High-Order Stochastic Gradient Thermostats for Bayesian Learning of Deep Models

Chunyuan Li[1], **Changyou Chen**[1], **Kai Fan**[2] **and Lawrence Carin**[1]
[1]Department of Electrical and Computer Engineering, Duke University
[2]Computational Biology and Bioinformatics, Duke University
chunyuan.li@duke.edu, cchangyou@gmail.com, kai.fan@stat.duke.edu, lcarin@duke.edu

Abstract

Learning in deep models using Bayesian methods has generated significant attention recently. This is largely because of the feasibility of modern Bayesian methods to yield scalable learning and inference, while maintaining a measure of uncertainty in the model parameters. Stochastic gradient MCMC algorithms (SG-MCMC) are a family of diffusion-based sampling methods for large-scale Bayesian learning. In SG-MCMC, multivariate stochastic gradient thermostats (mSGNHT) augment each parameter of interest, with a momentum and a thermostat variable to maintain stationary distributions as target posterior distributions. As the number of variables in a continuous-time diffusion increases, its numerical approximation error becomes a practical bottleneck, so better use of a numerical integrator is desirable. To this end, we propose use of an efficient symmetric splitting integrator in mSGNHT, instead of the traditional Euler integrator. We demonstrate that the proposed scheme is more accurate, robust, and converges faster. These properties are demonstrated to be desirable in Bayesian deep learning. Extensive experiments on two canonical models and their deep extensions demonstrate that the proposed scheme improves general Bayesian posterior sampling, particularly for deep models.

1 Introduction

The ability to learn abstract representations that support generalization to novel instances lies at the core of many problems in machine learning and computer vision. Human learners often can grasp concepts at multiple levels of abstraction from training examples, and make meaningful generalizations (Kemp, Perfors, and Tenenbaum 2007). Intuitively, appropriately employed prior knowledge and hierarchical reasoning are necessary in this task. Bayesian learning and inference applied to deep models may naturally possess such characterization, and potentially could take a step towards this ability (Salakhutdinov, Tenenbaum, and Torralba 2013).

Deep models come in two broad categories. The first uses stochastic hidden layers, typically deep latent variable models. This includes the deep sigmoid belief network (Mnih and Gregor 2014; Gan et al. 2015b), the variational autoencoder (Kingma and Welling 2014), and many others (Ranganath et al. 2015; Gan et al. 2015c; Pu, Yuan, and Carin 2015). The second category of deep models uses deterministic hidden layers. While the stochastic hidden units of the first category make this class of models naturally amenable to Bayesian learning (see the above references), for the second category appropriate priors on the weights of networks may be employed to consider weight uncertainty. Previous work has applied Bayesian methods to neural networks (MacKay 1992; Neal 1995), including feedforward neural networks (Blundell et al. 2015; Hernández-Lobato and Adams 2015; Korattikara et al. 2015) and convolutional neural networks (Gal and Ghahramani 2015; Li et al. 2016). Deep learning may often be interpreted as a stacking of such neural networks.

Bayesian learning and inference methods have generated significant recent research activity. Stochastic gradient Markov Chain Monte Carlo (SG-MCMC) methods (Welling and Teh 2011; Chen, Fox, and Guestrin 2014; Ding et al. 2014) are a family of Itô diffusion based algorithms that can efficiently sample target distributions, and can be applied to large datasets. In these algorithms, two approximations are made (Chen, Ding, and Carin 2015). (i) For *practical scalability*, stochastic gradients from minibatches of data are used to estimate the true gradient; (ii) For *numerical feasibility*, a numerical integration with small step is used to solve the corresponding Itô diffusion (a continuous-time Markovian process).

The first attempt at SG-MCMC was the Stochastic Gradient Langevin Dynamics (SGLD) (Welling and Teh 2011) . It is based on 1st-order Langevin dynamics. The Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) (Chen, Fox, and Guestrin 2014) extends SGLD with 2nd-order Langevin dynamics, where momentum variables are introduced into the system. In an attempt to address the problem of estimating stochastic gradient noise, the Stochastic Gradient Nośe-Hoover Thermostat (SGNHT) (Ding et al. 2014) was proposed, with one additional global thermostat variable. To further improve the efficiency of the SGNHT, a multivariate version of SGNHT (mSGNHT) was proposed by introducing multiple thermostat variables instead of a single one (Ding et al. 2014). It was shown that mSGNHT provides more adaptivity than SGNHT (Gan et al. 2015a).

By examining training with SG-MCMC algorithms, we note two issues. (i) as more variables are introduced, an accurate numerical method becomes more critical; and (ii) gradients in deep models often suffer from the vanishing/exploding problem (Bengio, Simard, and Frasconi 1994), which makes choosing a proper stepszie difficult in SG-MCMC. In this paper, we mitigate these concerns by utilizing a more accurate numerical integrator, the symmetric splitting integrator (SSI), to reduce discretization errors in mSGNHT. Furthermore, since SSI is more robust with respect to stepsizes than the default Euler integrator, it allows one to choose an appropriate stepsize much more easily. We justify that the Euler integrator used in mSGNHT is 1st-order, while the SSI is 2nd-order. Borrowing tools from (Chen, Ding, and Carin 2015), we show that mSGNHT with SSI (mSGNHT-S) converges faster and more accurately than mSGNHT with a Euler integrator (mSGNHT-E). Experiments across a wide range of model types demonstrate the utility of this method. Specifically, we consider latent Dirichlet allocation, logistic regression, deep neural networks and deep Poisson factor analysis.

2 Background

2.1 Itô Diffusion

Itó diffusion is a stochastic differential equation (SDE) defined as:

$$\mathrm{d}\mathbf{X}_t = F(\mathbf{X}_t)\mathrm{d}t + \sigma(\mathbf{X}_t)\mathrm{d}W_t \,, \qquad (1)$$

where $\mathbf{X}_t \in \mathbb{R}^n$, W_t is Brownian motion, and t is the time index. Functions $F : \mathbb{R}^n \to \mathbb{R}^n$ and $\sigma : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are assumed to satisfy the usual Lipschitz continuity condition (Knapp 2005). It has been shown that by designing appropriate functions F and σ, the stationary distribution, $\rho(\mathbf{X})$, of the Itô diffusion (1) has a marginal distribution that is equal to the posterior distribution of interest (Chen, Ding, and Carin 2015; Ma, Chen, and Fox 2015).

To formulate mSGNHT (Gan et al. 2015a; Ding et al. 2014) into the Itó diffusion (1), let $\mathbf{X} = (\boldsymbol{\theta}, \boldsymbol{p}, \boldsymbol{\xi})$, where $\boldsymbol{\theta} \in \mathbb{R}^n$ are the model parameters, $\boldsymbol{p} \in \mathbb{R}^n$ are momentums, and $\boldsymbol{\xi} \in \mathbb{R}^n$ represent the thermostats (Gan et al. 2015a)[1]. Define $U \triangleq -\left(\sum_{i=1}^N \log p(d_i|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})\right)$ as the unnormalized negative log-posterior, where $\{d_i\}$ represents the ith data sample, $p(d_i|\boldsymbol{\theta})$ the corresponding likelihood, and $p(\boldsymbol{\theta})$ the prior. For some constant $D > 0$, the mSGNHT in (Gan et al. 2015a) is shown to be in a form of Itô diffusion, with

$$F = \begin{bmatrix} \boldsymbol{p} \\ -\boldsymbol{\xi}\odot\boldsymbol{p} - \nabla_{\boldsymbol{\theta}} U \\ \boldsymbol{p}\odot\boldsymbol{p} - 1 \end{bmatrix}, \ \sigma = \sqrt{2D}\begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_n & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad (2)$$

where $\odot$ represents element-wise product, and $\mathbf{I}_n$ is the $n \times n$ identity matrix. Based on the Fokker-Planck equation (Risken 1989), the marginal stationary distribution over $\boldsymbol{\theta}$ can be shown to be $\rho(\boldsymbol{\theta}) \propto \exp(-U(\boldsymbol{\theta}))$, the posterior distribution we are interested in.

[1]$\mathbf{X}$ now is in $\mathbb{R}^{3n}$; for conciseness, we do not re-define the dimension for $\mathbf{X}$ in (1).

2.2 Euler Integrator

The continuous-time diffusion in (1) cannot be solved explicitly in general. As a result, numerical methods are required in SG-MCMCs to generate approximate samples. The standard numerical method used in SG-MCMC is the Euler integrator, which generates samples sequentially from a discrete-time approximation of (1). Specifically, conditioned on the current sample $\mathbf{X}_t$ and step size h, the next sample at time $t+1$ is generated via the rule:

$$\mathbf{X}_{t+1} = \mathbf{X}_t + F(\mathbf{X}_t)h + \sigma(\mathbf{X}_t)\boldsymbol{\zeta}_{t+1}, \ \ \zeta_{t+1} \sim \mathcal{N}(\mathbf{0}, h\mathbf{I}_n) \,.$$

In the case of mSGNHT, in each step, a stochastic gradient from a minibatch is used instead of the full gradient. We thus approximate U with $\tilde{U}_t \triangleq -\left(\frac{N}{|\mathcal{S}_t|}\sum_{i\in\mathcal{S}_t} \log p(d_i|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})\right)$ for the t-th iteration, where $\mathcal{S}_t \subset \{1, 2, \cdots, N\}$, and $|\cdot|$ is the cardinality of a set[2]. This results in the following sampling rules:

$$\begin{cases} \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \boldsymbol{p}_t h \\ \boldsymbol{p}_{t+1} &= \boldsymbol{p}_t - \nabla_{\boldsymbol{\theta}}\tilde{U}_t(\boldsymbol{\theta}_{t+1})h - \mathrm{diag}(\boldsymbol{\xi}_t)\boldsymbol{p}_t h + \sqrt{2D}\boldsymbol{\zeta}_{t+1} \\ \boldsymbol{\xi}_{t+1} &= \boldsymbol{\xi}_t + (\boldsymbol{p}_{t+1}\odot\boldsymbol{p}_{t+1} - 1)\,h \end{cases}$$

3 Symmetric Splitting Integrator for mSGNHT

The SSI has been studied in statistical physics (Leimkuhler and Matthews 2013; Leimkuhler and Shang 2015). It generalizes the idea of the *leap-frog* integrator used in the Hamiltonian Monte Carlo (Neal 2011) from the partial differential equation setting to the SDE setting. It was not until recently that SSI was introduced into machine learning to obtain a more accurate SGHMC algorithm (Chen, Ding, and Carin 2015). We adopt the idea and generalize it in this paper for mSGNHT.

The idea of SSI is to split the intractable SDE, *i.e.*, (1), into several sub-SDEs such that each sub-SDE can be solved analytically. For the mSGNHT represented in (2), it is readily split into the following sub-SDEs:

$$A: \begin{cases} \mathrm{d}\boldsymbol{\theta} &= \boldsymbol{p}\mathrm{d}t \\ \mathrm{d}\boldsymbol{p} &= 0 \\ \mathrm{d}\boldsymbol{\xi} &= (\boldsymbol{p}\odot\boldsymbol{p} - \mathbf{I})\,\mathrm{d}t \end{cases}, \ B: \begin{cases} \mathrm{d}\boldsymbol{\theta} &= 0 \\ \mathrm{d}\boldsymbol{p} &= -\boldsymbol{\xi}\odot\boldsymbol{p}\mathrm{d}t, \\ \mathrm{d}\boldsymbol{\xi} &= 0 \end{cases}$$

$$O: \begin{cases} \mathrm{d}\boldsymbol{\theta} &= 0 \\ \mathrm{d}\boldsymbol{p} &= -\nabla_{\boldsymbol{\theta}}\tilde{U}_t(\boldsymbol{\theta})\mathrm{d}t + \sqrt{2D}\mathrm{d}W. \\ \mathrm{d}\boldsymbol{\xi} &= 0 \end{cases} \qquad (3)$$

All the sub-SDEs can be solved analytically, leading to the following rules to generate samples $\{\boldsymbol{\theta}_{t+1}, \boldsymbol{p}_{t+1}, \boldsymbol{\xi}_{t+1}\}$ from mSGNHT for time $(t+1)$:

$$\begin{aligned} &A: \boldsymbol{\theta}_{t+1/2} = \boldsymbol{p}_t h/2, \ \ \boldsymbol{\xi}_{t+1/2} = \boldsymbol{\xi}_t + (\boldsymbol{p}_t\odot\boldsymbol{p}_t - 1)\,h/2 \to \\ &B: \boldsymbol{p}_{t+1/3} = \exp(-\boldsymbol{\xi}_{t+1/2}h/2)\odot\boldsymbol{p}_t \to \\ &O: \boldsymbol{p}_{t+2/3} = \boldsymbol{p}_{t+1/3} - \nabla_{\boldsymbol{\theta}}\tilde{U}_t(\boldsymbol{\theta}_{t+1/2})h + \sqrt{2D}\boldsymbol{\zeta}_{t+1} \to \\ &B: \boldsymbol{p}_{t+1} = \exp(-\boldsymbol{\xi}_{t+1/2}h/2)\odot\boldsymbol{p}_{t+2/3} \to \\ &A: \boldsymbol{\theta}_{t+1} = \boldsymbol{p}_{t+1}h/2, \ \ \boldsymbol{\xi}_{t+1} = \boldsymbol{\xi}_{t+1/2} + (\boldsymbol{p}_{t+1}\odot\boldsymbol{p}_{t+1} - 1)\,h/2 \end{aligned}$$

[2]We write $F(\mathbf{X}_t)$ from (2) as $\tilde{F}(\mathbf{X}_t)$ if a stochastic gradient is used in the rest of the paper.

From the update equations, SSI performs almost as efficiently as the Euler integrator. Furthermore, the splitting scheme for (2) is not unique. However, all of the schemes can be shown to have the same order of accuracy. In the following subsection, we show quantitatively that the SSI is more accurate than the Euler integrator in terms of approximation errors. To get an impression of how the SSI works, we illustrate it with a simple synthetic experiment.

Illustrations with a Double-well Potential To illustrate the proposed symmetric splitting scheme and its robustness to stepsize, fast convergence, and accurate parameter approximation, we follow (Ding et al. 2014), and consider the double-well potential with

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

and the target distribution $\rho(\theta) \propto \exp(-U(\theta))$. The unknown noise in the stochastic gradient is simulated as $\nabla\tilde{U}(\theta)h = \nabla U(\theta)h + \mathcal{N}(0, 2Bh)$, where $B = 1$. No injecting noise is added. We examine a large range of stepsize h from 10^{-3} to 0.3.

In Fig. 1, we plot the KL divergences between the true distributions and the estimated density, based on 10^6 samples, using two types of integrators. mSGNHT-S consistently provides a better approximation. The significant gap at larger stepsize reveals that mSGNHT-S allows large updates.

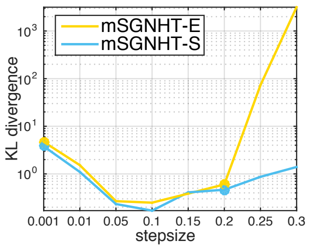

Figure 1: KL divergence for varying stepsize.

Furthermore, we visualize the results of the first 10^5 samples for $h = 10^{-3}$ and $h = 0.2$ in Fig. 2. When the stepsize is too small ($h = 10^{-3}$), conventional mSGNHT-E has not explored the whole parameter space; this is because it converges slower, as shown later. When the stepsize is large ($h = 0.2$), large numerical error is potentially brought in, and mSGNHT-E over-concentrates on the mode. In both cases, mSGNHT-S approaches the theoretical value of thermostat variable $\xi = 1$ more accurately.

3.1 Theoretical Justification

In (Chen, Ding, and Carin 2015), the authors formally studied the roles of numerical integrators in general SG-MCMCs. We adopt their framework, and justify the advantage of the proposed scheme for mSGNHT. We first define the local generator of the SDE (1) at the t-th iteration (*i.e.,* replacing the full gradient with the stochastic gradient from the t-th minibatch) as:

$$\tilde{\mathcal{L}}_t f(\mathbf{X}_t) \triangleq \left(\tilde{F}_t(\mathbf{X}_t)\cdot\nabla_{\mathbf{X}} + \frac{1}{2}\left(\sigma(\mathbf{X}_t)\sigma(\mathbf{X}_t)^\top\right) : \nabla_{\mathbf{X}}\nabla_{\mathbf{X}}^\top\right) f(\mathbf{X}_t)$$

where $\boldsymbol{a}\cdot\boldsymbol{b} \triangleq \boldsymbol{a}^\top\boldsymbol{b}$, $\mathbf{A} : \mathbf{B} \triangleq \mathrm{tr}(\mathbf{A}^\top\mathbf{B})$, $f : \mathbb{R}^n \to \mathbb{R}$ is any twice differentiable function. Based on the definition, according to the Kolmogorov backward equation, we have $\mathbb{E}[f(X)] = e^{h\tilde{\mathcal{L}}_t} f(\mathbf{X})$ where the expectation is taken over the randomness in the diffusion. The operator $e^{h\tilde{\mathcal{L}}_l}$ is called the Kolmogorov operator. Because a numerical integrator is adopted to solve the original SDE, the resulting Kolmogorov operator, denoted as $\tilde{P}_h^t$, approximates $e^{h\tilde{\mathcal{L}}_t}$. To characterise the accuracy of a numerical integrator, we use the following definition.

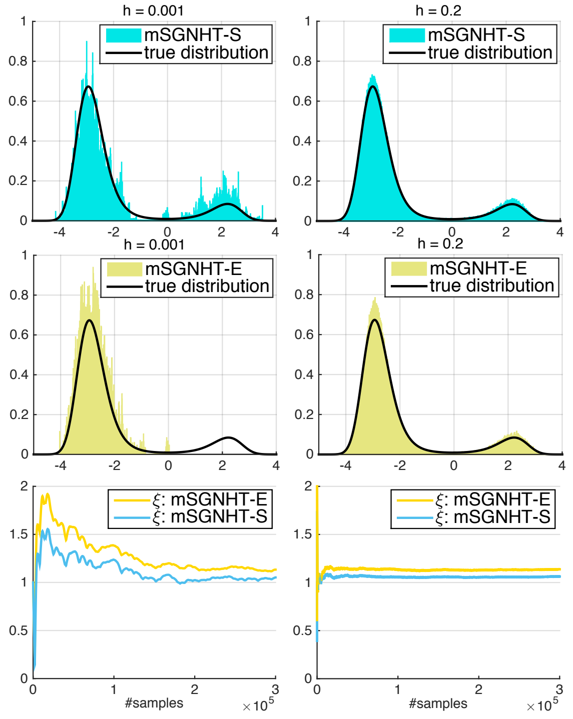

Figure 2: Samples of $\rho(\theta)$ with SSI (1st row) and Euler integrator (2nd row), and the estimated thermostat variable over iterations (3rd row).

Definition 1 *A numerical integrator is said to be a Kth-order local integrator if for any smooth and bounded function f, the corresponding Kolmogorov operator $\tilde{P}_h^t f(\boldsymbol{x})$ from the t-th minibatch with stepsize h satisfies the following relation:*

$$\tilde{P}_h^t f(\boldsymbol{x}) = e^{h\tilde{\mathcal{L}}_t} f(\boldsymbol{x}) + O(h^{K+1})\,. \tag{4}$$

We follow (Chen, Ding, and Carin 2015), and state in Lemma 1 that a Euler integrator satisfies Definition 1 with $K = 1$ when used in mSGNHT. Detailed proofs are provided in the Appendix.

Lemma 1 *The Euler integrator used in mSGNHT-E is a 1st-order local integrator, i.e.,*

$$\tilde{P}_h^l = e^{h\tilde{\mathcal{L}}_l} + O(h^2)\,. \tag{5}$$

Using the Baker–Campbell–Hausdorff formula for commutators (Rossmann 2002), the SSI can be shown to be a 2nd-order integrator in mSGNHT, stated in Lemma 2.

Lemma 2 *The symmetric splitting integrator used in mSGNHT-S is a 2nd-order local integrator, i.e.,*

$$\tilde{P}_h^l = e^{h\tilde{\mathcal{L}}_l} + O(h^3)\,. \tag{6}$$

The authors of (Chen, Ding, and Carin 2015) formalize the role of numerical integrators in terms of posterior *bias* and *mean square error* (MSE). Specifically, for a testing function $\phi(x)$, they study the difference between the posterior average $\bar{\phi} \triangleq \int \phi(x)\rho(x)\mathrm{d}x$ and the finite-time sample average $\hat{\phi} \triangleq \frac{1}{T}\sum_{t=1}^{T}\phi(x_t)$, where $\rho(x)$ denotes the true posterior of a Bayesian model, and $\{x_t\}$ denotes samples from a SG-MCMC algorithm. To study the role of the SSI applied in mSGNHT, we simplify the notation and conclude their results in the following lemma.

Lemma 3 (Roles of numerical integrators) *Under certain assumptions, the* Bias *and* MSE *of a SG-MCMC algorithm with stepsize h and a Kth-order integrator are:*

$$\text{Bias: } \left|\mathbb{E}\hat{\phi} - \bar{\phi}\right| = \mathcal{B}_{bias} + O(h^K)$$

$$\text{MSE: } \mathbb{E}\left(\hat{\phi} - \hat{\phi}\right)^2 = \mathcal{B}_{mse} + O(h^{2K}) \ ,$$

where $\mathcal{B}_{bias}$ and $\mathcal{B}_{mse}$ are functions depending on (h, T) but independent of K.

Based on Lemma 3 and (Chen, Ding, and Carin 2015), we summarize the properties of mSGNHT-S in the following remarks. The detailed are provided in Appendix.

Remark 1 (Robustness) *When applying Lemma 3 to mSGNHT, the bias and MSE of mSGNHT-S is bounded as: $\mathcal{B}_{bias} + O(h^2)$ and $\mathcal{B}_{mse} + O(h^4)$, compared to $\mathcal{B}_{bias} + O(h)$ and $\mathcal{B}_{mse} + O(h^2)$ for the mSGNHT-E, respectively. This indicates that mSGNHT-S is more robust to the stepsizes than mSGNHT-E.*

Remark 2 (Convergence Rate) *The higher order a numerical integrator is, the faster its optimal convergence rate is. Convergence rates in term of* bias *for mSGNHT-S and mSGNHT-E are $T^{-2/3}$ and $T^{-1/2}$, respectively, indicating mSGNHT-S converges faster.*

Remark 3 (Measure Accuracy) *In the limit of infinite time ($T \to \infty$), the terms $\mathcal{B}_{bias}$ and $\mathcal{B}_{mse}$ in Lemma 3 vanish, leaving only the $O(h^K)$ terms. This indicates mSGNHT-S is an order of magnitude more accurate than mSGNHT-E.*

3.2 Advantages of mSGNHT-S for Deep Learning

Compared to optimization-based methods (Martens 2010), the mSGNHT-S is able to more fully explore the parameter space. Therefore, it is less sensitive to initializations, a nontrivial issue in optimization (Sutskever et al. 2013). Second, the mSGNHT is related to stochastic gradient descent (SGD) with momentum in optimization (Chen, Fox, and Guestrin 2014; Ding et al. 2014), with the additional advantage that the momentum is updated element-wise to automatically adapt stepsizes. Additionally, (Sutskever et al. 2013) shows that momentum-accelerated SGD is capable of accelerating along directions of low-curvature in the parameter space, leading to faster convergence speed. As a result, mSGNHT is more favorable than other momentum-free SG-MCMC algorithms, such as the "vanilla" SGLD (Welling and Teh 2011).

Specifically for mSGNHT, we know from previous analysis that (i) mSGNHT-S is less sensitive to stepsize as shown in Remark 1, and thus can tolerate gradients of various magnitudes. This provides a potential solution to mitigate the *vanishing/exploding gradients* problem (Rumelhart, E., and Williams 1986). Our empirical results on deep neural networks verifies this in Section 5.2. (ii) Convergence speed is a critical criteria for learning, and mSGNHT-S clearly outperforms mSGNHT-E in this regard as discussed in Remark 2. (iii) mSGNHT-S converges to a solution one order magnitude more accurate than mSGNHT-E, as discussed in Remark 3, and it is more accurate in estimating model parameters. The number of parameters in large-scale models may be significant, and small numerical error in individual parameters can accumulate, causing noticeable inefficiency. For these reasons, we advocate mSGNHT-S for training large deep models.

4 Related Work

One direction for scalable Bayesian learning of deep models is stochastic variational inference. For deep models with stochastic hidden units, learning has been implemented using variational methods; when the latent variables are continuous, Stochastic Gradient Variational Bayes (SGVB) (Kingma and Welling 2014) has been employed, while models with discrete latent variables have been trained via Neural Variational Inference and Learning (NVIL) (Mnih and Gregor 2014). For deep models with deterministic hidden units, recent studies have shown that imposing uncertainty on global parameters helps prevent overfitting, yielding significant improvment on model performances. Representative works of this type include Bayes by Backprop (BBB) (Blundell et al. 2015) and Probabilistic Backpropagation (PBP) (Hernández-Lobato and Adams 2015), which approximate posteriors of network weights as a product of univariate Gaussian distributions.

Another direction for Bayesian deep learning is SG-MCMC, the line of work followed by this paper. These methods do not have to assume a simplifying form for the posterior, as in variational methods. SG-MCMC algorithms are closely related to optimization methods (*e.g.,* SGD). The difference with respect to optimization methods is the injection of Gaussian noise in the parameter update, allowing better exploration of parameter space when learning. Works of this type include the SGLD (Welling and Teh 2011), SGHMC (Chen, Fox, and Guestrin 2014), SGNHT (Ding et al. 2014), and mSGNHT (Gan et al. 2015a). It has been shown in (Sutskever et al. 2013) that carefully tuned momentum methods suffice for dealing with curvature issues in deep network training. mSGNHT belongs to the class of momentum-accelerated SG-MCMC algorithms. In terms of numerical integrators, recently work for HMC include (Chao et al. 2015; Shahbaba et al. 2014). For our SDE setting, (Leimkuhler and Shang 2015) proposes a symmetric splitting scheme for the SGNHT with a specific stochastic gradient noise, which is different from our setting of mSGNHT for deep models. Recently, (Chen, Ding, and Carin 2015) provides a theoretical foundation for rigorous study of numerical integrators for SG-MCMC. Our work is complementary, providing implementation guidance for the numerical integrator for other SG-MCMC, and investigating its roles in practical applications.

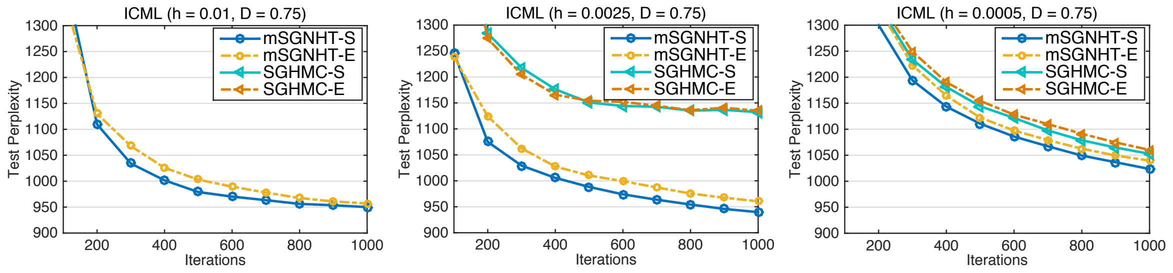

Figure 3: Learning curves of LDA on `ICML` dataset for different stepsize h.

5 Experiments

5.1 Canonical Models

We consider two representative Bayesian models to demonstrate that mSGNHT-S improves general posterior sampling: Latent Dirichlet Allocation (LDA) (Blei, Ng, and Jordan 2003) for latent variable models, and logistic regression.

Latent Dirichlet Allocation We first evaluate our method on the `ICML` dataset (Chen et al. 2015) using LDA. This dataset contains 765 documents from the abstracts of ICML proceedings from 2007 to 2011. After removing stopwords, we obtained a vocabulary size of 1918 and total words of 44140. We used 80% of the documents (selected at random) for training and the remaining 20% for testing. Similar to (Patterson and Teh 2013), we used the semi-collapsed LDA whose posterior is provided in the Appendix. Following (Ding et al. 2014), a Gaussian prior $\mathcal{N}(0.1, 1)$ is used for the reparametrized parameter. The Dirichlet prior parameter for topic distribution for each document is set to 0.1. The number of topics is set to 30. We use *perplexity* (Blei, Ng, and Jordan 2003) to measure the quality of algorithms.

To show the robustness of mSGNHT-S to stochastic gradient noise, we chose minibatch of size 5, and D in mSGNHT-S is fixed as 0.75. We test a wide range of values for step size h. Generally, larger h imposes larger gradient-estimation error and numerical error. Learning curves of the test perplexity for $h = 10^{-2}, 2.5\times10^{-3}, 5\times10^{-4}$ are shown in Fig. 3. We observe that the proposed SSI is consistently better than the Euler integrator. Furthermore, mSGNHT is shown to significantly outperform the SGHMC when h is large. We note that the gap between SSI and the Euler integrator is larger for mSGNHT's than SGHMC's, indicating the importance of numerical integrators in higher dimensional systems.

The best performances for each method are shown in Table 1. Note Gibbs sampling typically obtains the best perplexity because it uses the full dataset for each update. However, it is not scalable for large datasets. In our noisy gradient setup, we see that mSGNHT-S provides the lowest perplexity among the SG-MCMC methods, including a stochastic sampling method for simplex-structured distributions, Stochastic Gradient Riemannian Langevin Dyanmics (SGRLD) (Patterson and Teh 2013).

Logistic Regression We examine logistic regression (LR) on the `a9a` dataset[3]. The training and testing data consist of 32561 and 16281 data points, respectively, with parameter dimension 123. The minibatch size is set to 10, and the

[3]http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/binary.html

Table 1: LDA on `ICML`.

Method	Test Perplexity ↓
mSGNHT-S	**939.67**
mSGNHT-E	960.56
SGHMC-S	1004.73
SGHMC-E	1017.51
SGRLD	1154.68
Gibbs	907.84

Table 2: LR on `a9a`.

Method	Test Accuracy ↑
mSGNHT-S	**84.95%**
mSGNHT-E	84.72%
SGHMC-S	84.56%
SGHMC-E	84.51%
DSVI	84.80%
HFSGVI	84.84%

Gaussian prior on the parameters is $\mathcal{N}(0, 10)$. A thinning interval of 50 is used, with burn-in 300, and 3×10^3 total iterations. Similar to the experiments for LDA, we test a large range of h. We find that mSGNHT-S gives stable performances across varying h on the regression model. Test accuracies are compared in Table 2, from which mSGNHT-S also outperforms the recent doubly stochastic variational Bayes (SDVI) (Titsias and Lázaro-Gredilla 2014), and a higher-order variational autoencoder method (HFSGVI) (Fan et al. 2015). More details are provided in the Appendix.

5.2 Deep Models

To illustrate the advantages of the proposed algorithm for deep learning, two deep models are considered. Specifically, for the case of deterministic hidden layers, we consider deep Feedforward (Convolutional) Neural Networks (FNN), and for stochastic latent layers, we consider Deep Poisson Factor Analysis (DPFA) (Gan et al. 2015b).

Deep Neural Networks We evaluate FNN on the MNIST dataset for classification. The data contains 60000 training examples and 10000 testing examples, each being an 28×28 image of handwritten digit. A L-layer FNN is equivalent to compositing L times of a nonlinear function $g_{\boldsymbol{\theta}_\ell}$, *e.g.,* the sigmoid function used in the logistic regression model. At the top layer, a softmax function is used for multi-class classification, specifically

$$P(y|\boldsymbol{x}) \propto \text{softmax}\big(g_{\boldsymbol{\theta}_L} \circ \cdots \circ g_{\boldsymbol{\theta}_0}(\boldsymbol{x})\big) ,$$

where $\circ$ denotes function composition. For each data $\{\boldsymbol{x}, y\}$, $\boldsymbol{x} \in \mathbb{R}^{784}$ is the raw image, y is the label. A Gaussian prior is placed on model parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_0, \ldots, \boldsymbol{\theta}_L\} \propto \mathcal{N}(0, \sigma^2\mathbf{I})$ with $\sigma^2 = 1$ in our experiment.

We use the Rectified Linear Unit (ReLU) (Glorot, Bordes, and Bengio 2011) as $g_{\boldsymbol{\theta}_\ell}$ in each layer. The number of hidden units for each layer is 100, D is set to 5, stepsize h is set to 10^{-4}, 40 epochs are used. To reduce bias (Chen, Ding, and Carin 2015), h is decreased by half at epoch 20. We test the FNNs with depth $\{2, 3, 4\}$, respectively. Fig. 4 displays learning curves on testing accuracy and training negative

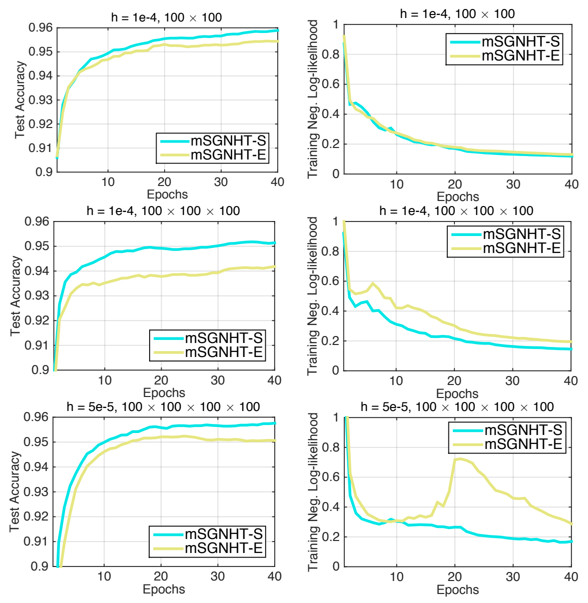

Figure 4: Learning curves of FNN with different depth on MNIST dataset.

log-likelihood. It can be seen that mSGNHT-S consistently converges faster, better than mSGNHT-E for both training and testing. The gaps between mSGNHT-S and mSGNHT-E becomes larger in deeper models. Notably, in the 4-layer FNN, mSGNHT-E failed when $h = 10^{-4}$, while mSGNHT-S worked well. We therefore plot the results for $h = 5{\times}10^{-5}$. From the training plot, mSGNHT-E is failing as learning progresses. It starts to work, only because the stepsize is decreased by half. This confirms that mSGNHT-S is robust to stepsizes, thus is able to mitigate the vanishing/exploding gradient problem in deep models. In addition, we also consider $g_{\boldsymbol{\theta}_\ell}$ as the sigmoid activation function, and the case of convolutional neural networks, empirical results are consistent with the ReLU case. More results are in the Appendix.

Deep Poisson Factor Analysis DPFA (Gan et al. 2015a) is a recently proposed framework for deep topic modeling, where interactions between topics are inferred through a deep latent binary hierarchy. We adopt the deep sigmoid belief networks (DSBN) (Gan et al. 2015b) as the deep architecture in the experiment. The generative process is

$$\mathbf{W} \sim \text{Pois}(\boldsymbol{\Phi}(\boldsymbol{\Psi} \odot \boldsymbol{h}^{(1)})), \quad \boldsymbol{h}^{(1)} \sim \text{DSBN}(\boldsymbol{h}^{\{(2),\cdots,(L)\}}),$$

where $\mathbf{W} \in \mathbb{Z}_+^{V \times J}$ is the observed word count matrix for J documents and vocabulary size V. $\boldsymbol{\Phi}$ is the word-topic matrix, column k, $\boldsymbol{\phi}_k \in \triangle_V$, encodes the relative importance of each word in topic k, with $\triangle_V$ representing the V-dimensional simplex. $\boldsymbol{\Psi}$ is the topic-document matrix, each of its column ψ_j contains relative topic intensities specific to document j. The latent binary feature matrix $\boldsymbol{h}^{(1)}$ indicates the usage of topic. We use mSGNHT to infer the parameters in DSBN, and the *Expanded-Natural* reparametrization method to sample from the probabilistic simplex (Patterson and Teh 2013). More details for model specification are in the Appendix.

We test the DPFA on a large dataset, *Wikipedia*, from which 10M randomly downloaded documents are used, using scripts provided in (Hoffman, Bach, and Blei 2010). We follow the setup in (Gan et al. 2015a), where 1K documents are randomly selected for testing and validation, respectively. The vocabulary size is 7702, and the minibatch size is set to 100, with one pass of the whole data in the experiments. We collect 300 posterior samples to calculate test perplexities, with a standard holdout technique. A three-layer DSBN is employed, with dimensions 128-64-32 (128 topics right above the data layer). Step sizes are chosen as 10^{-4} and 10^{-5}, and parameter $D = 40$.

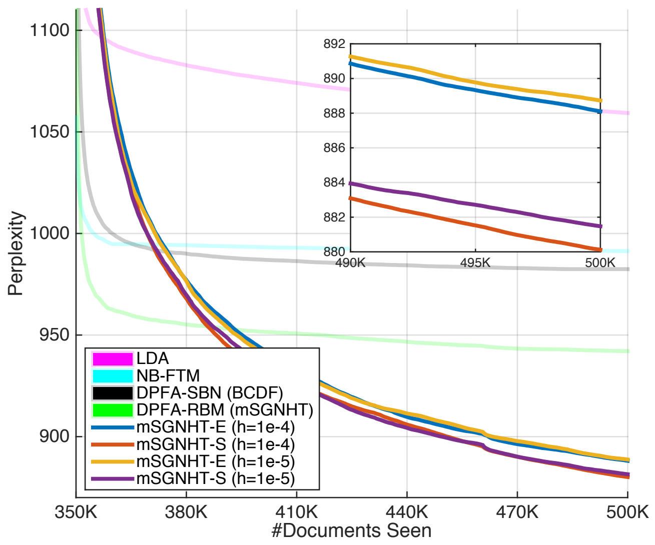

Figure 5: Perplexity on Wikipedia dataset.

The results are shown in Fig. 5, displaying the predictive perplexities on a held-out test set as a function of training documents seen. Clearly, mSGNHT-S converges faster than mSGNHT-E at both chosen stepsizes. A magnified plot is shown at the top-right corner of the figure as well, displaying perplexities for the last 10K documents.

mSGNHT-S outperforms other recent state-of-the-art methods (shown in semi-transparent plots). Specifically, we compare to, DPFA-SBN trained with Bayesian conditional density filtering (BCDF) (Guhaniyogi, Qamar, and Dunson 2014), DPFA with restricted Boltzmann machines (RBM) (Hinton 2002) trained with mSGNHT-E, and Negative Binomial Focused Topic Model (NB-FTM) (Zhou and Carin 2015) trained with BCDF. The shallow model LDA trained with BCDF is reported as the baseline.

6 Conclusion

A 2nd-order symmetric splitting integrator is proposed to solve the SDE within mSGNHT. This method is shown to be more accurate than the conventional Euler integrator, leading to higher robusness, faster convergence, and more accurate posterior samples. We apply the integrator on mSGNHT for four representative models, including latent Dirichlet allocation, logistic regression, deep neural networks, and deep Poisson factor analysis. Extensive experiments demonstrate that the proposed scheme improves large-scale sampling in terms of convergence speed and accuracy, particularly for deep models.

Acknowledgements This research was supported in part by ARO, DARPA, DOE, NGA, ONR and NSF.

References

Bengio, Y.; Simard, P.; and Frasconi, P. 1994. Learning long-term dependencies with gradient descent is difficult. *TNN*.

Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent Dirichlet allocation. *JMLR*.

Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; and Wierstra, D. 2015. Weight uncertainty in neural networks. In *ICML*.

Chao, W.-L.; Solomon, J.; Michels, D.; and Sha, F. 2015. Exponential integration for Hamiltonian Monte Carlo.

Chen, C.; Buntine, W.; Ding, N.; Xie, L.; and Du, L. 2015. Differential topic models. *TPAMI*.

Chen, C.; Ding, N.; and Carin, L. 2015. On the convergence of stochastic gradient MCMC algorithms with high-order integrators. In *NIPS*.

Chen, T.; Fox, E. B.; and Guestrin, C. 2014. Stochastic gradient Hamiltonian Monte Carlo. In *ICML*.

Ding, N.; Fang, Y.; Babbush, R.; Chen, C.; Skeel, R. D.; and Neven, H. 2014. Bayesian sampling using stochastic gradient thermostats. In *NIPS*.

Fan, K.; Wang, Z.; Beck, J.; Kwok, J.; and Heller, K. 2015. Fast second-order stochastic backpropagation for variational inference. In *NIPS*.

Gal, Y., and Ghahramani, Z. 2015. Bayesian convolutional neural networks with Bernoulli approximate variational inference. *arXiv:1506.02158*.

Gan, Z.; Chen, C.; Henao, R.; Carlson, D.; and Carin, L. 2015a. Scalable deep Poisson factor analysis for topic modeling. In *ICML*.

Gan, Z.; Henao, R.; Carlson, D.; and Carin, L. 2015b. Learning deep sigmoid belief networks with data augmentation. In *AISTATS*.

Gan, Z.; Li, C.; Henao, R.; Carlson, D.; and Carin, L. 2015c. Deep temporal sigmoid belief networks for sequence modeling. *NIPS*.

Glorot, X.; Bordes, A.; and Bengio, Y. 2011. Deep sparse rectifier neural networks. In *AISTATS*.

Guhaniyogi, R.; Qamar, S.; and Dunson, D. B. 2014. Bayesian conditional density filtering. *arXiv:1401.3632*.

Hernández-Lobato, J. M., and Adams, R. P. 2015. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *ICML*.

Hinton, G. E. 2002. Training products of experts by minimizing contrastive divergence. In *Neural computation*.

Hoffman, M.; Bach, F. R.; and Blei, D. M. 2010. Online learning for latent Dirichlet allocation. In *NIPS*.

Jarrett, K.; Kavukcuoglu, K.; Ranzato, M.; and LeCun, Y. 2009. What is the best multi-stage architecture for object recognition? In *ICCV*.

Kemp, C.; Perfors, A.; and Tenenbaum, J. B. 2007. Learning overhypotheses with hierarchical Bayesian models. *Developmental science*.

Kingma, D. P., and Welling, M. 2014. Auto-encoding variational Bayes. In *ICLR*.

Knapp, A. W. 2005. *Basic real analysis*. Sci. & B. M.

Korattikara, A.; Rathod, V.; Murphy, K.; and Welling, M. 2015. Bayesian dark knowledge. *NIPS*.

Leimkuhler, B., and Matthews, C. 2013. Rational construction of stochastic numerical methods for molecular sampling. *Applied Math. Research Express*.

Leimkuhler, B., and Shang, X. 2015. Adaptive thermostats for noisy gradient systems. *arXiv:1505.06889*.

Li, C.; Chen, C.; Carlson, D.; and Carin, L. 2016. Preconditioned stochastic gradient Langevin dynamics for deep neural networks. In *AAAI*.

Ma, Y. A.; Chen, T.; and Fox, E. B. 2015. A complete recipe for stochastic gradient MCMC. *NIPS*.

MacKay, D. 1992. A practical Bayesian framework for backpropagation networks. *Neural computation*.

Martens, J. 2010. Deep learning via Hessian-free optimization. In *ICML*.

Mnih, A., and Gregor, K. 2014. Neural variational inference and learning in belief networks. In *ICML*.

Neal, R. M. 1995. *Bayesian learning for neural networks*. PhD thesis, University of Toronto.

Neal, R. M. 2011. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*.

Patterson, S., and Teh, Y. W. 2013. Stochastic gradient Riemannian Langevin dynamics on the probability simplex. In *NIPS*.

Pu, Y.; Yuan, X.; and Carin, L. 2015. A generative model for deep convolutional learning. In *ICLR workshop*.

Ranganath, R.; Tang, L.; Charlin, L.; and Blei, D. M. 2015. Deep exponential families. *AISTATS*.

Risken, H. 1989. *The Fokker-Planck equation*. Springer-Verlag, New York.

Rossmann, W. 2002. *Lie Groups–An Introduction Through Linear Groups*. Oxford Graduate Texts in Mathematics, Oxford Science Publications.

Rumelhart, D. E.; E., H. G.; and Williams, R. J. 1986. Learning representations by back-propagating errors. *Nature*.

Salakhutdinov, R.; Tenenbaum, J. B.; and Torralba, A. 2013. Learning with hierarchical-deep models. *TPAMI*.

Shahbaba, B.; Lan, S.; Johnson, W. O.; and Neal, R. M. 2014. Split Hamiltonian Monte Carlo. *Stat. and Comp.*

Sutskever, I.; Martens, J.; Dahl, G.; and Hinton, G. 2013. On the importance of initialization and momentum in deep learning. In *ICML*.

Titsias, M., and Lázaro-Gredilla, M. 2014. Doubly stochastic variational Bayes for non-conjugate inference. In *ICML*.

Welling, M., and Teh, Y. W. 2011. Bayesian learning via stochastic gradient Langevin dynamics. In *ICML*.

Zhou, M., and Carin, L. 2015. Negative binomial process count and mixture modeling. *TPAMI*.

Supplementary Material of High-Order Stochastic Gradient Thermostats for Bayesian Learning of Deep Models

A The proof of Lemma 1

Proof In mSGNHT, $\mathbf{X}_t = (\boldsymbol{\theta}_t, \boldsymbol{p}_t, \boldsymbol{\xi}_t)$. The update equations with a Euler integrator are:

$$\begin{cases} \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \boldsymbol{p}_t h \\ \boldsymbol{p}_{t+1} &= \boldsymbol{p}_t - \nabla_{\boldsymbol{\theta}} \tilde{U}_t(\boldsymbol{\theta}_{t+1}) h - \text{diag}(\boldsymbol{\xi}_t) \boldsymbol{p}_t h + \sqrt{2D} \boldsymbol{\zeta}_{t+1} \\ \boldsymbol{\xi}_{t+1} &= \boldsymbol{\xi}_t + (\boldsymbol{p}_{t+1} \odot \boldsymbol{p}_{t+1} - 1)\, h \end{cases}$$

Based on the update equations, it is easily seen that the corresponding Kolmogorov operator $\tilde{P}_h^l$ for mSGNHT is

$$\tilde{P}_h^l = e^{h\mathcal{L}_1} \circ e^{h\mathcal{L}_2} \circ e^{h\mathcal{L}_3} \,, \tag{7}$$

where $\mathcal{L}_1 \triangleq (\boldsymbol{p} \odot \boldsymbol{p} - 1) \cdot \nabla_{\boldsymbol{\xi}}$, $\mathcal{L}_2 \triangleq -\text{diag}(\boldsymbol{\xi}) \boldsymbol{p}_t \cdot \nabla_{\boldsymbol{p}} - \nabla_{\boldsymbol{\theta}} \tilde{U}_l(\boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{p}} + 2DI : \nabla_{\boldsymbol{p}} \nabla_{\boldsymbol{p}}^T$, and $\mathcal{L}_3 \triangleq \boldsymbol{p} \cdot \nabla_{\boldsymbol{\theta}}$. Using the BCH formula, we have

$$\tilde{P}_h^l = e^{h(\mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3)} + O(h^2) \,. \tag{8}$$

On the other hand, the local generator of mSGNHT at the t-th iteration can be seen to be:

$$\tilde{\mathcal{L}}_t = \mathcal{L}_1 + \tilde{\mathcal{L}}_2 + \mathcal{L}_3 \,, \tag{9}$$

where $\mathcal{L}_1$ and $\mathcal{L}_3$ are defined previously, and $\tilde{\mathcal{L}}_2 = -\text{diag}(\boldsymbol{\xi}) \boldsymbol{p} \cdot \nabla_{\boldsymbol{p}} - \nabla_{\boldsymbol{\theta}} \tilde{U}_l(\boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{p}} + 2DI : \nabla_{\boldsymbol{p}} \nabla_{\boldsymbol{p}}^T$. According to the Kolmogorov's backward equation, we have

$$\mathbb{E}[f(\mathbf{X}_{t+1})] = e^{h\tilde{\mathcal{L}}_t} f(\mathbf{X}_t) \,. \tag{10}$$

Substitute (10) into (8) and use the fact that $\boldsymbol{p} = \boldsymbol{p}_t + O(h)$ by Taylor expansion, it is easily seen

$$\tilde{P}_h^l = e^{h\tilde{\mathcal{L}}_t + O(h^2)} + O(h^2) = e^{h\tilde{\mathcal{L}}_t} + O(h^2) \,.$$

This completes the proof.

B The proof of Lemma 2

Proof In symmetric splitting scheme for mSGNHT, according to the splitting in (4) in the main text, the generator $\tilde{\mathcal{L}}_t$ is split into the following sub-generators which can be solved analytically: $\tilde{\mathcal{L}}_l = \mathcal{L}_A + \mathcal{L}_B + \mathcal{L}_{O_l}$, where

$$\begin{aligned} \mathcal{A} &\triangleq \mathcal{L}_A = \boldsymbol{p} \cdot \nabla_{\boldsymbol{\theta}} + (\boldsymbol{p} \odot \boldsymbol{p} - 1) \cdot \nabla_{\boldsymbol{\xi}}, \\ \mathcal{B} &\triangleq \mathcal{L}_B = -\text{diag}(\boldsymbol{\xi}) \boldsymbol{p}_t \cdot \nabla_{\boldsymbol{p}}, \\ \mathcal{O}_l &\triangleq \mathcal{L}_{O_l} = -\nabla_{\boldsymbol{\theta}} \tilde{U}_l(\boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{p}} + 2D : \nabla_{\boldsymbol{p}} \nabla_{\boldsymbol{p}}^T \,. \end{aligned}$$

The corresponding Kolmogorov operator $\tilde{P}_h^l$ for the splitting integrator can be seen to be:

$$\tilde{P}_h^l \triangleq e^{\frac{h}{2}\mathcal{L}_A} \circ e^{\frac{h}{2}\mathcal{L}_B} \circ e^{h\mathcal{L}_{O_l}} \circ e^{\frac{h}{2}\mathcal{L}_B} \circ e^{\frac{h}{2}\mathcal{L}_A},$$

In the following we use the Baker–Campbell–Hausdorff (BCH) formula (Rossmann 2002) to show that $\tilde{P}_h^l$ is a 2nd-order integrator. Specifically,

$$e^{\frac{h}{2}\mathcal{A}} e^{\frac{h}{2}\mathcal{B}} = e^{\frac{h}{2}\mathcal{A} + \frac{h}{2}\mathcal{B} + \frac{h^2}{8}[\mathcal{A},\mathcal{B}] + \frac{1}{96}([\mathcal{A},[\mathcal{A},\mathcal{B}]] + [\mathcal{B},[\mathcal{B},\mathcal{A}]]) + \cdots} \tag{11}$$

$$= e^{\frac{h}{2}\mathcal{A} + \frac{h}{2}\mathcal{B} + \frac{h^2}{8}[\mathcal{A},\mathcal{B}]} + O(h^3) \,, \tag{12}$$

where $[X, Y] \triangleq XY - YX$ is the commutator of X and Y, (11) follows from the BCH formula, and (12) follows by moving high order terms $O(h^3)$ out of the exponential map using Taylor expansion. Similarly, for the other composition, we have

$$\begin{aligned} & e^{h\mathcal{O}_l} e^{\frac{h}{2}\mathcal{A}} e^{\frac{h}{2}\mathcal{B}} = e^{h\mathcal{O}_l} \left(e^{\frac{h}{2}\mathcal{A} + \frac{h}{2}\mathcal{B} + \frac{h^2}{8}[\mathcal{A},\mathcal{B}]} + O(h^3) \right) \\ =& e^{h\mathcal{O}_l + \frac{h}{2}\mathcal{A} + \frac{h}{2}\mathcal{B} + \frac{h^2}{8}[\mathcal{A},\mathcal{B}] + \frac{1}{2}[h\mathcal{O}_l, \frac{h}{2}\mathcal{A} + \frac{h}{2}\mathcal{B} + \frac{h^2}{8}[\mathcal{A},\mathcal{B}]]} + O(h^3) \\ =& e^{h\mathcal{O}_l + \frac{h}{2}\mathcal{A} + \frac{h}{2}\mathcal{B} + \frac{h^2}{8}[\mathcal{A},\mathcal{B}] + \frac{h^2}{4}[\mathcal{O}_l,\mathcal{A}] + \frac{h^2}{4}[\mathcal{O}_l,\mathcal{B}]} + O(h^3) \\ & e^{\frac{h}{2}\mathcal{A}} e^{h\mathcal{O}_l} e^{\frac{h}{2}\mathcal{A}} e^{\frac{h}{2}\mathcal{B}} \\ =& e^{\frac{h}{2}\mathcal{A}} \left(e^{h\mathcal{O}_l + \frac{h}{2}\mathcal{A} + \frac{h}{2}\mathcal{B} + \frac{h^2}{8}[\mathcal{A},\mathcal{B}] + \frac{h^2}{4}[\mathcal{O}_l,\mathcal{A}] + \frac{h^2}{4}[\mathcal{O}_l,\mathcal{B}]} + O(h^3) \right) \\ =& e^{h\mathcal{O}_l + h\mathcal{A} + \frac{h}{2}\mathcal{B} + \frac{h^2}{4}[\mathcal{A},\mathcal{B}] + \frac{h^2}{2}[\mathcal{O}_l,\mathcal{B}]} + O(h^3) \end{aligned}$$

As a result

$$\begin{aligned} & \tilde{P}_h^l \triangleq e^{\frac{h}{2}\mathcal{B}} e^{\frac{h}{2}\mathcal{A}} e^{h\mathcal{Z}} e^{\frac{h}{2}\mathcal{A}} e^{\frac{h}{2}\mathcal{B}} \\ =& e^{\frac{h}{2}\mathcal{B}} \left(e^{h\mathcal{O}_l + h\mathcal{A} + \frac{h}{2}\mathcal{B} + \frac{h^2}{4}[\mathcal{A},\mathcal{B}] + \frac{h^2}{2}[\mathcal{O}_l,\mathcal{B}]} + O(h^3) \right) \\ =& e^{h\mathcal{O}_l + h\mathcal{A} + h\mathcal{B} + \frac{h^2}{4}[\mathcal{A},\mathcal{B}] + \frac{h^2}{2}[\mathcal{O}_l,\mathcal{B}] + \frac{h^2}{4}[\mathcal{B},\mathcal{A}] + \frac{h^2}{4}[\mathcal{B},\mathcal{O}_l] + \frac{h^2}{8}[\mathcal{B},\mathcal{B}]} + O(h^3) \\ =& e^{h(\mathcal{B} + \mathcal{A} + \mathcal{O}_l)} + O(h^3) \\ =& e^{h(\mathcal{L} + \Delta V_l)} + O(h^3) = e^{h\tilde{\mathcal{L}}_l} + O(h^3) \,. \end{aligned}$$

This completes the proof.

C More details on Lemma 3

Our justification of the symmetric splitting integrator is based on Lemma 3, which is a simplification of the main theorems in (Chen, Ding, and Carin 2015). For completeness, we give details of their main theorems in this section.

To recap notation, for an Itó diffusion with an invariant measure $\rho(\mathbf{X})$, the posterior average is defined as: $\bar{\phi} \triangleq \int_{\mathcal{X}} \phi(\mathbf{X}) \rho(\mathbf{X}) \mathrm{d}x$ for some test function $\phi(\mathbf{X})$ of interest. Given samples $(x_t)_{t=1}^T$ from a SG-MCMC, we use the *sample average* $\hat{\phi} \triangleq \frac{1}{T} \sum_{t=1}^T \phi(x_t)$ to approximate $\bar{\phi}$.

In addition, we define an operator for the t-th iteration as:

$$\Delta V_t \triangleq (\nabla_\theta \tilde{U}_t - \nabla_\theta U) \cdot \nabla_{\boldsymbol{p}} \,. \tag{13}$$

Theorem 4 and Theorem 5 summarize the convergence of a SG-MCMC algorithm with a K-th order integrator with respect to the *Bias* and *MSE*, under certain assumptions. Please refer to (Chen, Ding, and Carin 2015) for detailed proofs.

Theorem 4 *Let $\|\cdot\|$ be the operator norm. The bias of an SG-MCMC with a Kth-order integrator at time $\mathcal{T} = hT$ can be bounded as:*

$$\left|\mathbb{E}\hat{\phi} - \bar{\phi}\right| = O\left(\frac{1}{Th} + \frac{\sum_t \|\mathbb{E}\Delta V_t\|}{T} + h^K\right) .$$

Theorem 5 *For a smooth test function ϕ, the MSE of an SG-MCMC with a Kth-order integrator at time $\mathcal{T} = hT$ is bounded, for some $C > 0$ independent of (T, h), as*

$$\mathbb{E}\left(\hat{\phi} - \bar{\phi}\right)^2 \leq C\left(\frac{\frac{1}{T}\sum_t \mathbb{E}\left\|\Delta V_t\right\|^2}{T} + \frac{1}{Th} + h^{2K}\right) .$$

We simplify Theorem 4 and Theorem 5 to Lemma 3, where the functions $\mathcal{B}_{\text{bias}} \triangleq O\left(\frac{1}{Th} + \frac{\sum_t \|\mathbb{E}\Delta V_t\|}{T}\right)$ and $\mathcal{B}_{\text{mse}} \triangleq C\left(\frac{\frac{1}{T}\sum_t \mathbb{E}\|\Delta V_t\|^2}{T} + \frac{1}{Th}\right)$, independent of the order of an integrator.

In addition, from Theorem 4 and Theorem 5, we can get the optimal convergence rate of a SG-MCMC algorithm with respect to the *Bias* and *MSE* by optimizing the bounds. Specifically, the optimal convergence rates of the *Bias* for the Euler integrator is $T^{-1/2}$ with optimal stepsize $\propto T^{-1/2}$, while this is $T^{-2/3}$ for the symmetric splitting operator with optimal stepsize $\propto T^{-1/3}$. For the MSE, the rate for the Euler integrator is $T^{-2/3}$ with optimal stepsize $\propto T^{-1/3}$, compared to $T^{-4/5}$ with optimal stepsize $\propto T^{-1/5}$ for the symmetric splitting integrator.

D Latent Dirichlet Allocation

Following (Ding et al. 2014), this section describes the semi-collapsed posterior of the LDA model, and the *Expanded-Natural* representation of the prabability simplexes used in (Patterson and Teh 2013).

Let $\mathbf{W} = \{w_{jv}\}$ be the observed words, $\mathbf{Z} = \{z_{jv}\}$ be the topic indicator variables, where j indexes the documents and v indexes the words. Let $(\pi)_{kw}$ be the topic-word distribution, n_{jkw} be the number of word w in document j allocated to topic k, $\cdot$ means marginal sum, *i.e.,* $n_{jk\cdot} = \sum_w n_{dkw}$. The semi-collapsed posterior of the LDA model is

$$p(\mathbf{W}, \mathbf{Z}, \pi|\alpha, \tau) = p(\pi|\tau)\prod_{j=1}^{J} p(\boldsymbol{w}_j, \boldsymbol{z}_j|\alpha, \pi), \tag{14}$$

where J is the number of documents, α is the parameter in the Dirichlet prior of the topic distribution for each document, τ is the parameter in the prior of π, and

$$p(\boldsymbol{w}_j, \boldsymbol{z}_j|\alpha, \pi) = \prod_{k=1}^{K}\frac{\Gamma\left(\alpha + n_{jk\cdot}\right)}{\Gamma\left(\alpha\right)}\prod_{w=1}^{W}\pi_{kw}^{n_{jkw}} . \tag{15}$$

The *Expanded-Natural* representation of the simplexes π_k's in (Patterson and Teh 2013) is used, where

$$\pi_{kw} = \frac{e^{\theta_{kw}}}{\sum_{w'} e^{\theta_{kw'}}} . \tag{16}$$

Following (Ding et al. 2014), a Gaussian prior on θ_{kw} is adopted,

$$p(\theta_{kw}|\tau = \{\beta, \sigma\}) = \mathcal{N}(\theta_{kw}, \sigma^2) .$$

The stochastic gradient of the log-posterior of parameter θ_{kw} with a mini-batch $\mathcal{S}_t$ becomes,

$$\begin{aligned}\frac{\partial\tilde{U}(\boldsymbol{\theta})}{\partial\theta_{kw}} &= \frac{\partial\log\tilde{p}(\boldsymbol{\theta}|\mathbf{W}, \tau, \alpha)}{\partial\theta_{kw}} \\ &= \frac{\beta - \theta_{kw}}{\sigma^2} + \frac{J}{|\mathcal{S}_t|}\sum_{i\in\mathcal{S}_t}\mathbb{E}_{\boldsymbol{z}_j|\boldsymbol{w}_j,\theta,\alpha}\left(n_{jkw} - \pi_{kw}n_{jk\cdot}\right) .\end{aligned} \tag{17}$$

for the t-th iteration, where $\mathcal{S}_t \subset \{1, 2, \cdots, J\}$, and $|\cdot|$ is the cardinality of a set.

When $\sigma = 1$, we obtain the same stochastic gradient as in (Patterson and Teh 2013) using Riemannian manifold. To calculate the expectation term, we use the same Gibbs sampling method as in (Patterson and Teh 2013),

$$p(z_{jv} = k|\boldsymbol{w}_d, \boldsymbol{\theta}, \alpha) = \frac{\left(\alpha + n_{jk\cdot}^{\backslash v}\right)e^{\theta_{kw_{jv}}}}{\sum_{k'}\left(\alpha + n_{jk'\cdot}^{\backslash v}\right)e^{\theta_{k'w_{jv}}}}, \tag{18}$$

where $\backslash v$ denotes the count excluding the v-th topic assignment variable. The expectation is estimated by the samples.

Running time For the results reported in the main text, to achieve reported results of LDA on ICML dataset, mSGNHT-E, mSGNHT-S and Gibbs take 1000 iterations, the running times are 161.99, 180.55 and 516.12 second, respectively.

E Logistic Regression

For each data $\{\boldsymbol{x}, y\}$, $\boldsymbol{x} \in \mathbb{R}^P$ is the input data, $y \in \{0, 1\}$ is the label. Logistic Regression gives the prabability

$$P(y|\boldsymbol{x}) \propto g_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1 + \exp\left(-(\mathbf{W}^\top\boldsymbol{x} + \boldsymbol{c})\right)} . \tag{19}$$

A Gaussian prior is placed on model parameters $\boldsymbol{\theta} = \{\mathbf{W}, \boldsymbol{c}\} \propto \mathcal{N}(0, \sigma^2\mathbf{I})$. We set $\sigma^2 = 10$ in our experiment.

We study the effectiveness of mSGNHT-S for different h and D. Learning curves of test errors and training log-likelihoods are shown in Fig. 6. Generally, the performances of the proposed mSGNHT-S are consistently more stable than the mSGNHT-E and SGHMC, across varying h and D. Furthermore, mSGNHT-S converges faster than mSGNHT-E, especially at the beginning of learning. Comparing column 1-2 (fixing h, varying D), mSGNHT-S is more robust to the choice of diffusion factor D. Comparing column 2-4 (fixing D, varying h), larger h potentially brings larger gradient-estimation errors and numerical errors. mSGNHT is shown to significantly outperform others when h is large.

F Deep Neural Networks

F.1 Sensitivity to Step-size

For feedforward neural nets (FNN) with a 2-layer model of 100-100 ReLU, we study the performance of mSGNHT-S for different h. $D = 5$. We test a wide range of h, and show

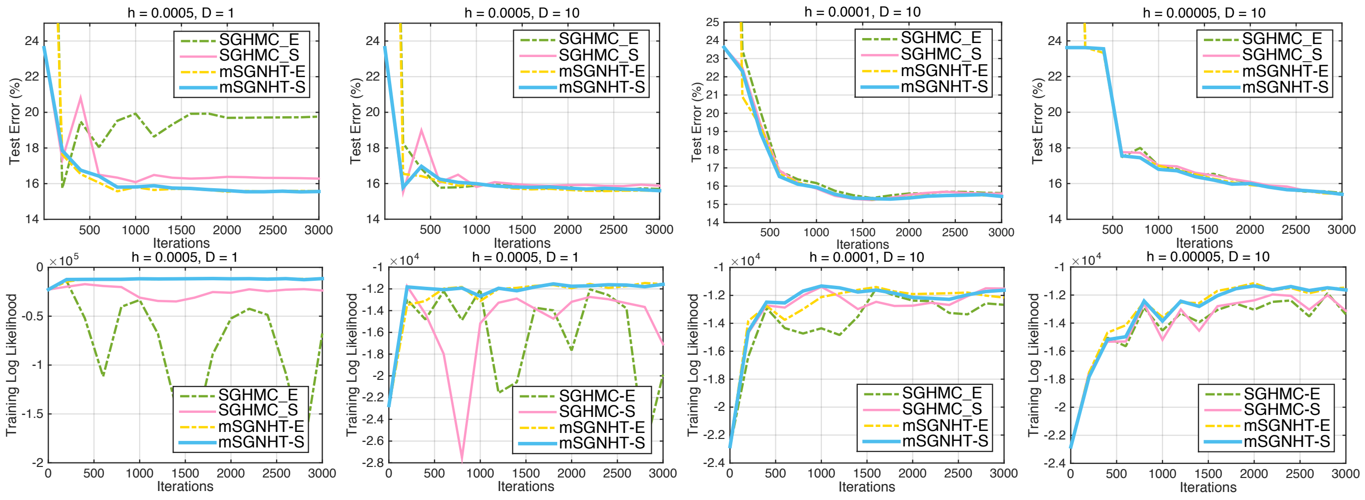
Figure 6: Learning curves for logistic regression on `a9a` dataset for varying h and D. Column 1-2 share the same h; column 2-4 share the same D. Top row is testing error; bottom row is training log-likelihood.

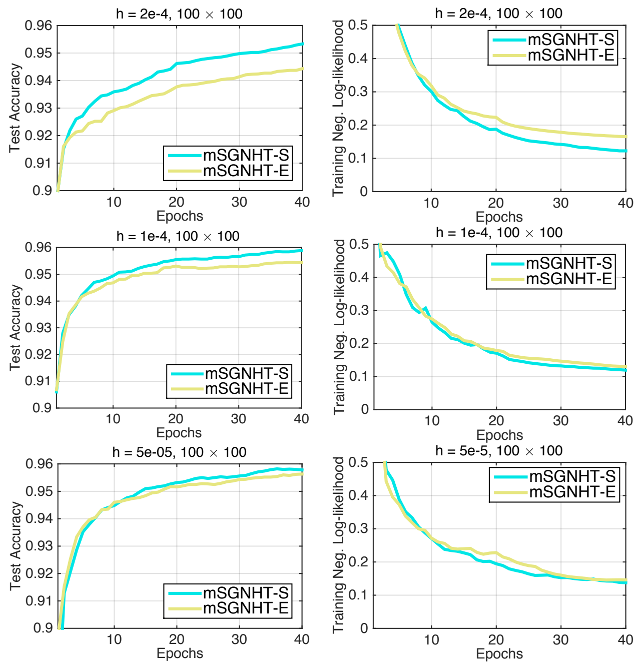
Figure 7: Learning curves for FNN (ReLU link) on MNIST dataset for varying stepsize h. Step size decreases top-down.

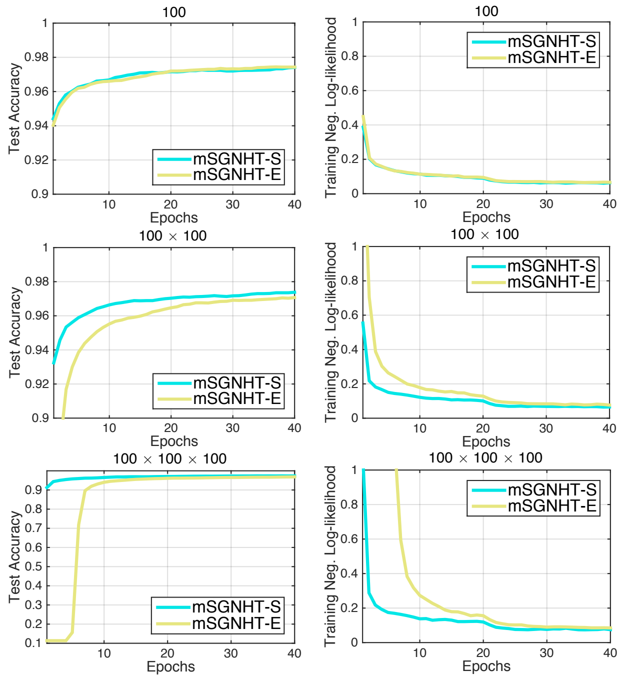
Figure 8: Learning curves for FNN (Sigmoid link) on MNIST dataset for varying network depth. Depth increases top-down.

in Fig. 7 the learning curves of the test accuracy and training negative log-likelihood for $h = 2\times10^{-4}, 1\times10^{-4}, 5\times10^{-5}$. mSGNHT-S is less sensitive to step size; it maintains fast convergence when h is large, while mSGNHT-S significantly slows down.

F.2 Sigmoid Activation Function

We compare different methods in the case of sigmoid link. Similar to the main paper, we test the FNNs with varying depths, *e.g.*, $\{1, 2, 3\}$, respectively. We set $D = 10$ and $h = 10^{-3}$. Fig. 8 displays learning curves on testing accuracy and training negative log-likelihood. The gaps between mSGNHT-S and mSGNHT-E becomes larger in deeper models. When a 3-layer network is emplyed, mSGNHT-E fails at the first 5 epochs, whilst mSGNHT-S converges pretty well. Moreover, mSGNHT-S converges more accurately, yielding an accuracy 97.36%, while mSGNHT-E gives 96.86%. This manifests the importance of numerical accuracy in SG-MCMC for practical applications, and mSGNHT-S is desirable in this regard.

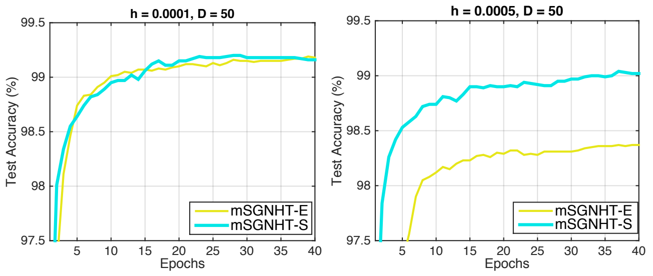

Figure 9: Learning curves of CNN for different step sizes.

F.3 Comparison with Other Methods

Based on network size 400-400 with ReLU, we also compared with a recent state-of-the-art inference method for neural nets, Bayes by Backprop (BBB) (Blundell et al. 2015). $h = 2 \times 10^{-4}$ and $D = 60$. The comparison is in Table 3. BBB and SGD are results taken from (Blundell et al. 2015).

Table 3: Classification accuracy on MNIST.

Method	Accuracy (%) ↑
mSGNHT-S	**98.25**
mSGNHT-E	98.20
BBB	98.18
SGD	98.17

F.4 Convolutional Neural Networks

We also performed the comparison with a standard network convolutional neural networks, LeNet (Jarrett et al. 2009), on MNIST dataset. It is 2-layer convolution networks followed by a 2-layer fully-connected FNN, each containing 200 hidden nodes that uses ReLU. Both convolutional layers use 5×5 filter size with 32 and 64 channels, respectively, 2×2 max pooling are used after each convolutional layer. 40 epochs are used, and L is set to 20. In Fig. 9, we tested the stepsizes $h = 10^{-4}$ and $h = 5 \times 10^{-4}$ for mSGNHT-S and mSGNHT-S, and $D = 50$.

Again, under the same network architecture, CNN trained with mSGNHT-S converges fater than mSGNHT-E. In particular, when a large stepsize used, mSGNHT-S has a significant improvement over mSGNHT-E.

G Deep Poisson Factor Analysis

G.1 Model Specification

We first provide model details of Deep Poisson Factor Analysis (DPFA) (Gan et al. 2015a). Given a discrete matrix $\mathbf{W} \in \mathbb{Z}_+^{V \times J}$ containing counts from J documents and V words, Poisson factor analysis (PFA) (Zhou and Carin 2015) assumes the entries of $\mathbf{W}$ are summations of $K < \infty$ latent counts, each produced by a latent factor (in the case of topic modeling, a hidden topic). For $\mathbf{W}$, the generative process of DPFA with L-layer Sigmoid Belief Networks (SBN), is as follows

$$p(h_{k_L}^{(L)}) = g(b_{k_L}^{(L)}) \quad (20)$$

$$p(h_{k_\ell n}^{(\ell)} = 1 | \boldsymbol{h}_n^{(\ell+1)}) = g\big((\boldsymbol{w}_{k_\ell}^{(\ell)})^\top \boldsymbol{h}_n^{(\ell+1)} + c_{k_\ell}^{(\ell)}\big) \quad (21)$$

$$\mathbf{W} \sim \text{Pois}(\mathbf{\Phi}(\mathbf{\Psi} \odot \boldsymbol{h}^{(1)})) \quad (22)$$

where $g(\cdot)$ is the Sigmoid link. Equation (20) and (21) define Deep Sigmoid Belief Networks (DSBN). $\mathbf{\Phi}$ is the factor loading matrix. Each column of $\mathbf{\Phi}$, $\boldsymbol{\phi}_k \in \Delta_V$, encodes the relative importance of each word in topic k, with Δ_V representing the V-dimensional simplex. $\mathbf{\Theta} \in \mathbb{R}_+^{K \times J}$ is the factor score matrix. Each column $\boldsymbol{\psi}_j$, contains relative topic intensities specific to document j. $\boldsymbol{h}^{(\ell)} \in \{0,1\}^{K \times 1}$ is a latent binary feature vector, which defines whether certain topics are associated with documents. The factor scores for document j at bottom layer are the element-wise multiplication $\boldsymbol{\psi}_j \odot \boldsymbol{h}^{(1)}$. DPFA is constructed by placing Dirichlet priors on $\mathbf{\Phi}_k$ and gamma priors on $\boldsymbol{\psi}_j$. This is summarized as,

$$x_{vj} = \sum_{k=1}^{K} x_{vjk}, \qquad x_{vjk} \sim \text{Pois}(\phi_{vk}\psi_{kj}h_k) \quad (23)$$

with priors specified as $\boldsymbol{\phi}_k \sim \text{Dir}(a_\phi, \ldots, a_\phi)$, $\psi_{kn} \sim \text{Gamma}(r_k, p_k/(1-p_k))$, $r_k \sim \text{Gamma}(\gamma_0, 1/c_0)$, and $\gamma_0 \sim \text{Gamma}(e_0, 1/f_0)$.

G.2 Model Inference

Following (Gan et al. 2015a), we use stochastic gradient Riemannian Langevin dynamics (SGRLD) (Patterson and Teh 2013) to sample the topic-word distributions $\{\boldsymbol{\phi}_k\}$. mSGNHT is used to sample the parameters in DSBN, *i.e.*, $\boldsymbol{\theta} = \left(\mathbf{W}^{(\ell)}, \boldsymbol{c}^{(\ell)}, \boldsymbol{b}^{(L)}\right)$, where $\ell = 1, \cdots, L-1$. Specifically, the stochastic gradients of $\mathbf{W}^{(\ell)}$ and $\boldsymbol{c}^{(\ell)}$ evaluated on a mini-batch of data (denote $\mathcal{S}$ as the index set of a mini-batch) are calculated,

$$\frac{\partial \tilde{U}}{\partial \boldsymbol{w}_{k_\ell}^{(\ell)}} = \frac{J}{|\mathcal{S}|} \sum_{i \in \mathcal{D}} \mathbb{E}_{\boldsymbol{h}_i^{(\ell)}, \boldsymbol{h}_i^{(\ell+1)}} \left[\left(\tilde{\sigma}_{k_\ell i}^{(\ell)} - h_{k_\ell i}^{(\ell)} \right) \boldsymbol{h}_i^{(\ell+1)} \right], \quad (24)$$

$$\frac{\partial \tilde{U}}{\partial c_{k_\ell}^{(\ell)}} = \frac{J}{|\mathcal{S}|} \sum_{i \in \mathcal{D}} \mathbb{E}_{\boldsymbol{h}_i^{(\ell)}, \boldsymbol{h}_i^{(\ell+1)}} \left[\tilde{\sigma}_{k_\ell i}^{(\ell)} - h_{k_\ell i}^{(\ell)} \right], \quad (25)$$

where $\tilde{\sigma}_{k_\ell i}^{(\ell)} = \sigma((\boldsymbol{w}_{k_\ell}^{(\ell)})^\top \boldsymbol{h}_i^{(\ell+1)} + c_{k_\ell}^{(\ell)})$, and the expectation is taken over posteriors. Monte Carlo integration is used to approximate this quantity.