IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 28, 2019, accepted February 4, 2020, date of publication February 10, 2020, date of current version February 20, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2972611

Neural Langevin Dynamical Sampling

MINGHAO GU[1] **AND SHILIANG SUN**[1,2]
[1]School of Computer Science and Technology, East China Normal University, Shanghai 200241, China
[2]Shanghai Institute of Intelligent Science and Technology, Tongji University, Shanghai 201804, China

Corresponding author: Shiliang Sun (slsun@cs.ecnu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61673179, and in part by the Shanghai Knowledge Service Platform Project under Grant ZF1213.

ABSTRACT Sampling technique is one of the asymptotically unbiased estimation approaches for inference in Bayesian probabilistic models. Markov chain Monte Carlo (MCMC) is a kind of sampling methods, which is widely used in the inference of complex probabilistic models. However, current MCMC methods can incur high autocorrelation of samples, which means that the samples generated by MCMC samplers are far from independent. In this paper, we propose the neural networks Langevin Monte Carlo (NNLMC) which makes full use of the flexibility of neural networks and the high efficiency of the Langevin dynamics sampling to construct a new MCMC sampling method. We propose the new update function to generate samples and employ appropriate loss functions to improve the performance of NNLMC during the process of sampling. We evaluate our method on a large diversity of challenging distributions and real datasets. Our results show that NNLMC is able to sample from the target distribution with low autocorrelation and rapid convergence, and outperforms the state-of-the-art MCMC samplers.

INDEX TERMS Hamiltonian dynamics, Langevin dynamics, Markov chain Monte Carlo, neural networks.

I. INTRODUCTION

In Bayesian machine learning, the inference of the complex probabilistic models generally needs the evaluation of the intractable integrals, which is challenging. Markov chain Monte Carlo (MCMC) [10] is a widely used approximation method for Bayesian probabilistic models [3]–[6]. MCMC methods construct the Markov chain in terms of the target distribution and iteratively sample from the Markov chain. If the number of samples is large enough, we can obtain the asymptotically unbiased estimation of the target distribution.

Recently, MCMC methods based on dynamics are widely used in Bayesian machine learning [18]–[24], and signal processing [1], [2]. Metropolis adjusted Langevin algorithm (MALA) [11] constructs the Markov chain through Langevin dynamics. Compared with the random-walk proposals [17], MALA sampler explores the state space more efficiently because it exploits the gradient information of the target distribution. Hamiltonian Monte Carlo (HMC) [24] utilizes Hamiltonian dynamics to construct the Markov chain. Compared with MALA, HMC introduces a momentum variable to adjust the range of exploration. Besides, HMC has various attractive properties. Since the total energy of the Hamiltonian system remains unchanged and the gradient information of the target distribution is utilized, HMC sampler has a high acceptance rate with rapid convergence. Magnetic Hamiltonian Monte Carlo (MHMC) [9] further adjusts the range of the exploration through the magnetic field, which reduces the autocorrelation of the samples and enhances the convergence speed. MCMC methods based on dynamics have multiple advantages, but when it comes to some challenging distributions, these methods tend to have poor performance because the neighbor samples can be very close, which results in high autocorrelation.

To improve the performance of the dynamics based MCMC methods, we design a new sampler, which is called neural networks Langevin Monte Carlo (NNLMC). The samples generated from NNLMC have low autocorrelation with rapid convergence. NNLMC takes advantage of the neural networks to adjust the range of exploration of the sampler. We design appropriate loss functions to train the sampler. The pure dynamics based MCMC methods only consider about the gradient information of the target distribution, while NNLMC not only uses the gradient information of the target distribution but also learns to improve the performance of the sampler during the process of sampling.

The contributions of this paper are summarized as follows. 1) We introduce the neural networks to Langevin dynamics to construct a novel Markov chain Monte Carlo sampler. The use of neural networks makes the sampler generate the

The associate editor coordinating the review of this manuscript and approving it for publication was Haiyong Zheng.

samples more flexible. 2) During the process of sampling, NNLMC is able to learn from the history information of the samples to generate new samples. 3) We design appropriate loss functions to train the sampler and further improve the performance of the sampler. 4) Experiments over challenging distributions and real datasets are conducted to estimate the performance of the proposed method. We first compare our method with MALA, HMC, and MHMC [9], [11], [24] on six challenging distributions in terms of the autocorrelation of the samples, maximum mean discrepancy (MMD) [26], the effective sample size (ESS) [24], and time consumption. We then sample from the posterior distributions of real datasets using Bayesian logistic regression to evaluate the performance of the samplers. The final results demonstrate that NNLMC is able to converge to the target distribution rapidly and generate more independent samples.

The rest of this article is organized as follows. In Section II, we introduce the related work which consists of Metropolis adjusted Langevin algorithm, Hamiltonian Monte Carlo, and magnetic Hamiltonian Monte Carlo. In Section III, we introduce our neural networks Langevin Monte Carlo and describe how to train this MCMC sampler. Experiments on sampling from various challenging distributions and posterior distribution of real datasets are given in Section IV. In Section V, we give a conclusion and discuss future work.

II. RELATED WORK

In this section we introduce the related dynamics based MCMC samplers which include MALA, HMC and MHMC.

A. METROPOLIS ADJUSTED LANGEVIN ALGORITHM

Metropolis adjusted Langevin algorithm (MALA) [11] is a popular MCMC sampler which aims to sample from the target distribution efficiently. Compared with the Metropolis-Hasting algorithm, MALA takes advantage of the gradient information to construct the Markov chain, while the Metropolis-Hasting algorithm proposes the new state through a random walk [17], which significantly reduces the efficiency of sampling.

The main idea of MALA is to use Langevin dynamics to construct the Markov chain. Langevin dynamics is a kind of stochastic process and it can be written as the stochastic differential equation (SDE) [13] which takes the form as:

$$\mathrm{d}\theta = \frac{1}{2}\nabla_\theta \ln\pi(\theta)\mathrm{d}t + \mathrm{d}W, \tag{1}$$

where π represents the probability density function, θ represents the random variable that is sampled, t represents the time, and W is a Wiener process. Since solving this SDE is very difficult, a first-order Euler-Maruyama discretization [15] is used to provide an approximate solution to the SDE, and the solution can be written as:

$$\theta_{t+1} = \theta_t + \frac{\epsilon^2}{2}\cdot\nabla_\theta \ln\pi(\theta_t) + \epsilon\cdot z_t, \tag{2}$$

where ϵ is the step size of discretization and $z_t \sim \mathcal{N}(z_t|0, I)$.

In order to satisfy the detailed balance [10] to make the MCMC converge to the target distribution, the Metropolis-Hasting procedure [12] is utilized. The transformation probability of MALA can be written as:

$$\begin{aligned} T(\theta'|\theta_n) &= \mathcal{N}(\theta'|\mu(\theta_n), \epsilon^2\cdot I),\\ T(\theta_n|\theta') &= \mathcal{N}(\theta_n|\mu(\theta'), \epsilon^2\cdot I),\\ \mu(\theta_n) &= \theta_n + \frac{\epsilon^2}{2}\cdot\nabla_\theta \ln\pi(\theta_n),\\ \mu(\theta') &= \theta' + \frac{\epsilon^2}{2}\cdot\nabla_\theta \ln\pi(\theta'), \end{aligned} \tag{3}$$

where $T(\theta'|\theta_n)$ and $T(\theta_n|\theta')$ are transformation probability, θ_n is the last sample and θ' is the new sample generated through (2). The final acceptance rate of MALA takes the form as:

$$\min\left[1, \frac{\pi(\theta')T(\theta'|\theta_n)}{\pi(\theta_n)T(\theta_n|\theta')}\right]. \tag{4}$$

MALA takes advantage of the gradient information of the target distribution to construct an efficient Markov chain, which makes the sampler converge to the target distribution rapidly. However, the first-order Euler-Maruyama discretization makes the samples remain in high correlation, which means that the samples are far from independent.

B. HAMILTONIAN MONTE CARLO

Hamiltonian Monte Carlo (HMC) is one of the most popular MCMC algorithm [17], [24]. HMC utilizes Hamiltonian dynamics to construct the sampler. Compared with MALA, HMC introduces a momentum variable p to control the scope of exploring the state space. For the Hamiltonian dynamics system, we describe the total energy through the following equation:

$$H(\theta, p) = U(\theta) + K(p), \tag{5}$$

where θ is the position variable that needs to be sampled, p is the auxiliary momentum variable, $U(\theta)$ is the potential energy at position θ, and $K(p)$ is the kinetic energy. $H(\theta, p)$ is the total energy of the Hamiltonian dynamics system. The Hamiltonian equations can be written as:

$$\begin{aligned} \frac{d\theta}{dt} &= \frac{\partial H(p,\theta)}{\partial p} = \nabla_p K(p),\\ \frac{dp}{dt} &= -\frac{\partial H(p,\theta)}{\partial \theta} = -\nabla_\theta U(\theta). \end{aligned} \tag{6}$$

Generally, solving the differential equations in (6) is very difficult. In practice, a discretization method called leapfrog [25] is used to simulate the Hamiltonian dynamics, which takes the form as:

$$\begin{aligned} p_{t+\frac{\epsilon}{2}} &= p_t - \frac{\epsilon}{2}\cdot\nabla_\theta U(\theta_t),\\ \theta_{t+\epsilon} &= \theta_t + \epsilon\cdot\nabla_p K(p_{t+\frac{\epsilon}{2}}),\\ p_{t+\epsilon} &= p_{t+\frac{\epsilon}{2}} - \frac{\epsilon}{2}\cdot\nabla_\theta(\theta_{t+\epsilon}), \end{aligned} \tag{7}$$

Algorithm 1 Hamiltonian Monte Carlo [24]

Input: step size ϵ, leapfrog size L, initial point θ_1, sample number N.
Output: samples θ.
for $n = 1$ **to** N **do**
  $p^{(n)} \sim \mathcal{N}(0, 1)$
  $(\theta_0, p_0) = (\theta^{(n)}, p^{(n)})$
  $p_0 = p_0 - \frac{\epsilon}{2} \cdot \nabla_\theta U(\theta_0)$
  $\theta_0 = \theta_0 + \epsilon \cdot \nabla_p K(p_0)$
  for $i = 1$ **to** L **do**
    $p_i = p_{i-1} - \epsilon \cdot \nabla_\theta U(\theta_{i-1})$
    $\theta_i = \theta_{i-1} + \epsilon \cdot \nabla_p K(p_i)$
  end for
  $p_L = p_L - \frac{\epsilon}{2} \cdot \nabla_\theta U(\theta_L)$
  $(\theta', p') = (\theta_L, p_L)$
  $u \sim$ Uniform[0, 1]
  $\alpha = \min\left[1, \exp\left(\frac{H(\theta^{(n-1)}, p^{(n-1)})}{H(\theta', p')}\right)\right]$
  if $\alpha > u$ **then**
    $(\theta^{(n+1)}, p^{(n+1)}) = (\theta', p')$
  else
    $(\theta^{(n+1)}, p^{(n+1)}) = (\theta^{(n)}, p^{(n)})$
  end if
end for

where ϵ represents the step size, p_t and θ_t represent the momentum and position at time t, respectively. Since the process of discretization introduces some errors, the Metropolis-Hasting procedure is used to adjust the acceptance rate of the sampler, which takes the form as:

$$\min\left[1, \exp\left(-\frac{H(\theta_{t+\epsilon}, p_{t+\epsilon})}{H(\theta_t, p_t)}\right)\right]. \tag{8}$$

Suppose the target distribution is $\pi(\theta)$. The potential energy function can be written as:

$$U(\theta) \propto -\ln\pi(\theta). \tag{9}$$

HMC introduces momentum variable p, and it samples from the joint distribution $\pi(\theta, p) \propto \exp[-H(\theta, p)]$. Through (7), the new state (θ', p') is generated. Since $\exp[-H(\theta, p)] = \exp[-U(\theta)] \cdot \exp[-K(p)]$, position variable θ and momentum variable p are independent, so HMC can alternatively sample θ and p. Alg. 1 demonstrates the main idea of HMC.

C. MAGNETIC HAMILTONIAN MONTE CARLO

Magnetic Hamiltonian Monte Carlo (MHMC) [9] exploits the non-canonical Hamiltonian dynamics to construct the MCMC sampler. Different from HMC, MHMC adds a magnetic field to the Hamiltonian dynamics.

We can imagine the magnetic energy as a kind of potential energy. To make the concept of magnetic concrete, we suppose that there is a charged particle in the magnetic field. If a particle is not charged, then only gravity acts on the particle. If the particle is charged, the magnetic force may act on the particle. Furthermore, the value of force may change along with the change of the value of the momentum. The larger the momentum is, the larger the magnetic force will act on the particle. Moreover, the direction of the force is always perpendicular to the direction of motion.

MHMC can be viewed as a special case of HMC. When the direction of the magnetic field is the same with the direction of the motion of the particle, MHMC sampler degenerates to the HMC sampler. The magnetic Hamiltonian equations are formulated as follows:

$$\begin{aligned} \frac{d\theta}{dt} &= M^{-1}p, \\ \frac{dp}{dt} &= -\nabla U(\theta) + Gp, \end{aligned} \tag{10}$$

where G represents the magnetic filed, M is the mass matrix and the update equations are defined as:

$$\begin{aligned} \theta_{t+1} &= \theta_t + G^{-1}(\exp(G \cdot \epsilon) - I)p_t, \\ p_{t+1} &= \exp(G \cdot \epsilon)p_t. \end{aligned} \tag{11}$$

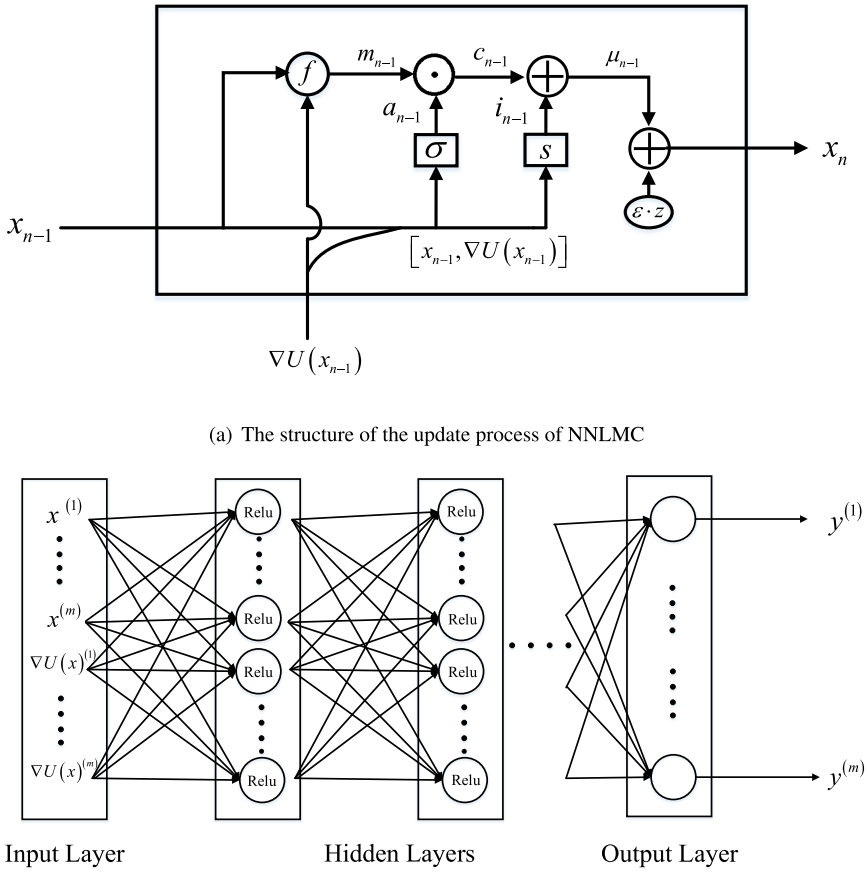
With the help of the magnetic field, the charged particle can explore its state space more efficiently. MHMC sampler may reduce the autocorrelation rate of the samples, which makes the samples more independent. Since MHMC has significant performance, in the experiment part, we compare our method with it.

III. NEURAL NETWORKS LANGEVIN MONTE CARLO

MALA iteratively samples from the target distribution through the Markov chain defined on the Langevin dynamics. However, MALA makes the samples remain highly autocorrelated, which makes the samples far from independent. In order to sample from the target distribution independently and rapidly, we propose neural networks Langevin Monte Carlo (NNLMC), which adjust the Langevin dynamics through neural networks to obtain samples, instead of obtaining samples using (2). In this section, we introduce the NNLMC in detail, then we propose appropriate loss functions to train and optimize the sampler.

Although HMC is an efficient sampler, utilizing neural networks to improve HMC sampler is difficult, because the introduction of the neural networks breaks the detailed balance. In other words, the transformation probabilities $T(x'|x)$ and $T(x|x')$ in HMC sampler are the same. However, the introduction of the neural networks makes the transformation probabilities different, where x is the last sample and x' is the new sample. Although we can adjust the acceptance probability by Metropolis-Hasting procedure, the transformation probability is difficult to calculate. In MALA, the transformation probability is designed as a Gaussian distribution, which provides us a chance to design the model on it.

Assume that $\pi(x)$ represents the target distribution, $U(x) = -\ln\pi(x)$ is the potential energy function, and $\nabla_x U(x)$ represents the gradient of the potential energy function. x is the variable that needs to be sampled. The architecture of NNLMC is demonstrated in Figure (1(a)). NNLMC computes a mapping from the input x_{n-1} and $\nabla_x U(x_{n-1})$ to the output



(a) The structure of the update process of NNLMC

(b) The structure of the neural networks of NNLMC

FIGURE 1. The architecture of the NNLMC.

x_n by using the following equations:

$$\begin{aligned} m_{n-1} &= f(x_{n-1}, \nabla_x U(x_{n-1})) \\ &= x_{n-1} - \frac{\epsilon^2}{2} \cdot \nabla_x U(x_{n-1}), \\ a_{n-1} &= \sigma([x_{n-1}, \nabla_x U(x_{n-1})]), \\ c_{n-1} &= m_{n-1} \odot a_{n-1}, \\ i_{n-1} &= s([x_{n-1}, \nabla_x U(x_{n-1})]), \\ \mu_{n-1} &= c_{n-1} + i_{n-1}, \\ x_n &= \mu_{n-1} + \epsilon \cdot z_n, \end{aligned} \tag{12}$$

where "$\odot$" is the element-wise product of the vectors, ϵ is the step size of the discretization, and z_n is the standard Gaussian distribution. $\sigma(x) = y_\sigma = W_1^\top Relu(W_\sigma^\top x)$ and $s(x) = y_s = W_2^\top Relu(W_s^\top x)$ are the neural networks parameterized by W_1, W_σ, W_2, and W_s, respectively, where y is the output of the neural networks and we use the Relu as the active function. We do not use the active function at the output layer. The structure of the neural networks is demonstrated in Figure (1(b)). We concatenate x_{n-1} and $\nabla_x U(x_{n-1})$, and take $[x_{n-1}, \nabla_x U(x_{n-1})]$ as the input of the neural networks. Assume that x_{n-1} is a m-D vector, then $[x_{n-1}, \nabla_x U(x_{n-1})]$ is a 2m-D vector. Through the above neural networks, we obtain a_{n-1} and i_{n-1} which are m-D vectors.

It is noted that (12) can be divided into four main parts. In the first part, we exploit the gradient information directly to obtain the proposal state m_{n-1}. In the second part, we adjust the proposal state through the neural networks a_{n-1}, and then we obtain c_{n-1}. In the third part, we additional add i_{n-1} to provide more flexibility for the proposal state, which can be viewed as the bias. Finally, we add the Gaussian term to obtain the proposal sample. Compared with MALA, the proposed method exploits neural networks to add the flexibility of generating samples.

We then talk about how to accept the proposal samples. Since we take advantages of the framework of Langevin dynamics, we can utilize the Metropolis-Hasting procedure to maintain the detailed balance, which ensures that the samples can finally converge to the target distribution. So the transformation function of NNLMC can also be viewed as a Gaussian distribution, and the mean of this transformation function can be written as follows:

$$\begin{aligned} \mu(x_n) = \left(x_n - \frac{\epsilon^2}{2} \cdot \nabla_x U(x_n)\right) \odot \sigma\left([x_n, \nabla_x U(x_n)]\right) \\ + s\left([x_n, \nabla_x U(x_n)]\right). \end{aligned} \tag{13}$$

The variance of this transformation function is ϵ^2. Finally, the transformation probability can be defined as:

$$T(x'|x_n) = \mathcal{N}\left(x'|\mu(x_n), \epsilon^2 \cdot I\right), \tag{14}$$

where x' is the new sample generated through (12) and the acceptance rate α takes the form as:

$$\begin{aligned}\alpha &= \min\left[1, \frac{\pi(x')T(x_n|x')}{\pi(x_n)T(x'|\theta_n)}\right] \\ &= \min\left[1, \frac{\pi(x')\mathcal{N}(x_n|\mu(x'), \epsilon^2 \cdot I)}{\pi(x_n)\mathcal{N}\left(x'|\mu(x_n), \epsilon^2 \cdot I\right)}\right].\end{aligned} \tag{15}$$

We take (15) as the acceptance rate to accept the samples so that we can ensure that the samples are able to converge to the target distribution.

In order to improve the performance of NNLMC sampler to have lower autocorrelation, rapid convergence, and higher acceptance rate during the process of sampling, we next discuss loss functions used in NNLMC.

A. LOSS FUNCTION OF THE TRAINING PROCEDURE

We design two loss functions to train the parameters of W_σ and W_s.

The first loss function is defined as:

$$l_1(\omega) = \exp\left(-\left||x' - x_n|\right|\right), \tag{16}$$

where $\left||x' - x_n|\right| = \sqrt{(x' - x_n)^\top (x' - x_n)}$ and ω represents the parameters of W_σ and W_s. Our purpose is to maximize the distance between x' and x_n to reduce the autocorrelation among samples. Although we can directly minimizing $-\left||x' - x_n|\right|$ to enlarge the distance between x' and x_n, we find it difficult to optimize. With the enhancement of the distance, this loss function will dominate the optimization direction, which will lead to the unexpected results.

The second loss function is defined as:

$$l_2(\omega) = \exp\left(-\frac{\pi(x')}{\pi(x_n)}\right). \tag{17}$$

We wish to use (17) to raise the acceptance rate of the samples. The final loss function takes the form as:

$$L(\omega) = w_1 \cdot l_1(\omega) + w_2 \cdot l_2(\omega), \tag{18}$$

where w_1 and w_2 are two weight parameters and $w_1 + w_2 = 1$.

B. THE COMPLETE ALGORITHM

We then elaborately introduce the details of the procedure of training the NNLMC sampler on a batch of samples. Assume that π represents the target distribution, $\boldsymbol{x}_1^{(1:b)} = \{x_1^i | i \in [1, b]\}$ represents the initial samples and $\boldsymbol{x}_n^{(1:b)}$ represents the samples at time n, where b represents the batch size. Provided that we have obtained $\boldsymbol{x}_n^{(1:b)}$, then new samples $\boldsymbol{x}_t^{(1:b)}$ can be calculated through (12) which takes the form as:

$$\begin{aligned}\boldsymbol{x}_t^{(1:b)} &= \left(\boldsymbol{x}_n^{(1:b)} - \frac{\epsilon^2}{2} \cdot \nabla_{\boldsymbol{x}} U\left(\boldsymbol{x}_n^{(1:b)}\right)\right) \\ &\odot \sigma\left(\left[\boldsymbol{x}_n^{(1:b)}, \nabla_{\boldsymbol{x}} U\left(\boldsymbol{x}_n^{(1:b)}\right)\right]\right) \\ &+ s\left(\left[\boldsymbol{x}_n^{(1:b)}, \nabla_{\boldsymbol{x}} U\left(\boldsymbol{x}_n^{(1:b)}\right)\right]\right) + \epsilon \cdot \boldsymbol{z}_n^{(1:b)}.\end{aligned} \tag{19}$$

We then calculate the transformation probability $T\left(\boldsymbol{x}_t^{(1:b)}|\boldsymbol{x}_n^{(1:b)}\right)$ and $T\left(\boldsymbol{x}_n^{(1:b)}|\boldsymbol{x}_t^{(1:b)}\right)$ which take the form as:

$$\begin{aligned}T\left(\boldsymbol{x}_t^{(1:b)}|\boldsymbol{x}_n^{(1:b)}\right) &= \mathcal{N}\left(\boldsymbol{x}_t^{(1:b)}|\mu\left(\boldsymbol{x}_n^{(1:b)}\right), \epsilon^2 \cdot I^{(1:b)}\right), \\ T\left(\boldsymbol{x}_n^{(1:b)}|\boldsymbol{x}_t^{(1:b)}\right) &= \mathcal{N}\left(\boldsymbol{x}_n^{(1:b)}|\mu\left(\boldsymbol{x}_t^{(1:b)}\right), \epsilon^2 \cdot I^{(1:b)}\right),\end{aligned} \tag{20}$$

where $\mu(\boldsymbol{x}_n^{(1:b)})$ and $\mu(\boldsymbol{x}_t^{(1:b)})$ can be calculated through (13).

After calculating the transformation probability, we can get the acceptance rate as follows:

$$\begin{aligned}&\alpha^{(1:b)} \\ &= \min\left[1, \frac{\pi\left(\boldsymbol{x}_t^{(1:b)}\right)\mathcal{N}\left(\boldsymbol{x}_t^{(1:b)}|\mu\left(\boldsymbol{x}_n^{(1:b)}\right), \epsilon^2 \cdot I^{(1:b)}\right)}{\pi\left(\boldsymbol{x}_n^{(1:b)}\right)\mathcal{N}\left(\boldsymbol{x}_n^{(1:b)}|\mu\left(\boldsymbol{x}_t^{(1:b)}\right), \epsilon^2 \cdot I^{(1:b)}\right)}\right].\end{aligned} \tag{21}$$

For each sample $x_t^{(i)}$ in $\boldsymbol{x}_t^{(1:b)}$, we accept the sample through the acceptance rate $\alpha^{(i)}$, which can be written as:

$$\begin{cases} x_{n+1}^{(i)} = x_t^{(i)} & \text{if } x_t^{(i)} \text{ is accepted, } i \in [1, b] \\ x_{n+1}^{(i)} = x_n^{(i)} & \text{if } x_t^{(i)} \text{ is not accepted, } i \in [1, b]. \end{cases} \tag{22}$$

After accepting these samples, we further improve the performance of NNLMC sampler by optimizing the parameters W_σ and W_s. Since a batch of the samples is used, the loss functions take the form as:

$$\begin{aligned}l_1(\omega) &= \exp\left(-\frac{1}{b}\sum_{i=1}^{b}\left|x_t^{(i)} - x_n^{(i)}\right|\right), \\ l_2(\omega) &= \exp\left(-\frac{1}{b}\sum_{i=1}^{b}\frac{\pi\left(x_t^{(i)}\right)}{\pi\left(x_n^{(i)}\right)}\right), \\ L(\omega) &= w_1 \cdot l_1(\omega) + w_2 \cdot l_2(\omega).\end{aligned} \tag{23}$$

Finally, we minimize the loss function through gradient descent, which can be written as:

$$\omega = \omega - \gamma\frac{\partial L(\omega)}{\omega}, \tag{24}$$

where γ represents the learning rate. The complete algorithm is given in Alg. 2.

IV. EXPERIMENTS

In this section, we show the performance of NNLMC. We conduct the experiments on six distributions which consist of ring distribution (RING), the strongly correlated Gaussian (SCG), the ill-conditioned Gaussian (ICG), the mixtures of Gaussian (MOG), the rough well (RW), and the Gaussian funnel (GF). We compare NNLMC with HMC, MALA, and MHMC on autocorrelation, the maximum mean discrepancy, effective sample size, and time consumption. Next, we conduct the experiments on six real datasets using Bayesian logistic regression. We sample from the posterior distributions and compare our methods with logistic regression (LR), variational Bayesian logistic regression (VBLR), HMC and MALA on some performance indexes.

Algorithm 2 Neural Networks Langevin Monte Carlo

Input: target probability density function π, step size ϵ, learning rate γ, optimization steps L, sample number N, batch size b, initial batch samples $\boldsymbol{x}_1^{(1:b)}$, energy function $U(\cdot)$, gradient of energy function $\nabla U(\cdot)$.
Output: samples $x_{1:N}$.
Initializing the parameters $\omega = (W_\sigma, W_s)$ of the neural networks.
for $n = 1$ **to** N **do**
  for $iter = 1$ **to** L **do**
    Obtaining the proposal samples $\boldsymbol{x}_t^{(1:b)}$ through (19).
    Calculating $\mu\left(\boldsymbol{x}_n^{(1:b)}\right)$ and $\mu\left(\boldsymbol{x}_t^{(1:b)}\right)$ through (13).
    Calculating $T\left(\boldsymbol{x}_t^{(1:b)}|\boldsymbol{x}_n^{(1:b)}\right)$ and $T\left(\boldsymbol{x}_n^{(1:b)}|\boldsymbol{x}_t^{(1:b)}\right)$ through (20).
    Obtaining the acceptance rate $\alpha^{(1:b)}$ through (21).
    Calculating loss function $L(\omega)$ through (23).
    $\omega = \omega - \gamma\nabla_\omega L(\omega)$
  end for
  $u^{(1:b)} \sim$ Uniform[0, 1]
  for $i = 1$ **to** b **do**
    if $\alpha^{(i)} > u^{(i)}$ **then**
      $x_{n+1}^{(i)} = x_t^{(i)}$
    else
      $x_{n+1}^{(i)} = x_n^{(i)}$
    end if
  end for
end for

The whole experiments are conducted on a server with eight GPUs. Batch size is set to be 3000, the hidden layers of the neural networks are set to be [512, 512, 512]. The activation functions are set to be ReLU. The learning rate of NNLMC is set to be 1e-5. w_1 and w_2 are both set to be 0.5 in all experiments.

A. PERFORMANCE INDEXES

First, we introduce the performance indexes.

Effective sample size (ESS) [24] is a common method to measure the variance of the MCMC samplers, which takes the form as:

$$\text{ess} = N/(1 + 2 \times \sum_{s=1}^{M} \rho(s)), \tag{25}$$

where ess represents the effective sample size, $\rho(\cdot)$ is the autocorrelation function, s represents the step size of autocorrelation, N is the number of the samples, and M is the maximum step size of autocorrelation. Since our method almost has no autocorrelation after 30 steps, we set $M = 30$.

Autocorrelation is widely used to measure the correlation between a series of samples. Assume that X represents a series of samples, and t represents the position of the sample and $t \in (1, N - s)$, where N is the number of samples and s represents the step size of autocorrelation.

TABLE 1. Six distributions and their acronyms.

DISTRIBUTIONS	ACRONYM
RING-SHAPED DISTRIBUTION	RING
MIXTURES OF GAUSSIAN	MOG
2-D STRONGLY CORRELATED GAUSSIAN	SCG
ILL-CONDITIONED GAUSSIAN	ICG
ROUGH WELL	RW
GAUSSIAN FUNNEL	GF

The autocorrelation function $\rho(\cdot)$ can be written as:

$$\rho(s) = \frac{\mathbb{E}[(X_t - \mu)(X_{t+s} - \mu)]}{\sigma^2}, \tag{26}$$

where function $\mathbb{E}(\cdot)$ is the mathematical expectation operator, $\mu = \mathbb{E}[X]$ and $\sigma^2 = \mathbb{E}[(X - \mu)^2]$. Autocorrelation is one of the most effective ways to measure the correlation between samples. The lower the autocorrelation of the samples is, the more independent the samples will be. Kullback-Leibler divergence is used to calculate the difference between two distributions represented as the function, while Maximum mean discrepancy (MMD) [26] is commonly used to measure the discrepancy between two distributions represented as a series of samples. The lower the value of MMD is, the closer the two distribution will be. We assume that X represents the samples generated from the sampler and Y represents samples generated from the actual distribution. M and N are the number of samples for X and Y, respectively. The definition of MMD takes the form as:

$$R^2[X, Y] = \frac{1}{M^2}\sum_{i,j=1}^{M}\kappa(\mathbf{x}_i, \mathbf{x}_j) - \frac{2}{MN}\sum_{i,j=1}^{M,N}\kappa(\mathbf{x}_i, \mathbf{y}_j) + \frac{1}{N^2}\sum_{i,j=1}^{N}\kappa(\mathbf{y}_i, \mathbf{y}_j), \tag{27}$$

where $\kappa(\cdot, \cdot)$ is the kernel function which takes the form as:

$$\kappa(\mathbf{x}, \mathbf{y}) = \left(1.0 + \mathbf{x}^\top\mathbf{y}\right)^2. \tag{28}$$

In our experiments, we utilize MMD to measure the converge speed of the samplers.

B. VARIETIES OF CHALLENGING DISTRIBUTIONS

In this section, we compare NNLMC with MALA, HMC, and MHMC on six distributions in terms of autocorrelation, MMD, ESS, and time consumption. For RING, ICG, MOG, and SCG, we set step size of NNLMC $\epsilon_p = 0.8$. For GF, we set step size $\epsilon_p = 1.0$. For RW, we set step size $\epsilon_p = 0.8$. We set optimization length $L = 2$ for NNLMC. For HMC and MHMC, we set $\epsilon = \frac{\epsilon_p}{lf}$, where lf represents the leapfrog length. We set $lf = 40$ in the experiments. The initial samples are all sampled from the standard normal distribution. For all methods, we sample 20000 samples with 10000 burn-in samples. The distributions are introduced below.

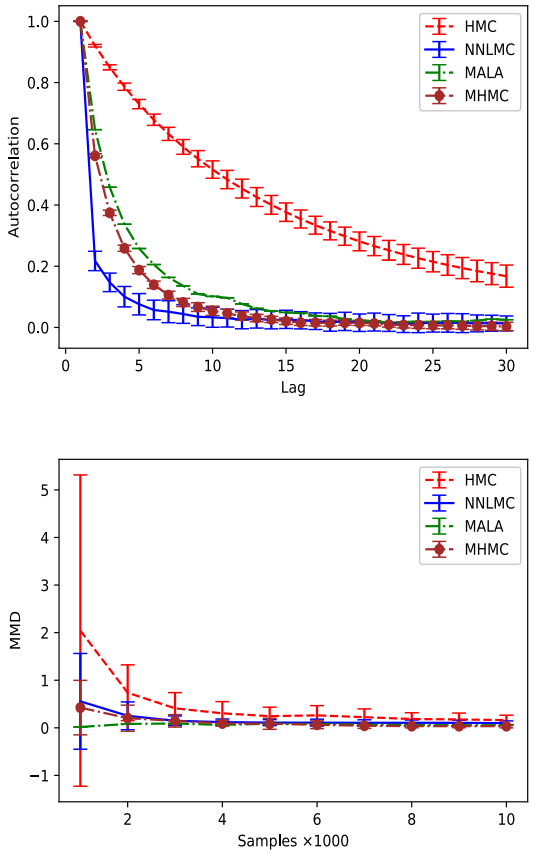


FIGURE 2. The performance of NNLMC, HMC, MALA, and MHMC on Gaussian funnel. The upper figure demonstrates the relationship between autocorrelation and lag (the interval between samples). The bottom figure shows the relationship between MMD and the number of samples.

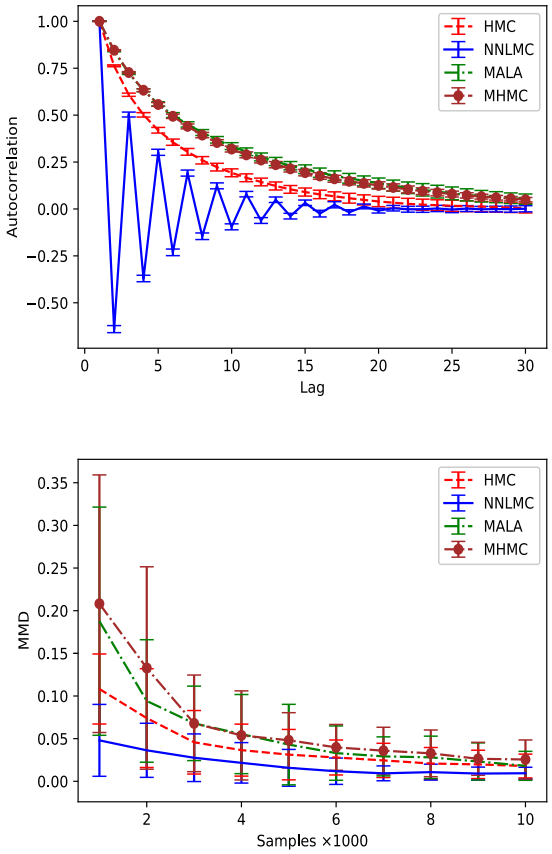


FIGURE 3. The performance of NNLMC, HMC, MALA, and MHMC on mixtures of Gaussian. The upper figure demonstrates the relationship between autocorrelation and lag (the interval between samples). The bottom figure shows the relationship between MMD and the number of samples.

The energy function of the ring-shaped distribution can be written as:

$$U(x) = \frac{(\sqrt{x_1^2 + x_2^2} - 2)^2}{0.32}. \tag{29}$$

We sample from a 2-D ill-conditioned Gaussian whose diagonal covariance is set to be 10.05 and 0.105. We sample from a 2-D strongly correlated Gaussian (which is similar to the case in [24]) whose covariance matrix is as follows:

$$\sum = \begin{bmatrix} 5.05 & -4.95 \\ -4.95 & 5.05 \end{bmatrix}. \tag{30}$$

We sample from the Rough well (which is similar to the case in [8]) whose energy function takes the form as:

$$U(x) = \frac{1}{2}x^{\mathrm{T}}x + \eta \sum_i \cos(\frac{x_i}{\eta}), \tag{31}$$

where we set $\eta = 10^{-2}$. We sample from the Gaussian funnel whose energy function can be defined as:

$$U(x) = \frac{1}{2}\left(\left(\frac{x_1}{\sigma}\right)^2 + \frac{x_2^2}{\exp(x_1)} + \ln\left(2\pi \cdot \exp(x_1)\right)\right), \tag{32}$$

where we set $\sigma = 1.0$. We sample from the mixtures of Gaussian whose probability density function takes the form as:

$$\pi(x) = \frac{1}{2}\mathcal{N}(x|\mu, I) + \frac{1}{2}\mathcal{N}(x|-\mu, I), \tag{33}$$

where $\mu = [-1.0, 1.0]$. The distributions and their acronyms are showed in Table. 1.

Fig. 2 and Fig. 3 illustrate the performance on Gaussian funnel and mixtures of Gaussian, respectively, which show that the samples generated from NNLMC sampler have lower autocorrelation and more rapid convergence than other methods.

As Fig. 4, Fig. 5 and Fig. 6 illustrate, in ICG, SCG and RING distributions, the autocorrelation of samples generated from NNLMC is significantly lower than other methods, and NNLMC is able to converge to the target distribution rapidly, which suggests that the proposed loss functions do make the neural networks adjust the sampler, and thus the performance of the sampler is significantly improved.

In rough well, we find that HMC, MALA, and MHMC still have high autocorrelation after 30-step autocorrelation, while NNLMC has much lower autocorrelation. However, NNLMC has large fluctuation. We further inquire the reason

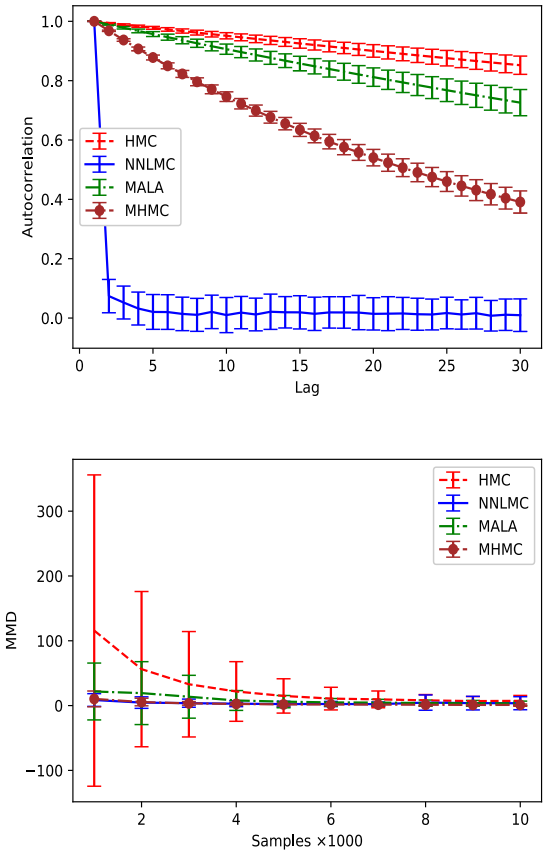


FIGURE 4. The performance of NNLMC, HMC, MALA, and MHMC on ill-conditioned Gaussian. The upper figure demonstrates the relationship between autocorrelation and lag (the interval between samples). The bottom figure shows the relationship between MMD and the number of samples.

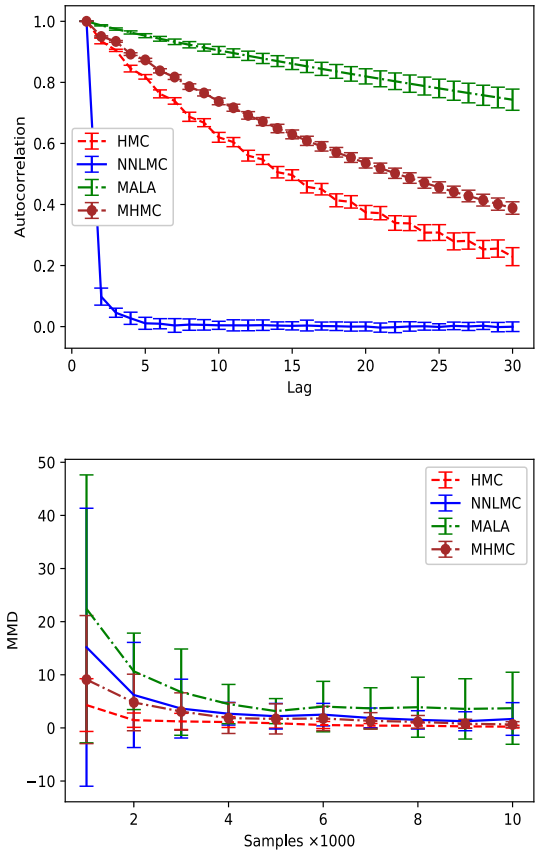


FIGURE 5. The performance of NNLMC, HMC, MALA, and MHMC on strongly correlated Gaussian. The upper figure demonstrates the relationship between autocorrelation and lag (the interval between samples). The bottom figure shows the relationship between MMD and the number of samples.

TABLE 2. The effective sample size (ESS) of HMC, MALA, MHMC and NNLMC on six distributions.

DATA	HMC	MALA	MHMC	NNLMC
RING	240.48	210.98	261.27	**3592.46**
MOG	804.69	540.71	556.57	**4231.47**
SCG	307.46	189.84	252.23	**2878.76**
ICG	177.18	190.85	250.10	**2429.14**
RW	166.60	173.25	176.66	**5478.01**
GF	367.64	1079.88	1376.07	**1897.74**

TABLE 3. The time consumption (seconds) of HMC, MALA, MHMC and NNLMC on six distributions.

DATA	HMC	MALA	MHMC	NNLMC
RING	286	32	343	296
MOG	384	45	449	461
SCG	382	29	462	297
ICG	277	24	322	289
RW	291	33	343	296
GF	617	65	715	463

why the autocorrelation of NNLMC has such fluctuation. It is noted that the neighbour samples have large distance. For example, we have sampled 5 samples a, b, c, d, e, where (a, b), (b, c), (c, d), and (d, e) have long distance, while (a, c), (c, e), and (b, d) have close distance. It is the loss function $l_1(\omega)$ defined in (23) that causes this phenomenon. The final result is demonstrated in Fig. 7. Although NNLMC has large fluctuation in autocorrelation on rough well, the result is competitive, for the autocorrelation of NNLMC declines much more fast than other methods.

Although in some distributions, NNLMC has large fluctuation on autocorrelation, alternate positive correlation and negative correlation imply that NNLMC has excellent performance in ESS, and the final result is demonstrated in Table. 2.

It is noted that compared with other methods, NNLMC has better performance on ESS, which demonstrates the effectiveness of the training procedure. The time consumption of HMC, MALA, MHMC, and NNLMC is shown in Table. 3. It is noted that MALA has the lowest computational cost, and leapfrog length makes HMC and MHMC slower than MALA. Although the computational cost of NNLMC is much higher than that of MALA due to the optimization of the parameters during the process of sampling, it is competitive with HMC and MHMC, for it achieves better performance in autocorrelation and MMD but not consume extra time.

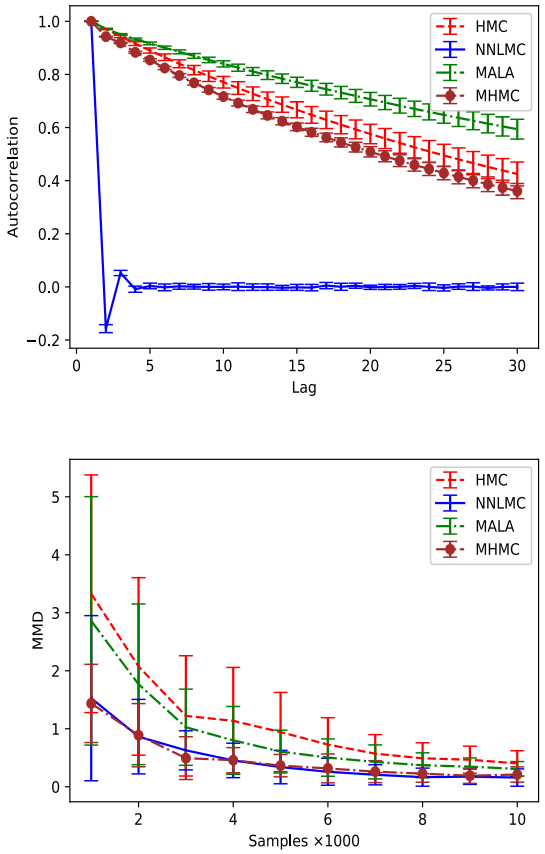


FIGURE 6. The performance of NNLMC, HMC, MALA, and MHMC on the ring distribution. The upper figure demonstrates the relationship between autocorrelation and lag (the interval between samples). The bottom figure shows the relationship between MMD and the number of samples.

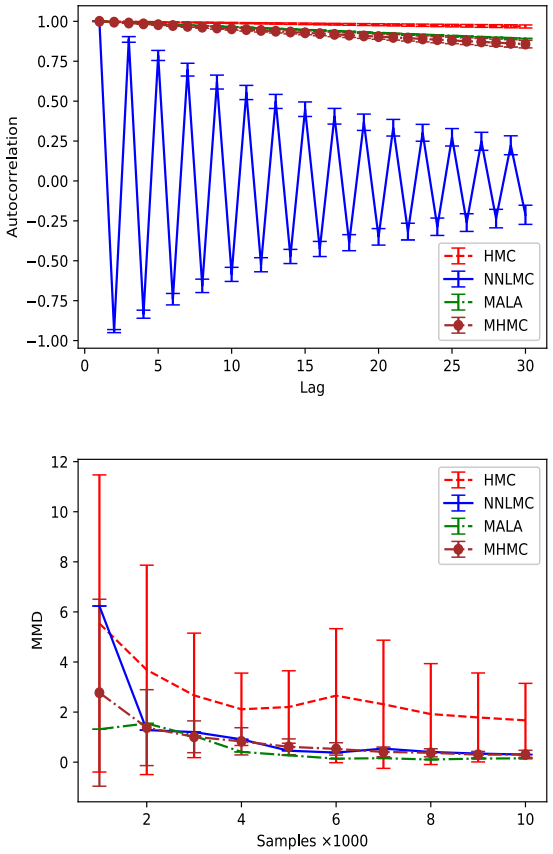


FIGURE 7. The performance of NNLMC, HMC, MALA, and MHMC on rough well. The upper figure demonstrates the relationship between autocorrelation and lag (the interval between samples). The bottom figure shows the relationship between MMD and the number of samples.

In order to further explore the effect in the autocorrelation of changing w_1 and w_2, we conduct an experiment on MOG with different values of w_1 and w_2. The result is demonstrated in Fig. 8.

C. BAYESIAN LOGISTIC REGRESSION

Logistic regression (LR) [14] can be defined as a conditional distribution $p(Y|X)$ which is parameterized by the logistic distribution. The goal of LR is to maximize the likelihood function. After obtaining the parameters of $p(Y|X)$, we can calculate the label of the data. Bayesian logistic regression [27] (BLR) is a also classification model. For the two-class classification problem, BLR can be defined as a posterior distribution $p(\mathbf{w}|t) \propto p(t|\mathbf{w})p(\mathbf{w})$, where $p(t|\mathbf{w}) = \prod_{n=1}^{N}[1 - y_n]^{1-t_n}$, $p(\mathbf{w})$ is the prior distribution of the parameters $\mathbf{w}$, $t = (t_1, ..., t_N)^\top$ is the label of data, and y_n is the predicted value. The goal of BLR is to calculate the predicted value under the expectation of the posterior distribution to predict the label of the data, which takes the form as:

$$\int \sigma(\mathbf{w}^\top \mathbf{x}) p(\mathbf{w}|t) \mathrm{d}\mathbf{w} = \mathbb{E}_{p(\mathbf{w}|t)}[\sigma(\mathbf{w}^\top \mathbf{x})], \tag{34}$$

where $\sigma(\cdot)$ is the logistic function. Evaluating this expectation is intractable, for the normalization term in the posterior distribution is unknown. In order to calculate this expectation,

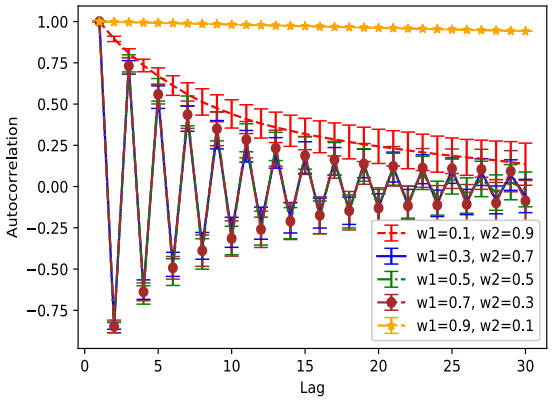


FIGURE 8. The Autocorrelation of NNLMC with different values of w_1 and w_2.

variational Bayesian logistic regression [29] (VBLR) utilizes the variational distribution to approximate the posterior distribution to calculate the integral. In our experiment, we directly sample from the posterior distribution $p(\mathbf{w}|t)$ and evaluate the integral through Monte Carlo methods, which can be written as:

$$\mathbb{E}_{p(\mathbf{w}|t)}[\sigma(\mathbf{w}^\top \mathbf{x})] = \frac{1}{N} \sum_{i=1, \mathbf{w}_i \sim p(\mathbf{w}|t)}^{N} \sigma(\mathbf{w}_i^\top \mathbf{x}). \tag{35}$$

TABLE 4. Classification accuracy of VBLR, LR, HMC, MALA, and NNLMC.

DATA	LR	VBLR	HMC	MALA	NNLMC
HA	69.3±0.2	69.3±0.1	69.3±0.2	**70.16 ±0.2**	69.7±0.6
PI	76.6±0.2	76.2±0.1	**76.6±0.1**	76.2±0.2	74.1±0.6
MA	82.5±0.3	83.1±0.1	83.1±0.1	80.0±0.5	**84.6±0.7**
HE	75.9±0.2	75.9±0.2	75.9±0.2	69.4±1.2	**78.3±2.0**
GE	71.5±0.1	71.5±0.1	**72.5±0.2**	63.17±1.8	70.0±1.3
AU	86.9±0.2	87.6±0.2	87.6±0.2	87.3±1.0	**87.7±1.0**

TABLE 5. Area under the receiver operating characteristic curve of VBLR, LR, HMC, MALA and NNLMC.

DATA	LR	VBLR	HMC	MALA	NNLMC
HA	62.7±0.1	63.2±0.1	63.0±0.2	63.1±0.2	**63.5±0.1**
PI	79.2±0.2	79.3±0.1	79.3±0.1	80.1±0.3	**80.4±0.6**
MA	89.9±0.1	89.8±0.1	89.8±0.1	86.9±0.4	**90.3±0.3**
HE	80.1±0.2	81.3±0.2	82.2±0.3	79.1±0.6	**87.3±0.3**
GE	74.7±0.2	75.5±0.2	76.7±0.3	65.8±0.7	**77.4±1.1**
AU	92.5±0.2	93.9±0.2	93.9±0.3	92.9±0.6	**94.0±0.6**

TABLE 6. The mean value of effective sample size of each dimension for HMC, MALA, and NNLMC.

DATA	HMC	MALA	NNLMC
HA	107.69	1149.43	**1738.59**
PI	73.08	354.62	**882.38**
HE	1093.10	177.80	**1600.42**
MA	**670.72**	290.95	301.20
GE	7.19	9.67	**355.76**
AU	220.60	17.92	**284.95**

Utilizing (35), we can calculate the label of the data. We use the performance of classification to evaluate the quality of the samples.

We sample from the posterior distributions on six real-world datasets from UCI repository [16]: Pima Indian (Pi), Haberman (Ha), Immunotherapy (Im), Heart (He), German (Ge) and Australian (Au). To improve the stability of the model, we normalize all datasets to have zero mean value and unit variance. In all datasets, we set a standard normal distribution $\mathcal{N}(0, I)$ as the prior distribution for the parameters. In each experiment, we sample 8000 samples and repeatedly conduct the experiments ten times to calculate the mean value and the standard deviation for each performance index. In this experiment, we have not compared the proposed method with MHMC, because the magnetic parameter G of MHMC is difficult to set [9] when the dimension of the variable is large.

Table 4 shows the classification accuracy, and Table 5 shows the area under the receiver operating characteristic curve (AUC) [28]. The results demonstrate that NNLMC obtains better performance than LR, VBLR, and similar performance to HMC on classification accuracy. Besides, NNLMC achieves the best performance on AUC. All these performance indexes demonstrate that NNLMC can sample from the posterior distribution accurately. Finally, we utilize the mean ESS of each dimension to further measure the quality of the samples generated from MALA, HMC, and NNLMC. As Table 6 suggests, NNLMC has better performance than HMC and MALA in ESS.

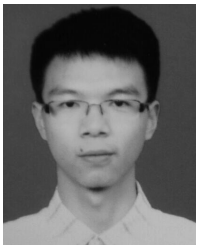
V. CONCLUSION

In this study, we propose a new sampler called neural Langevin Monte Carlo, which takes advantage of the flexibility of neural networks to construct an effective sampler. The appropriate loss functions are designed to train and improve the performance of the proposed sampler. NNLMC is able to generate samples with low autocorrelation and rapid convergence especially for the challenging distributions. The experiments results in various distributions, and real datasets show that our method can provide superior performance compared with pure dynamics based MCMC methods. We plan to introduce reinforcement learning to our model to obtain further improvement for future work.

REFERENCES

[1] L. Martino, "A review of multiple try MCMC algorithms for signal processing," *Digit. Signal Process.*, vol. 75, pp. 134–152, Apr. 2018.
[2] M. F. Bugallo, L. Martino, and J. Corander, "Adaptive importance sampling in signal processing," *Digit. Signal Process.*, vol. 47, pp. 36–49, Dec. 2015.
[3] L. Hou, X. Chen, K. Lan, R. Rasmussen, and J. Roberts, "Volumetric next best view by 3D occupancy mapping using Markov chain Gibbs sampler for precise manufacturing," *IEEE Access*, vol. 7, pp. 121949–121960, 2019.
[4] Z. Wang, S. Lyu, and L. Liu, "Learnable Markov chain Monte Carlo sampling methods for lattice Gaussian distribution," *IEEE Access*, vol. 7, pp. 87494–87503, 2019.
[5] X. Zhou and M. Xu, "Model-based proposal learning for Monte Carlo optimization of redundancy allocation problem," *IEEE Access*, vol. 6, pp. 47953–47958, 2018.
[6] M. Xu, X. Zhou, Q. Huo, and H. Liu, "Efficient stochastic approximation Monte Carlo sampling for heterogeneous redundancy allocation problem," *IEEE Access*, vol. 4, pp. 7383–7390, 2016.
[7] R. M. Neal, "Slice sampling," *Ann. Statist.*, vol. 31, no. 3, pp. 705–767, Jun. 2003.

[8] J. Sohl-Dickstein, M. Mudigonda, and M. R. DeWeese, "Hamiltonian Monte Carlo without detailed balance," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 719–726.
[9] N. Tripuraneni, M. Rowland, Z. Ghahramani, and R. Turner, "Magnetic Hamiltonian Monte Carlo," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3453–3461.
[10] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*. Springer, 2013.
[11] U. Grenander and M. I. Miller, "Representations of knowledge in complex systems," *J. Roy. Stat. Soc., B Methodol.*, vol. 56, no. 4, pp. 549–581, Nov. 1994.
[12] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, Apr. 1970.
[13] B. Øksendal, "Stochastic differential equations," in *Stochastic Differential Equations*. Cham, Switzerland: Springer, 2003, pp. 65–84.
[14] D. A. Freedman, *Statistical Models: Theory and Practice*. Cambridge, U.K.: Cambridge Univ. Press, 2009.
[15] P. E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*. Springer, 2013.
[16] K. Bache and M. Lichman. (2013). UCI Machine Learning Repository. School of Information and Computer Science. University of California. Irvine, CA, USA. [Online]. Available: http://archive.ics.uci.edu/ml
[17] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid Monte Carlo," *Phys. Lett. B*, vol. 195, pp. 216–222, Sep. 1987.
[18] M. Betancourt, "The fundamental incompatibility of scalable Hamiltonian Monte Carlo and naive data subsampling," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 533–540.
[19] M. Girolami and B. Calderhead, "Riemann manifold Langevin and Hamiltonian Monte Carlo methods," *J. Roy. Stat. Soc., B (Stat. Methodol.)*, vol. 73, no. 2, pp. 123–214, Mar. 2011.
[20] G. Celeux, M. Hurn, and C. P. Robert, "Computational and inferential difficulties with mixture posterior distributions," *J. Amer. Stat. Assoc.*, vol. 95, no. 451, pp. 957–970, Sep. 2000.
[21] Y. Zhang, X. Wang, C. Chen, R. Henao, K. Fan, and L. Carin, "Towards unifying Hamiltonian Monte Carlo and slice sampling," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 1462–1470.
[22] Z. Wang, S. Mohamed, and N. Freitas, "Adaptive Hamiltonian and Riemann manifold Monte Carlo," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1462–1470.
[23] M. D. Homan and A. Gelman, "The no-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1593–1623, Jan. 2014.
[24] S. A. G. G. Brooks and X. Jones Meng, *Handbook Markov Chain Monte Carlo*. Boca Raton, FL, USA: CRC Press, 2011.
[25] R. M. Neal, *Probabilistic Inference Using Markov Chain Monte Carlo Methods*. Toronto, ON, Canada: Univ. of Toronto Press, 1993.
[26] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *J. Mach. Learn. Res.*, vol. 13, pp. 723–773, Mar. 2012.
[27] D. J. C. Mackay, "The evidence framework applied to classification networks," *Neural Comput.*, vol. 4, no. 5, pp. 720–736, Sep. 1992.
[28] J. A. Hanley and B. J. Mcneil, "A method of comparing the areas under receiver operating characteristic curves derived from the same cases," *Radiology*, vol. 148, no. 3, pp. 839–843, Sep. 1983.
[29] T. Jaakkola and M. Jordan, "A variational approach to Bayesian logistic regression models and their extensions," *Statist. Comput.*, vol. 1553, no. 3, pp. 147–154, 2000.



MINGHAO GU received the bachelor's degree in computer science in 2017. He is currently pursuing the master's degree with the Pattern Recognition and Machine Learning Research Group, School of Computer Science and Technology, East China Normal University, Shanghai, China. His research interests include machine learning and Monte Carlo methods.

SHILIANG SUN received the Ph.D. degree in pattern recognition and intelligent systems from the Department of Automation and the State Key Laboratory of Intelligent Technology and Systems, Tsinghua University, Beijing, China, in 2007.

He is currently a Professor with the School of Computer Science and Technology and the Head of the Pattern Recognition and Machine Learning Research Group, East China Normal University, Shanghai, China. From 2009 to 2010, he was a Visiting Researcher with the Department of Computer Science, Centre for Computational Statistics and Machine Learning, University College London, London, U.K. In 2014, he was a Visiting Researcher with the Department of Electrical Engineering, Columbia University, New York, NY, USA. His current research interests include kernel methods, multiview learning, learning theory, approximate inference, sequential modeling, and their applications. His research results have expounded in more than 100 publications at peer-reviewed journals and conferences, such as JMLR, IEEE T-NNLS, IEEE T-Cybernetics, IEEE T-MM, NIPS, ICML, IJCAI, and ECML.

Dr. Sun is on the Editorial Board of multiple international journals, including *Neurocomputing* and the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.