STEIN NEURAL SAMPLER

Tianyang Hu[*], **Zixiang Chen**[†], **Hanxi Sun**[*], **Jincheng Bai**[*], **Mao Ye**[*], **Guang Cheng**[*]

ABSTRACT

We propose two novel samplers to produce high-quality samples from a given (un-normalized) probability density. The sampling is achieved by transforming a reference distribution to the target distribution with neural networks, which are trained separately by minimizing two kinds of Stein Discrepancies, and hence our method is named as Stein neural sampler. Theoretical and empirical results suggest that, compared with traditional sampling schemes, our samplers share the following three advantages: 1. Being asymptotically correct; 2. Experiencing less convergence issue in practice; 3. Generating samples instantaneously.

1 Introduction

A core problem in machine learning and Bayesian statistics is to approximate a complex target distribution given its probability density function up to an (unknown) normalizing constant. Discrete approximation of a given distribution has been studied extensively using Markov Chain Monte Carlo (MCMC) (Gamerman and Lopes, 2006). Even though being asymptotically correct, MCMC suffers from several practical issues such as local trap, which is commonly seen in practice and often due to the multimodality of the target. Another classical approach is Variational Bayes (VB) (Kingma and Welling, 2013; Blei et al., 2017), where the target density is approximated using a tractable parametric family. Although the optimization of VB tends to have no convergence issue, the asymptotic correctness is usually not guaranteed. Another alternative is the recently proposed Stein Variational Gradient Descent (SVGD) (Liu and Wang, 2016), which is a particle-based sampling framework that sequentially updates a set of particles. SVGD is able to approximate the target distribution using a small number of particles (samples) but the local trap still remains an issue. Besides, SVGD can not generate samples instantaneously given a trained SVGD chain: if additional samples are required, we will need to start a new SVGD chain from scratch.

Motivated by the weaknesses and advantages of MCMC, VB and SVGD, it is desirable to construct a sampler that is both asymptotically correct with less convergence issue and able to generate sample instantaneously after being well trained. Villani (2008) showed that between any two non-atomic distributions, there always exist a measurable transform T. Therefore, a natural route to achieve our goals is through learning a preservable transformation T that transforms an easy-to-sample distribution $p_z(z)$ to the target distribution $q(x)$. One way to learn such a transformation is to model it within a sufficiently rich family of functions such as Neural Network (Raghu et al., 2016). The success of Generative Adversarial Network (GAN) indicates that Neural networks have very strong expressive power in modeling complex distributions (Goodfellow et al., 2014; Radford et al., 2015; Brock et al., 2018).

Although both GAN and our sampler aim at generating samples from complicated distributions, GAN learns from a set of true samples, while our sampler is trained with the un-normalized true density q. Given an explicit form of q seems more informative, however, it may involve intractable integrations when measuring the distance between the samples and the target. To bypasses these difficulties, we turn to Stein discrepancy, which can serve as a measurement of sample quality.

Our Contribution In this paper, we propose two new sampling schemes based on Stein discrepancy that can directly learn preservable transformations constructed by neural networks: Kernelized Stein Discrepancy Neural Sampler (KSD-NS) and Fisher Divergence Neural Sampler (Fisher-NS).

[*]Department of Statistics, Purdue University
[†]Department of Statistics, Tsinghua University

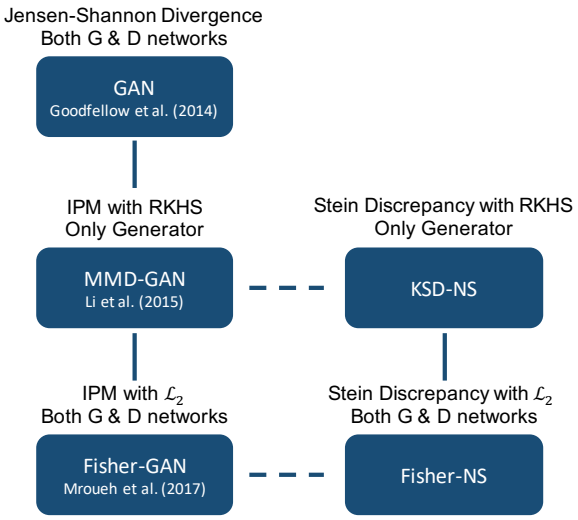

Figure 1: Summary of Our Proposed Approaches and the Relationship to Existing GAN Models

1. KSD-NS: The sampler is trained by directly minimizing the Kernelized Stein Discrepancy (KSD) between the generated samples and the target density. KSD as loss can be interpreted as a one-sample goodness-of-fit testing statistic. Our training procedure is based on an estimator of KSD with the mini-batch error bound of $\mathcal{O}\left(1/\sqrt{n}\right)$.
2. Fisher-NS: KSD distinguish two distributions based on discriminator function in Reproducing Kernel Hilbert Space (RKHS), which is a relatively small function space. To enhance the power of the sampler, we expand the space of discriminator function from RKHS to $\mathcal{L}_2$, where functions are represented by $\mathcal{L}_2$-regularized deep neural networks. This training scheme optimally corresponds to minimizing the Fisher divergence, which is stronger than KSD. This extension also leads to a better empirical performance.

The proposed schemes aim to train a transformation that is asymptotically correct and enables instantaneous sampling. Extensive empirical studies conducted in section 6 demonstrates that our neural sampler suffers less convergence issue such as local trap.

2 Background

Stein's Identity Let $q(x)$ be a continuously differentiable density supported on $\mathcal{X} \subseteq \mathbb{R}^d$ and $\boldsymbol{f}(x) = [f_1(x), \cdots, f_d(x)]^\top$ be a smooth vector function satisfying some mild boundary conditions. Then, we have the following Stein's identity:

$$\mathbb{E}_{x\sim q}\left[\nabla_x \log q(x)\boldsymbol{f}(x)^\top + \nabla_x \boldsymbol{f}(x)\right] = 0, \tag{2.1}$$

Denote $S_q(x) = \nabla_x \log q(x)$, which is usually referred as the score function of $q(x)$. Note that calculating $S_q(x)$ does not require the normalization constant in $q(x)$, which is often intractable in practice. This property makes Stein's identity an ideal tool for handling un-normalized distributions to be sampled in machine learning and statistics.

Stein Discrepancy Let $p(x)$ be another smooth density supported on $\mathcal{X}$ and if we replace the expectation $\mathbb{E}_q[\cdot]$ in (2.1) with $\mathbb{E}_p[\cdot]$, the equality will not hold in general. This property naturally induces a distance between two densities $p(x)$ and $q(x)$ by optimizing the right hand side of (2.1) over all functions $\boldsymbol{f}$ within a function space $\mathcal{F}$,

$$\mathcal{D}(p, q, \mathcal{F}) = \sup_{\boldsymbol{f}\in\mathcal{F}} \left\{\mathbb{E}_{x\sim p} \operatorname{tr}\left(S_q(x)\boldsymbol{f}(x)^\top + \nabla_x \boldsymbol{f}(x)\right)\right\}, \tag{2.2}$$

If $\mathcal{F}$ is large enough, $\mathcal{D}(p, q, \mathcal{F}) = 0$ if and only if $p = q$. However, $\mathcal{F}$ cannot be too large. Otherwise, $\mathcal{D}(p, q, \mathcal{F}) = \infty$ for any $p \neq q$.

Kernelized Stein Discrepancy If the function space $\mathcal{F} = \left\{\boldsymbol{f} \in \mathcal{H}^d \,\middle|\, ||\boldsymbol{f}||_{\mathcal{H}^d} = 1\right\}$ is a unit ball in an RKHS $\mathcal{H}^d$, with $k(\cdot, \cdot)$ being the associated positive definite kernel function, then the supremum in (2.2) has a closed form solution

(Liu et al., 2016),

$$\mathrm{KSD}(p, q) = \mathbb{E}_{x,x' \sim p}\left[u_q(x, x')\right]$$

where

$$\begin{aligned} u_q(x, x') = \; & S_q(x)^\top k(x, x') S_q(x') \\ & + S_q(x)^\top \nabla_x k(x, x') + \nabla_x k(x, x')^\top S_q(x') \\ & + \mathrm{tr}(\nabla_{x,x'} k(x, x')). \end{aligned} \tag{2.3}$$

The corresponding optimal discriminative function $\boldsymbol{f}^*$ satisfies $||\boldsymbol{f}^*||_{\mathcal{H}^d} = 1$ and

$$\boldsymbol{f}^*(\cdot) \propto \mathbb{E}_{x\sim p}\left[S_q(x)k(x, \cdot) + \nabla_x k(x, \cdot)\right],$$

Empirical KSD measures the goodness-of-fit of a given sample $X = \{x_1, \cdots, x_n\}$ to the target density $q(x)$. The minimum variance unbiased estimator can be written as

$$\widehat{\mathrm{KSD}}(p, q) = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq i}^{n} [u_q(x_i, x_j)] \tag{2.4}$$

Despite the ease of computation, RKHS is relatively small and may fail to detect non-convergence in higher dimensions (Gorham and Mackey, 2017).

Integral Probability Metrics (IPM) IPM measures the distance between two distributions p and q via the largest discrepancy in expectation over a class of well behaved witness functions $\mathcal{F}$:

$$\mathrm{IPM}\left(p, q, \mathcal{F}\right) = \sup_{f \in \mathcal{F}} \left\{\mathbb{E}_{x\sim p}\left[f(x)\right] - \mathbb{E}_{x\sim q}\left[f(x)\right]\right\}. \tag{2.5}$$

Note that the function space $\mathcal{F}$ needs to have some constraint. Otherwise the corresponding IPM will be trivial with value 0 if $p = q$ and ∞ if $p \neq q$. A broad class of distances can be viewed as special cases of IPM (Müller, 1997). For instance, if we choose all the functions whose integration under q is zero, then we get Stein discrepancy (Gorham and Mackey, 2017).

3 Related Work

SVGD Given an initial distribution $p_0(x)$, the idea of SVGD is to learn a nonlinear transformation T such that the distribution of $T(x)$ approximates the target distribution q in the sense of Kullback-Leibler (KL) divergence. However, directly learning the nonlinear transformation T is difficult and SVGD circumvents this difficulty by constructing T with incremental linear updates $x' = x + \epsilon \boldsymbol{f}(x)$. The key observation is that

$$\nabla_\epsilon \mathrm{KL}(p' \,||\, q)\Big|_{\epsilon=0} = -\,\mathbb{E}_{x\sim p}[\mathrm{tr}(S_q \boldsymbol{f}^\top(x) + \nabla_x \boldsymbol{f}(x))] \tag{3.1}$$

where $x \sim p$ and $x' \sim p'$. Confining $\boldsymbol{f}$ inside a unit ball of RKHS gives the optimal $\boldsymbol{f}^*$ in a closed form, in a similar spirit to KSD.

However, one weakness of SVGD is that the nonlinear transformation T can not be stored in any form after finishing an SVGD chain. As a consequence, we have to run extra SVGD chains when extra samples are needed. In comparison, our proposed neural sampler learns a preservable nonlinear transformation T trained by neural networks. In this way, our framework is able to generate new samples without additional efforts once the transformation is learned.

Generative Adversarial Network GAN also learns to transform random noises to high-quality samples. Its objective is to implicitly capture the underlying distribution of given samples by building a generator to sample from it. In GANs, the generator is constructed using deep neural networks and trained adversarially with another discriminator network. The discriminator takes both true samples and generated samples as input and essentially conducts two sample test between them. The min-max game between the two networks potentially corresponds to minimizing the Jenson-Shannon divergence in the vanilla GAN (Goodfellow et al., 2014). Other choices of divergence lead to variants of GAN such as Maximum Mean Discrepancy (MMD) (Li et al., 2015), Wasserstein distance (Arjovsky et al., 2017), Chi-squared distance (Mroueh and Sercu, 2017), etc. The aforementioned distances can all be seen as special cases of IPM; see an overview by Mroueh et al. (2017).

Fusion of GANs and Sampling The fusion of deep learning and sampling is not new. Song et al. (2017) proposed A-NICE-MC, where the proposal distribution in MCMC is, instead of domain-agnostic, adversarially trained using neural networks. Stein GAN (Wang and Liu, 2016) also proposes to train a neural network to draw samples from given target distributions for probabilistic inference. Their method is by iteratively adjusting the weights according to the SVGD updates. From a GAN perspective, in each iteration of Stein GAN, the discriminator is performing a two sample test between the currently generated samples and the one-step updated samples by SVGD. This method generalized SVGD to training neural networks and it is minimizing the Kullback-Leibler divergence between the sampling distribution and the target inside a RKHS.

Although Stein GAN shares a similar objective with our proposed neural samplers, two approaches are fundamentally different. Instead of KL divergence, we incorporate Stein discrepancy, a special case of Integral Probability Metrics, which serves as a bridging tool between true samples and true density. This enables various frameworks in IPM based GAN to be directly developed in parallel (Figure 1).

4 KSD Neural Sampler

Let $q(x)$ denote the un-normalized target density with support on $\mathcal{X} \subset \mathbb{R}^d$ and $Q(x)$ be the corresponding distribution function. Denote the noise by z with density $p_z(z)$ supported on $\mathbb{R}^{d_0}$. Let G_θ denote our sampler, which is a multi-layer neural network parametrized by θ. Let $x = G_\theta(z)$ be our generated samples and denote its underlying density by $p_\theta(x)$. In summary, our setup is as follows:

$$z \sim p_z(z), \quad G_\theta(z) = x \sim p_\theta(x)$$

We want to train the network parameters θ so that $p_\theta(x)$ is a good approximation to the target $q(x)$.

4.1 Methodology

Evaluating how close is the generated samples $X = \{G_\theta(z_i)\}_{i=1}^n$ to the target $q(x)$ is equivalent to conducting one-sample goodness-of-fit test. When $q(x)$ is un-normalized, one well-defined testing framework is based on kernelized Stein discrepancy.

KSD is the counterpart of maximum mean discrepancy (MMD) in two-sample test (Gretton et al., 2012). By choosing $\mathcal{F}$ in IPM (2.5) to be an unit-ball in RKHS, Li et al. (2015) proposed MMD-GAN, which simplifies the GAN framework by eliminating the need of training a discriminator network. As a result, MMD-GAN is more stable and easier to train.

Motivated by MMD-GAN, we propose to train our neural sampler G_θ by directly minimizing KSD with respect to θ using gradient-based optimization. At each iteration, we sample a batch of noise $\{z_1, \cdots, z_n\} \sim p_z(z)$ and calculate the corresponding samples $\{G_\theta(z_1), \cdots, G_\theta(z_n)\}$. Plugging in the samples to formula (2.4) gives the empirical KSD estimator, which can serve as an indicator of how well our current samples are approximating $q(x)$. Iteratively updating θ to minimize the empirical KSD until convergence. Algorithm 1 summarizes our training procedure.

Algorithm 1 KSD-NS

1: **Input:** un-normalized density $q(x)$, noise density $p_z(z)$, number of iterations T, learning rate α, mini-batch size n.
2: **Initialize** parameter θ for the generator network.
3: **For** iteration $t = 1, \ldots, T$, **Do**
4: Generate i.i.d. noise inputs $z_1, \ldots, z_n$ from $N(0, I_{d_0})$
5: Obtain fake sample $G_\theta(z_1), \cdots, G_\theta(z_n)$
6: Compute empirical $\widehat{\mathrm{KSD}}(p_\theta, q)$
7: Compute gradients $\nabla_\theta \widehat{\mathrm{KSD}}(p_\theta, q)$
8: update $\theta \leftarrow \theta - \alpha \nabla_\theta \widehat{\mathrm{KSD}}(p_\theta, q)$
9: **End For**

Comparing to Stein GAN, KSD-NS has the following advantages:

1. While the loss of Stein GAN is not interpretable, the loss in our KSD-NS directly shows the sample quality. KSD is always non-negative and a smaller KSD indicates a better sample quality.
2. We show KSD-NS is theoretically sound in section 4.2: with sufficient batch size, empirical KSD loss converging to zero implies weak convergence of the sampling distribution.

3. Even though both methods used RKHS and kernel trick, empirical results show that our KSD-NS tends to capture better global structures and less likely to drop mode. Stein GAN is more sensitive to initialization and suffers local trap more severely.

4.2 Mini-Batch Error Bound

The optimization described above involves evaluating the expectation under p_θ and it is approximated by the mini-batch sample mean. Natural questions to ask include when the empirical KSD is minimized and what we can say about the population KSD? In the following, we demonstrate that the generalization error is bounded when mini-batch sample size is sufficiently large.

Let $X_\theta = \{x_1, \cdots, x_n\}$ be the generated samples from our generator $G_\theta(\cdot)$ with Θ being the parameter space. Denote $\widehat{\theta}$ and θ^* as the value minimizing the empirical KSD and population KSD:

$$\widehat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmin}} \widehat{\mathrm{KSD}}(X_\theta, q),$$
$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}} \mathrm{KSD}(p_\theta, q),$$

We are interested in bounding the difference

$$\widehat{\mathrm{KSD}}\left(p_{\widehat{\theta}}, q\right) - \mathrm{KSD}\left(p_{\theta^*}, q\right),$$

whose upper bound is given in the following theorem.

Theorem 4.1. Assume q and $k(\cdot, \cdot)$ satisfy some smoothness conditions so that the newly defined kernel u_q in (2.3) is L_1-Lipschitz with one of the arguments fixed. Under some norm constraints on the weight matrix of each layer of the generator G_θ, then for any $\epsilon > 0$, the following bound holds with probability at least $\exp(-\epsilon^2 n/2)$,

$$\mathrm{KSD}\left(p_{\theta^*}, q\right) \leq \widehat{\mathrm{KSD}}\left(p_{\widehat{\theta}}, q\right) + \mathcal{O}\left(\frac{C_d}{\sqrt{n}}\right) + \epsilon,$$

where C_d is a function of the dimension d.

Remark The norm constraints on neural networks in the above theorem require the norm of each weight matrices to be bounded. The usual norms used are Frobenius norm, $W_{p,q}$ norm and other matrix norms. More details about the conditions and the results are in the supplementary material.

Theorem 4.1 implies that in practice, with enough batch size, the generator G_θ can be trusted if we observe a small KSD loss. But we want to raise the following two points.

1. When KSD is small, what we can tell is that in the support of the samples, the score function of p_θ matches the target S_q well. An almost-zero KSD doesn't necessarily imply p_θ captures all the modes or recovers all the support of the true density. Admittedly, local trap is a common problem across various sampling methods, but our KSD-NS demonstrates strong resistance to this issue in simulations.
2. KSDs based on the commonly used kernels, such as Gaussian kernel, Matern kernel, fail to detect non-convergence when $d \geq 3$ (Gorham and Mackey, 2017). However, KSD used in our neural sampler is exempt from such curse of dimensionality and we show that with some mild constraints, convergence to zero of KSD-NS does imply weak convergence of p_θ to q.

4.3 Metrization of Weak Convergence

The issue of KSD with Gaussian kernel in higher dimensions can be traced back to the fast decaying kernel function. If we choose a heavy-tail kernel, such as Inverse Multi-Quadratic (IMQ) kernel, the corresponding KSD can detect non-convergence. The following theorem is from Gorham and Mackey (2017).

Theorem 4.2. Under IMQ kernel $k(x, y) = \left(c^2 + ||x - y||_2^2\right)^\beta$ where $c > 0$ and $\beta \in (-1, 0)$, $\mathrm{KSD}(p_\theta, q) \to 0$ implies $p_\theta \xrightarrow{d} q$.

The above theorem shows that IMQ KSD detects non-convergence. Since IMQ is a bounded kernel, the corresponding KSD is well-defined as long as $F(p_\theta, q) < \infty$. If we use Gaussian kernel or other popular kernels, we can still ensure weak convergence if we enforce uniformly tightness (Merolla et al., 2016). One simple approach is through weight clipping of the generator, see appendix for details.

If we choose the appropriate kernel and prevent our generated samples from going to infinity, the KSD-NS is theoretically sound. However, in practice, the performance of our model usually deteriorate as dimension goes higher. In the next section, we introduce the Fisher divergence neural sampler, which expands RKHS to $\mathcal{L}_2$ space to better deal with the curse of dimensionality.

5 Fisher Neural Sampler

The ease of computation for kernel methods does not come free. RKHS is a relatively small function space and the expressive power of kernel function in RKHS decays when dimension goes higher. In generating images, empirical performance of MMD-GAN (Li et al., 2015) is usually not comparable to more computationally intensive GANs like Wasserstein GAN (Arjovsky et al., 2017; Gulrajani et al., 2017).

5.1 Methodology

Instead of an unit-ball in RKHS, we choose the function space $\mathcal{F}$ in Stein discrepancy (2.2) to be $\mathcal{L}_2$. Next, we approximate $\mathcal{L}_2$ functions by another multi-layer neural network $\boldsymbol{f}_\eta(x)$:

$$\mathcal{D}_\eta(p_\theta, q) = \sup_\eta \left\{ \mathbb{E}_{x\sim p_\theta} \operatorname{tr}\left(S_q(x) \boldsymbol{f}_\eta(x)^\top + \nabla_x \boldsymbol{f}_\eta(x) \right) \right\}.$$

Neural networks as functions are not square integrable by nature, since they don't vanish at infinity by default. To impose the $\mathcal{L}_2$ constraint, we add an $\mathcal{L}_2$ penalty term and thus our loss function becomes

$$L_{\eta,\lambda}(p_\theta, q) = \mathcal{D}_\eta(p_\theta, q) - \lambda \, \mathbb{E}_{x\sim p_\theta} \left[\boldsymbol{f}_\eta^\top \boldsymbol{f}_\eta \right]$$

where λ is a tuning parameter. Our training objective is

$$\min_\theta \max_\eta L_{\eta,\lambda}(p_\theta, q)$$

The *ideal* training scheme is:

step 1 Initialize generator network G_θ and the discriminator network f_η

step 2 Fix θ, train η to *optimal*

step 3 Fix η, train θ with one step

step 3 Repeat step 2 and 3 until *convergence*

The ideal part mainly refers to training the discriminator to optimal and the discriminator itself has large enough capacity. The proposed training scheme is similar to that in Wasserstein GAN (Arjovsky et al., 2017) and Fisher GAN (Mroueh and Sercu, 2017). Under the optimality assumptions, we now show the extension from RKHS to $\mathcal{L}_2$ indeed introduces a stronger mode of convergence.

5.2 Optimal Discriminator

The Fisher divergence between two densities p and q is defined as

$$\mathrm{F}(p \,||\, q) = \mathbb{E}_{x\sim p} \left[||\nabla_x \log(p) - \nabla_x \log(q)||_2^2 \right].$$

We now show that Fisher divergence is the corresponding loss of our ideal training scheme, provided that the discriminator network has enough capacity.

Theorem 5.1. The optimal discriminator function is

$$\frac{1}{2\lambda} \left(S_q(x) - S_p(x) \right).$$

Training the generator with the optimal discriminator corresponds to minimizing the fisher divergence between p_θ and q. The corresponding optimal loss for training θ is

$$L(\theta) = \frac{1}{4\lambda} \mathbb{E}_{x\sim p_\theta} ||S_q(x) - S_{p_\theta}(x)||_2^2.$$

One observation is that when our sampling distribution p_θ is close to the target q, the discriminator function f_η tends to zero. Naturally, f_η can be used as an diagnostic tool to evaluate how well our neural sampler is working.

Fisher Divergence vs. KSD Fisher divergence dominates KSD in the following sense (Liu et al., 2016):

$$\mathrm{KSD}(p,q) \leq \sqrt{\mathbb{E}_{x,x'\sim p}[k(x,x')^2]} \cdot \mathrm{F}(p||q).$$

Fisher divergence is stronger than KSD, and lot of other distances between distributions, such as total variation, Hellinger distance, Wasserstein distance, etc (Ley et al., 2013).

Fisher Divergence vs. KL Divergence KL divergence is not symmetric and usually not stable for optimization due to its division format, while KSD and Fisher divergence are more robust in contrast. Under mild conditions, according to Sobolev inequality, Fisher divergence is a stronger distance than KL divergence, which serves as the objective distance in SVGD. It implies that, theoretically, our framework has higher potentiality than SVGD.

In SVGD or Stein GAN, the normalizing constant is unknown so it is hard to quantify how well the KL divergence is being minimized. In comparison, both KSD and Fisher divergence only rely on the score function and hence, the values are directly interpretable as goodness-of-fit test statistics.

Remark The optimality assumption on discriminator may seem unrealistic. However,

1. Optimality of discriminator is an usual assumption for all GAN models mentioned in this paper. Optimization in deep neural networks are highly non-convex and the mini-max game in GAN model is extremely hard to characterize. Losing the assumption require tremendous amount of work (Arora et al., 2017).
2. Many results suggest that deep neural networks with large capacity usually generalize well. Bad local minimum is scarce and more efficient optimization tools to escape saddle points are being developed (Kawaguchi, 2016; LeCun et al., 2015; Jin et al., 2017).

In practice, we suggest choosing a large enough discriminator network and after each iteration of θ, we train η for 5 times, as suggested in Wasserstein GAN (Arjovsky et al., 2017). Algorithm 2 summarizes our training procedure.

Algorithm 2 Fisher-NS

Input: un-normalized density $q(x)$, noise density $p_z(z)$, number of step 2 iterations m, number for step 4 iterations T, tuning parameter λ, learning rate α_1, α_2, mini-batch size n.
Initialize parameter θ and η for both neural networks.
For iteration $t = 1, \ldots, T$, **Do**
  Generate i.i.d. noise inputs $z_1, \ldots, z_n$ from $N(0, I_d)$
  Obtain fake sample $G_\theta(z_1), \cdots, G_\theta(z_n)$
  For $h = 1, \ldots, m$, **Do**
    Compute empirical loss $L_{\eta,\lambda}(p_\theta, q)$
    Compute gradient $\nabla_\eta L_{\eta,\lambda}(p_\theta, q)$
    $\eta \leftarrow \eta + \alpha_1 \nabla_\eta L_{\eta,\lambda}(p_\theta, q)$
  End For
  Compute empirical loss $L_{\eta,\lambda}(p_\theta, q)$
  Compute gradient $\nabla_\theta L_{\eta,\lambda}(p_\theta, q)$
  $\theta \leftarrow \theta - \alpha_2 \nabla_\theta L_{\eta,\lambda}(p_\theta, q)$
End For

5.3 Training the Generator

After the training cycle for the discriminator, we fix η and train the generator G_θ. Denote the loss function to be $L(\theta)$ and ideally, we would want $L(\theta)$ to be continuous with respect to θ. Wasserstein GAN (Arjovsky et al., 2017) gives a very intuitive explanation of the importance of this continuity. We now give some sufficient conditions, under which our training scheme satisfies the continuity condition with respect to θ for any discriminator function f_η.

Theorem 5.2. If the following conditions are satisfied: 1) both the generator's weights and the noises are bounded, 2) discriminator uses smooth activate function i.e. tanh, sigmoid, etc., 3) target score function s_q is continuously differentiable. Then $L(\theta)$ is continuous everywhere and differentiable almost everywhere w.r.t θ.

Remark These conditions are to impose some Lipschitz continuity (details in appendix). The first condition is trivially satisfied if we choose uniform as random noise and apply weight clipping to the generator. Except for θ being bounded, the other conditions are mild. It is true that procedures like weight clipping will make the function space smaller. But we can make the clipping range large enough to reach a fixed accuracy (Merolla et al., 2016). The empirical difference should be negligible if the range is sufficiently large.

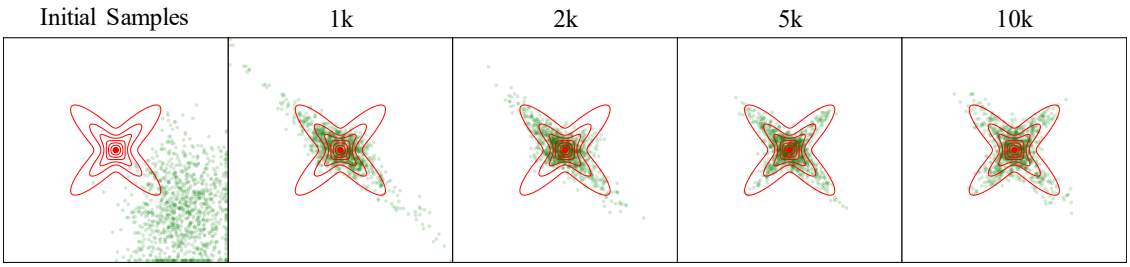

Figure 2: Toy example with 2D Gaussian mixture. Trained with KSD. The red contours represent the target distribution and the green dots are generated samples. From left to right are the initialization, 1k, 2k, 5k, 10k, iterations correspondingly.

Table 1: MSE of h_1 and h_2 and MMD of each method in 30 independent runs.

	MSE of h_1	MSE of h_2	MMD
KSD-NS	0.3410	0.4258	0.004737
Fisher-NS	17.02	4.726	0.02140
SteinGAN	220.23	182.5	0.2928
SVGD	229.2	115.3	0.1992
LD	224.7	201.0	0.5217

6 Experiments

We test our neural samplers on both toy examples and real world problems. We compare them to Stein GAN, SVGD and other commonly used sampling methods such as Langevin dynamics (LD) and variational inference method. From the simulation results on Gaussian mixtures, our methods demonstrate superior ability to handle **multimodality** as well as avoiding **local trap** compared to the benchmarking methods. When applied to real world data, which is high dimensional, our method shows comparable performance. All the experiment details are attached in the Appendix.

Gaussian Mixtures We start with a toy example to illustrate how our sampler transform a reference distribution to match the target. let the target distribution to be a 2-dimensional mixture normal

$$q(x) = 0.5 \cdot \mathcal{N}(x; 0, I_2(0.8)) + 0.5 \cdot \mathcal{N}(x; 0, I_2(-0.8))$$

where $\mathcal{N}(x; \mu, \Sigma)$ denotes the density function of $N(\mu, \Sigma)$ and $I_2(\rho)$ denotes 2-dimension identity matrix with ρ as off-diagonal elements. Figure [2] shows the how the sampling distribution evolves during the training.

Above is a simple case where the local trap phenomenon would not occur and our method successfully captures the detailed shape of the distribution. However, often times in practice, the target distribution is multimodal which usually causes the sampling method to be trapped in certain modes.

In order to demonstrate the capability of our method in escaping the local mode and exploring the global space, in the next example, we consider a 2-dimensional Gaussian mixture model with modes far from each other. Specifically, the target is a mixture of 8 standard Gaussian components equally spaced on a circle of radius 15 with equal mixing weights. To make the task more difficult, we set the initial particles to be far away from the true modes. For fair comparison, the network configurations for Stein GAN, KSD-NS and Fisher-NS are exactly the same. Figure [3] shows the contour of the target distribution and the evolution of the particles of each compared method. The result suggests that the proposed method is far more powerful in exploring the global structure.

Consider $X = (X_1, X_2)^\top \sim q$ with $\mathbb{E}(X) = \mu = (\mu_1, \mu_2)^\top$. To measure the quality of the sample quantitatively, we consider estimating the following statistics based the particles generated by each method.

$$h_1 = \mathbb{E}(X_1) + \mathbb{E}(X_1) \quad \text{and} \quad h_2 = \sqrt{\mathbb{E}(X_1 - \mu_1)^2} + \sqrt{\mathbb{E}(X_2 - \mu_2)^2}$$

We run 30 independent runs of each method and compared the mean square error in estimating the two quantities. We also include the average of the estimated Maximum Mean Discrepancy (MMD) (Gretton et al., 2012) in 30 runs to measure the sample quality. Table [1] summarizes the results. All the quantities are the smaller, the better.

Bayesian Logistic Regression on Covertype Data We apply Stein neural sampler to Bayesian logistic regression model for binary classification and test our methods on the Covertype dataset. The dataset has 581,012 observations,

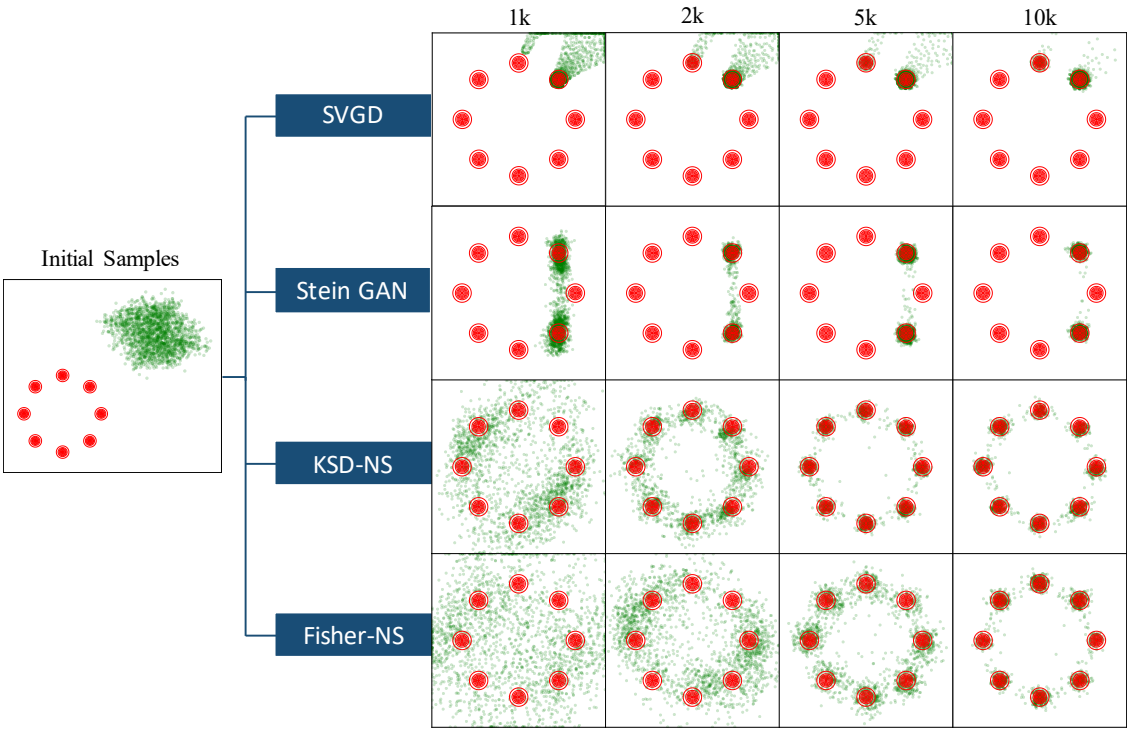

Figure 3: 2D Gaussian mixture with 8 components. The red dashed lines are the target density function and the green dots are generated samples at 0, 1000, 2000, 5000, 10000 iterations respectively.

Table 2: Test Accuracy for Covertype.

	SVGD	SGLD	DSVI	Fisher-NS	SteinGAN
Accuracy	74.57%	74.90%	73.69%	75.35%	75.33%

54 features and a binary response. We use the same setting as in Wang and Liu (2016), where we randomly split the whole dataset into the training set and the testing set by a ratio of 4:1. We compare our method to Stein GAN, SVGD, stochastic gradient Langevin dynamics (SGLD) Welling and Teh (2011) and doubly stochastic variational inference (DSVI) Titsias and Lazaro-Gredilla (2014). Table [2] reports the classification accuracy on test set.

7 Conclusion

In this paper, we propose two novel frameworks that directly learns preservable transformations from random noise to target distributions. KSD-NS enjoys theoretical guarantee and demonstrates strong ability to capture multi-modal distributions. Fisher-NS extends KSD-NS and potentially achieves convergence with respect to Fisher divergence.

The introduction of GAN to sampling is exciting. Neural network as generator has great capacity to model the transformation and the adversarial training can optimally correspond to minimizing all kinds of distances between distributions. Using Stein discrepancy as a bridge, numerous variants of GAN and their related techniques can be potentially applied in parallel to sampling.

References

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Arora, S., Ge, R., Liang, Y., Ma, T., and Zhang, Y. (2017). Generalization and equilibrium in generative adversarial nets (gans). *arXiv preprint arXiv:1703.00573*.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

Brock, A., Donahue, J., and Simonyan, K. (2018). Large Scale GAN Training for High Fidelity Natural Image Synthesis. *ArXiv e-prints*.

Durán, R. G. and López García, F. (2010). Solutions of the divergence and analysis of the stokes equations in planar hölder-α domains. *Mathematical Models and Methods in Applied Sciences*, 20(01):95–120.

Fasshauer, G. E. (2011). Positive definite kernels: past, present and future. *Dolomite Research Notes on Approximation*, 4:21–63.

Gamerman, D. and Lopes, H. F. (2006). *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*. Chapman and Hall/CRC.

Golowich, N., Rakhlin, A., and Shamir, O. (2017). Size-independent sample complexity of neural networks. *arXiv preprint arXiv:1712.06541*.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems*.

Gorham, J. and Mackey, L. (2017). Measuring sample quality with kernels. *arXiv preprint arXiv:1703.01717*.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Scholkopf, B., and Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30.

Jin, C., Ge, R., Netrapalli, P., Kakade, S. M., and Jordan, M. I. (2017). How to escape saddle points efficiently. *arXiv preprint arXiv:1703.00887*.

Kawaguchi, K. (2016). Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*, pages 586–594.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

Ley, C., Swan, Y., et al. (2013). Stein's density approach and information inequalities. *Electronic Communications in Probability*, 18.

Li, Y., Swersky, K., and Zemel, R. (2015). Generative moment matching networks. In *International Conference on Machine Learning*.

Liu, Q., Lee, J., and Jordan, M. (2016). A kernelized stein discrepancy for goodness-of-fit tests. In *International Conference on Machine Learning*, pages 276–284.

Liu, Q. and Wang, D. (2016). Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances In Neural Information Processing Systems*, pages 2378–2386.

Mendelson, S. (2003). A few notes on statistical learning theory. In *Advanced lectures on machine learning*, pages 1–40. Springer.

Merolla, P., Appuswamy, R., Arthur, J., Esser, S. K., and Modha, D. (2016). Deep neural networks are robust to weight binarization and other non-linear distortions. *arXiv preprint arXiv:1606.01981*.

Mroueh, Y., Li, C.-L., Sercu, T., Raj, A., and Cheng, Y. (2017). Sobolev gan. *arXiv preprint arXiv:1711.04894*.

Mroueh, Y. and Sercu, T. (2017). Fisher gan. In *Advances in Neural Information Processing Systems*, pages 2513–2523.

Müller, A. (1997). Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., and Sohl-Dickstein, J. (2016). On the expressive power of deep neural networks. *arXiv preprint arXiv:1606.05336*.

Song, J., Zhao, S., and Ermon, S. (2017). A-nice-mc: Adversarial training for mcmc. In *Advances in Neural Information Processing Systems*, pages 5140–5150.

Titsias, M. and Lazaro-Gredilla, M. (2014). Doubly stochastic variational bayes for non-conjugate inference. *ICML*.

van de Geer, S. (2016). Symmetrization, contraction and concentration. In *Estimation and Testing Under Sparsity*, pages 233–238. Springer.

Villani, C. (2008). *Optimal transport: old and new*, volume 338. Springer Science & Business Media.

Wang, D. and Liu, Q. (2016). Learning to draw samples: With application to amortized mle for generative adversarial learning. *arXiv preprint arXiv:1611.01722*.

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. *ICML*.

APPENDIX

A.1 Proof of Theorem 4.1

Lemma A.1. (Theorem 3.7 from Liu et al. (2016)) Assume $k(x, x')$ is a positive definite kernel in the Stein class of p, with positive eigenvalues λ_j and eigenfunctions $e_j(x)$, then $u_q(x, x')$ is also a positive definite kernel, and can be rewritten into

$$u_q(x, x') = \sum_j \lambda_j [\mathcal{A}_q e_j(x)]^\top [\mathcal{A}_q e_j(x')], \tag{A.1}$$

where $\mathcal{A}_q$ is the Stein operator acted on e_j that

$$\mathcal{A}_q(f) = \nabla \log q(x) \cdot f + \nabla f \tag{A.2}$$

Gaussian Kernel(Fasshauer, 2011) Gaussian kernel is a popular characteristic kernel written as

$$k(x, x') = \exp(-\frac{||x - x'||^2}{2\sigma^2})$$

Its eigenexpansion is

$$\lambda_j \propto b^j, \quad b < 1 \tag{A.3}$$

$$e_j(x) \propto \frac{\exp(-a||x||^2)}{\sqrt{2^j j!}} \prod_{i=1}^{d} H_j(x_i\sqrt{2c}) \tag{A.4}$$

where $a, b, c > 0$ are some constants depending on σ, and H_k is k-th order Hermite polynomial. The eigenfunctions are $L2$-orthonorm. For details, please refer to section 6.2 of (Fasshauer, 2011).

Lemma A.2. (McDiarmids inequality, Mendelson (2003)) Let $X_1, \cdots, X_n \in \mathcal{X}$ be independent random variables and let $f : \mathcal{X}^n \to \mathbb{R}$ be a function of $X_1, \cdots, X_n$. Assume there exists $c_1, \cdots, c_n \geq 0$ such that $\forall i, x_1, \cdots, x_n, x_i' \in \mathcal{X}$,

$$|f(x_1, \cdots, x_i, \cdots, x_n) - f(x_1, \cdots, x_i', \cdots, x_n)| \leq c_i$$

Then, for all $\epsilon > 0$,

$$\mathbb{P}\left(f - \mathbb{E}(f) \geq \epsilon\right) \leq \exp\left(-\frac{2\epsilon}{\sum_{i=1}^n c_i^2}\right)$$

Lemma A.3. (Norm-based Sample Complexity Control (Golowich et al. (2017))) Let $\mathcal{H}_d$ be the class of real-valued neural networks of depth D over domain $\mathcal{Z}$, where each weight matrix W_j has Frobenius norm at most $M_F(j)$. Let the activation function be 1-Lipschitz, positive-homogeneous (such as the ReLU). Denote $\widehat{\mathfrak{R}}_n(\mathcal{H})$ to be the empirical Rademacher complexity of $\mathcal{H}$. Then,

$$\widehat{\mathfrak{R}}_n(\mathcal{H}_d) \leq \frac{B\left(\sqrt{2D\log 2} + 1\right)\prod_{j=1}^D M_F(j)}{\sqrt{n}}$$

where $B > 0$ is the range of the input distribution such that $||z|| \leq B$ almost surely.

Lemma A.4. (Extension of Ledoux-Talagrand contraction inequality (van de Geer, 2016)) Let $u : \mathbb{R}^d \to \mathbb{R}$ be L-Lipschitz functions w.r.t. L_1 norm, i.e. $\forall x, y \in \mathbb{R}^d, |u(x) - u(y)| \leq L||x - y||_1$. For some function space $\mathcal{F} = \{f = (g_1(x), \cdots, g_d(x))^\top : \mathbb{R}^{d_0} \to \mathbb{R}^d\}$, denote $\mathcal{F}_i = \{g_i(x) : \mathbb{R}^{d_0} \to \mathbb{R}\}$ for $i = 1, 2, \cdots, d$, accordingly. Then,

$$\widehat{\mathfrak{R}}_n(u \circ \mathcal{F}) \leq 2^{d-1} L \sum_{i=1}^d \widehat{\mathfrak{R}}_n(\mathcal{F}_i)$$

where $\circ$ means composition and $u \circ \mathcal{F} = \{u \circ f : f \in \mathcal{F}\}$.

Theorem 4.1 Assume q and $k(\cdot, \cdot)$ satisfy some smoothness conditions so that the newly defined kernel u_q in lemma A.1 is L_1-Lipschitz with one of the argument fixed. If generator G_θ satisfy the conditions in A.3. Then, For any $\epsilon > 0$, with probability at least $\exp(-\epsilon^2 n/2)$ the following bound holds,

$$\text{KSD}(p_{\theta^*}, q) \leq \widehat{\text{KSD}}(p_{\hat{\theta}}, q) + \mathcal{O}\left(\frac{2^d dB\sqrt{D}\prod_{j=1}^D M_F(j)}{\sqrt{n}}\right) + \epsilon \tag{A.5}$$

Proof. For the ease of notation, let's denote

$$\mathcal{E}(\theta) = \widehat{\mathrm{KSD}}(X_\theta, q), \quad \mathcal{T}(\theta) = \mathrm{KSD}(p_\theta, q)$$

By applying the large deviation bound on U-statistics of (Hoeffding, 1963), we have that for any $\theta \in \Theta$

$$\mathbb{P}\left(|\mathcal{E}(\theta) - \mathbb{E}(\mathcal{E}(\theta))| > \epsilon\right) \leq 2\exp\left(-\frac{\epsilon^2 n}{16}\right) \tag{A.6}$$

Note that (A.6) holds for any fixed θ. Since θ^* is the population MMD minimizer that doesn't depend on samples, we have $\mathbb{E}(\mathcal{E}(\theta^*)) = \mathcal{T}(\theta^*)$, which yields

$$\mathbb{P}\left(|\mathcal{E}(\theta^*) - \mathcal{T}(\theta^*)| > \epsilon\right) \leq 2\exp\left(-\frac{\epsilon^2 n}{16}\right)$$

On the other hand, $\widehat{\theta}$ is the empirical MMD minimizer and to bound it, we want to show that for some β_ϵ s.t.

$$\mathbb{P}(\sup_\theta |\mathcal{E}(\theta) - \mathcal{T}(\theta)| > \epsilon) < \beta_\epsilon \tag{A.7}$$

Apply (2.4), we can write

$$\sup_\theta |\mathcal{E}(\theta) - T(\theta)| \leq \sup_\theta \left| \frac{1}{n(n-1)} \sum_{i=1}^n \sum_{j \neq i}^n u(X_i, X_j) - \mathbb{E}\left(u(X_i, X_j)\right) \right| \tag{A.8}$$

$$:= \sup_\theta I(X_\theta) \tag{A.9}$$

For $I(X_\theta)$, notice that the bounded condition for McDiarmid's inequality still holds that

$$|\sup_\theta I(X_\theta) - \sup_\theta I(X'_\theta)| \leq \sup_\theta |I(X_\theta) - I(X'_\theta)| \leq \frac{2}{n}$$

where X_θ and X'_θ only differ in one element. Then McDiarmid's inequality gives us that

$$\mathbb{P}\bigg(\sup_\theta I(X_\theta) - \mathbb{E}(\sup_\theta I(X_\theta)) > \epsilon \bigg) \leq \exp\left(-\frac{\epsilon^2 n}{2}\right) \tag{A.10}$$

With high probability, $\sup_\theta I(X_\theta)$ can be bounded by $\mathbb{E}(\sup_\theta I(X_\theta)) + \epsilon$. Now we give a bound for $\mathbb{E}(\sup_\theta I(X_\theta))$.

$$\begin{aligned}
\mathbb{E}\left(\sup_\theta I(X_\theta)\right) &= \mathbb{E}_{p_\theta}\left(\sup_\theta \left| \frac{1}{n(n-1)} \sum_{i=1}^n \sum_{j \neq i}^n u(X_i, X_j) - \mathbb{E}_{p_\theta}\left(u(X_i, X_j)\right) \right|\right) \\
&= \mathbb{E}_{p_\theta}\left(\sup_\theta \left| \frac{1}{n} \sum_{i=1}^n \frac{1}{n-1} \sum_{j \neq i}^n \Big(u(X_i, X_j) - \mathbb{E}_{p_\theta}\left(u(X_i, X_j)\right) \Big) \right|\right) \\
&\leq \frac{1}{n} \sum_{i=1}^n \mathbb{E}_{p_\theta}\left(\sup_\theta \left| \frac{1}{n-1} \Big(\sum_{j \neq i}^{n-1} u(X_i, X_j) - \mathbb{E}_{p_\theta}\left(u(X_i, X_j)\right) \Big) \right|\right) \\
&= \mathbb{E}_{p_\theta}\left(\sup_\theta \left| \frac{1}{n-1} \sum_{j=1}^{n-1} u(X_n, X_j) - \mathbb{E}_{p_\theta}\left(u(X_n, X_j)\right) \right|\right) \\
&= \mathbb{E}_{X_n}\left[\mathbb{E}_{p_\theta}\bigg(\sup_\theta \left| \frac{1}{n-1} \sum_{j=1}^{n-1} u(X_n, X_j) - \mathbb{E}_{p_\theta}\left(u(X_n, X_j)\right) \right| \bigg) \middle| X_n \right]
\end{aligned}$$

Once X_n is fixed, $u(X_n, X_j)$ for different j's are independent. By standard argument of Rademacher complexity, we have

$$\mathbb{E}_{p_\theta}\bigg(\sup_\theta \left| \frac{1}{n-1} \sum_{j=1}^{n-1} k(X_n, X_j) - \mathbb{E}_{p_\theta}\left(u(X_n, X_j)\right) \right| \bigg) \middle| X_n \quad \leq \quad 2\mathfrak{R}_{n-1}\Big(\mathcal{F}_{\theta, X_n}\Big) \tag{A.11}$$

where

$$\mathcal{F}_{\theta,X_n} = \{u(X_n, G_\theta(z)) : z \sim p_z \text{ and independent of } X_n\}$$

Combine the assumption of u_q being Lipschitz with lemma A.4, we have

$$\mathfrak{R}_n\Big(\mathcal{F}_{\theta,X_n}\Big) \leq 2^{d-1}\sum_{i=1}^{d}\mathfrak{R}_n(\mathcal{F}_{\theta,X_n,i}) \tag{A.12}$$

Applying lemma A.3 yields

$$2\mathfrak{R}_n\Big(\mathcal{F}_{\theta,X_n}\Big) \leq \frac{2^d dL\cdot B\left(\sqrt{2D\log 2}+1\right)\prod_{j=1}^{D}M_F(j)}{\sqrt{n}} := \eta(n,d) \tag{A.13}$$

$$= \mathcal{O}\left(\frac{2^d dB\sqrt{D}\prod_{j=1}^{D}M_F(j)}{\sqrt{n}}\right) \tag{A.14}$$

Together with (A.10) and (A.11), we can further get

$$\begin{aligned}
&\mathbb{P}\Big(\sup_\theta I(X_\theta) > \eta(n,d)+\epsilon\Big)\\
\leq&\mathbb{P}\Big(\sup_\theta I(X_\theta) > 2\mathfrak{R}_{n-1}\Big(\mathcal{F}_{\theta,X_n}\Big)+\epsilon\Big)\\
\leq&\mathbb{P}\Big(\sup_\theta I(X_\theta) > \mathbb{E}(\sup_\theta I(Y_\theta))+\epsilon\Big) \leq \exp\left(-\frac{\epsilon^2 n}{2}\right)
\end{aligned}$$

Now we can get

$$\begin{aligned}
\mathbb{P}\left(\mathcal{T}(\widehat{\theta})-\mathcal{T}(\theta^*) > 4\epsilon+\eta\right) &= \mathbb{P}\left(\mathcal{T}(\widehat{\theta})-\mathcal{E}(\widehat{\theta})+\mathcal{E}(\widehat{\theta})-\mathcal{T}(\theta^*) > 4\epsilon+\eta\right)\\
&\leq \mathbb{P}\left(\mathcal{T}(\widehat{\theta})-\mathcal{E}(\widehat{\theta})+\mathcal{E}(\theta^*)-\mathcal{T}(\theta^*) > 4\epsilon+\eta\right)\\
&\leq \mathbb{P}\left(|\mathcal{E}(\widehat{\theta})-\mathcal{T}(\widehat{\theta})| > \epsilon+\eta\right)+\mathbb{P}\left(|\mathcal{E}(\theta^*)-\mathcal{T}(\theta^*)| > 2\sqrt{2}\,\epsilon\right)\\
&\leq 2\exp\left(-\frac{\epsilon^2 n}{2}\right)
\end{aligned}$$

Together with (A.7), the theorem is proved and the bound goes to zero if $\epsilon^2 n$ go to infinity. □

Remark The Lipschitz condition for kernel u_q is not hard to satisfy. From Lemma A.1, if we use Gaussian kernel, as long as S_q doesn't have exponential tails, the Lipschitz condition is satisfied.

In our application, we can choose a wide range of noise distributions as long as it is easy to sample and regular enough. If we choose uniform distribution, then $B = \mathcal{O}(\sqrt{d})$. If assume $M_F(j) \leq M \leq \infty$ for any $j = 1, 2, \cdots, D$. Then (A.5) becomes

$$\text{KSD}(p_{\theta^*}, q) \leq \widehat{\text{KSD}}(p_{\widehat{\theta}}, q) + \mathcal{O}\left(\frac{2^d d^{3/2}}{\sqrt{n}}\right)+\epsilon$$

A.2 Proof of Theorem 5.1

Lemma A.5.

$$\mathbb{E}_{x\sim q}\,\text{tr}\left(\nabla_x \log p(x)\boldsymbol{f}(x)^\top + \nabla_x \boldsymbol{f}(x)\right) = \mathbb{E}_{x\sim q}\,\text{tr}\left((\nabla_x \log p(x) - \nabla_x \log q(x))\boldsymbol{f}(x)^\top\right)$$

Proof.

$$\begin{aligned}
&\mathbb{E}_{x\sim q}\,\text{tr}\left(\nabla_x \log q(x)\boldsymbol{f}(x)^\top + \nabla_x \boldsymbol{f}(x)\right) = 0\\
&\qquad\Rightarrow \mathbb{E}_{x\sim q}\,\text{tr}\left(\nabla_x \boldsymbol{f}(x)\right) = -\mathbb{E}_{x\sim q}\,\text{tr}\left(\nabla_x \log q(x)\boldsymbol{f}(x)^\top\right)
\end{aligned}$$

□

Lemma A.6.

$$|\mathbb{E}_{x\sim q}\operatorname{tr}\left(\boldsymbol{g}(x)\boldsymbol{f}(x)^\top\right)| \le \sqrt{\mathbb{E}_{x\sim q}\operatorname{tr}\left(\boldsymbol{g}(x)^\top\boldsymbol{g}(x)\right) * \mathbb{E}_{x\sim q}\operatorname{tr}\left(\boldsymbol{f}(x)^\top\boldsymbol{f}(x)\right)}$$

The equality holds iff $f \propto g$ a.s x $\sim q$.

Proof. Firstly, we have $tr(\boldsymbol{f}(x)\boldsymbol{f}(x)^\top) = tr(\boldsymbol{f}(x)^\top\boldsymbol{f}(x)) = \boldsymbol{f}(x)^\top\boldsymbol{f}(x) = \|\boldsymbol{f}(x)\|_2^2$

$$\mathbb{E}_{x\sim q}\operatorname{tr}\left((\boldsymbol{f}-t*\boldsymbol{g})(\boldsymbol{f}-t*\boldsymbol{g})^\top\right) \ge 0$$
$$\Rightarrow \mathbb{E}_{x\sim q}\operatorname{tr}\left(\boldsymbol{f}^\top\boldsymbol{f}\right) + t^2\mathbb{E}_{x\sim q}\operatorname{tr}\left(\boldsymbol{g}^\top\boldsymbol{g}\right) \ge 2t\mathbb{E}_{x\sim q}\operatorname{tr}\left(\boldsymbol{g}\boldsymbol{f}^\top\right)$$

Because inequality hold for all t, so the lemma is proved. □

Theorem 5.1 The optimum discriminator is

$$\frac{1}{2\lambda}(S_q - S_p)$$

Training generator equals minimize the fisher divergence of p and q

$$\frac{1}{4\lambda}\mathbb{E}_{x\sim p}\operatorname{tr}\left((S_q-S_p)^\top(S_q-S_p)\right)$$

Proof. Let our loss function be L. Because $\operatorname{tr}(\boldsymbol{f}(x)\boldsymbol{f}(x)^\top) = \operatorname{tr}(\boldsymbol{f}(x)^\top\boldsymbol{f}(x)) = \boldsymbol{f}(x)^\top\boldsymbol{f}(x) = \|\boldsymbol{f}(x)\|_2^2$. Then we have

$$L = \mathbb{E}_{x\sim p}\operatorname{tr}\left(S_q(x)\boldsymbol{f}(x)^\top + \nabla_x\boldsymbol{f}(x)\right) - \lambda\mathbb{E}_{x\sim p}[\operatorname{tr}(\boldsymbol{f}(x)^\top\boldsymbol{f}(x))] \tag{A.15}$$
$$= \mathbb{E}_{x\sim p}\operatorname{tr}\left((S_q(x)-S_p(x))\boldsymbol{f}(x)^\top\right) - \lambda\mathbb{E}_{x\sim p}[\operatorname{tr}(\boldsymbol{f}(x)^\top\boldsymbol{f}(x))] \tag{A.16}$$
$$\le \sqrt{\mathbb{E}_{x\sim p}\operatorname{tr}\left((S_q-S_p)^\top(S_q-S_p)\right)\cdot\mathbb{E}_{x\sim p}\operatorname{tr}\left(\boldsymbol{f}(x)^\top\boldsymbol{f}(x)\right)} - \lambda\mathbb{E}_{x\sim p}\operatorname{tr}\left(\boldsymbol{f}(x)^\top\boldsymbol{f}(x)\right) \tag{A.17}$$
$$\le \frac{1}{4\lambda}\mathbb{E}_{x\sim p}\operatorname{tr}\left((S_q-S_p)^\top(S_q-S_p)\right) \tag{A.18}$$

Equality sign in (A.17) holds iff $f \propto S_q - S_p, a.s\ x \sim p$.

Equality sign in (A.18) holds iff

$$\mathbb{E}_{x\sim p}\operatorname{tr}\left(\boldsymbol{f}(x)^\top\boldsymbol{f}(x)\right) = \frac{1}{4\lambda^2}\mathbb{E}_{x\sim p}\operatorname{tr}\left((S_q-S_p)^\top(S_q-S_p)\right)$$

So argmax of L is $\frac{1}{2\lambda}(\nabla\log p - \nabla\log q)a.s x \sim p$. □

Theorem 5.2 If the following conditions are satisfied: 1) both the generator's weights and the noises are bounded, 2) discriminator uses smooth activate function i.e. tanh, sigmoid, etc., 3) target score function s_q is continuously differentiable. Then $L(\theta)$ is continuous everywhere and differentiable almost everywhere w.r.t θ.

Proof. Using generator with weight clipping and uniform noise, we have a transform function G_θ which is lipschitz. As we can see later in Theorem A.10, there exist a compact set Ω. $P(G_\theta(z) \in \Omega) = 1, \forall\theta$.

From the condition of discriminator we know that f is smooth. So $\mathcal{A}_q f$ is continuously differentiable on Ω, $f^T f$ is continuously differentiable on Ω. So we have $\|\mathcal{A}_q f\|_{lip,\Omega} + \|f^T f\|_{lip,\Omega} < \infty$

$$\begin{aligned}
&\mathbb{E}_{p_\theta}(\mathcal{A}_q f(x) - \lambda f(x)^T f(x)) - \mathbb{E}_{p_{\theta'}}(\mathcal{A}_q f(x) - \lambda f(x)^T f(x)) \\
=&\left(\mathbb{E}_z(\mathcal{A}_q f(G_\theta(z)) - \lambda f(G_\theta(z))^T f(G_\theta(z))) - \mathbb{E}_z(\mathcal{A}_q f(G_{\theta'}(z)) - \lambda f(G_{\theta'}(z))^T f(G_\theta(z)'))\right) \\
\le&(\|\mathcal{A}_q f\|_{lip,\Omega} + \lambda\|f^T f\|_{lip,\Omega})\mathbb{E}_z(\|G_\theta - G_{\theta'}\|)
\end{aligned}$$

Because z and θ all bounded, We know that G is locally lipschitz. For a given pair (θ, z) there is a constant $C(\theta, z)$ and an open set U_θ such that for every $(\theta', z) \in U_\theta$ we have

$$\|G_\theta(z) - G_{\theta'}(z)\| \le C(\theta, z)\|\theta - \theta'\|$$

Under the condition mentioned before, $E_z|C(\theta, z)| < \infty$, so we achieve

$$\|L(\theta) - L(\theta')\| \le (\|\mathcal{A}_q f\|_{lip,\Omega} + \lambda\|f^T f\|_{lip,\Omega})\mathbb{E}_z|C(\theta, z)|\|\theta - \theta'\|$$

Therefore, $L(\theta)$ is locally Lipschitz and continuous everywhere. Last, applying Radamachers theorem proves $L(\theta)$ is differentiable almost everywhere, which completes the proof. □

A.3 Relationship to Wasserstein GAN

Denote $\phi = \mathrm{tr}(\mathcal{A}_q f)$, then $\phi = \mathbf{div}(qf)/q$ and our loss function without penalty can be re-written as $E_p(\phi) - E_q(\phi)$.

Lemma A.7. (Durán and López García, 2010) If Ω is a John domain, for any $v \in L_0^l(\Omega), l > 1$ there exists $u \in \mathcal{W}_0^{1,l}$ such that $\mathbf{div}\ u = v$ in Ω

In Wasserstein GAN, if we constrain the functions to be compactly supported and the expectation under the target distribution $\mathbb{E}_q(f)$ to be zero, the result doesn't change.

Theorem A.8. Suppose there exists $l > 1$ s.t $\|xq\|, \|q\| \in L_0^l$, then if we constrain $\phi = \mathrm{tr}(\mathcal{A}_q f)$ to be Lip-1 and compacted supported. Then the optimal loss function is Wasserstein-1 distance.

Proof. For every function ϕ which is Lip-1 and has compact support, $\|\phi q\| \leq \|xq\| + c \cdot \|q\|$, where $c > 0$ is some constant. So the equation has a solution, there exist f which has a compact support s.t $\phi = tr(\mathcal{A}_q f)$. □

Remark Firstly, $\|xq\|, \|q\| \in L_0^l$ is extremely weak even for Cauchy distribution this condition holds. Secondly, we can apply weight clipping to f to ensure f has a compact support and $\mathrm{tr}(\mathcal{A}_p f)$ is Lip-1.

A.4 Weak convergence

Theorem A.9. If kernel $k(x, y)$ if bounded by constant c. Then $S(p_\theta, q) \leq c \cdot F(p_\theta, q)$

Proof.

$$\begin{aligned}
S(p_\theta, q)^2 &= |\mathbb{E}_{x,x'}((S_{p_\theta}(x) - S_q(x))^T k(x, x')(S_{p_\theta}(x') - S_q(x')))|^2 \\
&\leq \mathbb{E}_{x,x'}(k(x, x')^2) \cdot \mathbb{E}_{x,x'}(|(S_{p_\theta}(x) - S_q(x))^T (S_{p_\theta}(x') - S_q(x'))|^2) \\
&\leq \mathbb{E}_{x,x'}(k(x, x')^2) \cdot \mathbb{E}_{x,x'}(\|(S_{p_\theta}(x) - S_q(x))\|_2^2 \|(S_{p_\theta}(x') - S_q(x'))\|_2^2) \\
&= \mathbb{E}_{x,x'}(k(x, x')^2) \cdot F(p_\theta, q)^2 \\
&\leq c^2 \cdot F(p_\theta, q)^2
\end{aligned}$$

□

Theorem A.10. Suppose we use uniform or Gaussian noise, tanh or relu activate function for generator. Then p_θ is uniformly tight, if we clip the weight to $(-c, c)$ for any $c > 0$.

Proof. Denote the transform function of generator is $G_\theta(x)$. Fix z_0 in the space. Then we know that there exist R, s.t $\|G_\theta(z_0)\| < R$ for all θ. In addition, G_θ is a lipschitz function because the weight is clipped to $(-c, c)$. So there exist k s.t $\|G_\theta(x) - G_\theta(y)\| \leq k\|x - y\|$. So we have

$$P(\|G_\theta(z)\| > A) \leq P(\|G_\theta(z) - G_\theta(z_0)\| > A - R) \leq P(\|z - z_0\| > (A - R)/k)$$

Notice that z $\sim$ normal or uniform. For all $\epsilon > 0$, there exist $\widehat{A}$ s.t $(\|z - z_0\| > \widehat{A}/k) < \epsilon$.Therefore $P(\|G_\theta(z)\| > \widehat{A} + R) < \epsilon$ holds for all θ, which means G_θ are uniformly tight. Moreover if noise is uniform, there exist $\widehat{A}$ s.t $P(\|G_\theta(z)\| > \widehat{A} + R) = 0$ for all θ.

□

A.5 Simulation Details

Experiment Setting in Gaussian Mixtures To ensure the comparison is fair, the neural samplers structure in Stein GAN, KSD-NS, Fisher-NS are the same in all the 2-dim Gaussian mixture experiments shown in the simulation section. The generator/sampler is a plain network that has two hidden layers with width 200 and $\tanh(\cdot)$ as activation function. The discriminator network in Fisher-NS is set to be of the same structure as the generator. The noise is chosen to be uniform$(-10, 10)$. The optimization is done in TensorFlow via RMSProp. The learning rate is 0.001 for all Stein GAN and KSD-NS and Fisher-NS. The step size for SVGD is 0.3.

Experiment Setting in Bayesian Logistic Regression Across all the methods, we use a mini-batch of 100 data points for each iteration (for each stage in DSVI). The setting for SVGD is the same as in (Liu and Wang, 2016). For SGLD, as suggested in Welling and Teh (2011), the learning rate is chosen to be $0.1/(t+1)^{0.55}$.

Codes available at https://github.com/HanxiSun/SteinNS