entropy

MDPI

Article

Adversarially Training MCMC with Non-Volume-Preserving Flows

Shaofan Liu and Shiliang Sun *

School of Computer Science and Technology, East China Normal University, Shanghai 200062, China; 51194506021@stu.ecnu.edu.cn

* Correspondence: slsun@cs.ecnu.edu.cn

Abstract: Recently, flow models parameterized by neural networks have been used to design efficient Markov chain Monte Carlo (MCMC) transition kernels. However, inefficient utilization of gradient information of the target distribution or the use of volume-preserving flows limits their performance in sampling from multi-modal target distributions. In this paper, we treat the training procedure of the parameterized transition kernels in a different manner and exploit a novel scheme to train MCMC transition kernels. We divide the training process of transition kernels into the exploration stage and training stage, which can make full use of the gradient information of the target distribution and the expressive power of deep neural networks. The transition kernels are constructed with non-volume-preserving flows and trained in an adversarial form. The proposed method achieves significant improvement in effective sample size and mixes quickly to the target distribution. Empirical results validate that the proposed method is able to achieve low autocorrelation of samples and fast convergence rates, and outperforms other state-of-the-art parameterized transition kernels in varieties of challenging analytically described distributions and real world datasets.

Keywords: Hamiltonian Monte Carlo; flow models; Markov chain Monte Carlo; statistical pattern recognition; Bayesian machine learning

Citation: Liu, S.; Sun, S. Adversarially Training MCMC with Non-Volume-Preserving Flows. *Entropy* **2022**, *24*, 415. https://doi.org/10.3390/e24030415

Academic Editors: Udo von Toussaint and Mateu Sbert

Received: 16 December 2021
Accepted: 14 March 2022
Published: 16 March 2022

1. Introduction

Markov chain Monte Carlo (MCMC) is one of the most powerful approaches to sample from complex target distributions in statistical pattern recognition and Bayesian machine learning. It has been widely employed in probabilistic modeling and inference [1,2]. MCMC methods approximate target distributions by generating samples from a proposal distribution depending on the last sample, and ensure that the samples converge to the target distribution by satisfying the detailed balance [3]. In theory, we can arbitrarily choose a proposal distribution and employ the Metropolis–Hastings (MH) [4] algorithm to satisfy the detailed balance. However, the specific choice of the proposal distribution has a significant effect on the convergence and mixing speed [5,6]. For instance, a Gaussian distribution centered on the current state is a regular choice for the proposal distribution, which is also called a random walk proposal [7,8]. Although it has a particularly nice intuition, the proposal scales poorly with the increase of dimensions and complexity of the target distribution [9,10].

Hamiltonian Monte Carlo (HMC) [7] introduces auxiliary momentum variables v to extend the state space to (x, v). In every update step of HMC, the leapfrog discretization scheme is used to update x and v alternately [11]. Before a new update step, the auxiliary variables are resampled to explore in the new equal-energy surface (or iso-probability contour) so that x could change greatly in a systematic way. As a result, HMC can traverse a long distance in the state space with a single MH step and prevent the random walk behavior [7,12]. However, HMC can suffer dramatically in highly complex, multi-modal distributions, for it is difficult to traverse low-density regions [13,14].

The core problem of MCMC is to build transition kernels (also known as the proposal distribution) that can explore and sample from the target distribution efficiently. Recently, training MCMC transition kernels parameterized by deep neural networks achieves great success [13,15,16]. A-NICE-MC [15] uses volume-preserving flows [17] as transition kernels and trains via generative adversarial network (GAN) [18,19]. L2HMC [13] uses non-volume-preserving flows incorporating gradient information of the target distribution as transition kernels, and the sampler is trained to maximize the expected squared jumped distance which is equivalent to minimizing the lag-one autocorrelation [20].

A-NICE-MC uses volume-preserving flows as the transition kernels bring the random walk behavior when traversing between different energy levels, and then they can get correct samples through one MH step. They maintain a buffer to save the sample points, use samples from the buffer as the learning objective of their transition kernels, and then the buffer is updated with the new sample points [15]. In this way, they do not need to compute the gradient of the target distribution since the sample points in the buffer already contain the geometric information of the target distribution. However, although eliminating the calculation of the gradient reduce the computational burden, without the guidance of gradient information, this exploration is almost random. L2HMC introduces neural networks into the leapfrog integrator to construct more flexible transition kernels, where the gradient information of the target distribution is used as the input of neural networks. Consequently, when exploring the same region twice, the gradient information of the region also needs to be calculated twice since we can not record all gradients of the target distribution. Thus the gradient information can not be exploited efficiently in the training process, which may result in unnecessarily long training time.

To solve these issues, we design a new method that uses non-volume-preserving flows as transition kernels to mix faster for the target distribution. Specifically, we divide the training process of the parameterized transition kernels into the exploration stage and training stage. In the exploration stage, we use a gradient-based transition kernel to explore the target distribution as much as possible, and we can design powerful exploration operators without the restriction of convergence. We save samples of the exploration stage and use these samples to record the geometry of the target distribution. In the adversarial training stage, we optimize the parameters of the transition kernels by minimizing the distance between samples generated from transition kernels and samples from the exploration stage. Then samples generated from the trained transition kernels are accepted through MH steps [21]. The accepted samples will be used to replace some of the samples collected in the previous exploration stage. With the increase of the training iterations, we can generate samples with better quality at a low computation cost since the collected samples record the geometry of the target distribution. As a result, in the training stage, we need not compute the gradient to get a high acceptance rate like L2HMC or vanilla HMC. Gradients are only needed in the exploration stage.

The rest of this paper is organized as follows. In Section 2, we firstly introduce the necessary background on MCMC methods and non-volume-preserving flows. We present the proposed method in detail in Section 3. In Section 4, we describe the core ideas of A-NICE-MC and L2HMC, we also introduce the motivation of our method in this section. Experiments are given in Section 5. Finally, we conclude this paper in Section 6.

2. Background

2.1. Markov Chain Monte Carlo and Metropolis–Hasting Algorithm

MCMC methods [2] aim to construct an ergodic Markov chain converging to $p(x)$ under a target density $p(x) = \frac{\tilde{p}(x)}{Z_p}$, where $\tilde{p}(x)$ can be readily evaluated and Z_p is an unknown constant. At each step of the algorithm, the new sample x' is obtained from the transition kernel (or proposal distribution) $K_\theta(x'|x)$ which depends on the current state x. The MH step is utilized to make the Markov chain satisfy the detailed balance which can be written as:

$$p(x')K_\theta(x|x') = p(x)K_\theta(x'|x). \tag{1}$$

Specifically, a new sample x' generated from a proposal distribution $q_\theta(x'|x)$ is accepted with probability $A_\theta(x'|x)$ which takes the form as:

$$A_\theta(x'|x) = \min\left(1, \frac{p(x')q_\theta(x|x')}{p(x)q_\theta(x'|x)}\right). \tag{2}$$

For vanilla HMC, assume that ξ is a state in the Hamiltonian dynamics. The transition from ξ to $\xi^\star$ is deterministic, invertible and volume-preserving, which means that $q_\theta(\xi|\xi^\star) = q_\theta(\xi^\star|\xi)$ [15]. According to the change of variable formula which takes the form as:

$$p_X(x) = p_Z(f(x))\left|\det\left(\frac{\partial f(x)}{\partial x^\top}\right)\right|, \tag{3}$$

the MH acceptance probability for the HMC proposal [22] can be simplified as:

$$A_\theta(\xi^\star|\xi) = \min\left(1, \frac{p(\xi^\star)}{p(\xi)}\left|\frac{\partial q_\theta(\xi^\star|\xi)}{\partial \xi^\top}\right|\right), \tag{4}$$

where $\left|\frac{\partial q_\theta(\xi^\star|\xi)}{\partial \xi^\top}\right| = \frac{q_\theta(\xi|\xi')}{q_\theta(\xi'|\xi)} = 1$.

2.2. Hamiltonian Monte Carlo and Exploration on Total Energy Function

In Hamiltonian Monte Carlo (HMC) [7], we assume that the target distribution $p(x)$ takes the form as:

$$p(x) = \frac{1}{Z_U}\exp[-U(x)],$$

where $U(x)$ is interpreted as the potential energy of the dynamics, and Z_U is an unknown constant. Auxiliary variables v can be interpreted as the momentum of the dynamics, and the kinetic energy takes the form as:

$$K(v) = \frac{1}{2}v^\top v.$$

The total energy H is the sum of potential and kinetic energies:

$$H(x, v) = U(x) + K(v),$$

and the joint distribution can be written as:

$$p(x, v) = \frac{1}{Z_H}\exp[-H(x, v)].$$

When simulating Hamiltonian dynamics for a finite time, the value of x and v will change with the total energy conserving. In practice, x and v are updated through the leapfrog discretization, which for a single time step consists of:

$$\begin{aligned} v &= v - \frac{\epsilon}{2}\partial_x U(x); \\ x' &= x + \epsilon v; \\ v' &= v - \frac{\epsilon}{2}\partial_x U(x'), \end{aligned} \tag{5}$$

where ϵ is the step size.

In this way, x can change in a systematic way, which prevents the random walk behavior [23]. Samples of the joint distribution are obtained by traversing along the equal-energy surface of the extended state space. To explore other regions of the target distribution, the momentum variables v should be resampled from an isotropic Gaussian.

Consequently, as Figure 1 shows, the entire HMC can be divided into two stages: the deterministic exploration of an energy level which is represented by the blue arrow, and

the random walk between energy levels which is represented by the green arrow. The contours represent different energy levels, where the energy level of the outer contour is higher [24,25]. For the volume-preserving flows, the exploration across energy levels is achieved by resampling a new momentum variable v' generated from the Gaussian distribution, and thus it can only explore regions of different energy levels in a random walk behavior which is inefficient [26,27]. Although the process of the leapfrog discretization can introduces numerical errors inevitably, the MH step can help HMC converge to the target distribution. Since HMC can well preserve the total energy, it has a high acceptance rate.

Schematic representation of deterministic sampling at a given energy level (blue arrow) and Gaussian redistribution of kinetic energy (green arrows).

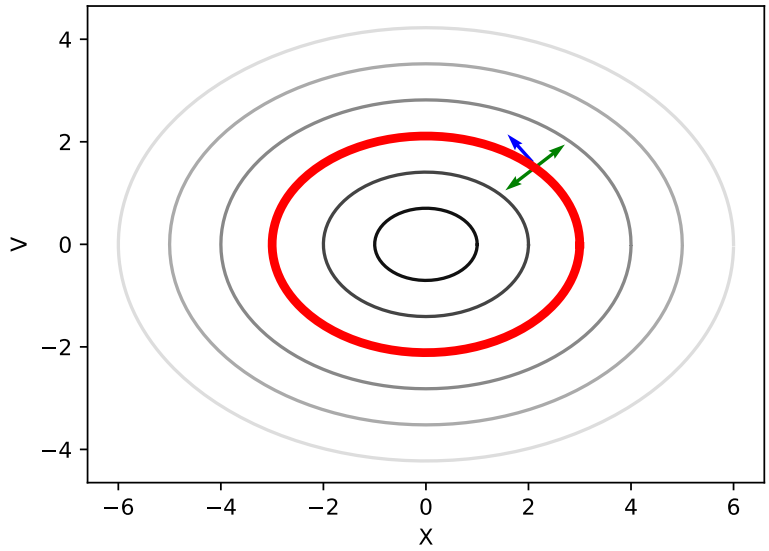


Figure 1. Traversing between energy levels. The blue arrow represents the deterministic exploration of an energy level. Green arrows represents the random walk between energy levels.

If we can change the total energy greatly during exploration and guarantee that the transition kernels can converge to the target distribution, the random walk behavior of the exploration between energy levels can be prevented [28]. This idea encouraged us to develop the non-volume-preserving sampler. In our algorithm, we utilize the non-volume-preserving flows as the transition kernels, which means that $\left|\frac{\partial q_\theta(\xi^\star|\xi)}{\partial \xi^\top}\right| \neq 1$. Thus, to avoid a huge computational burden, the Jacobian determinant of the transition kernels should be easy to compute. Next we will introduce the non-volume-preserving flow used in this paper, which has the desired property.

2.3. Real-Valued Non-Volume-Preserving Flows

The main idea of the real-valued non-volume-preserving (RNVP) flows [13,29–31] is to construct flexible and invertible architectures that enable the computation of log-likelihood on continuous data using the change of variable formula [30]. By stacking a sequence of carefully designed bijection functions which have tractable Jacobian determinant, flexible and expressible transition functions can be constructed. As illustrated in Figure 2, we use only four layers non-volumn-preserving flows to map a normal distribution to a complex distribution. This simple experiment proves the powerful expression ability of RNVP. Each bijection function of RNVP is called coupling layers with the form as:

$$\begin{aligned} y_{1:d} &= x_{1:d} \\ y_{d+1:D} &= x_{d+1:D} \odot \exp[S(x_{1:d})] + T(x_{1:d}), \end{aligned} \tag{6}$$

where the Jacobian of this function is:

$$\frac{\partial y}{\partial x^\top} = \begin{bmatrix} I_d & 0 \\ \frac{\partial y_{d+1:D}}{\partial x_{1:d}^\top} & \mathrm{diag}(\exp[S(x_{1:d})]) \end{bmatrix}, \tag{7}$$

which is triangular and where determinant can be computed simply by $\exp\left[\sum_j S(x_{1:d})\right]$. S rescales the inputs and T is a translation. Arbitrarily complex transition functions can be built by stacking numerical layers. Moreover, computing the inverse of the coupling layers does not require computing the inverse of S and T [13]:

$$\begin{aligned} x_{1:d} &= y_{1:d} \\ x_{d+1:D} &= [y_{d+1:D} - T(y_{1:d})] \odot \exp[-S(y_{1:d})], \end{aligned} \tag{8}$$

and its Jacobian determinant is also tractable.

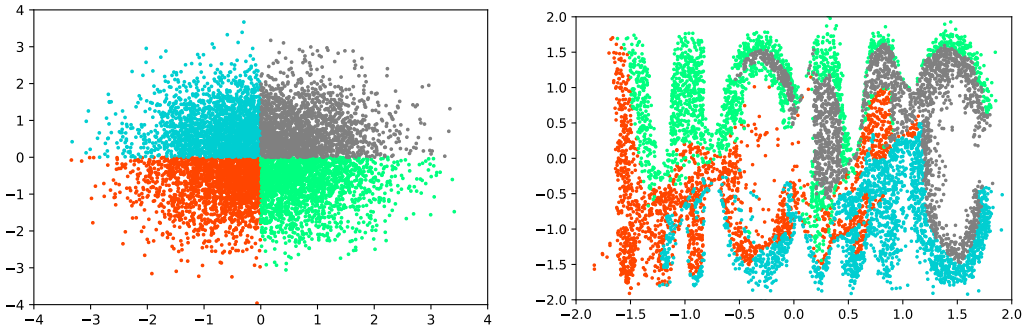


Figure 2. Scatter plots of normal distribution (**left**) and complex distribution of "MCMC" (**right**). We use 6 layers RNVP flows constructed by neural networks with each layer contains 512 hidden units and training with maximum likelihood estimation. The result is observed after 100,000 iterations with a learning rate of 0.0001. The batch size is set to 512.

Since the computation of the determinant requires $\exp\left[\sum_j S(x_{1:d})\right]$ (or $\exp\left[-\sum_j S(x_{1:d})\right]$ for inverse transition), we can exploit arbitrarily flexible functions such as deep neural networks as S and T. Therefore, we use RNVP to construct the transition kernels.

3. The Proposed Method

To overcome the problem of inefficient exploration of the volume-preserving flows, and inefficient utilization of gradient information of the target distribution, we propose NVP-MC, which exploits non-volume-preserving flows to construct transition kernels, and repeatedly utilize gradient information of the target distribution through adversarially training. In the following subsections, we will first describe the main idea of the proposed method, and then introduce how to sample from a target distribution p_d specified by an analytic expression:

$$p_d(x) = \frac{1}{Z}\exp[-U(x)], \tag{9}$$

where Z is an unknown normalization constant and $U(x)$ is the potential energy function in Hamiltonian dynamics.

3.1. Using Non-Volume-Preserving Flows as Generator

We construct three layers of the non-volume-preserving flow model as the generator. At each layer of the generator, the update of position and momentum variables are correlative.

We firstly update auxiliary momentum variables with the following transition function:

$$v' = v + \epsilon T(x), \tag{10}$$

where $T(x)$ is a translation item determined by x. We do not multiply a scale item to v like RNVP in Equation (6), because we find that it can bring much worse results, and is difficult to train. The Jacobian determinant of this transition function is $\mathbf{I}$. Next we update x through the transition function:

$$x' = x \odot \exp\left[\epsilon S(v')\right] + T(v') \odot \exp\left[\epsilon S(v')\right], \tag{11}$$

where $S(v')$ is the scale item and $T(v')$ is the translation item that all determined by v'. To simplify the calculation, we use the same item to rescale x and $T(v')$, which makes the training process more stable in practice. The determinant of this transition function is $\exp[\epsilon S(v')]$. We note that our transition kernels update x without dividing the x into two parts like other methods. When a part of x is updated through another part of x, the correlation between data will inevitably increase. Thus, we gave up this approach and only updated x based on v.

Finally, we use the same function of Equation (10) to update v again. The full forward update steps are shown below:

$$\begin{aligned} v' &= v + \frac{\epsilon}{2} T(x); \\ x' &= x \odot \exp\left[\epsilon S(v')\right] + T(v') \odot \exp\left[\epsilon S(v')\right]; \\ v'' &= v' + \frac{\epsilon}{2} T(x'). \end{aligned} \tag{12}$$

The (x', v'') represents the next state. Then we use MH algorithm to ensure p_d is the stationary distribution of the transition kernels. Assume that f_θ represents forward transition function and $\xi = (x, v)$ represents the previous state, the MH acceptance probability takes the form as:

$$A_\theta(\xi^\star, \xi) = \min(1, \frac{p(\xi^\star)}{p(\xi)} \left| \frac{\partial f_\theta(\xi^\star)}{\partial \xi^\top} \right|), \tag{13}$$

where $\xi^\star$ denotes the new state obtained from the transition kernels depending on ξ. Because the update steps of v and v' are volume preserving, we only need to compute Jacobian determinant of Equation (11) which is $\exp[\sum_i \epsilon S(v_i)]$, where v_i represents the i-th dimension of v. To obtain the symmetric transition behavior and satisfy the detailed balance, we need to build the inverse transition function f_θ^{-1} and choose froward transition and backward transition functions with the same probability. The inverse transition function f_θ^{-1} takes the form as:

$$\begin{aligned} v' &= v'' - \frac{\epsilon}{2} T(x'); \\ x &= \left(x' - T(v') \odot \exp\left[\epsilon S(v')\right]\right) \odot \exp\left[-\epsilon S(v')\right]; \\ v &= v' - \frac{\epsilon}{2} T(x). \end{aligned} \tag{14}$$

The Jacobian determinant of Equation (14) is $\exp[\sum_i -\epsilon S(v_i)]$ which is similar to Equation (12). The proposed method is similar to the past work of L2HMC and A-NICE-MC. Our approach incorporates adversarial training of A-NICE-MC to prevent the usage of gradient information and uses non-volume-preserving flows to construct flexible transition kernels.

3.2. Loss Function and Training Procedure

Now, we describe the training procedure of the proposed method. Firstly, we run a gradient-based exploration operator to explore the energy function of the target distribution and get some samples to initialize the buffer. In A-NICE-MC, HMC is chosen as the exploration operator. However, in our scheme, we do not need the exploration operator to sample exactly from the target distribution and only need to explore more regions of the target distribution.

One of the characteristic properties of high-dimensional spaces is that there is much more volume outside any given neighborhood than inside of it [9]. In other words, exploration towards the uniform distribution is inefficient in high-dimensional spaces [32]. In the D-dimensional spaces, the additional computational cost of evaluating a gradient compared with evaluating the function itself will typically be a fixed factor independent of D, whereas the D-dimensional gradient vector conveys D pieces of information compared with the one piece of information given by the function itself [23]. Therefore, we also use the gradient-based sampler as the exploration operator. Some studies have shown that the MCMC methods implemented without the detailed balance can achieve acceleration of convergence [33,34]. Thus we exploit HMC without MH steps as the exploration operator to get higher mixing performance.

At each epoch of training, we generate samples through the transition kernels and use the MH algorithm to compute acceptance probability. We treat all the samples from the transition kernels as the "fake samples", and treat the samples from the buffer as the "true samples". A discriminator will be trained to distinguish the "true samples" from the "fake samples". Our transition kernel will be trained to fool the discriminator. After getting new samples, we drop some samples of the buffer in a constant ratio and insert the new accepted samples, and thereby the quality of samples in the buffer can be improved. We train our parameters in the framework of GANs, and the training objective can be formulated as:

$$\begin{aligned} \min_K \max_D V(D,K) = \min_T \max_D \mathbb{E}_{x\sim B(x)}[D(x)] \\ - \mathbb{E}_{z\sim\mathcal{N}(0,1)}[D(K(z))] \end{aligned} \tag{15}$$

where K is our transition kernel, D is the discriminator network, and B is the buffer used to save the correct samples. To reduce the autocorrelation of samples obtained from the generator, we use the pairwise discriminator in [15]. We initialize B with HMC without the MH step. In fact, the samples of the uniform distribution can also be used to initialize the buffer. Because a part of samples will be dropped after every training epoch, and some new samples will be added into the buffer, the quality of samples in the buffer will improve stably during the training process. However, initial samples from the uniform distribution can not scale well in high-dimensional state spaces, it can only perform well in low-dimensional state spaces. We exactly describe our training procedure in Algorithm 1.

Unlike L2HMC, we do not exploit the gradient information of the target distribution to train the transition kernels. We have actually tried to do that, and find that the introduction of the gradient items increases the training time by tenfold but not get better results than the proposed method. Moreover, we find that L2HMC can not exploit the gradient information of the target distribution properly. We will discuss the phenomenon in detail in the next section.

Algorithm 1 Traing NVP-MC

Input: Energy function $U(x)$, batch size M, learning rate α, number of iterations N, empty buffer B, transition kernels K_θ and K_θ^{-1}.

Initialize B using HMC without the MH step. Initialize the parameters of the transition kernel K_θ and parameters of the discriminator D_ϕ.
for $i = 1 \to i = N$ **do**
  Sample a batch $\{(x, v)_{(i)}\}_{i \leqslant N}$ of Gaussian noise as the start points.
  for $i = 1 \to i = M$ **do**
    Randomly sample a number u in open interval $(0, 1)$.
    Choose transition kernel: $K_\theta(x, v) = \begin{cases} K_\theta(x, v), & 0 < u < 0.5 \\ K_\theta^{-1}(x, v), & 1 > u > 0.5. \end{cases}$
    Generate the new sample $\{(x', v'')_{(i)}\}$ through K_θ.
    Accept the new sample with probability computed by Equation (4),
    and replace the samples in B with the accepted samples.
  end for
  Sample a batch $\{(x'')_{(i)}\}_{i \leqslant N}$ from B as the correct samples.
  Update the discriminator:
  $\phi \leftarrow \phi - \alpha \nabla_\phi \frac{1}{M} \sum_{i=1}^{M} \left[\log D_\phi \left(x''_{(i)} \right) + \log \left(1 - D_\phi \left(K_\theta \left(x_{(i)} \right) \right) \right) \right]$.
  Update the transition kernel:
  $\theta \leftarrow \theta - \alpha \nabla_\theta \frac{1}{M} \sum_{i=1}^{M} \log \left(1 - D_\phi \left(K_\theta \left(x_{(i)} \right) \right) \right)$.
end for

4. Related Work

Since the choice of the proposal distribution determines the effect of the MH algorithm, many works have focused on this research. Recently, some works [13,15] exploit flexible deep neural networks or flow models [17,30] to build the proposal distribution that can mix fast for the target distribution. These algorithms outperform the vanilla Hamiltonian Monte Carlo (HMC) [35], and are the state-of-the-art methods.

4.1. Getting MCMC Transition Kernels through Adversarially Training

A-NICE-MC [15] aims to obtain parameterized MCMC transition kernels as the proposal distribution of MH algorithm through adversarially training. The training principle is similar to Wasserstein GANs [18]. They use a novel pairwise discriminator to reduce autocorrelation of samples by scoring two samples at a time. For every sample pair, "real data" (x_{r_1}, x_{r_2}) are drawn from bootstrapped samples. Assume $K_\theta(x|x')$ represents the transition kernel, the "fake data" (x_{f_1}, x_{f_2}) are generated by $x_{f_1} \sim K_\theta(x|x_{f_2})$, where x_{f_2} is either drawn from the data distribution or generated from the transition kernel with initial noise samples. Compared with the normal discriminator, the pairwise discriminator is more sensitive among the correlation of samples. For the generation process of "fake data", the introduction of true data makes the training procedure more stable. Inspired by HMC, A-NICE-MC also introduces auxiliary variables into the transition kernels, but without gradient information of the target distribution. They leverage volume-preserving flows as the transition kernels, which have tractable Jacobian determinants and are the same as vanilla HMC. The inputs into the transition kernels of A-NICE-MC will first be partitioned into two parts x_1 and x_2, then the transition kernels take the form as:

$$\begin{aligned} x'_1 &= x_1 \\ x'_2 &= x_2 + m(x_1), \end{aligned} \tag{16}$$

where m is a neural network. Intuitively, the expression ability of the transfer kernels is not sufficient. However, the transition function can be stacked multiple layers to get an expressive model [17]. In HMC, the introduction of auxiliary variables can prevent the random walk behavior. Auxiliary variables in A-NICE-MC have a similar purpose and

add randomness to the generator. In practice, they obtain exact initial samples by running HMC and use a bootstrap process [36] to generate samples.

However, A-NICE-MC suffers from the same difficulties in mixing across energy levels as HMC [13], especially when facing multi-modal distributions. For gradient-based MCMC methods, effective exploration is achieved by exploiting the differential structure of the energy function. In addition, when a region is explored twice, the gradient is also calculated twice, which is inefficient. In other words, they only exploring, but not making full use of the information of the area that has been explored.

Using gradient-based MCMC methods to get some initial samples, we can repeatedly utilize gradient information of the target distribution through adversarially training, which will bring considerable performance improvement. Therefore, we develop a new method for training flexible MCMC kernels, which not only has the powerful exploration ability of non-volume-preserving flows, but also can effectively utilize gradient information of the target distribution in an adversarially training form.

4.2. Parameterized Non-Volume-Preserving Transition Kernels

Vanilla HMC can be seen as an invertible volume-preserving flow, which has difficulties in mixing across energy levels [13]. Inspired by RNVP, L2HMC introduces neural networks as scale and translation items into leapfrog integrator to construct the parameterized non-volume-preserving transition kernels, which can explore the target distribution efficiently. The training objective of L2HMC aims to maximize the acceptance rate and expected squared jumped distance [20], and the loss function takes the form as:

$$\ell_\lambda[\xi, \xi^\star, A(\xi^\star|\xi)] = \frac{\lambda^2}{\delta(\xi, \xi^\star)A(\xi^\star|\xi)} - \frac{\delta(\xi, \xi^\star)A(\xi^\star|\xi)}{\lambda^2}, \tag{17}$$

where $\xi = (x, v)$ and $\xi^\star = (x', v')$ represent the last state and the new state, respectively. λ is a scale parameter, and $\delta(\xi, \xi^\star)$ is the expected squared jumped distance between these two states. $A(\xi^\star|\xi)$ denotes the acceptance probability for the new state $\xi^\star$. Intuitively, the training objective encourages the parameterized transition kernels to get low autocorrelation of samples and meanwhile keep a high acceptance rate. Maximizing expected squared jumped distance is equivalent to minimizing the lag-one autocorrelation [20].

In L2HMC, scale and transition items are all controlled by neural networks. The parameterized transition kernels take the form:

$$\begin{aligned}
v' &= v \odot \exp\left[\frac{\epsilon}{2} S_v(\zeta_1)\right] - \frac{\epsilon}{2}(\partial_x U(x) \odot \exp[\epsilon Q_v(\zeta_1)] + T_v(\zeta_1)) \\
x' &= x_{\bar{m}^t} + m^t \odot \left(x \odot \exp[\epsilon S_x(\zeta_2)] + \epsilon\left(v' \odot \exp[\epsilon Q_x(\zeta_2)] + T_x(\zeta_2)\right)\right) \\
x'' &= x'_{m^t} + \bar{m}^t \odot \left[x' \odot \exp[\epsilon S_x(\zeta_3)] + \epsilon\left(v' \odot \exp[\epsilon Q_x(\zeta_3)] + T_x(\zeta_3)\right)\right] \\
v'' &= v' \odot \exp\left[\frac{\epsilon}{2} S_v(\zeta_4)\right] - \frac{\epsilon}{2}\left(\partial_x U(x'') \odot \exp[\epsilon Q_v(\zeta_4)] + T_v(\zeta_4)\right).
\end{aligned} \tag{18}$$

Here, $\zeta = (x, \partial_x U(x), t)$ represents the full state and excludes v. The three newly introduced functions T, Q and S are all neural networks. Scale items $\exp(S(\zeta))$ and $\exp(Q(\zeta))$ utilize gradient information of the target distribution as input, which will take up much computational budget. m^t is a fixed random binary mask that determines which variables are updated. The update scheme first updates a subset of the coordinates of x [13], and then updates the rest subset to make transition kernels more expressive.

However, L2HMC has poor performance in far-distance multi-modal distributions. When using a small batch size, it will be hard to converge to the target distribution or traverse between correct modes in our empirical experiments, which indicates that L2HMC can not effectively and correctly use the gradient information of the target distribution to help build high-performance transition kernels.

Since the large step size will introduce more errors when traversing along iso-probability contours, there is a trade-off between a high acceptance rate and high diversity of samples.

For the training objective (Equation (17)) of L2HMC, the two aspects are all considered, but they do not have a relationship of adversarial form here. Moreover, the order of magnitude of $\delta(\xi, \xi^\star)$ and $A(\xi^\star|\xi)$ seems to be different. $A(\xi^\star|\xi)$ represents the acceptance rate which is no more than 1, while $\delta(\xi, \xi^\star)$ represents the Euclidean distance between samples that can be much greater than 1. When we optimizer the parameters through gradient descent with the objective of larger $\delta(\xi, \xi^\star)A(\xi^\star|\xi)$, the influence of the two items on the gradient are obviously different. In other words, the trained sampler may have a high acceptance rate but high autocorrelation or vice versa under the loss function, which will bring bad empirical results. We will discuss the situation in detail in Section 5.

5. Experiment

In this section, we evaluate the performance of NVP-MC. We compare our method with A-NICE-MC(NICE), L2HMC and vanilla RNVP on four challenging distributions using the effective sample size (ESS) [35]. For vanilla RNVP, we firstly sample some correct sample points of the target distribution, then directly train the RNVP flows to capture the target distribution. Since the method is firstly inspired by Normalizing Flow(NF) [37], we use NF to represent the baseline method in our experiments. Then we compare with the three methods by evaluating maximum mean discrepancy (MMD) [38]. Similar to experiments in [13], we build a challenging mixture of Gaussians to show the shortcomings of parameterized gradient-based transition kernels. We demonstrate the sample density plots generated by the four different methods. Finally, we conduct experiments on nine real datasets using Bayesian logistic regression [39] to showcase the practicability of the proposed method.

5.1. Performance Indexes

ESS can reflect the magnitude of the autocorrelation or the number of "effective samples" of the samples, which is defined as:

$$\text{ESS} = N / \left[1 + 2 \times \sum_{s=1}^{\infty} \rho(s) \right], \tag{19}$$

where N represents the total sample number. We use the same computation method as [15] to estimate ρ_s:

$$\hat{\rho}_s = \frac{1}{\hat{\sigma}^2 (N - s)} \sum_{n=s+1}^{N} (x_n - \hat{\mu})(x_{n-s} - \hat{\mu}), \tag{20}$$

where $\hat{\mu}$ and $\hat{\sigma}$ are the empirical mean and variance obtained from the independent sampler. Similar to [40], when the autocorrelation goes below 0.05 we truncate the sum to reduce the noise of large lags.

MMD can measure the difference between samples drawn from two distributions X and Y, which takes the form:

$$\begin{aligned} MMD^2[X, Y] = & \frac{1}{M^2} \sum_{i,j=1}^{M} \kappa(x_i, x_j) - \frac{2}{MN} \sum_{i,j=1}^{M,N} \kappa(x_i, y_j) \\ & + \frac{1}{N^2} \sum_{i,j=1}^{N} \kappa(y_i, y_j), \end{aligned} \tag{21}$$

where M represents the sample number in X while N represents the sample number in Y, and $\kappa(\cdot, \cdot)$ is the kernel function which takes the form:

$$\kappa(x, y) = \left(1.0 + x^\top y\right)^2. \tag{22}$$

Through evaluating *MMD*, we can justify the convergence of the proposed method. The lower the *MMD* value is, the more rapid convergence the sampler has. We run the

same computation procedure of *MMD* for 10 times and record the mean and variance as our final results.

5.2. Varieties of Challenging Energy Functions

We present an empirical evaluation for our trained sampler on varieties of synthetic 2D energy functions. For all energy functions, the largest ESS for all dimensions is shown in Table 1 and the MMD value is demonstrated in Figure 3. Then we further utilize the MoG experiment, to show the powerful exploration efficiency of the proposed method, compared with L2HMC.

For synthetic 2D target distributions we choose: Mixture of six Gaussians (MoG6): The analytic form of $p(x)$ for MoG6 is:

$$p(x) = \frac{1}{6}\sum_{i=1}^{6}\mathcal{N}(x|\mu_i, \sigma_i),$$

where $\mu_i = \left[\sin\frac{i\pi}{3}, \cos\frac{i\pi}{3}\right]$ and $\sigma_i = [0.5, 0.5]$.

Gaussian funnel (GF): The energy function of the 2D funnel is

$$U(x) = \frac{1}{2}\left[\left(\frac{x_1}{\sigma}\right)^2 + \frac{x_2^2}{\exp(x_1)} + \ln[2\pi\cdot\exp(x_1)]\right]$$

and we set $\sigma = 1.0$.

Strongly correlated Gaussian (SCG): We rotate a diagonal Gaussian with variance $[10^2, 10^{-2}]$ by $\frac{\pi}{4}$, which takes the form:

$$p(x) = \mathcal{N}\left(0, B\Sigma B^T\right), \quad \Sigma = \begin{bmatrix} 10^{-2} & 0 \\ 0 & 10^2 \end{bmatrix}, \quad B = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}.$$

The case is an extreme version of the example in [35].

Mixture of five rings (Ring5): The analytic form of the energy function of the mixture of 5 ring shaped target distributions is:

$$U(x) = \min(u_1, u_2, u_3, u_4, u_5),$$

where $u_i = (\sqrt{x_1^2 + x_2^2} - i)^2/0.04$.

Mixture of Gaussians (MoG2): This is a mixture of two isotropic Gaussians separated by the distance of $10\sqrt{2}$:

$$p(x) = 0.5 * \mathcal{N}(x|\mu_1, \sigma_1) + 0.5 * \mathcal{N}(x|\mu_2, \sigma_2),$$

where $\mu_1 = [5, 5]$, $\mu_2 = [-5, -5]$ and $\sigma_1^2 = 3, \sigma_2^2 = 0.05$. The MoG2 distribution is more challenging than the example used to show that L2HMC has better mixing performance than A-NICE-MC in [13].

Ill-Conditioned Gaussian (ICG): This is a Gaussian distribution with diagonal covariance spaced log-linearly between 0.01 and 100, which has the same analytic expression with SCG.

We use the same hyperparameters for all density-based experiments. Specifically, we construct the transition kernels with three coupling layers to ensure that both x and v could get fully updated. We get initial samples through HMC without MH steps with 6 leapfrog steps and step size $\epsilon = 0.3$, and discard the first 1000 steps as burn-in steps. The remain samples are saved in a buffer, after each training epoch of transition kernels, the samples in the buffer will be discard by 0.5 and add the new accepted samples into the buffer. At each training epoch, the transition kernels will move 5000 steps. In each coupling layer, the S and T are 3 layers neural networks. The discriminator is also 3 layer neural networks with

400 units and activated with leaky rectified linear units. We train our model by Adam [41] optimizer with batch size of 32 and $beta_1 = 0.5$, $beta_2 = 0.9$ for D_{loss} and G_{loss}. The learning rate is set to be 0.0005 for the D_{loss}, and 0.0003 for the G_{loss}. To make the training process more stable, we clip the gradient of all the neural networks by -8 and 8.

As shown in Figure 3, A-NICE-MC and NVP-MC have lower MMD among the two distributions. When we use less than 4000 samples to compute MMD, the MMDs variance of NVP-MC is larger than A-NICE-MC. Its because that NVP-MC explores the target distributions in a more extensive way, which brings more noise, however, the exploration can bring NVP-MC samples with higher quality. When we use more than 4000 samples to compute MMD, the variance of MMD of NVP-MC is almost the same as A-NICE-MC, while the two methods all achieve the lowest MMD compared with NF and L2HMC.

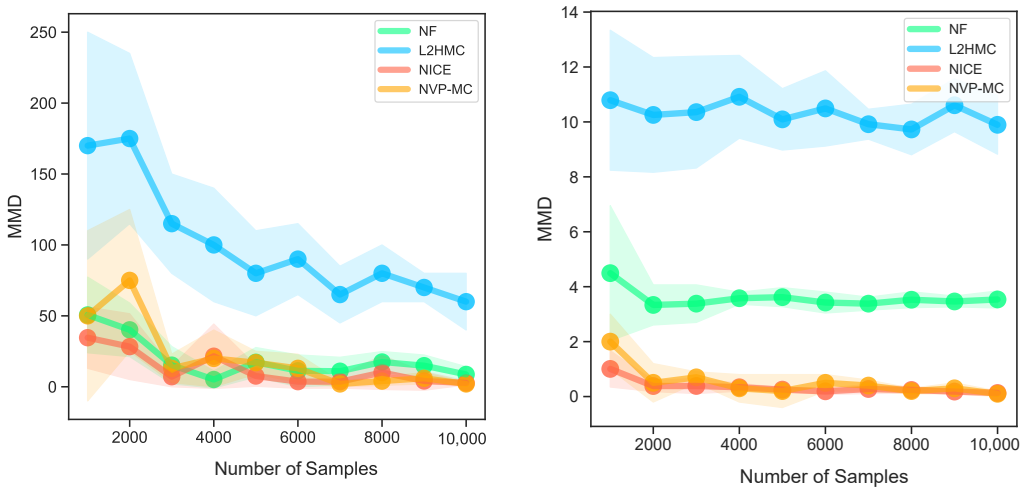


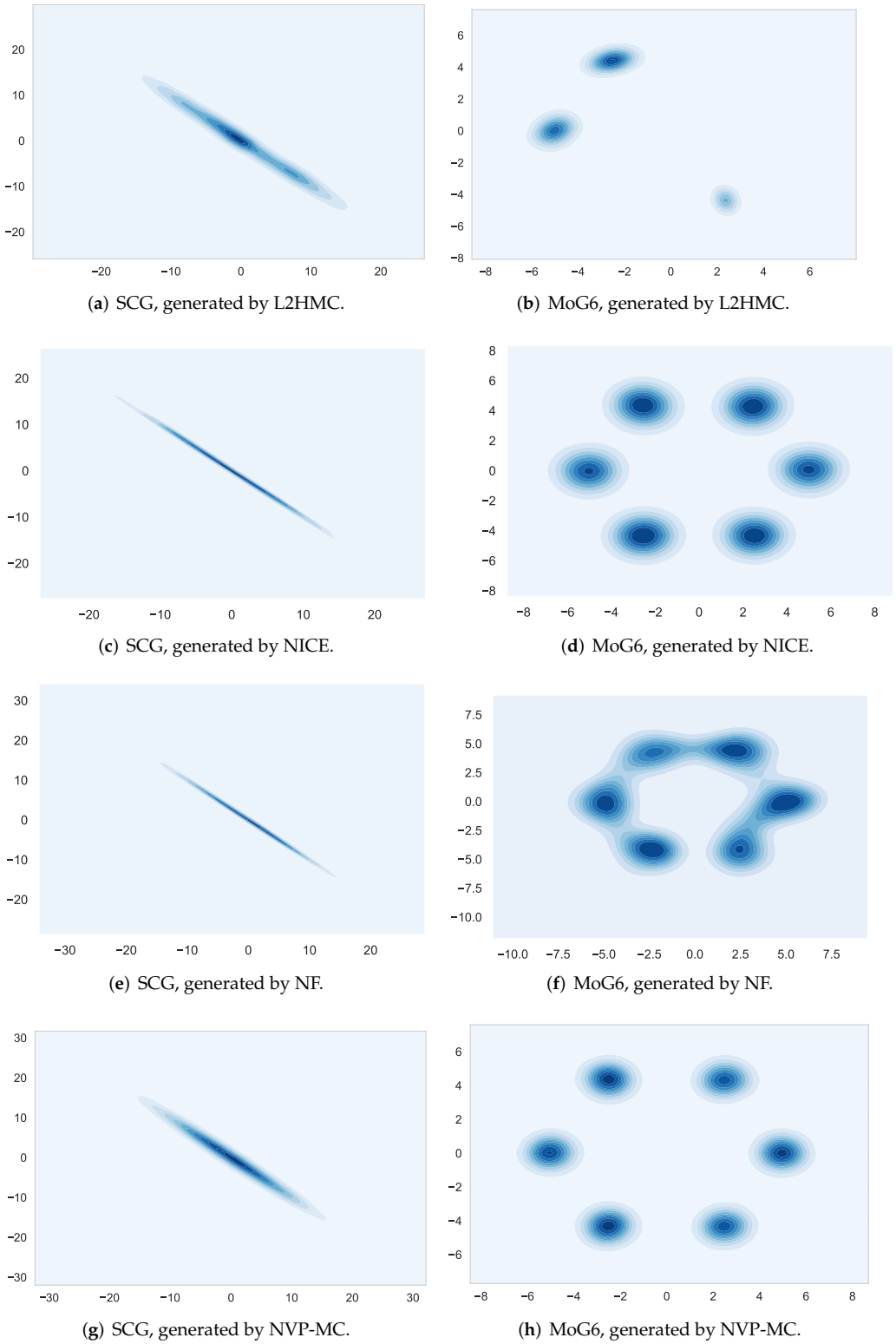
Figure 3. The performance of NF, L2HMC, NICE, NVP-MC on SCG (**left**) and MoG6 (**right**) distributions. In limited training steps, NF and L2HMC can not sample correctly from the target distributions compared with A-NICE-MC and NVP-MC.

We further demonstrate the sample density plots generated by the four different methods. As Figure 4 shows, L2HMC and NF cannot find the target distributions of SCG and MoG6, while NVP-MC still performs well, for NVP-MC exploits HMC without MH steps as the exploration operator to get higher mixing performance. Although NICE is able to find the target distributions of SCG and MoG6, it has a large error (lower ESS) while NVP-MC is able to sample from the target distribution precisely.

To estimate the quality of samples, we train NVP-MC, A-NICE-MC and L2HMC for 10,000 iterations with the same batch size and record the minimum value of ESS for all dimensions. Because we choose the same model size as A-NICE-MC (neural networks with 3 hidden layers with 400 (1024 for the 50-d ICG), the training time is similar. Therefore, there is no need for us to construct experiments to evaluate ESS per second (ESS/S) which is mentioned in [15]. L2HMC consumes far more time than A-NICE-MC and NVP-MC but gets the ESS value that does not match the effort. As Table 1 illustrates, NVP-MC can get significant improvement in ESS in all distributions compared with A-NICE-MC.

Table 1. The ESS of NVP-MC, A-NICE-MC, L2HMC, and HMC. Data in bold represent best results.

Target	NVP-MC	A-NICE-MC	L2HMC	HMC
MoG6	**568.2**	320.0	311.2	1.0
GF	**834.3**	270.0	304.1	8.0
SCG	**1000.0**	539.4	497.0	0.48
Ring5	**200.3**	155.6	69.1	0.43
50-d ICG	**0.83**	0.29	0.78	0.02



(**a**) SCG, generated by L2HMC. (**b**) MoG6, generated by L2HMC.

(**c**) SCG, generated by NICE. (**d**) MoG6, generated by NICE.

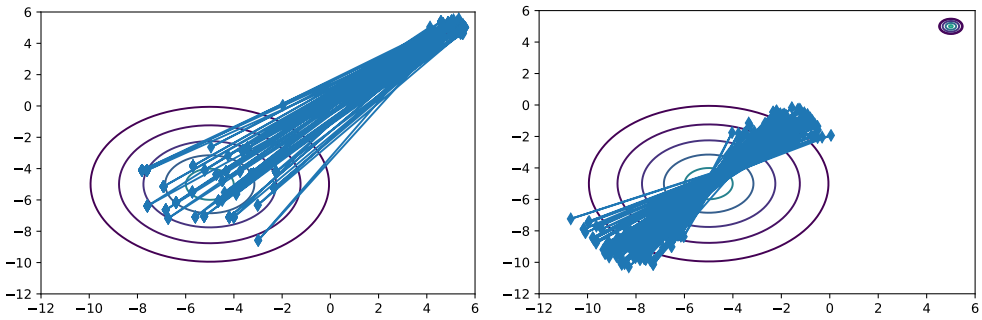
(**e**) SCG, generated by NF. (**f**) MoG6, generated by NF.

(**g**) SCG, generated by NVP-MC. (**h**) MoG6, generated by NVP-MC.

Figure 4. Density plots of samples from L2HMC, A-NICE-MC, NF and NVP-MC.

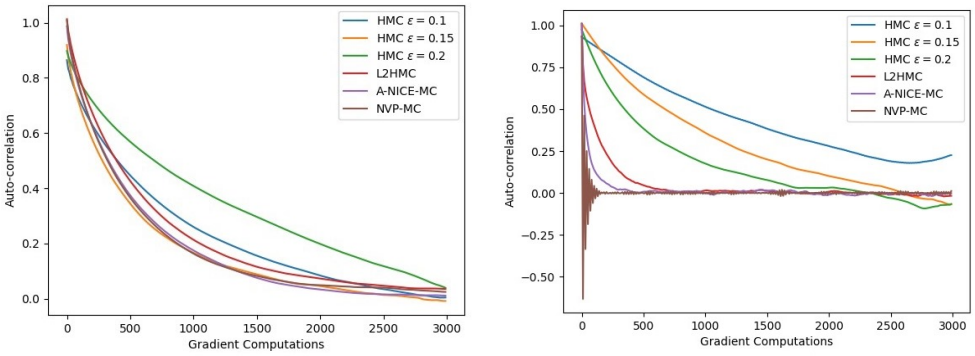
To further validate the performance of the proposed method in far-distance multi-modal distributions, we build on the MoG experiment presented in [13]. We increase the distance between modes by $\sqrt{2}$ and use the same variance. We run L2HMC and NVP-MC 10,000 iterations respectively, and use the same batch size for the two algorithms. As shown in Figure 5, although L2HMC introduces the gradient information of the target distribution, it can not correctly sample from the target distribution while NVP-MC can mix quickly

between the modes, which indicates that it may be inappropriate to introduce gradient information directly into parameterized transition kernels. Gradient information is informative in the exploration stage, but not the training stage. By dividing the training process into the two stages, we can remove the restriction of exact sampling in the exploration stage and the proposed method can have powerful exploration ability. In the training stage, adversarially training can learn the geometry of the target distribution by minimizing the loss function. The exact samples can be obtained through accepting the samples generated from the transition kernels by using MH steps. Moreover, L2HMC needs twice times than NVP-MC to train the transition kernels in the training stage, and a trained NVP-MC can sample from the target distribution six times faster than HMC. We also report the autocorrelation of NVP-MC and the compared methods. As seen in Figure 6, the samples collected by NVP-MC have the fastest drop in autocorrelation with respect to gradient evaluations in target distribution of 2d-SCG. In high-dimensional target distribution of 50d-ICG, the rate of autocorrelation decline of NVP-MC is similar to that of NICE-MC and slightly faster than that of L2HMC.



(**a**) NVP-MC (**b**) L2HMC

Figure 5. NVP-MC (**a**) can correctly and faster mix between modes in limited training steps and batch sizes compared with L2HMC (**b**).



(**a**) 50d-ICG (**b**) 2d-SCG

Figure 6. Autocorrelation with respect to gradient evaluation steps for 50-d ICG and 2d-SCG.

5.3. Bayesian Logistic Regression

In this section, we conduct the experiments on nine real datasets using Bayesian logistic regression. To show the practicability of the proposed method, we sample from the posterior distribution and compare NVP-MC with HMC, logistic regression (LR) [42] and variational Bayesian logistic regression (VBLR), which are all the widely applicable methods in Bayesian logistic regression [43]. Given a conditional distribution $p(Y|X)$ parameterized by the logistic distribution, the goal of LR is to maximize the likelihood function and get the optimized parameters to predict the class of the data. We consider nine datasets from UCI repository [44]: Pima (Pi), Haberman (HA), Blood (BL), Immunotherapy (IM), Indian

(IN), Mammographic (MA), Heart(HE), German (GE) and Australian (AU) and evaluate the accuracy rate and area under the receiver operating characteristic curve (AUC) [45]. To improve the stability of the models, we normalize all datasets to have zero mean and unit variance. We set the normal distribution $\mathcal{N}(0, I)$ as the prior distribution of parameters, and use the same data partition for all experiments.

We set the dimension of the auxiliary variable v to 35 for every real dataset experiment. At each training epoch, the transition kernels will move 5000 steps. In each coupling layer, the S and T are 3 layers neural networks with 400 units for input and output layer, 800 units for the hidden layer, and each layer is not activated. The discriminator is 3 layer neural networks with 800 units and activated with leaky rectified linear units. As for the optimizer, we use the same setting as in the synthetic 2D target distribution experiments.

As Tables 2 and 3 illustrated, NVP-MC can obtain better performance in almost all datasets, which indicates that the proposed method can sample from the posterior distribution more accurately.

Table 2. Classification accuracy for LR, VBLR, HMC and NVP-MC. Data in bold represent best results.

Dataset	LR	VBLR	HMC	NVP-MC
HA	69.3 ± 0.2	69.3 ± 0.1	69.3 ± 0.2	$\mathbf{69.4 \pm 0.1}$
PI	76.6 ± 0.2	76.2 ± 0.1	76.6 ± 0.1	$\mathbf{76.8 \pm 0.1}$
MA	82.5 ± 0.3	$\mathbf{83.1 \pm 0.1}$	$\mathbf{83.1 \pm 0.1}$	$\mathbf{83.1 \pm 0.1}$
BL	76.0 ± 0.2	76.0 ± 0.2	76.0 ± 0.3	$\mathbf{76.1 \pm 0.2}$
IM	77.7 ± 0.3	77.8 ± 0.4	83.2 ± 0.2	$\mathbf{83.3 \pm 0.4}$
IN	$\mathbf{75.8 \pm 0.3}$	73.2 ± 0.2	73.2 ± 0.2	73.8 ± 0.2
HE	75.9 ± 0.2	75.9 ± 0.2	75.9 ± 0.2	$\mathbf{76.1 \pm 0.2}$
GE	71.5 ± 0.1	71.5 ± 0.1	72.5 ± 0.2	$\mathbf{73.2 \pm 0.1}$
AU	86.9 ± 0.2	87.6 ± 0.2	87.6 ± 0.2	$\mathbf{87.7 \pm 0.2}$

Table 3. AUC for LR, VBLR, HMC and NVP-MC. Data in bold represent best results.

Dataset	LR	VBLR	HMC	NVP-MC
HA	62.7 ± 0.1	63.2 ± 0.1	63.0 ± 0.2	$\mathbf{63.3 \pm 0.1}$
PI	79.2 ± 0.2	79.3 ± 0.1	79.3 ± 0.1	$\mathbf{79.4 \pm 0.1}$
MA	$\mathbf{89.9 \pm 0.1}$	89.8 ± 0.1	$\mathbf{89.9 \pm 0.1}$	$\mathbf{89.9 \pm 0.1}$
BL	73.5 ± 0.3	73.4 ± 0.3	$\mathbf{74.4 \pm 0.3}$	73.6 ± 0.2
IM	76.7 ± 0.3	78.5 ± 0.5	89.2 ± 0.2	$\mathbf{89.3 \pm 0.4}$
IN	73.2 ± 0.3	$\mathbf{73.2 \pm 0.2}$	72.4 ± 0.2	72.7 ± 0.2
HE	80.1 ± 0.2	81.3 ± 0.2	$\mathbf{82.2 \pm 0.3}$	81.9 ± 0.2
GE	74.7 ± 0.2	75.5 ± 0.2	76.7 ± 0.3	$\mathbf{76.7 \pm 0.1}$
AU	92.5 ± 0.2	$\mathbf{93.9 \pm 0.2}$	93.9 ± 0.3	$\mathbf{93.9 \pm 0.2}$

6. Conclusions

In this study, we develop the adversarially training MC, non-volume-preserving transition kernels, and exploit a novel scheme to train MCMC kernels with promising mixing performance. Compared with existing gradient-based sample methods, the proposed method can leverage gradient information more efficiently, and explore the target distribution faster. The experiments in various challenging distributions and real datasets show that the proposed method outperforms other state-of-the-art MCMC methods and is promising for practical uses.

Author Contributions: Conceptualization, S.L. and S.S.; methodology, S.L.; experiments, S.L.; formal analysis, S.L. and S.S.; writin—original draft preparation, S.L.; writing—review and editing, S.L. and S.S.; visualization, S.L.; supervision, S.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the NSFC Projects 62076096 and 62006078, the Shanghai Municipal Project 20511100900, Shanghai Knowledge Service Platform Project ZF1213, the Shanghai Chenguang Program under Grant 19CG25, the Open Research Fund of KLATASDS-MOE and the Fundamental Research Funds for the Central Universities.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available in a publicly accessible repository. The data presented in this study are openly available in UCI Machine Learning Repository at http://archive.ics.uci.edu/ml/index.php (accessed on 13 March 2022).

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

References

1. Robert, C.; Casella, G. *Monte Carlo Statistical Methods*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013.
2. Neal, R.M. *Probabilistic Inference Using Markov Chain Monte Carlo Methods*; Technical Report; Department of Computer Science, University of Toronto: New Brunswick, MA, Canada, 1993.
3. Martino, L.; Read, J. On the flexibility of the design of multiple try Metropolis schemes. *Comput. Stat.* **2013**, *28*, 2797–2823. [CrossRef]
4. Hastings, W.K. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* **1970**, *57*, 97–109. [CrossRef]
5. Wang, Z.; Mohamed, S.; Freitas, N. Adaptive Hamiltonian and riemann manifold Monte Carlo. In Proceedings of the 30th International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013.
6. Wang, J.; Sun, S. Decomposed slice sampling for factorized distributions. *Pattern Recognit.* **2020**, *97*, 107021. [CrossRef]
7. Duane, S.; Kennedy, A.D.; Pendleton, B.J.; Roweth, D. Hybrid Monte Carlo. *Phys. Lett. B* **1987**, *195*, 216–222. [CrossRef]
8. Simsekli, U.; Yildiz, C.; Nguyen, T.H.; Richard, G.; Cemgil, A.T. Asynchronous Stochastic Quasi-Newton MCMC for Non-Convex Optimization. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018.
9. Betancourt, M. A conceptual introduction to Hamiltonian Monte Carlo. *arXiv* **2017**, arXiv:1701.02434.
10. Psutka, J.V.; Psutka, J. Sample size for maximum-likelihood estimates of Gaussian model depending on dimensionality of pattern space. *Pattern Recognit.* **2019**, *91*, 25–33. [CrossRef]
11. Betancourt, M.; Byrne, S.; Girolami, M. Optimizing the integrator step size for Hamiltonian Monte Carlo. *arXiv* **2014**, arXiv:1411.6669.
12. Zou, D.; Xu, P.; Gu, Q. Stochastic Variance-Reduced Hamilton Monte Carlo Methods. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018.
13. Levy, D.; Hoffman, M.D.; Sohl-Dickstein, J. Generalizing Hamiltonian Monte Carlo with Neural Networks. In Proceedings of the 6th International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
14. Liu, C.; Zhuo, J.; Zhu, J. Understanding MCMC Dynamics as Flows on the Wasserstein Space. *arXiv* **2019**, arXiv:1902.00282.
15. Song, J.; Zhao, S.; Ermon, S. A-NICE-MC: Adversarial training for MCMC. In Proceedings of the 31st Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
16. Azadi, S.; Olsson, C.; Darrell, T.; Goodfellow, I.; Odena, A. Discriminator rejection sampling. *arXiv* **2018**, arXiv:1810.06758.
17. Dinh, L.; Krueger, D.; Bengio, Y. Nice: Non-linear independent components estimation. *arXiv* **2014**, arXiv:1410.8516.
18. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein GAN. *arXiv* **2017**, arXiv:1701.07875.
19. Wei, G.; Luo, M.; Liu, H.; Zhang, D.; Zheng, Q. Progressive generative adversarial networks with reliable sample identification. *Pattern Recognit. Lett.* **2020**, *130*, 91–98. [CrossRef]
20. Pasarica, C.; Gelman, A. Adaptively scaling the Metropolis algorithm using expected squared jumped distance. *Stat. Sin.* **2010**, *20*, 343–364.
21. Yang, J.; Roberts, G.O.; Rosenthal, J.S. Optimal scaling of Metropolis algorithms on general target distributions. *arXiv* **2019**, arXiv:1904.12157.
22. Green, P.J. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika* **1995**, *82*, 711–732. [CrossRef]
23. Bishop, C.M. *Pattern Recognition and Machine Learning*; Springer: New York City, NY, USA, 2006.
24. Cong, Y.; Chen, B.; Liu, H.; Zhou, M. Deep Latent Dirichlet Allocation with Topic-Layer-Adaptive Stochastic Gradient Riemannian MCMC. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017.
25. Betancourt, M.; Byrne, S.; Livingstone, S.; Girolami, M. The geometric foundations of Hamiltonian Monte Carlo. *Bernoulli* **2017**, *23*, 2257–2298. [CrossRef]
26. Tripuraneni, N.; Rowland, M.; Ghahramani, Z.; Turner, R. Magnetic Hamiltonian Monte Carlo. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017.
27. Huang, L.; Wang, L. Accelerated Monte Carlo simulations with restricted Boltzmann machines. *Phys. Rev.* **2017**, *95*, 035105. [CrossRef]

28. Li, C.; Chen, C.; Carlson, D.; Carin, L. Preconditioned Stochastic Gradient Langevin Dynamics for deep neural networks. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 Feburary 2016
29. Kingma, D.P.; Salimans, T.; Jozefowicz, R.; Chen, X.; Sutskever, I.; Welling, M. Improved variational inference with inverse autoregressive flow. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016.
30. Dinh, L.; Sohl-Dickstein, J.; Bengio, S. Density estimation using Real NVP. *arXiv* **2016**, arXiv:1605.08803.
31. Ma, F.; Ayaz, U.; Karaman, S. Invertibility of convolutional generative networks from partial measurements. In Proceedings of the Advances in Neural Information Processing Systems, Montréal, QC, Canada, 3–8 December 2018.
32. Dinh, V.; Bilge, A.; Zhang, C.; Matsen, F.A., IV. Probabilistic path Hamiltonian Monte Carlo. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017.
33. Zhang, Y.; Ghahramani, Z.; Storkey, A.J.; Sutton, C.A. Continuous relaxations for discrete Hamilton Monte Carlo. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012.
34. Ichiki, A.; Ohzeki, M. Violation of detailed balance accelerates relaxation. *arXiv* **2013**, arXiv:1306.6131.
35. Neal, R.M. MCMC using Hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*; Chapman & Hall/CRC: Boca Raton, FL, USA, 2011.
36. Efron, B.; Tibshirani, R.J. *An Introduction to the Bootstrap*; CRC Press: Boca Raton, FL, USA, 1994.
37. Rezende, D.J.; Mohamed, S. Variational Inference with Normalizing Flows. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015.
38. Borgwardt, K.M.; Gretton, A.; Rasch, M.J.; Kriegel, H.; Scholkopf, B.; Smola, A.J. Integrating structured biological data by Kernel Maximum Mean Discrepancy. *IBM J. Res. Dev.* **2006**, *22*, 49–57. [CrossRef]
39. MacKay, D.J.C. The Evidence Framework Applied to Classification Networks. *Neural Comput.* **1992**, *4*, 720–736. [CrossRef]
40. 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.
41. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
42. Freedman, D.A. *Statistical Models: Theory and Practice*; Cambridge University Press: Berkeley, CA, USA, 2009.
43. Tóth, J.; Tomán, H.; Hajdu, A. Efficient sampling-based energy function evaluation for ensemble optimization using simulated annealing. *Pattern Recognit.* **2020**, *107*, 107510. [CrossRef]
44. Dua, D.; Graff, C. UCI Machine Learning Repository. 2017. Available online: https://archive.ics.uci.edu/ml/index.php (accessed on 13 March 2022).
45. 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. [CrossRef]