entropy

MDPI

Article

Dynamical Sampling with Langevin Normalization Flows

Minghao Gu [1], Shiliang Sun [1,2,*] and Yan Liu [3]

[1] School of Computer Science and Technology, East China Normal University, 3663 North Zhongshan Road, Shanghai 200241, China; guminghao1081@gmail.com
[2] Shanghai Institute of Intelligent Science and Technology, Tongji University, Shanghai 201804, China
[3] School of Data Science and Engineering, East China Normal University, 3663 North Zhongshan Road, Shanghai 200241, China; yliu@cc.ecnu.edu.cn
* Correspondence: slsun@cs.ecnu.edu.cn

Received: 7 October 2019; Accepted: 6 November 2019; Published: 10 November 2019

Abstract: In Bayesian machine learning, sampling methods provide the asymptotically unbiased estimation for the inference of the complex probability distributions, where Markov chain Monte Carlo (MCMC) is one of the most popular sampling methods. However, MCMC can lead to high autocorrelation of samples or poor performances in some complex distributions. In this paper, we introduce Langevin diffusions to normalization flows to construct a brand-new dynamical sampling method. We propose the modified Kullback-Leibler divergence as the loss function to train the sampler, which ensures that the samples generated from the proposed method can converge to the target distribution. Since the gradient function of the target distribution is used during the process of calculating the modified Kullback-Leibler, which makes the integral of the modified Kullback-Leibler intractable. We utilize the Monte Carlo estimator to approximate this integral. We also discuss the situation when the target distribution is unnormalized. We illustrate the properties and performances of the proposed method on varieties of complex distributions and real datasets. The experiments indicate that the proposed method not only takes the advantage of the flexibility of neural networks but also utilizes the property of rapid convergence to the target distribution of the dynamics system and demonstrate superior performances competing with dynamics based MCMC samplers.

Keywords: Normalization flows; Langevin diffusions; Langevin normalization flows; Monte Carlo sampling

1. Introduction

In machine learning, Bayesian inference [1] and Bayesian optimization [2], complex probabilistic models typically require the calculation of intractable and high-dimension integrals. For example, for a classification task, we need to predict the class of instances. We assume that $p(y_*|x_*, \mathcal{D}) = \int p(y_*|x_*, \theta)p(\theta|\mathcal{D})\mathrm{d}\theta$ is the prediction model, where x_* represents the instance, y_* represents the class, $\mathcal{D}$ represents the data, $p(y_*|x_*, \theta)$ is the likelihood function and $p(\theta|\mathcal{D})$ is the posterior distribution. When the probabilistic model becomes complex, this integral is intractable. Generally, two kinds of methods are used to approximate the integral, which are Markov chain Monte Carlo (MCMC) [3,4] and variational inference (VI) [5,6]. MCMC is a powerful framework, which is widely used to deal with the complex and intractable probabilistic models [7–9]. MCMC methods approximate the complex probability distributions by a large number of samples which are sampled from a Markov chain iteratively. They serve as a fundamental approach in probabilistic inference, which provides the asymptotically unbiased estimation for the probabilistic models, while VI gives the deterministic approximation for the target distributions [10].

Recent MCMC methods can be divided into two aspects. One class of the MCMC methods is slice sampling [11] and the other one is dynamical sampling [12,13]. The main problem of the slice sampler is that when sampling from the distributions with high dimensions, solving the slice interval can be very difficult. Utilizing the dynamics system to construct an efficient Markov chain is commonly employed [14–16]. Hamiltonian Monte Carlo (HMC) [14] is one of the dynamics based methods, which has multiple attractive properties concerning rapid explorations of the state space and high acceptance rate of the samples. HMC exploits Hamiltonian dynamics to construct efficient Markov chain Monte Carlo, which has become increasingly popular in machine learning and statistics. Since HMC uses the gradient information of the target distribution, it can explore the state space much more efficiently than the random-walk proposals [17], which ensures the rapid convergence of the sampler. Since it has the property of volume conservation, HMC is able to propose large moves with a higher acceptance rate.

HMC and its further developments [18–23] exploit the gradient information of the target distribution to explore the state space. Nevertheless, since the step size of the leapfrog is difficult to choose, there exists the correlation between neighbor samples and thus the high autocorrelation may occur. Though we can enlarge the step size of the leapfrog, it will waste a lot of computation resources. Moreover, they tend to fail when the target distributions are multi-modal [21,24–26]. These MCMC methods usually fail to move from one mode to another because such a move requires passing through low probability regions. These places have large boundary gradients which prevent samplers from traveling through the modes. Therefore, designing an effective sampler for multi-modal distributions has remained a significant challenge.

The disadvantages of the current methods motivate us to design a powerful sampler which can have not only low autocorrelation but also accurate estimation for the target distribution. In this paper, a new sampling method called Langevin normalization flows Monte Carlo (NFLMC) is proposed. We introduce Langevin diffusions to the normalization flows (NFs) [27] to construct a new sampler. The main idea of this method is to train a variational distribution to approximate the target distribution, whose parameters are determined by the neural networks. With the idea of Langevin diffusions, we design new transformation functions for NFs which have the properties of rapid convergence to the target distribution and better approximation to the target distributions. Since we exploit the gradient information of the target distributions, the calculation of the integrals of the Kullback-Leibler ($\mathbb{KL}$) divergence is intractable. So we use the Monte Carlo estimator to calculate the $\mathbb{KL}$ divergence. However, the $\mathbb{KL}$ divergence calculated by Monte Carlo estimator may be negative in the process of training, which would mislead the final results, so we propose a new loss function to train the NFLMC sampler.

The main contributions of this paper can be summarized as follows. (1) We introduce Langevin diffusions to normalization flows to construct a novel Monte Carlo sampler. (2) We propose the modified $\mathbb{KL}$ divergence as the loss function to train the sampler, which ensures that the proposed method can converge to the target distribution. (3) The proposed method achieves better performances in multi-modal sampling and varieties of complex distributions. (4) we do not need the Metropolis-Hasting procedure [28] to adjust the sampler compared with MCMC samplers. (5) A number of experiments verify the theoretical results and practical value. We apply the proposed method to varieties of distributions and supervised classification tasks using Bayesian logistic regression. The proposed method is compared with state-of-the-art dynamics based MCMC methods [24,29,30] in the autocorrelation rate and convergence speed. The experiments demonstrate that the NFLMC method has a superior performance in sampling complex posterior distributions.

The rest of this article is organized as follows. In Section 2, we review the preliminary of our study, including the introduction of variational inference with normalization flows and Langevin diffusions. In Section 3, we introduce our Langevin normalization flows and describe the transformation functions. In Section 4, we propose the Langevin normalization flows Monte Carlo sampler. Experiments and analysis are given in Section 5. In Section 6, we conclude this paper and discuss the future work.

2. Preliminary

2.1. Normalization Flows

The normalization flows [27] were first introduced to deal with the flexible and complex posterior distributions in the context of variational inference. It is a powerful approach to generate arbitrary posterior distributions utilizing a sequence of invertible transformation. In other words, the initial density will transform to a valid probability distribution through iteratively applying the normalization flows. Given the observed data $\boldsymbol{x}$, the normalization flows start with an initial variable $\boldsymbol{z}_0$ generated from a simple distribution q, which has the analytical probability density and then repeatedly apply an invertible transformation function f_θ which is parameterized by θ. After a sequence of iterations, a complex and flexible distribution of z_T will be obtained. It takes the form as follows:

$$\boldsymbol{z}_0 \sim q(\boldsymbol{z}_0|\boldsymbol{x}), \boldsymbol{z}_t \sim f_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{x}), \forall t = 1 \dots T. \tag{1}$$

Since the Jacobian determinant of each transformation f_θ can be calculated, we can obtain the final distribution π_{u_T} through the following equation.

$$\ln\left[\pi_{u_T}(\boldsymbol{z}_T|\boldsymbol{x})\right] = \ln\left[q(\boldsymbol{z}_0|\boldsymbol{x})\right] - \sum_{t=1}^{T} \ln\left(\det\left|\frac{\partial \boldsymbol{z}_t}{\partial \boldsymbol{z}_{t-1}}\right|\right). \tag{2}$$

To make Equation (2) tractable, the Jacobian determinant of each transformation function f_θ should be carefully designed to satisfy two main properties. First, the transformation function f_θ is easy to invert. Second, the Jacobian determinant should be tractable. We assume that z_0 comes from a simple distribution $q(z_0|x)$ and $z_T = f_\theta(z_0)$. When calculating the probability of z_T in Equation (2), we need to calculate the Jacobian determinant and use $f^{-1}(z_T)$ to calculate z_0. So the transformation function f_θ should be easy to invert and the Jacobian determinant should be tractable. Generally, the invertible transformation function f_θ with known Jacobian determinant [27] is defined as:

$$f_\theta(\boldsymbol{z}_{t-1}) = \boldsymbol{z}_{t-1} + mh(w^{\mathrm{T}}\boldsymbol{z}_{t-1} + b), \tag{3}$$

where $h(\cdot)$ represents the nonlinear function, $m = [m^1, m^2, \dots, m^n]$ and $w = [w^1, w^2, \dots, w^n]$ are parameter vectors and b is the scalar and n is the dimension of the parameter vectors. So $mh(w^{\mathrm{T}}\boldsymbol{z}_{t-1} + b)$ can be viewed as a multi-layer perceptron with one hidden layer and a single unit, which is demonstrated in Figure 1.

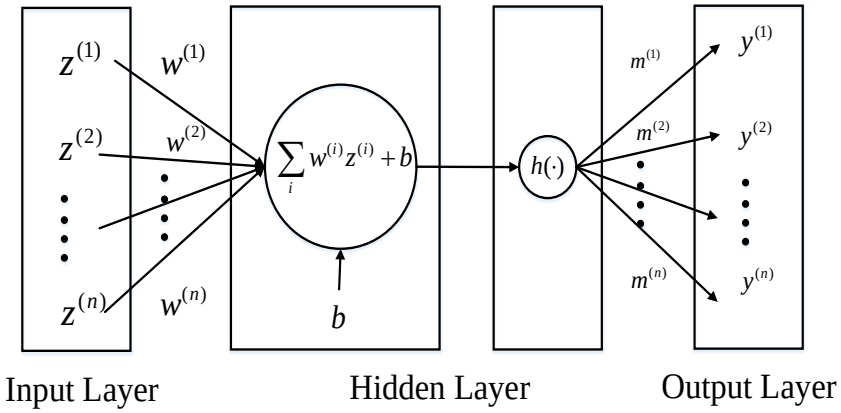

Figure 1. The multi-layer perceptron with one hidden layer and a single unit.

Real-valued non-volume preserving (RNVP) [31] develops a new transformation function, which makes the model more flexible. The main idea of RNVP is that coupling layers are used to construct the normalization flows. Assume that $\boldsymbol{x}$ is the original variable. The coupling layers can be defined as:

$$\begin{aligned} \boldsymbol{y}_{1:d} &= \boldsymbol{x}_{1:d} \\ \boldsymbol{y}_{d+1:D} &= \boldsymbol{x}_{d+1:D} \odot \exp(s(\boldsymbol{x}_{1:d})) + t(\boldsymbol{x}_{1:d}), \end{aligned} \tag{4}$$

where function s represents the scale and t represents the translation. Both of them are neural networks. RNVP provides a more powerful and flexible posterior distribution for density estimation.

Recently, neural density estimators are widely used in the approximation of data distributions [32,33] and variational inference [27]. NFs and their further studies performs very well in modeling images, videos and audio [31,34–37].

2.2. Langevin Diffusions

Langevin dynamics is a common method to model molecular dynamics systems. A D-dimension Langevin diffusions are a time based stochastic process $\boldsymbol{x} = (\boldsymbol{x}_t), t \geq 0$ with stochastic sample paths, which can be defined as a solution to the stochastic differential equation taking the form as follows:

$$\mathrm{d}\boldsymbol{x}_t = \boldsymbol{b}(\boldsymbol{x}_t)\mathrm{d}t + \boldsymbol{\sigma}(\boldsymbol{x}_t)\mathrm{d}W_t, \tag{5}$$

where $\boldsymbol{b}(\boldsymbol{x})$ represents the drift vector, $\boldsymbol{\sigma}(\boldsymbol{x})$ represents the volatility matrix and $W = (W_t), t \geq 0$ represents a standard Wiener process [38]. Equation (5) gives the evolution of a random variable under Langevin diffusions but when it comes to the evolution of the probability density function, the diffusions should be described by Fokker-Planck equation [39]. We assume that $u(\boldsymbol{x}, t)$ represents the evolution of the probability density function, $\boldsymbol{x} = [x_1, x_2, \ldots, x_D]^{\mathrm{T}}$ and $V(\boldsymbol{x}) = \boldsymbol{\sigma}(\boldsymbol{x})\boldsymbol{\sigma}(\boldsymbol{x})^{\mathrm{T}}$. We set $b_i(\boldsymbol{x})$ to be the i-th term of the vector $\boldsymbol{b}(\boldsymbol{x})$ and $V_{ij}(\boldsymbol{x})$ to be the i-th row and the j-th column's term of the matrix $V(\boldsymbol{x})$. So the Fokker-Planck equation can be defined as follows:

$$\frac{\partial}{\partial t}u(\boldsymbol{x}, t) = -\sum_{i=1}^{D}\frac{\partial}{\partial x_i}[b_i(\boldsymbol{x})u(\boldsymbol{x}, t)] + \frac{1}{2}\sum_{i=1,j=1}^{D}\frac{\partial^2}{\partial x_i \partial x_j}[V_{ij}(\boldsymbol{x})u(\boldsymbol{x}, t)]. \tag{6}$$

If we have $u(\boldsymbol{x}, t) = \pi_g(\boldsymbol{x}), \forall t \in T$, then this process is stationary and π_g can be viewed as the stationary distribution of the diffusion, which means that if $x_t \sim \pi_g(x)$, then $x_{t+\epsilon} \sim \pi_g(x)$, $\forall\, \epsilon > 0$ [40]. Langevin diffusion with stationary distribution π_g can be defined by the stochastic differential equation [23]:

$$\mathrm{d}\boldsymbol{x}_t = \frac{1}{2}\nabla \ln \pi_g(\boldsymbol{x}_t)\mathrm{d}t + \mathrm{d}W_t. \tag{7}$$

The setting of $\boldsymbol{b}$, $\boldsymbol{\sigma}$ and $u(x, t)$ in Equation (7) makes $\frac{\partial u}{\partial t} = 0$, which suggests that the invariant measure of Langevin diffusion is related to $\pi_g(x)$ [40].

Generally, solving the stochastic differential equations exactly is intractable. Since stochastic differential equations usually have strong coupling and nonlinearity, it is difficult to calculate the exact expression of its solution. So it is necessary to utilize the numerical discretization methods to approximate the solution to stochastic differential equations. Euler-Maruyama discretization [41] is one of the common approaches to obtain the approximate solution to the stochastic differential equation, which takes the form as:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \frac{\epsilon^2}{2}\nabla_{\boldsymbol{x}} \ln \pi_g(\boldsymbol{x}_t) + \epsilon \boldsymbol{z}_t, \tag{8}$$

where $\boldsymbol{z}_t \sim \mathcal{N}(\boldsymbol{z}|0, \mathrm{I})$ and ϵ represents the step size.

It is noted that Langevin diffusions take advantage of the gradient information of the target distribution. The gradient information makes Langevin diffusions explore the state space efficiently. What's more, Langevin diffusions contain the Wiener process that can be viewed as the random work. The random work helps to explore the state space extensively. The idea of Langevin diffusions are widely used in MCMC methods. Metropolis adjusted Langevin algorithm (MALA) [40] is one of the applications of Langevin diffusions. The main idea of MALA is to give the proposed state through Langevin diffusions, whose equation is given in Equation (8). MALA exploits the Metropolis-Hasting correction [28] to satisfy the detailed balance [42], which ensures that the samples generated from Langevin diffusions will converge to the target distribution. It is the gradient information of the target distribution that accelerates the convergence rate to the stationary distribution of MCMC. Although MALA do provide an efficient way for MCMC to sample from the target distribution, the autocorrelation among samples remains high.

Since NFs provide a more powerful and flexible posterior distribution for density estimation and MALA achieves rapid convergence to the target distribution, we maintain their advantages to develop a new sampler with appropriate training strategy, which can accurately sample from the target distribution with low autocorrelation.

3. Langevin Normalization Flows

3.1. Main Idea

Normalization flows [27,31] approximate the target distributions through a series of transformation functions. In order to approximate the target distributions efficiently and accurately, we utilize the information of the target distributions. Through exploiting the advantages of efficient exploration of Langevin diffusions, we propose a new normalization flow which is called Langevin normalization flows (NFL). We redesign the transformation functions through the gradient information of the target distribution, which helps us to approximate the target distributions precisely and efficiently.

Constructing the Langevin normalization flows has to satisfy two primary conditions. The first one is that the update of each step of the transformation function should be approximately invertible. The second one is that the determinant of the Jacobian and the inverse Jacobian of the transformation function must be tractable. In this way, we can ensure that the distribution obtained through the flows is able to converge to the target distribution.

We then describe the details of our proposed transformation functions for a single Langevin step. We assume that $\boldsymbol{x}_{1:D}$ is the initial sample, where D is the number of the dimension of the sample. We first update a half of the sample. The transformation functions are as follows:

$$\begin{cases} \boldsymbol{y}_{1:d} &= \boldsymbol{x}_{1:d}, \\ \boldsymbol{y}_{d+1:D} &= (\boldsymbol{x}_{d+1:D} - \frac{\epsilon^2}{2}\nabla U(\boldsymbol{x}_{1:D})_{d+1:D} + \epsilon \cdot \exp(\sigma(\boldsymbol{x}_{1:d}))) \\ & \quad \odot \exp(S(\boldsymbol{x}_{1:d})) + T(\boldsymbol{x}_{1:d}), \end{cases} \tag{9}$$

where $\sigma(\boldsymbol{x})$ can be viewed as the Wiener process in Langevin diffusions. $S(\boldsymbol{x})$ represents the logarithmic scale of the sample which is able to rescale the position of the sample. $T(\boldsymbol{x})$ is the shift of the sample. $\sigma(\boldsymbol{x})$, $S(\boldsymbol{x})$ and $T(\boldsymbol{x})$ are all controlled by the neural networks, where W_σ, W_S and W_T are their parameters. U is the energy function of the probability density function. In addition, ϵ represents the step size of the Langevin diffusions. It is noted that in Equation (9), we first utilize Langevin diffusions to generate samples and then we use neural networks to further adjust the samples. Since we only update $\boldsymbol{x}_{d+1:D}$ and $\boldsymbol{y}_{1:D}$ is the intermediate variable, $\boldsymbol{x}_{1:d}$ should be updated then. It takes the form as:

$$\begin{cases} \boldsymbol{z}_{d+1:D} &= \boldsymbol{y}_{d+1:D}, \\ \boldsymbol{z}_{1:d} &= (\boldsymbol{y}_{1:d} - \frac{\epsilon^2}{2}\nabla U(\boldsymbol{y}_{1:D})_{1:d} + \epsilon \cdot \exp(\sigma(\boldsymbol{y}_{d+1:D}))) \\ & \quad \odot \exp(S(\boldsymbol{y}_{d+1:D})) + T(\boldsymbol{y}_{d+1:D}), \end{cases} \tag{10}$$

where $z_{1:D}$ represents the final obtained state after applying the above transformation functions to $y_{1:D}$. The advantage of dividing x into two part is that Equation (9) generates y_{d+1} and affects only x_{d+1} while Equation (10) generates $z_{1:d}$ and affects only $y_{1:d}$. At the same time, the determinant of the Jacobian is tractable, which relies on the fact that:

$$\frac{\partial(f_b \circ f_a)}{\partial \alpha_a^{\mathrm{T}}}(\alpha_a) = \frac{f_a}{\partial \alpha_a^{\mathrm{T}}}(\alpha_a) \cdot \frac{f_b}{\partial \alpha_b^{\mathrm{T}}}(\alpha_b = f_a(\alpha_a)),$$
$$\det(A \cdot B) = \det(A) \cdot \det(B).$$

The Jacobian matrices of these transformation functions are as follows:

$$\frac{\partial f_\theta}{\partial x} = \begin{bmatrix} \mathrm{I}_{1:\mathrm{d}} & 0 \\ \frac{\partial y_{d+1:D}}{\partial x_{1:d}^{\mathrm{T}}} & \frac{\partial y_{d+1:D}}{\partial x_{d+1:D}^{\mathrm{T}}} \end{bmatrix}, \frac{\partial f_\theta}{\partial y} = \begin{bmatrix} \frac{\partial z_{1:d}}{\partial y_{1:d}^{\mathrm{T}}} & \frac{\partial z_{d+1:D}}{\partial y_{d+1:D}^{\mathrm{T}}} \\ 0 & \mathrm{I}_{\mathrm{d+1:D}} \end{bmatrix}.$$

It is noted that the Jacobian matrices of the transformation functions are upper triangular matrix and lower triangular matrix respectively, which simplify the calculation of the Jacobian determinants. In order to calculate the logarithmic probability of the transformation distribution, we need the help of inverse transformation functions and the inverse logarithmic Jacobian determinants. The logarithmic probability can be computed as follows.

$$\pi_u(x) = q(f_\theta^{-1}(x))\det\left|\frac{\partial f_\theta^{-1}}{\partial x}\right|, \tag{11}$$

where q represents the initial distribution. The inverse transformation functions f_θ^{-1} take the form as:

$$\begin{cases} y_{d+1:D} = z_{d+1:D}, \\ y_{1:d} \quad = (z_{1:d} - T(y_{d+1:D})) \odot \exp(-Sy_{d+1:D})) \\ \qquad -\epsilon \cdot \exp(\sigma(y_{d+1:D})) + \frac{\epsilon^2}{2} \cdot \nabla U((t_1, y_{d+1:D}))_{1:d}, \\ t_1 \quad = (z_{1:d} - T(y_{d+1:D})) \odot \exp(-S(y_{d+1:D})) - \epsilon \cdot \exp(\sigma(y_{d+1:D}). \end{cases} \tag{12}$$

It is noted that Equation (12) is approximately invertible. Since we introduce gradient information to the transformation functions, the inverse transformation function f_θ^{-1} is difficult to obtain. For instance, in Equation (12), $z_{1:D}$ is known and we wish to use $z_{1:D}$ to calculate $y_{1:D}$. Although we can easily obtain $y_{d+1:D}$ through the first equation of Equation (12), when it comes to calculating $y_{1:d}$, we have to calculate $\nabla U(y_{1:D})_{1:d}$ to update $y_{1:d}$. However achieving the closed-form solution for $y_{1:d} = \frac{\epsilon^2}{2} \cdot \nabla U(y_{1:D})_{1:d} + const$ is difficult especially when the gradient function is complex, where $const = (z_{1:d} - T(y_{d+1:D})) \odot \exp(-S(y_{d+1:D})) - \epsilon \cdot \exp(\sigma(y_{d+1:D}))$. In order to calculate $y_{1:d}$, we have additionally introduced a variable t_1 in the process of calculating the inverse transformation function. We set $y_{1:D}$ in $\nabla U(y_{1:D})_{1:d}$ to be $(t_1, y_{d+1:D})$ and we calculate t_1 without using gradient information. Finally we update $y_{1:d}$ through $\nabla U((t_1, y_{d+1:D}))_{1:d}$. The error of this approximation is $\frac{\epsilon^2}{2}[\nabla U(y_{1:D})_{1:d} - \nabla U((t_1, y_{d+1:D}))_{1:d}]$ which depends on the product of $\frac{\epsilon^2}{2}$ and $\nabla U((\xi, y_{d+1:D})), \xi \in (y_{1:d}, t_1)$. This approach is also exploited in the calculation of $x_{1:D}$, which takes the form as:

$$\begin{cases} x_{1:d} \quad = y_{1:d}, \\ x_{d+1:D} = (y_{d+1:D} - T(x_{1:d})) \odot \exp(-S(x_{1:d})) \\ \qquad -\epsilon \cdot \exp(\sigma(x_{1:d})) + \frac{\epsilon^2}{2} \cdot \nabla U((x_{1:d}, t_2))_{d+1:D}, \\ t_2 \quad = (y_{d+1:D} - T(x_{1:d})) \odot \exp(-S(x_{1:d})) - \epsilon \cdot \exp(\sigma(x_{1:d})). \end{cases} \tag{13}$$

In order to calculate the logarithmic probability of the transformation distribution, we have to compute the inverse logarithmic Jacobian determinants. The final formulas are defined as follows:

$$\begin{aligned}\ln\left|\frac{\partial f_\theta^{-1}}{\partial \boldsymbol{z}}\right| &= \ln|\exp(-S(\boldsymbol{y}_{d+1:D})) + \frac{\epsilon^2}{2}\cdot\nabla\nabla U((t_1,\boldsymbol{y}_{d+1:D}))_{1:d}\odot\exp(-S(\boldsymbol{y}_{d+1:D}))|,\\ \ln\left|\frac{\partial f_\theta^{-1}}{\partial \boldsymbol{y}}\right| &= \ln|\exp(-S(\boldsymbol{y}_{1:d})) + \frac{\epsilon^2}{2}\cdot\nabla\nabla U((\boldsymbol{x}_{1:d},t_2))_{d+1:D}\odot\exp(-S(\boldsymbol{y}_{1:d}))|.\end{aligned} \tag{14}$$

Particularly, we introduce Langevin diffusions to normalization flows to construct the transformation function. Since the Langevin diffusions exploit the gradient of the target distribution, the transformation function is able to explore the state space efficiently.

Hamiltonian dynamics introduce the auxiliary momentum variable to explore the state space efficiently. Through the transformation of the energy over potential energy and kinetic energy, the total energy remains unchanged. Since the change of the state is associated with the transformation of the energy, designing normalization flows which are based on Hamiltonian dynamics becomes complex, which will be our future work.

3.2. Difference between Normalization Flows and Langevin Normalization Flows

There are two main differences between normalization flows and Langevin normalization flows. First, NFL cooperates with Langevin diffusions to construct an efficient and accurate approximation for the target distributions competing with the normalization flows. Second, when approximating the target distribution, the normalization flows are trained to minimize $\mathbb{KL}(q|p)$, where q represents the approximation distribution and p represents the target distribution. Since the transformation function is invertible, the integral of $\mathbb{KL}(q|p)$ can be calculated precisely. However, for NFL, the transformation functions demonstrated in Equations (12) and (13) are only approximately invertible because of the usage of the gradient information of the target distribution. Since the precise value of $\mathbb{KL}(q|p)$ cannot be obtained through integration, Monte Carlo estimation is used to calculate $\mathbb{KL}(q|p)$.

4. Dynamical Sampling Using Langevin Normalization Flows

Probabilistic inference involving multi-modal distributions is very difficult for dynamics based MCMC samplers. Besides, samples generated from these samplers are still highly auto-correlated. In order to solve these problems, we develop a new Monte Carlo sampler using Langevin normalization flows which are called Langevin normalization flows Monte Carlo (NFLMC). Given the target distribution and the initial distribution, NFLMC learns the parameters of the conversion of the initial distribution to the target distribution of the sampler. In the following subsections, we begin to describe the main idea of the method and then we introduce how our method works. Finally, we give the loss function of the training procedure and the algorithm. When the value of loss function converges, NFLMC can precisely sample from the target distribution.

4.1. Main Idea

The procedure of NFLMC is elaborated here. Assume that the target distribution is denoted as π_t, the initial distribution is denoted as π_q, θ represents the parameters of the transformation functions, π_u represents the transformation distribution and Ls represents the Langevin step length. First, we generate N samples $\boldsymbol{X} = \{\boldsymbol{x}_{(t)}\}_{t=0}^{N}, \boldsymbol{x} \in \mathbb{R}^D$ from π_q and initialize the parameters θ in the transformation functions. For each sample $\boldsymbol{x} \in \boldsymbol{X}$, the update equation takes the form as:

$$\begin{cases} \boldsymbol{x}_{1:d} &= \boldsymbol{x}_{1:d},\\ \boldsymbol{x}_{d+1:D} &= \boldsymbol{x}_{d+1:D} - \frac{\epsilon^2}{2}\nabla U(\boldsymbol{x}_{1:D})_{d+1:D} + \epsilon\cdot\exp(\sigma(\boldsymbol{x}_{1:d})).\end{cases} \tag{15}$$

We repeatedly utilize Equation (15) *Ls* times to update $\boldsymbol{x}_{d+1:D}$, where ϵ is the step size of Langevin diffusions, $\nabla U(\boldsymbol{x}_{1:D})_{d+1:D}$ is the gradient of the energy function of the target distribution. It is noted that the second term in Equation (15) is similar with Equation (8). After applying *Ls* steps of Langevin diffusions, we rescale $\boldsymbol{x}_{d+1:D}$ through Equation (9) and we obtain $\boldsymbol{y}_{1:D}$ which is a half update of $\boldsymbol{x}_{1:D}$. We then update $\boldsymbol{x}_{1:d}$ which takes the form as:

$$\begin{cases} \boldsymbol{y}_{d+1:D} = \boldsymbol{y}_{d+1:D}, \\ \boldsymbol{y}_{1:d} \quad = \boldsymbol{y}_{1:d} - \frac{\epsilon^2}{2}\nabla U(\boldsymbol{y}_{1:D})_{1:d} + \epsilon \cdot \exp(\sigma(\boldsymbol{y}_{d+1:D})). \end{cases} \tag{16}$$

We also repeatedly utilize Equation (16) *Ls* times to update $\boldsymbol{x}_{1:d}$. After that we rescale $\boldsymbol{x}_{1:d}$ through Equation (10) and finally we obtain $\boldsymbol{z}_{1:D} = f_\theta(\boldsymbol{x}_{1:D})$, where we define the transformation function as f_θ. Now we gain the samples $\boldsymbol{z}_{1:D}, \boldsymbol{z}_{1:D} \sim \pi_u$. In order to optimize the parameters θ in f_θ to close to the target distribution. Through minimizing $\mathbb{KL}(\pi_u|\pi_t)$, we are able to obtain the optimal parameters of f_θ. Since the integral of $\mathbb{KL}(\pi_u|\pi_t)$ for Langevin normalization flow is intractable, we use the Monte Carlo integral to calculate $\mathbb{KL}(\pi_u|\pi_t)$. The objective function is as follows:

$$\min_\theta \mathbb{KL}(\pi_u|\pi_t) = \frac{1}{N}\min_\theta \sum_{i=1, \boldsymbol{x}_{(i)}\sim\pi_u}^{N} \ln \frac{\pi_u\left(\boldsymbol{x}_{(i)}\right)}{\pi_t\left(\boldsymbol{x}_{(i)}\right)}. \tag{17}$$

As Equation (17) shows, we need the samples generated from π_u and the probability of each sample to calculate the loss function. Since we have already had $\boldsymbol{z}_{1:D}$ generated from π_u, we only need to calculate the logarithmic probability for $\pi_u(\boldsymbol{z}_{1:D})$ which takes the form as:

$$\ln\left[\pi_u(\boldsymbol{z}_{1:D})\right] = \ln\left[\pi_q(f_\theta^{-1}(\boldsymbol{z}_{1:D}))\right] + \ln\left(\det\left|\frac{\partial f_\theta^{-1}}{\partial \boldsymbol{z}_{1:D}}\right|\right), \tag{18}$$

where f_θ^{-1} is the inverse transformation function which can be calculated through Equation (12) and Equation (13). Since the update of $\boldsymbol{x}_{1:D}$ is divided into two parts, the calculation of $\ln\left(\det\left|\frac{\partial f_\theta^{-1}}{\partial \boldsymbol{z}_{1:D}}\right|\right)$ takes the form as:

$$\ln\left(\det\left|\frac{\partial f_\theta^{-1}}{\partial \boldsymbol{z}_{1:D}}\right|\right) = \ln\left(\det\left|\frac{\partial f_\theta^{-1}}{\partial \boldsymbol{z}_{1:d}}\right|\right) + \ln\left(\det\left|\frac{\partial f_\theta^{-1}}{\partial \boldsymbol{z}_{d+1:D}}\right|\right), \tag{19}$$

where $\ln\left(\det\left|\frac{\partial f_\theta^{-1}}{\partial \boldsymbol{z}_{1:d}}\right|\right)$ and $\ln\left(\det\left|\frac{\partial f_\theta^{-1}}{\partial \boldsymbol{z}_{d+1:D}}\right|\right)$ can be written as:

$$\begin{aligned} \ln\left|\frac{\partial f_\theta^{-1}}{\partial \boldsymbol{z}_{1:d}}\right| &= \ln|\exp(-S(\boldsymbol{z}_{d+1:D})) + \frac{\epsilon^2}{2}\cdot\nabla\nabla U((t_1, \boldsymbol{z}_{d+1:D}))_{1:d} \odot \exp(-S(\boldsymbol{z}_{d+1:D}))|, \\ \ln\left|\frac{\partial f_\theta^{-1}}{\partial \boldsymbol{z}_{d+1:D}}\right| &= \ln|\exp(-S(\boldsymbol{z}_{1:d})) + \frac{\epsilon^2}{2}\cdot\nabla\nabla U((\boldsymbol{y}_{1:d}, t_2))_{d+1:D} \odot \exp(-S(\boldsymbol{y}_{1:d}))|, \end{aligned} \tag{20}$$

where $\boldsymbol{y}_{1:d}$, t_1 and t_2 can be calculated through Equations (12) and (13).

However, in the progress of optimizing Equation (17), we find that the $\mathbb{KL}$ divergence may not be strictly non-negative because of the Monte Carlo integral, so we introduce a new objective function to overcome this problem. The detailed content is discussed in the next subsection.

4.2. Loss Function of the Training Procedure

As we have already discussed the transformation function in Langevin normalization flows, we do need a criterion to ensure that the final transformation distribution π_u will converge to the target

distribution π_t. In order to train the parameters θ which control the function σ, S and T, we choose to minimize $\mathbb{KL}(\pi_u|\pi_t)$ as the loss function to guarantee that π_u will be the expected distribution. Specifically, we take the advantage of Monte Carlo sampling to calculate the integral in $\mathbb{KL}$ divergence. Although the $\mathbb{KL}$ divergence is non-negative in theory, Monte Carlo integral may cause the abnormal of the result which means that the $\mathbb{KL}$ divergence is negative. In that case, minimizing Equation (17) will enable the loss to be smaller and thus the transformation distribution will not converge to the correct direction. To address this problem, we propose a new loss function which is defined as follows:

$$\mathcal{L}_{\pi_u \to \pi_t}(\theta) = \int \pi_u(\boldsymbol{x}) \left(\ln \frac{\pi_u(\boldsymbol{x})}{\pi_t(\boldsymbol{x})} \right)^2 \mathrm{d}\boldsymbol{x} = \mathbb{E}_{\pi_u} \left[\left(\ln \frac{\pi_u(\boldsymbol{x})}{\pi_t(\boldsymbol{x})} \right)^2 \right]. \tag{21}$$

Since we have $\mathbb{E}_{\pi_u} \left[\left(\ln \frac{\pi_u(\boldsymbol{x})}{\pi_t(\boldsymbol{x})} \right)^2 \right] \geq \mathbb{E}^2_{\pi_u} \left[\ln \frac{\pi_u(\boldsymbol{x})}{\pi_t(\boldsymbol{x})} \right]$, it is reasonable for us to minimize $\mathbb{E}_{\pi_u} \left[\left(\ln \frac{\pi_u(\boldsymbol{x})}{\pi_t(\boldsymbol{x})} \right)^2 \right]$ to achieve the purpose of minimizing the $\mathbb{KL}$ divergence.

4.3. Unnormalized Probability Distributions

In Bayesian machine learning, we generally require sampling from the posterior distribution to approximate the complex probabilistic modal. Since $p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta)$, the posterior distribution is an unnormalized distribution. So, we discuss the unnormalized probability distributions in this section. We assume that the unnormalized probability distribution $p_{unt}(\boldsymbol{x})$ equals to $\pi_t(\boldsymbol{x})Z$, where Z is the true normalization constant and π_t is the probability density function. After utilizing the Equation (21), we observe that:

$$\begin{aligned} \mathcal{L}_{\pi_u \to p_{unt}}(\theta) = & \int \pi_u(\boldsymbol{x}) \ln^2 \frac{\pi_u(\boldsymbol{x})}{\pi_t(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} + \ln^2 Z \cdot \int \pi_u(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \\ & + 2\ln Z \cdot \int \pi_u(\boldsymbol{x}) \ln \frac{\pi_u(\boldsymbol{x})}{\pi_t(\boldsymbol{x})} \mathrm{d}\boldsymbol{x}. \end{aligned} \tag{22}$$

It is noted that the third term $2\ln Z \cdot \int \pi_u(\boldsymbol{x}) \ln \frac{\pi_u(\boldsymbol{x})}{\pi_t(\boldsymbol{x})} \mathrm{d}\boldsymbol{x}$ in Equation (22) can be simplified as:

$$2\ln Z \cdot \int \pi_u(\boldsymbol{x}) \ln \frac{\pi_u(\boldsymbol{x})}{\pi_t(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} = 2\ln Z \cdot \mathbb{KL}(\pi_u|\pi_t). \tag{23}$$

The object of the optimization is to minimize the loss function $\mathcal{L}_{\pi_u \to p_{unt}}$, which is equivalent to minimize $\int \pi_u(\boldsymbol{x}) \ln^2 \frac{\pi_u(\boldsymbol{x})}{\pi_t(\boldsymbol{x})} \mathrm{d}x$ and $2\ln Z \cdot \mathbb{KL}(\pi_u|\pi_t)$, for $\ln^2 Z$ is a constant. Since the $\mathbb{KL}$ divergence is nonnegative, if $Z \in (0, 1)$, then $2\ln Z \cdot \mathbb{KL}(\pi_u|\pi_t)$ is negative. Minimizing $2\ln Z \cdot \mathbb{KL}(\pi_u|\pi_t)$ is to maximizing $\mathbb{KL}(\pi_u|\pi_t)$, which will mislead the direction of the optimization.

So as to solve this problem, we introduce a scale parameter γ. We assume $p_{unt}(\boldsymbol{x}) = \frac{\pi_t(\boldsymbol{x})Z}{\gamma}$, so the loss function can be written as:

$$\begin{aligned} \mathcal{L}_{\pi_u \to p_{unt}}(\theta) = & \int \pi_u(x) \ln^2 \frac{\pi_u(\boldsymbol{x})}{\pi_t(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} + \ln^2 \frac{Z}{\gamma} \cdot \int \pi_u(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} + 2\ln \frac{Z}{\gamma} \cdot \int \pi_u(\boldsymbol{x}) \ln \frac{\pi_u(\boldsymbol{x})}{\pi_t(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} \\ = & \frac{1}{N} \sum_{\boldsymbol{x}_{(i)} \sim \pi_u}^{N} \ln^2 \frac{\pi_u\left(\boldsymbol{x}_{(i)}\right)}{\pi_t\left(\boldsymbol{x}_{(i)}\right)} + 2\ln \frac{Z}{\gamma} \cdot \frac{1}{N} \sum_{\boldsymbol{x}_{(i)} \sim \pi_u}^{N} \ln \frac{\pi_u\left(\boldsymbol{x}_{(i)}\right)}{\pi_t\left(\boldsymbol{x}_{(i)}\right)} + \ln^2 \frac{Z}{\gamma} \\ = & \mathbb{E}_{\pi_u} \left[\ln^2 \frac{\pi_u(\boldsymbol{x})}{\pi_t(\boldsymbol{x})} \right] + 2\ln \frac{Z}{\gamma} \cdot \mathbb{E}_{\pi_u(\boldsymbol{x})} \left[\ln \frac{\pi_u(\boldsymbol{x})}{\pi_t(\boldsymbol{x})} \right] + \ln^2 \frac{Z}{\gamma}. \end{aligned} \tag{24}$$

As Equation (24) hinted, the function is composed of three terms. The first term is the same as Equation (21). The second term is the scaling term and the last term is a constant term. If $\gamma = Z$, then

we recover the Equation (21). If $\gamma < Z$, then $2\ln\frac{Z}{\gamma} \cdot \mathbb{E}_{\pi_u}\left[\ln\frac{\pi_u}{\pi_t}\right]$ is nonnegative, which not only ensures that the loss function will optimize towards the right direction but also cooperates with the information of $\mathbb{KL}$ divergence. In addition, the parameter γ is able to control the force of the optimization of $\mathbb{KL}$ divergence.

Furthermore, it is noted that the gradient of the loss function is:

$$\nabla_\theta \mathcal{L}_{\pi_u \to p_{unt}}(\theta) = 2 \sum_{i=1, \boldsymbol{x}_{(i)} \sim \pi_u}^{N} \ln\frac{\pi_u\left(\boldsymbol{x}_{(i)}\right)}{\pi_t\left(\boldsymbol{x}_{(i)}\right)} \nabla_\theta \left[\ln\frac{\pi_u\left(\boldsymbol{x}_{(i)}\right)}{\pi_t\left(\boldsymbol{x}_{(i)}\right)}\right] + 2\ln\frac{Z}{\gamma} \cdot \sum_{i=1, \boldsymbol{x}_{(i)} \sim \pi_u}^{N} \nabla_\theta \left[\ln\frac{\pi_u\left(\boldsymbol{x}_{(i)}\right)}{\pi_t\left(\boldsymbol{x}_{(i)}\right)}\right], \tag{25}$$

where $2\sum_{i=1, \boldsymbol{x}_{(i)} \sim \pi_u}^{N} \ln\frac{\pi_u(\boldsymbol{x}_{(i)})}{\pi_t(\boldsymbol{x}_{(i)})} \nabla_\theta \left[\ln\frac{\pi_u(\boldsymbol{x}_{(i)})}{\pi_t(\boldsymbol{x}_{(i)})}\right]$ gives the importance weight $\ln\frac{\pi_u(\boldsymbol{x}_{(i)})}{\pi_t(\boldsymbol{x}_{(i)})}$ for $\nabla_\theta \left[\ln\frac{\pi_u(\boldsymbol{x}_{(i)})}{\pi_t(\boldsymbol{x}_{(i)})}\right]$, for each sample $\boldsymbol{x}_{(i)}$, so it can be viewed as the rescale of the gradient of the $\mathbb{KL}$ divergence, which proves the correctness of the loss function. The complete algorithm is given in Algorithm 1.

Algorithm 1 Training NFLMC

Input: target distribution π_t, step size ϵ, learning rate β, scale parameter γ, Langevin step length Ls, number of iterations K_{iters}, sample number N, the initial distribution π_q, the transformation distribution π_u, the energy function U, the gradient of energy function ∇U and the second order gradient $\nabla\nabla U$.

Output: the parameters $\theta = (W_\sigma, W_S, W_T)$ of the sampler.

Initializing the parameters θ of the neural network.

for $k = 1$ **to** K_{iters} **do**

Sample N samples from the proposal distribution π_q.

$\boldsymbol{x} \sim \pi_q, \boldsymbol{X} = \{\boldsymbol{x}_{(n)}\}_{n=1}^{N}$

for $i = 1$ **to** Ls **do**

$\boldsymbol{x}_{d+1:D} = \boldsymbol{x}_{d+1:D} - \frac{\epsilon^2}{2}\nabla U(\boldsymbol{x}_{1:D})_{d+1:D} + \epsilon \cdot \exp(\sigma(\boldsymbol{x}_{1:d}))$

Obtaining $\boldsymbol{y}_{1:D}$ through Equation (9).

for $i = 1$ **to** Ls **do**

$\boldsymbol{x}_{1:d} = \boldsymbol{y}_{1:d} - \frac{\epsilon^2}{2}\nabla U(\boldsymbol{y}_{1:D})_{1:d} + \epsilon \cdot \exp(\sigma(\boldsymbol{y}_{d+1:D}))$

Obtaining $\boldsymbol{z}_{1:D}$ through Equation (10).

Calculating the loss $L_{\pi_u \to p_{unt}}(\theta)$ through Equation (24).

Obtaining $\ln\pi_u$ by using Equation (18).

$L_{\pi_u \to p_{unt}}(\theta) = \frac{1}{N}\sum_{n=1}^{N}\left(\ln\frac{\gamma\pi_u(\boldsymbol{z}_{(n)})}{p_{unt}(\boldsymbol{z}_{(n)})}\right)^2$

Calculating $\nabla_\theta L_{\pi_u \to p_{unt}}(\theta)$ through Equation (25).

Updating θ in the transformation functions.

$\theta = \theta - \beta\nabla_\theta L_{\pi_u \to p_{unt}}(\theta)$

In practice, there are several important points to note about the implementation of Algorithm 1. First, the proposal distribution π_q should be a simple distribution which is easy to analyze. We suggest to use the Gaussian distribution as the proposal distribution. Second, the number of the samples N should be set to a large value. In our experiments, we set $N = 8000$. Third, the scale parameter γ can be estimated through importance sampling. $\gamma = \sum_{x \sim q(x)} \frac{Z\pi_t}{q(x)}$, where $Z\pi_t$ represents the target distribution and $q(x)$ represents the proposal distribution. We have built a demo program which is available at: https://github.com/Emcc81/NFLMC.

5. Applicability of NFLMC

In this section, we will demonstrate the performance of NFLMC. We present a detailed analysis of our trained sampler on varieties of target distributions. First, we will compare the proposed sampler

with RNVP and HMC on five different distributions which are composed of the ring (the ring-shaped density), the ill-conditioned Gaussian, the strongly correlated Gaussian, the Gaussian funnel and the rough Well. After that, we present the results on two multi-modal distributions. Finally, we demonstrate the results on a task from machine learning —Bayesian logistic regression.

All our experiments are conducted on a standard computer with eight Nvidia RTX2080Ti GPUs. The nodes of each layer of the neural networks are set to be 512 with ReLU as the activation function. The number of the layer of the neural networks is set to be 3. Langevin steps are set to be 2 to 5. The number of transformation functions is set to be 8. The learning rate is set to be 0.05. The maximum iteration is set to be 10,000. We estimate scale parameter γ through importance sampling. Now, we introduce the performance index which will be used in the following parts.

Effective sample size—The variance of a Monte Carlo sampler is determined by its effective sample size (ESS) [14] which is defined as:

$$\mathrm{ESS} = N/(1 + 2 \times \sum_{s=1}^{M} \rho(s)), \tag{26}$$

where N represents the total sampling number, M is set to be 30 in our experiments and $\rho(s)$ represents the s-step autocorrelation. Autocorrelation is an index which considers the correlation between two samples. Let X be a set of samples and t be the number of iteration (t is an integer). Then X_t is the sample at time t of X. The definition of the autocorrelation between time s and t is:

$$R(s,t) = \frac{\mathbb{E}[(X_t - \mu_t)(X_s - \mu_s)]}{\sigma_t \sigma_s}, \tag{27}$$

where $\mathbb{E}$ is the expected value operator. Autocorrelation can measure the correlation between two nearby samples. If the value of autocorrelation is high, the samples are far from independent and vice versa.

Maximum mean discrepancy—The difference between samples drawn from two distributions can be measured as maximum mean discrepancy (MMD) [43] which is defined as follows:

$$MMD^2[X,Y] = \frac{1}{M^2} \sum_{i,j=1}^{M} k(x_i, x_j) - \frac{2}{MN} \sum_{i,j=1}^{M,N} k(x_i, y_j) + \frac{1}{N^2} \sum_{i,j=1}^{N} k(y_i, y_j), \tag{28}$$

where M represents the sample number in X, N represents the sample number in Y and k represents the kernel function. Through MMD, we can analyze the convergence speed of the proposed methods.

5.1. Varieties of Unimodal Distributions

Since RNVP performs well in density estimation, we utilize the loss function proposed in Equation (21) to train RNVP to sample from the target distribution. This kind of method is called the naive normalization flows Monte Carlo (NNFMC). We then compare the NFLMC with NNFMC and HMC on convergence rate and autocorrelation, respectively. In each experiment, we set the same learning rate for NFLMC and NNFMC. The initial distributions are all set to be the standard normal distribution. We next introduce the distributions used in the experiment.

Ring: The ring shaped target density. The analytic form of the energy function of the ring is: $U(x) = \frac{(\sqrt{x_1^2 + x_2^2} - 2)^2}{0.32}$.

Ill-conditioned Gaussian: Gaussian distribution with diagonal covariance spaced log-linearly between 10^{-2} and 10^2.

Strongly correlated Gaussian: We rotate a diagonal Gaussian with variances $[10^2, 10^{-2}]$ by $\frac{\pi}{4}$. This is an extreme version of an example from Brooks [14].

Rough well: A similar example from Sohl-Dickstein et al. [44] and its energy function is: $U(x) = \frac{1}{2} x^{\mathrm{T}} x + \eta \sum_i \cos(\frac{x_i}{\eta})$. We set $\eta = 10^{-2}$.

Gaussian funnel: We conduct our experiment on a 2-*D* funnel, whose energy function takes the form as: $U(\mathbf{x}) = \frac{1}{2}\left[\left(\frac{x_1}{\sigma}\right)^2 + \frac{x_2^2}{\exp(x_1)} + \ln\left(2\pi \cdot \exp(x_1)\right)\right]$ and we set $\sigma = 1.0$.

As Figure 2 illustrates, our method performs better in all these distributions in terms of convergence rate. In ill conditioned Gaussian, rough well, Gaussian funnel and strongly corrected Gaussian with $\mu = [0, 0]$, NFLMC gains fast convergence, which indicates that the Langevin diffusions do help the normalization flows to find the correct direction. In strongly corrected Gaussian with $\mu = [10, 10]$, NNFMC is unable to converge to the target distribution, since the loss remains high during the training procedure. It is the utilization of gradient information of the target distribution that aids NFLMC to converge the target distribution rapidly.

In ring-shaped distribution, NNFMC has a significant fluctuation in the process of training, while NFLMC converges rapidly during the training procedure, which shows the stability of NFLMC. Since the loss of NNFMC has large fluctuation, we carefully tune the learning rate for NNFMC. As Figure 3 illustrates, NFLMC converges to the target distribution more quickly than NNFMC. Besides, NNFMC has a large error while NFLMC is able to sample from the target distribution precisely.

So what causes the large fluctuation of NNFMC? We think that the lack of strong guidance when exploring the state space makes NNFMC difficult to converge. Since the initial distribution is a standard normal distribution, samples from the initial distribution have large distance with the samples of ring-shaped distribution. So it is challenging for NNFMC to explore state space and the value of the loss function has large fluctuation. In order to verify this thought, we enlarge the size of the ring distribution whose energy function has the form: $U(x) = \frac{(\sqrt{x_1^2 + x_2^2} - 3)^2}{0.32}$ and we observe that NNFMC fails to sample from this distribution while NFLMC can still converge to the target distribution. As Figure 3 shows, NNFMC cannot find the target distribution, while NFLMC still performs well, for NFLMC utilizes the gradient information of the target distribution.

We then compare our method with HMC in terms of the autocorrelation on five different distributions.

Figure 4 demonstrates that NFLMC obtains better performance in autocorrelation, which indicates that NFLMC overcomes the defects of the MCMC samplers. HMC (0.05) and HMC (0.1) represent the HMC sampler with different step size.

5.2. Mixtures of Gaussian Distributions

We conduct our second experiment on two multi-modal distributions where we consider two simple 2-*D* mixtures of Gaussian distributions (MOG) whose probability density function are analytically available. First, we consider a MOG whose modes have the same probability and then we consider a MOG whose modes have different probabilities and further distance. The first distribution is defined as: $p(x) = \frac{1}{2}\mathcal{N}(x|\mu, \mathbf{I}) + \frac{1}{2}\mathcal{N}(x| - \mu, \mathbf{I})$, where $\mu = (2.5, -2.5)$. The second distribution is defined as: $p(x) = 0.88\mathcal{N}(x|\mu, \mathbf{I}) + 0.12\mathcal{N}(x| - \mu, \mathbf{I})$, where $\mu = (4, -4)$. The experiment settings is the same with Tripuraneni et al. [24]. The purpose of the experiments is to sample points which are i.i.d. distributed in multi-modal distributions correctly.

We compare HMC [14], MHMC [24], MGHMC [30] and NICE-MC [29] against NFLMC. First, we compare the MMD of these methods and then averaged autocorrelation is used to compare the performance of each method further. Each MCMC method is run 32 times and 20,000 iterations with 11,000 burn-in samples. The number of leap-frog steps is uniformly drawn from $(100 - l, 100 + l)$ with $l = 20$, which is suggested by Livingstone et al. [45]. We set step size $\epsilon = 0.05$ and the initiate position $x = (0, 0)$. The initial distribution for NFLMC is a Gaussian distribution with $\mu = [0, 0]$ and $\mathrm{diag}(\sigma) = [2, 2]$. As Figure 5 illustrates, NFLMC obtains excellent performance compared with MHMC, HMC and MGHMC regarding MMD and autocorrelation. In addition, NFLMC has a smaller variance of MMD compared with NICE-MC. However, when it comes to autocorrelation, NICE-MC shows the huge fluctuation, while NFLMC remains steady, which manifests the stability of NFLMC.

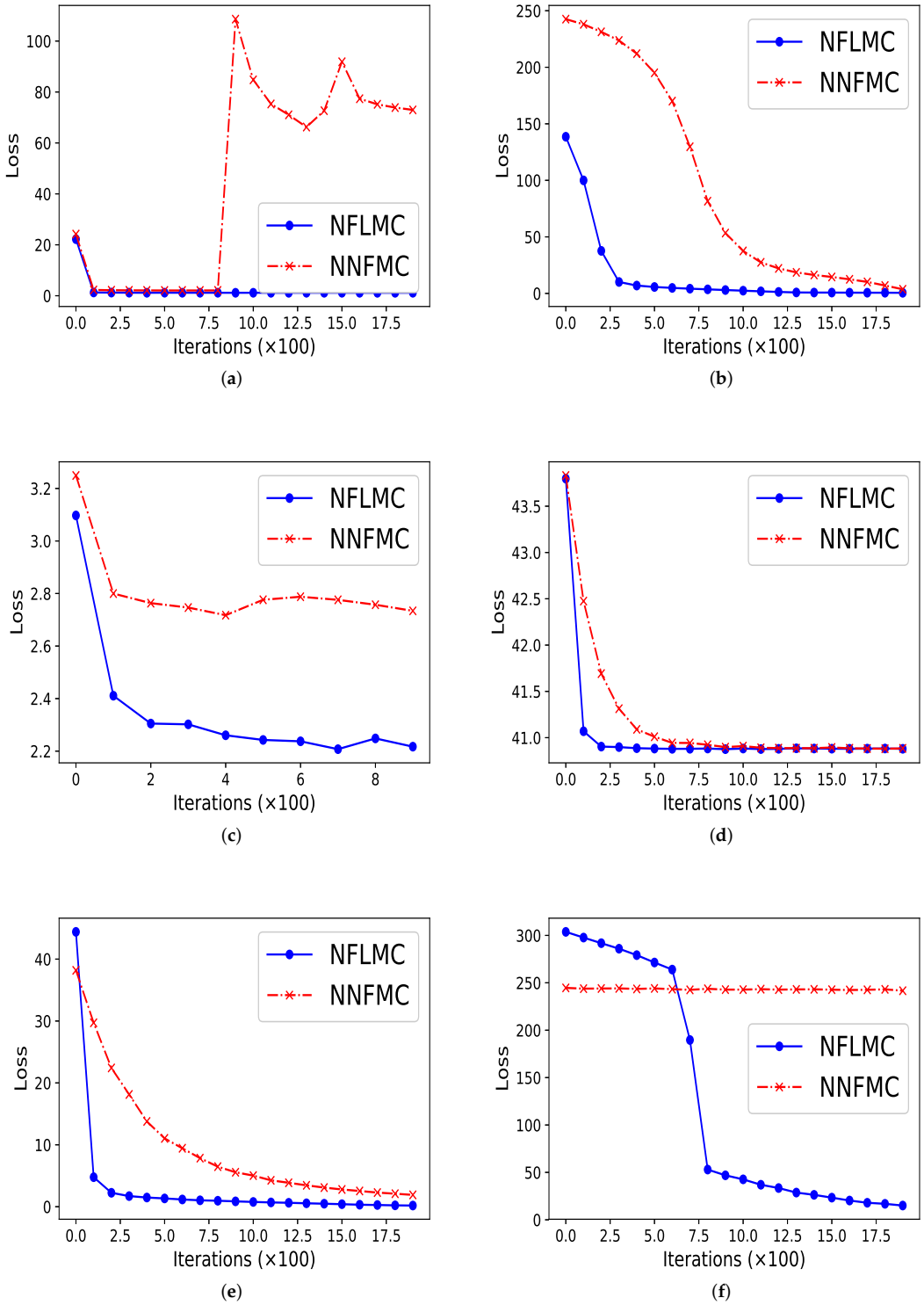

Figure 2. The comparison of normalization flows Monte Carlo (NFLMC) and naive normalization flows Monte Carlo sampler (NNFMC) on six different distributions. The abscissa represents the number of iterations and the ordinate represents the value of loss in training procedure. (**a**) The performance of NFLMC and NNFMC on ring. (**b**) The performance of NFLMC and NNFMC on ill conditioned Gaussian. (**c**) The performance of NFLMC and NNFMC on rough well. (**d**) The performance of NFLMC and NNFMC on Gaussian funnel. (**e**) The performance of NFLMC and NNFMC on strongly correlated Gaussian with $\mu = [0, 0]$. (**f**) The performance of NFLMC and NNFMC on strongly correlated Gaussian with $\mu = [10, 10]$.

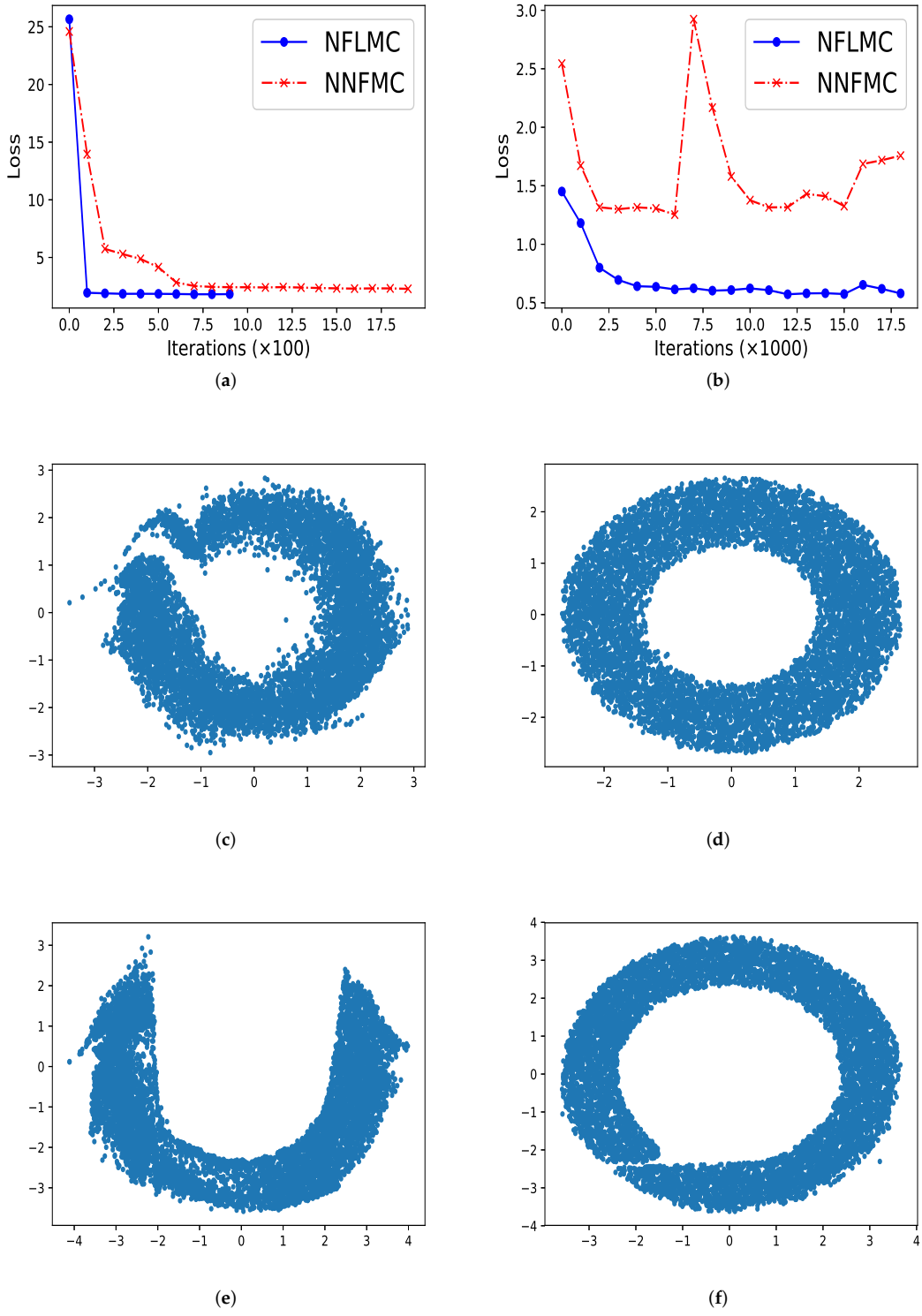

Figure 3. The performance of naive normalization flows Monte Carlo (NNFMC) and Langevin normalization flows Monte Carlo (NFLMC) on two ring distributions. Specially, for the ring distributions with large radius, we show the change of the loss with respect to iterations after 1000 iterations. (**a**) The loss of NFLMC and NNFMC on a ring with small radius. (**b**) The loss of NFLMC and NNFMC on a ring with large radius. (**c**) The scatter diagram of NNFMC on a ring with small radius. (**d**) The scatter diagram of NFLMC on a ring with small radius. (**e**) The scatter diagram of NNLMC on a ring with large radius. (**f**) The scatter diagram of NFLMC on a ring with large radius.

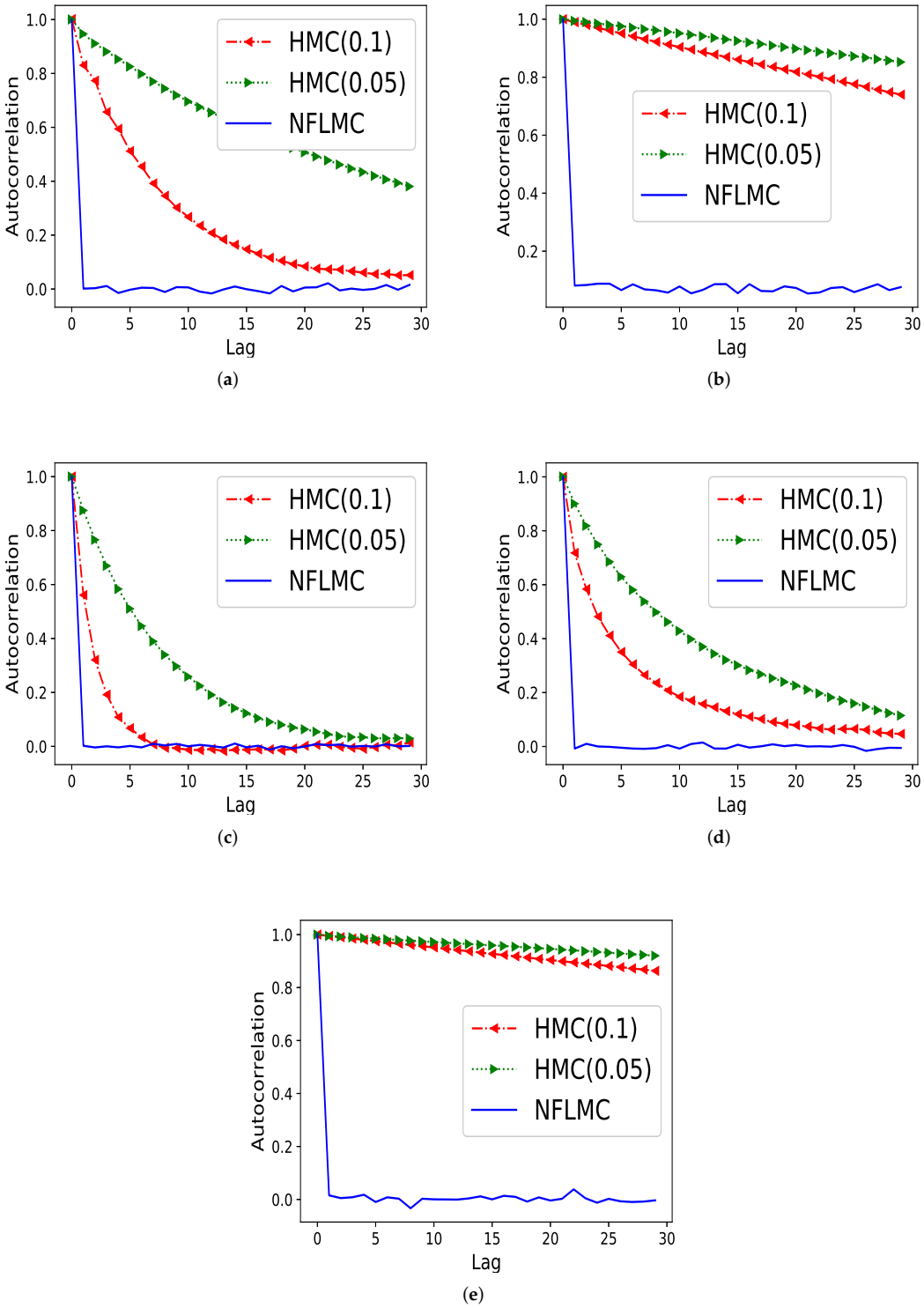

Figure 4. The comparison of Hamiltonian Monte Carlo (HMC) and NFLMC on the autocorrelation on five different distributions. The abscissa represents the steps between the samples and the ordinate represents the autocorrelation between samples. (**a**) The autocorrelation of HMC and NFLMC on ring. (**b**) The autocorrelation of HMC and NFLMC on ill conditioned Gaussian. (**c**) The autocorrelation of HMC and NFLMC on rough well. (**d**) The autocorrelation of HMC and NFLMC on Gaussian funnel. (**e**) The autocorrelation of HMC and NFLMC on strongly correlated Gaussian with $\mu = [0, 0]$.

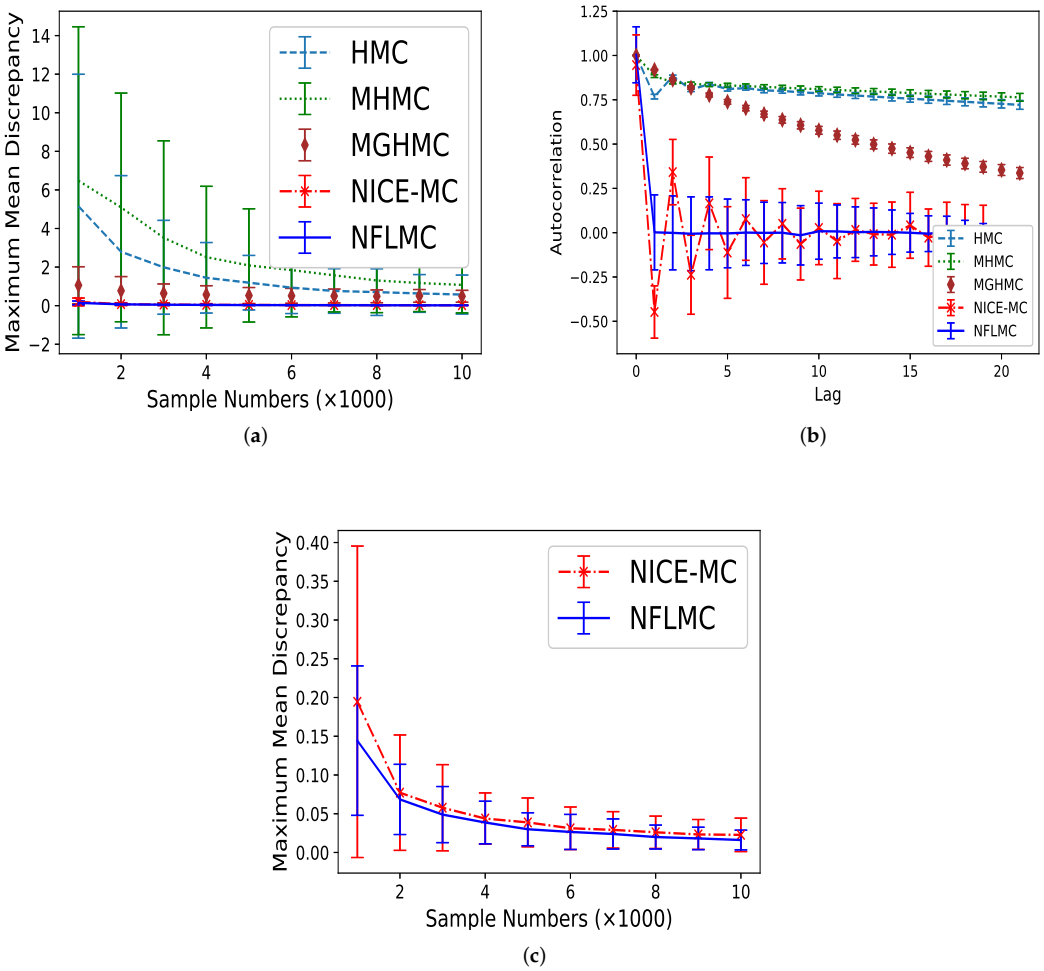

Figure 5. The comparison of five different methods on maximum mean discrepancy (MMD) and autocorrelation on Gaussian mixtures distribution. (**a**) The relationship between autocorrelation and sample numbers. (**b**) The relationship between autocorrelation and lag. (**c**) The detailed comparison of NFLMC and NICE-MC on MMD.

We then discuss the circumstance in which the modes are far from each other and with different probabilities. When μ in MOG become larger, for instance, $\mu = (4, -4)$. In Hamiltonian dynamics, there exists a significant force in this low probability regions which hinder samplers from jumping out of the current mode. In other words, the gradients in boundary regions are tremendous and the momentum will increasingly decrease until it changes its direction which makes HMC and MHMC challenging to sample from the target distribution. So we compare NFLMC with parallel HMC and NICE-MC. The scatter diagram of both parallel HMC and NFLMC is demonstrated in Figure 6. We observe that parallel HMC can sample from the multi-modal distribution but it cannot precisely estimate the probability of each mode.

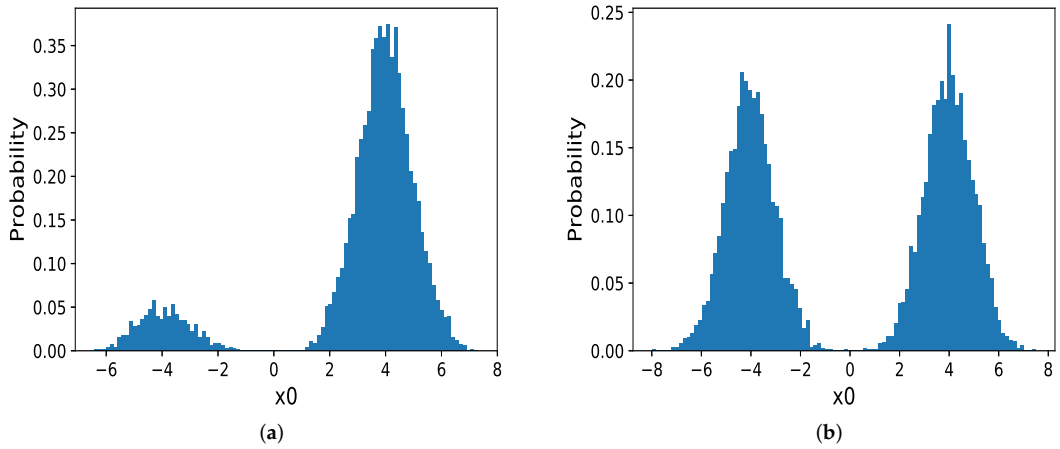

Figure 6. The performance of parallel HMC and NFLMC on the mixtures of Gaussian with different probabilities of the modes. (**a**) The histogram of NFLMC on x_0. (**b**) The histogram of parallel HMC on x_0.

For parallel HMC, it seems that two modes have the same probability. However, the real probability of each mode is $\pi_1 = 0.12, \pi_2 = 0.88$. As Figure 7 illustrates, compared with NICE-MC, NFLMC converges quickly to the target distribution while gains the lower autocorrelation. It is the fact that NFLMC takes advantage of the neural networks to explore the phase space, which results in good performance.

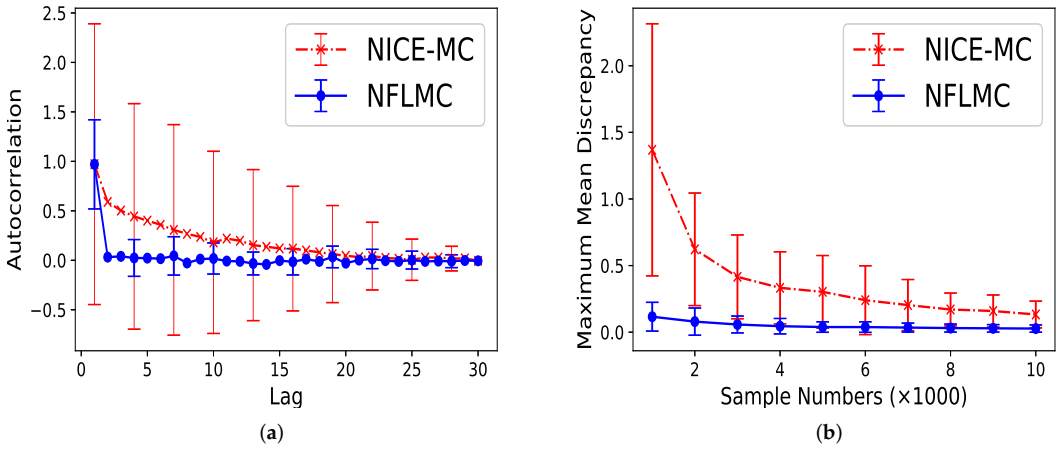

Figure 7. The performance of NFLMC and NICE-MC on the mixtures of Gaussians with different probabilities of the modes. (**a**) The autocorrelation of the samples generated from NFLMC and NICE-MC. (**b**) The change of MMD on NFLMC and NICE-MC.

5.3. Bayesian Logistic Regression

Logistic regression (LR) [46] is a traditional way for classification. Employing maximizing the logistic likelihood function, we can get the optimized parameters. Through the parameters, we can predict the class of the data. Bayesian logistic regression [47] is also a classic model for classification which takes advantage of logistic sigmoid function as the likelihood function. For the two-class classification, the likelihood function is defined as: $p(\mathrm{t}|w) = \prod_{n=1}^{N}[1-y_n]^{1-t_n}$, where $\mathrm{t} = (t_1, \ldots, t_N)^\top$ and $y_n = p(\mathcal{C}_1|\phi_n) = \sigma(w^\top\phi)$. t_n represents the category of the data and y_n represents the probability of the data belonging to one class. Through integrating the logistic function on the posterior distribution, we can get the class of the data. However, sometimes the integral

is difficult to calculate, variational Bayesian logistic regression (VBLR) substitute the real posterior distribution to the variational distribution. Instead of using variational inference, we apply Monte Carlo sampling technology to this model. Through sampling from the posterior distribution, the class of the data can be estimated.

We evaluate our methods on nine real-world datasets from UCI repository [48]—Pima Indian (Pi), Haberman (Ha), Blood (Bl), Immunotherapy (Im), Indian (In), Mammographic (Ma), Heart (He), German (Ge) and Australian (Au) using Bayesian logistic regression. Feature dimensions are from 3 to 25 and the data instances are from 306 to 1086. All datasets are normalized to have zero mean value and unit variance. First, we set the standard normal distribution $\mathcal{N}(0, \mathbf{I})$ as the prior distribution for the parameters. In each experiment, we run 9000 iterations with 1000 burn-in samples for HMC. For NFLMC, we set the standard normal distribution as the initial distribution. We train the NFLMC sampler until the value of loss function converges and then sampling 8000 samples using the well-trained sampler. The maximum number of iterations is set to be 10^5. We set the step size $\epsilon = 0.001$ and we run ten times to calculate the mean and the standard deviation.

Results regarding the accurate rate of prediction and area under the receiver operating characteristic curve (AUC) [49] are summarized in Tables 1 and 2, respectively. The results show that in these nine datasets, NFLMC yields good performance in accurate rate and AUC. In order to further compare the quality of the samples, we calculate the mean of ESS of each dimension for both HMC and NFLMC. We use 30 steps autocorrelation to calculate this value. Table 3 demonstrates that NFLMC achieves higher ESS than HMC, which suggests that NFLMC has lower autocorrelation than HMC for each dimension.

Table 1. Classification accuracy for variational Bayesian logistic regression (VBLR), logistic regression (LR), HMC and NFLMC on nine different datasets.

Data	LR	VBLR	HMC	NFLMC
Ha	69.3 ± 0.2	69.3 ± 0.1	69.3 ± 0.2	69.4 ± 0.1
Pi	76.6 ± 0.2	76.2 ± 0.1	76.6 ± 0.1	76.6 ± 0.1
Ma	82.5 ± 0.3	83.1 ± 0.1	83.1 ± 0.1	83.1 ± 0.2
Bl	76.0 ± 0.2	76.0 ± 0.2	76.0 ± 0.3	76.0 ± 0.1
Im	77.7 ± 0.3	77.75 ± 0.4	83.2 ± 0.2	83.3 ± 0.2
In	75.8 ± 0.3	73.2 ± 0.2	73.2 ± 0.2	74.1 ± 0.2
He	75.9 ± 0.2	75.9 ± 0.2	75.9 ± 0.2	75.9 ± 0.1
Ge	71.5 ± 0.1	71.5 ± 0.1	72.5 ± 0.2	73.0 ± 0.1
Au	86.9 ± 0.2	87.6 ± 0.2	87.6 ± 0.2	87.7 ± 0.1

Table 2. Area under the receiver operating characteristic curve (AUC) for VBLR, LR, HMC and NFLMC on nine different datasets.

Data	LR	VBLR	HMC	NFLMC
Ha	62.7 ± 0.1	63.2 ± 0.1	63.0 ± 0.2	63.2 ± 0.1
Pi	79.2 ± 0.2	79.3 ± 0.1	79.3 ± 0.1	79.5 ± 0.1
Ma	89.9 ± 0.1	89.8 ± 0.1	89.89 ± 0.1	89.9 ± 0.2
Bl	73.5 ± 0.3	73.4 ± 0.3	74.4 ± 0.3	73.5 ± 0.2
Im	76.7 ± 0.3	78.5 ± 0.5	89.2 ± 0.2	89.3 ± 0.3
In	73.2 ± 0.3	73.2 ± 0.2	72.4 ± 0.2	72.8 ± 0.4
He	80.1 ± 0.2	81.3 ± 0.2	82.2 ± 0.3	84.8 ± 0.2
Ge	74.7 ± 0.2	75.5 ± 0.2	76.7 ± 0.3	76.9 ± 0.1
Au	92.5 ± 0.2	93.9 ± 0.2	93.9 ± 0.3	94.0 ± 0.2

Table 3. The mean of effective sample size of each dimension for HMC and NFLMC on nine differen datasets.

Data	HMC	NFLMC	Data	HMC	NFLMC
Ha	107.69	2503.75	In	408.87	3590.34
Pi	73.08	3534.50	He	1093.10	3200.00
Ma	670.72	2570.69	Ge	7.19	2842.92
Bl	808.84	2824.87	Au	220.60	2538.25
Im	1879.78	1917.54			

6. Discussion and Conclusions

In this study, we propose Langevin normalization flows and develop Langevin normalization flows Monte Carlo, a novel scalable sampling algorithm which exploits the flexibility of the neural networks and efficient exploration of Langevin diffusions. We design the appropriate loss function to train the sampler to ensure that the sampler is able to converge to the target distribution. We also discuss the unnormalized probability distributions and propose the appropriate loss function to these distributions. The experiments conducted on synthetic and real datasets suggest that our method is able to sample from the target distributions precisely and independently.

Although HMC has various advantages, it is difficult for us to design the model based on HMC, because the auxiliary momentum variable should be carefully concerned in the transformation function of NFs. In the future, we plan to design the neural network sampler based on Hamiltonian dynamics.

Author Contributions: Conceptualization, M.G. and S.S.; methodology, M.G. and S.S.; experiments, M.G. and Y.L.; formal analysis, M.G.; writing–original draft preparation, M.G.; writing–review and editing, S.S. and Y.L.; supervision, S.S.

Funding: This work is supported by the National Natural Science Foundation of China under Project 61673179 and the Strategic Priority Research Program of ECNU.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gunji, Y.P.; Murakami, H.; Tomaru, T.; Basios, V. Inverse Bayesian inference in swarming behaviour of soldier crabs. *Philos. Trans. R. Soc. Math. Phys. Eng. Sci.* **2018**, *376*, 1–17.
2. Imani, M.; Ghoreishi, S.F.; Allaire, D.; Braga-Neto, U.M. MFBO-SSM: Multi-fidelity Bayesian optimization for fast inference in state-space models. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27–28 January 2019; pp. 7858–7865.
3. Livingstone, S.; Girolami, M. Information-geometric Markov chain Monte Carlo methods using diffusions. *Entropy* **2014**, *16*, 3074–3102.
4. Robert, C.P.; Casella, G. *Monte Carlo Statistical Methods*; Springer: Berlin/Heidelberg, Germany, 2013.
5. Altieri, N.; Duvenaud, D. Variational Inference with Gradient Flows. Available online: http://approximateinference.org/accepted/AltieriDuvenaud2015.pdf (10 November 2015).
6. Blei, D.; Kucukelbir, A.; McAuliffe, J. Variational inference: a review for statisticians. *J. Am. Stat. Assoc.* **2017**, *112*, 859–887.
7. Hock, K.; Earle, K. Markov chain Monte Carlo used in parameter inference of magnetic resonance spectra. *Entropy* **2016**, *18*, 57–69.
8. Seo, J.; Kim, Y. Approximated information analysis in Bayesian inference. *Entropy* **2015**, *17*, 1441–1451.
9. Imani, M.; Ghoreishi, S.F.; Braga-Neto, U.M. Bayesian control of large MDPs with unknown dynamics in data-poor environments. In Proceedings of the Advances in Neural Information Processing Systems (NIPS 2018), Montreal, QC, Canada, 3–8 December 2018; pp. 8146–8156.
10. Sun, S. A review of deterministic approximate inference techniques for Bayesian machine learning. *Neural Comput. Appl.* **2013**, *23*, 2039–2050.
11. Neal, R.M. Slice sampling. *Ann. Stat.* **2003**, *31*, 705–741.

12. Li, Q.; Newton, K. Diffusion equation-assisted Markov chain Monte Carlo methods for the inverse radiative transfer equation. *Entropy* **2019**, *21*, 291–315.
13. Skeel, R.; Fang, Y. Comparing Markov chain samplers for molecular simulation. *Entropy* **2017**, *19*, 561–576.
14. Brooks, S.; Gelman, A.; Jones, G.; Meng, X. *Handbook of Markov chain Monte Carlo*; CRC Press: Boca Raton, FL, USA, 2011.
15. Hokman, M.D.; Gelman, A. The no-u-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *J. Mach. Learn. Res.* **2014**, *15*, 1593–1623.
16. Wang, Z.; Mohamed, S.; Freitas, N. Adaptive Hamiltonian and Riemann manifold Monte Carlo. In Proceedings of the International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013; pp. 1462–1470.
17. Duane, S.; Kennedy, A.D.; Pendleton, B.J.; Roweth, D. Hybrid Monte Carlo. *Phys. Lett. B* **1987**, *195*, 216–222.
18. Celeux, G.; M.Hurn.; Robort, C.P. Computational and inferential difficulties with mixture posterior distributions. *J. Am. Stat. Assoc.* **2000**, *95*, 957–970.
19. Neal, R.M. Annealed importance sampling. *Stat. Comput.* **2001**, *11*, 125–139.
20. Rudoy, D.; Wolfe, P.J. Monte Carlo methods for multi-modal distributions. In Proceedings of the Fortieth Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 6–9 November 2006; pp. 2019–2023.
21. Sminchisescu, C.; Welling, M. Generalized darting Monte Carlo. In Proceedings of the Artificial Intelligence and Statistics, San Juan, Puerto Rico, 21–24 March 2007; pp. 516–523.
22. Craiu, R.V. Learn from thy neighbor: parallel-chain and regional adaptive MCMC. *J. Am. Stat. Assoc.* **2009**, *104*, 1454–1466.
23. Girolami, M.; Calderhead, B. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *J. R. Stat. Soc.* **2011**, *73*, 123–214.
24. Tripuraneni, N.; Rowland, M.; Ghahramani, Z.; Turner, R. Magnetic Hamiltonian Monte Carlo. In Proceedings of the International Conference on Machine Learning, Sydney, NSW, Australia, 6–11 August 2017; pp. 3453–3461.
25. Ahn, S.; Chen, Y.; Welling, M. Distributed and adaptive darting Monte Carlo through regenerations. In Proceedings of the Artificial Intelligence and Statistics, Scottsdale, AZ, USA, 29–30 April 2013; pp. 108–116.
26. Lan, S.; Streets, J.; Shahbaba, B. Wormhole Hamiltonian Monte Carlo. In Proceedings of the AAAI Conference on Artificial Intelligence, Quebec City, QC, Canada, 27–31 July 2014; pp. 1953–1959.
27. Rezende, D.; Mohamed, S. Variational inference with normalizing flows. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 1530–1538.
28. Hastings, W.K. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* **1970**, *57*, 97–109.
29. Song, J.; Zhao, S.; Ermon, S. A-nice-mc: Adversarial training for MCMC. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 5140–5150.
30. Zhang, Y.; Wang, X.; Chen, C.; Henao, R.; Fan, K.; Carin, L. Towards unifying Hamiltonian Monte Carlo and slice sampling. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 1741–1749.
31. Dinh, L.; Sohl-Dickstein, J.; Bengio, S. Density estimation using real NVP. *arXiv* **2016**, arXiv:1605.08803.
32. Paige, B.; Wood, F. Inference networks for sequential Monte Carlo in graphical models. In Proceedings of the International Conference on Machine Learning, New York City, NY, USA, 19–24 June 2016; pp. 3040–3049.
33. Papamakarios, G.; Murray, I. Fast ε-free inference of simulation models with Bayesian conditional density estimation. In *Advances in Neural Information Processing Systems*; Barcelona, Spain, 5–10 December 2016; pp. 1028–1036.
34. Ballé, J.; Laparra, V.; Simoncelli, E.P. Density modeling of images using a generalized normalization transformation. *arXiv* **2015**, arXiv:1511.06281.
35. Kingma, D.P.; Salimans, T.; Jozefowicz, R.; Chen, X.; Sutskever, I.; Welling, M. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*; Barcelona, Spain, 5–10 December 2016; pp. 4743–4751.
36. Dinh, L.; Krueger, D.; Bengio, Y. NICE: Non-linear independent components estimation. *arXiv* **2014**, arXiv:1410.8516.

37. Oord, A.V.D.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A.; Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv* **2016**, arXiv:1609.03499.
38. Durrett, R. *Stochastic Calculus: A Practical Introduction*; CRC Press: Boca Raton, FL, USA, 2018.
39. Øksendal, B. Stochastic differential equations. In *Stochastic Differential Equations*; Springer: Berlin/Heidelberg, Germany, 2003.
40. Roberts, G.O.; Stramer, O. Langevin diffusions and Metropolis-Hastings algorithms. *Methodol. Comput. Appl. Probab.* **2002**, *4*, 337–357.
41. Kloeden, P.E.; Platen, E. *Numerical Solution of Stochastic Differential Equations*; Springer: Berlin/Heidelberg, Germany, 2013.
42. Martino, L.; Read, J. On the flexibility of the design of multiple try Metropolis schemes. *Comput. Stat.* **2013**, *28*, 2797–2823.
43. Roberts, G.O.; Stramer, O. A kernel two-sample test. *J. Mach. Learn. Res.* **2012**, *13*, 723–773.
44. Sohl-Dickstein, J.; Mudigonda, M.; DeWeese, M.R. Hamiltonian Monte Carlo without detailed balance. In Proceedings of the International Conference on Machine Learning, Beijing, China, 21–26 June 2014; pp. 719–726.
45. Livingstone, S.; Betancourt, M.; Byrne, S.; Girolami, M. On the geometric ergodicity of Hamiltonian Monte Carlo. *arXiv* **2016**, arXiv:1601.08057.
46. Freedman, D.A. *Statistical Models: Theory and Practice*; Cambridge University Press: Cambridge, UK, 2009.
47. MacKay, D.J.C. The evidence framework applied to classification networks. *Neural Comput.* **1992**, *4*, 720–736.
48. Dua, D.M.; Graff, C. UCI Machine Learning Repository. 2017. Available online: https://archive.ics.uci.edu/ml/index.php (accessed on 10 November 2017).
49. Hanley, J.A.; McNeil, B.J. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology* **1983**, *148*, 839–843.