proceedings of Machine Learning Research*, pages 96–105, Playa Blanca, Lanzarote, Canary Islands, 09–11 Apr 2018. PMLR.

[115] Renfrey Burnard Potts. Some generalized order-disorder transformations. In *Mathematical proceedings of the cambridge philosophical society*, volume 48, pages 106–109. Cambridge University Press, 1952.

[116] Matias Quiroz, Robert Kohn, Mattias Villani, and Minh-Ngoc Tran. Speeding up MCMC by efficient data subsampling. *Journal of the American Statistical Association*, 114(526):831–843, 2019.

[117] Matias Quiroz, Minh-Ngoc Tran, Mattias Villani, Robert Kohn, and Khue-Dung Dang. The block-poisson estimator for optimally tuned exact subsampling mcmc. *arXiv preprint arXiv:1603.08232*, 2016.

[118] Adrian E Raftery, Michael A Newton, Jaya M Satagopan, and Pavel N Krivitsky. Estimating the integrated likelihood via posterior simulation using the harmonic mean identity. 2006.

[119] Maxim Raginsky, Alexander Rakhlin, and Matus Telgarsky. Non-convex learning via stochastic gradient Langevin dynamics: a nonasymptotic analysis. *arXiv preprint arXiv:1702.03849*, 2017.

[120] Tom Rainforth, Adam R Kosiorek, Tuan Anh Le, Chris J Maddison, Maximilian Igl, Frank Wood, and Yee Whye Teh. Tighter variational bounds are not necessarily better. *International Conference on Machine Learning*, 2018.

[121] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine.

Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems*, pages 113–124, 2019.

[122] Rajesh Ranganath, Dustin Tran, and David Blei. Hierarchical variational models. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 324–333, 2016.

[123] Sachin Ravi and Alex Beatson. Amortized Bayesian meta-learning. *International Conference on Learning Representations*, 2019.

[124] Alfréd Rényi et al. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. The Regents of the University of California, 1961.

[125] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1530–1538, 2015.

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

[127] Gareth O Roberts and Osnat Stramer. Langevin diffusions and Metropolis-Hastings algorithms. *Methodology and computing in applied probability*, 4(4):337–357, 2002.

[128] Gareth O Roberts, Richard L Tweedie, et al. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363, 1996.

[129] Daniel Rudolf. Explicit error bounds for markov chain monte carlo. *arXiv preprint arXiv:1108.3201*, 2011.

[130] Yunus Saatchi and Andrew Gordon Wilson. Bayesian GAN. *NIPS*, 2017.

[131] Tim Salimans, David A Knowles, et al. Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8(4):837–882, 2013.

[132] Daniel Seita, Xinlei Pan, Haoyu Chen, and John Canny. An efficient minibatch

acceptance test for Metropolis-Hastings. *Uncertainty in Artificial Intelligence*, 2017.

[133] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of residual networks using large learning rates. *arXiv preprint arXiv:1708.07120*, 2017.

[134] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[135] O Stramer and RL Tweedie. Langevin-type models i: Diffusions with given stationary distributions and their discretizations. *Methodology and Computing in Applied Probability*, 1(3):283–306, 1999.

[136] Yee Whye Teh, Alexandre H Thiery, and Sebastian J Vollmer. Consistency and fluctuations for stochastic gradient Langevin dynamics. *The Journal of Machine Learning Research*, 17(1):193–225, 2016.

[137] Lloyd N Trefethen. *Approximation theory and approximation practice*, volume 128. Siam, 2013.

[138] Constantino Tsallis. Possible generalization of Boltzmann-Gibbs statistics. *Journal of statistical physics*, 52(1-2):479–487, 1988.

[139] Konstantin S Turitsyn, Michael Chertkov, and Marija Vucelja. Irreversible Monte Carlo algorithms for efficient sampling. *Physica D: Nonlinear Phenomena*, 240(4-5):410–414, 2011.

[140] Douglas N VanDerwerken and Scott C Schmidler. Parallel Markov Chain Monte Carlo. *arXiv preprint arXiv:1312.7479*, 2013.

[141] S. J. Vollmer, K. C. Zygalakis, and Y. W. Teh. (Non-)asymptotic properties of stochastic gradient Langevin dynamics. Technical Report arXiv:1501.00438, University of Oxford, UK, January 2015.

[142] Sebastian J Vollmer, Konstantinos C Zygalakis, and Yee Whye Teh. Exploration of the (non-) asymptotic bias and variance of stochastic gradient Langevin dynamics. *The Journal of Machine Learning Research*, 17(1):5504–5548, 2016.

[143] Sebastian J. Vollmer, Konstantinos C. Zygalakis, and Yee Whye Teh. Exploration of the (non-)asymptotic bias and variance of stochastic gradient langevin dynamics. *Journal of Machine Learning Research*, 17(159):1–48, 2016.

[144] Dilin Wang, Hao Liu, and Qiang Liu. Variational inference with tail-adaptive f-divergence. In *Advances in Neural Information Processing Systems*, pages 5737–5747, 2018.

[145] Tongzhou Wang, Yi Wu, Dave Moore, and Stuart J Russell. Meta-learning MCMC proposals. In *Advances in Neural Information Processing Systems*, pages 4146–4156, 2018.

[146] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688, 2011.

[147] Andrew Gordon Wilson. The case for bayesian deep learning. *arXiv preprint arXiv:2001.10995*, 2020.

[148] Andrew Gordon Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *arXiv preprint arXiv:2002.08791*, 2020.

[149] Pan Xu, Jinghui Chen, Difan Zou, and Quanquan Gu. Global convergence of Langevin dynamics based algorithms for nonconvex optimization. *arXiv preprint arXiv:1707.06618*, 2017.

[150] Zhongwen Xu, Hado P van Hasselt, and David Silver. Meta-gradient reinforcement learning. In *Advances in neural information processing systems*, pages 2396–2407, 2018.

[151] Mingzhang Yin and Mingyuan Zhou. Semi-implicit variational inference. *arXiv preprint arXiv:1805.11183*, 2018.

[152] Cheng Zhang, Judith Butepage, Hedvig Kjellstrom, and Stephan Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[153] Jianyi Zhang, Ruiyi Zhang, and Changyou Chen. Stochastic Particle-Optimization Sampling and the Non-Asymptotic Convergence Theory. *arXiv e-prints*, page arXiv:1809.01293, Sep 2018.

[154] Ruqi Zhang, A Feder Cooper, and Christopher De Sa. AMAGOLD: Amortized Metropolis adjustment for efficient stochastic gradient MCMC. *International Conference on Artificial Intelligence and Statistics*, 2020.

[155] Ruqi Zhang, A Feder Cooper, and Christopher M De Sa. Asymptotically optimal exact minibatch Metropolis-Hastings. *Advances in Neural Information Processing Systems*, 2020, **Spotlight Presentation**.

[156] Ruqi Zhang and Christopher De Sa. Poisson-minibatching for Gibbs sampling with convergence rate guarantees. *Advances in Neural Information Processing Systems*, 2019, **Spotlight Presentation**.

[157] Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. Cyclical stochastic gradient MCMC for Bayesian deep learning. *International Conference on Learning Representations*, 2020, **Oral Presentation**.

[158] Ruqi Zhang, Yingzhen Li, Christopher De Sa, Sam Devlin, and Cheng Zhang. Meta-learning divergences for variational inference. *International Conference on Artificial Intelligence and Statistics*, 2021.

[159] Y. Zhang, P. Liang, and M. Charikar. A hitting time analysis of stochastic gradient Langevin dynamics. In *COLT*, 2017.