Stein Variational Inference for Discrete Distributions

Jun Han[1] **Fan Ding**[2] **Xianglong Liu**[2] **Lorenzo Torresani**[1] **Jian Peng**[3] **Qiang Liu**[4]
[1]Dartmouth College [2]Beihang University [3]UIUC [4]UT Austin

Abstract

Gradient-based approximate inference methods, such as Stein variational gradient descent (SVGD) [19], provide simple and general purpose inference engines for differentiable continuous distributions. However, existing forms of SVGD cannot be directly applied to discrete distributions. In this work, we fill this gap by proposing a simple yet general framework that transforms discrete distributions to equivalent piecewise continuous distributions, on which the gradient-free SVGD is applied to perform efficient approximate inference. The empirical results show that our method outperforms traditional algorithms such as Gibbs sampling and discontinuous Hamiltonian Monte Carlo on various challenging benchmarks of discrete graphical models. We demonstrate that our method provides a promising tool for learning ensembles of binarized neural network (BNN), outperforming other widely used ensemble methods on learning binarized AlexNet on CIFAR-10 dataset. In addition, such transform can be straightforwardly employed in gradient-free kernelized Stein discrepancy to perform goodness-of-fit (GOF) test on discrete distributions. Our proposed method outperforms existing GOF test methods for intractable discrete distributions.

1 INTRODUCTION

Discrete probabilistic models provide a powerful framework for capturing complex phenomenons and patterns, especially in conducting logic and symbolic reasoning. However, probabilistic inference of high dimensional discrete distribution is in general NP-hard and requires highly efficient approximate inference tools.

Traditionally, approximate inference in discrete models is performed by either Gibbs sampling and Metropolis-Hastings algorithms, or deterministic variational approximation, such as belief propagation, mean field approximation and variable elimination methods [26, 6]. However, both of these two types of algorithms have their own critical weaknesses: Monte Carlo methods provides theoretically consistent sample-based (or particle) approximation, but are typically slow in practice, while deterministic approximation are often much faster in speed, but does not provide progressively better approximation like Monte Carlo methods offers. New methods that integrate the advantages of the two methodologies is a key research challenge; see, for example, [17, 20, 2].

Recently, Stein variational gradient descent (SVGD, [19]) provides a combination of deterministic variational inference with sampling, for the case of *continuous distributions*. The idea is to directly optimize a particle-based approximation of the intractable distributions by following a functional gradient descent direction, yielding both practically fast algorithms and theoretical consistency. However, because SVGD only works for continuous distributions, a key open question is if it is possible to exploit it for more efficient inference of discrete distributions.

In this work, we leverage the power of SVGD for inference of discrete distributions. Our idea is to transform discrete distributions to piecewise continuous distributions, on which gradient-free SVGD, a variant of SVGD that leverages a differentiable surrogate distribution to sample non-differentialbe continuous distributions, is applied to perform inference. To do so, we design a simple yet general framework for transforming discrete distributions to equivalent continuous distributions, which is specially tailored for our purpose, so that we can conveniently construct differentiable surrogates when applying GF-SVGD.

We apply our proposed algorithm to a wide range of discrete distributions, such as Ising models and re-

Proceedings of the 23[rd]International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

stricted Boltzmann machines. We find that our proposed algorithm significantly outperforms traditional inference algorithms for discrete distributions. In particular, our algorithm is shown to be provide a promising tool for ensemble learn of binarized neural network (BNN) in which both weights and activation functions are binarized. Learning BNNs have been shown to be a highly challenging problem, because standard backpropagation cannot be applied. We cast learning BNN as a Bayesian inference problem of drawing a set of samples (which forms an ensemble predictor) of the posterior distribution of weights, and apply our SVGD-based algorithm for efficient inference. We show that our method outperforms other widely-used ensemble methods such as bagging and AdaBoost in achieving highest accuracy with the same ensemble size on the binarized AlexNet.

In addition, we develop a new goodness-of-fit test for intractable discrete distributions based on gradient-free kernelized Stein discrepancy on the transformed continuous distributions using the simple transform constructed before. Our proposed algorithm outperforms discrete KSD (DKSD, [27]) and maximum mean discrepancy (MMD, [8]) on various benchmarks.

Related work on Sampling The idea of transforming the inference of discrete distributions to continuous distributions has been widely studied, which, however, mostly concentrates on leveraging the power of Hamiltonian Monte Carlo (HMC); see, for example, [1, 22, 23, 28, 7]. Our framework of transforming discrete distributions to piecewise continuous distribution is similar to [22], but is more general and tailored for the application of GF-SVGD. **Related work on goodness-of-fit test** Our goodness-of-fit testing is developed from KSD [18, 3], which works for differentiable continuous distributions. Some forms of goodness-of-fit tests on discrete distributions have been recently proposed such as [21, 5, 25]. But they are often model-specialized and require the availability of the normalization constant. [27, 8] is related to ours and will be empirically compared.

Outline Our paper is organized as follows. Section 2 introduces GF-SVGD and GF-KSD. Section 3 proposes our main algorithms for sampling and goodness-of-fit testing on discrete distributions. Section 4 provides empirical experiments. We conclude the paper in Section 5.

2 STEIN VARIATIONAL GRADIENT DESCENT

We first introduce SVGD [19], which provides deterministic sampling but requires the gradient of the target distribution. We then introduce gradient-free SVGD and gradient-free KSD [10], which can be applied to the target distribution with unavailable or intractable gradient.

Let $p(\boldsymbol{x})$ be a differentiable density function supported on $\mathbb{R}^d$. The goal of SVGD is to find a set of samples $\{\boldsymbol{x}_i\}_{i=1}^n$ (called "particles") to approximate p in the sense that

$$\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{x}_i) = \mathbb{E}_p[f(\boldsymbol{x})],$$

for general test functions f. SVGD achieves this by starting with a set of particles $\{\boldsymbol{x}_i\}_{i=1}^n$ drawn from any initial distribution, and iteratively updates the particles by

$$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \epsilon \boldsymbol{\phi}^*(\boldsymbol{x}_i), \quad \forall i = 1, \dots, n, \tag{1}$$

where ϵ is a step size, and $\boldsymbol{\phi}\colon \mathbb{R}^d \to \mathbb{R}^d$ is a velocity field chosen to drive the particle distribution closer to the target. Assume the distribution of the particles at the current iteration is q, and $q_{[\epsilon\boldsymbol{\phi}]}$ is the distribution of the updated particles $\boldsymbol{x}' = \boldsymbol{x} + \epsilon\boldsymbol{\phi}(\boldsymbol{x})$. The optimal choice of $\boldsymbol{\phi}$ can be framed as the following optimization problem:

$$\boldsymbol{\phi}^* = \arg\max_{\boldsymbol{\phi}\in\mathbb{F}} \left\{ -\frac{d}{d\epsilon}\mathrm{KL}(q_{[\epsilon\boldsymbol{\phi}]} \,||\, p)\big|_{\epsilon=0} = \mathbb{E}_{\boldsymbol{x}\sim q}[\mathcal{A}_p^\top \boldsymbol{\phi}(\boldsymbol{x})] \right\},$$
$$\text{with} \quad \mathcal{A}_p^\top \boldsymbol{\phi}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}}^\top \boldsymbol{\phi}(\boldsymbol{x}), \tag{2}$$

where $\mathbb{F}$ is a set of candidate velocity fields, $\boldsymbol{\phi}$ is chosen in $\mathbb{F}$ to maximize the decreasing rate on the KL divergence between the particle distribution and the target, and $\mathcal{A}_p$ is a linear operator called *Stein operator* and is formally viewed as a column vector similar to the gradient operator $\nabla_{\boldsymbol{x}}$.

In SVGD, $\mathbb{F}$ is chosen to be the unit ball of a vector-valued reproducing kernel Hilbert space (RKHS) $\mathcal{H} = \mathcal{H}_0 \times \cdots \times \mathcal{H}_0$, where $\mathcal{H}_0$ is an RKHS formed by scalar-valued functions associated with a positive definite kernel $k(\boldsymbol{x}, \boldsymbol{x}')$, that is, $\mathbb{F} = \{\boldsymbol{\phi} \in \mathcal{H}\colon ||\boldsymbol{\phi}||_{\mathcal{H}} \le 1\}$. This choice of $\mathbb{F}$ makes it possible to consider velocity fields in infinite dimensional function spaces while still obtaining computationally tractable solution.

[19] showed that (2) has a simple closed-form solution:

$$\boldsymbol{\phi}^*(\boldsymbol{x}') \propto \mathbb{E}_{\boldsymbol{x}\sim q}[\mathcal{A}_p k(\boldsymbol{x}, \boldsymbol{x}')], \tag{3}$$

where $\mathcal{A}_p$ is applied to variable $\boldsymbol{x}$. With the optimal form $\boldsymbol{\phi}^*(\boldsymbol{x}')$, the objective in (2) equals to

$$\mathcal{D}_{\mathbb{F}}(q||p) \stackrel{def}{=} \max_{\boldsymbol{\phi}\in\mathbb{F}} \left\{ \mathbb{E}_{\boldsymbol{x}\sim q}\left[\mathcal{A}_p^\top \boldsymbol{\phi}(\boldsymbol{x})\right] \right\}, \tag{4}$$

where $\mathbb{D}_{\mathbb{F}}(q \,||\, p)$ is the kernelized Stein discrepancy (KSD) defined in [18, 3].

In practice, SVGD iteratively update particles $\{\boldsymbol{x}_i\}$ by $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \frac{\epsilon}{n}\Delta\boldsymbol{x}_i$, where,

$$\Delta\boldsymbol{x}_i = \sum_{j=1}^{n}[\nabla \log p(\boldsymbol{x}_j)k(\boldsymbol{x}_j, \boldsymbol{x}_i) + \nabla_{\boldsymbol{x}_j} k(\boldsymbol{x}_j, \boldsymbol{x}_i)]. \quad (5)$$

Gradient-free SVGD GF-SVGD [10] extends SVGD to the setting when the gradient of the target distribution does not exist or is unavailable. The key idea is to replace it with the gradient of the differentiable surrogate $\rho(\boldsymbol{x})$ whose gradient can be calculated easily, and leverage it for sampling from $p(\boldsymbol{x})$ using a mechanism similar to importance sampling.

The derivation of GF-SVGD is based on the following key observation,

$$w(\boldsymbol{x})\mathcal{A}_\rho^\top \boldsymbol{\phi}(\boldsymbol{x}) = \mathcal{A}_p^\top \big(w(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x})\big). \quad (6)$$

where $w(\boldsymbol{x}) = \rho(\boldsymbol{x})/p(\boldsymbol{x})$. Eq. (6) indicates that the Stein operation w.r.t. p, which requires the gradient of the target p, can be transferred to the Stein operator of a surrogate distribution ρ, which does not depends on the gradient of p. Based on this observation, GF-SVGD modifies to optimize the following object,

$$\boldsymbol{\phi}^* = \arg\max_{\boldsymbol{\phi}\in\mathbb{F}}\{\mathbb{E}_q[\mathcal{A}_p^\top (w(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x}))]\}. \quad (7)$$

Similar to the derivation in SVGD, the optimization problem (7) can be analytically solved; in practice, GF-SVGD derives a gradient-free update as $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \frac{\epsilon}{n}\Delta\boldsymbol{x}_i$, where

$$\Delta\boldsymbol{x}_i \propto \sum_{j=1}^{n} w_j \big[\nabla \log \rho(\boldsymbol{x}_j)k(\boldsymbol{x}_j, \boldsymbol{x}_i) + \nabla_{\boldsymbol{x}_j} k(\boldsymbol{x}_j, \boldsymbol{x}_i)\big], \quad (8)$$

which replaces the true gradient $\nabla \log p(\boldsymbol{x})$ with a surrogate gradient $\nabla \log \rho(\boldsymbol{x})$, and then uses an importance weight $w_j := \rho(\boldsymbol{x}^j)/p(\boldsymbol{x}^j)$ to correct the bias introduced by the surrogate. [10] observed that GF-SVGD can be viewed as a special case of SVGD with an "importance weighted" kernel, $\widetilde{k}(\boldsymbol{x}, \boldsymbol{x}') = \rho(\boldsymbol{x})/p(\boldsymbol{x})k(\boldsymbol{x}, \boldsymbol{x}')\rho(\boldsymbol{x}')/p(\boldsymbol{x}')$. Therefore, GF-SVGD inherits the theoretical justifications of SVGD [16]. GF-SVGD is proposed to apply to continuous-valued distributions.

Gradient-Free KSD As shown in [10], the optimal decrease rate of the KL divergence in (2) is

$$\mathcal{D}^2(q, p) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim q}[w(\boldsymbol{x})k_\rho(\boldsymbol{x}, \boldsymbol{x}')w(\boldsymbol{x}')], \quad (9)$$

where $\kappa_\rho(\boldsymbol{x}, \boldsymbol{x}')$ is defined as,

$$\begin{aligned}\kappa_\rho(\boldsymbol{x}, \boldsymbol{x}') =& \boldsymbol{s}_\rho(\boldsymbol{x})^\top k(\boldsymbol{x}, \boldsymbol{x}')\boldsymbol{s}_\rho(\boldsymbol{x}') + \boldsymbol{s}_\rho(\boldsymbol{x})^\top \nabla_{\boldsymbol{x}'} k(\boldsymbol{x}, \boldsymbol{x}') \\ &+ \boldsymbol{s}_\rho(\boldsymbol{x}')^\top \nabla_{\boldsymbol{x}} k(\boldsymbol{x}, \boldsymbol{x}') + \nabla_{\boldsymbol{x}} \cdot (\nabla_{\boldsymbol{x}'} k(\boldsymbol{x}, \boldsymbol{x}')),\end{aligned} \quad (10)$$

where $\boldsymbol{s}_\rho(\boldsymbol{x})$ is score function of the surrogate $\rho(\boldsymbol{x})$. Note that in order to estimate the KSD between q and p, we only need samples $\{\boldsymbol{x}_i\}$ from q, $w(\boldsymbol{x})$ and the gradient of $\rho(\boldsymbol{x})$. Therefore, we obtain a form of *gradient-free KSD*.

The goal of this paper is to develop a tool for goodness-of-fit testing on discrete distribution based on gradient-free KSD and a method for sampling on discrete-valued distributions by exploiting gradient-free SVGD.

3 MAIN METHOD

This section introduces the main idea of this work. We first provides a simple yet powerful way to transform the discrete-valued distributions to the continuous-valued distributions. Then we leverage the gradient-free SVGD to sample from the transformed continuous-valued distributions. Finally, we leverage the constructed transform to perform goodness-of-fit test on discrete distributions.

Assume we are interested in sampling from a given discrete distribution $p_*(\boldsymbol{z})$, defined on a finite discrete set $\mathcal{Z} = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_K\}$. We may assume each $\boldsymbol{a}_i$ is a d-dimensional vector of discrete values. Our idea is to construct a piecewise continuous distribution $p_c(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathbb{R}^d$, and a map $\Gamma\colon \mathbb{R}^d \to \mathcal{Z}$, such that the distribution of $\boldsymbol{z} = \Gamma(\boldsymbol{x})$ is p_* when $\boldsymbol{x} \sim p_c$. In this way, we can apply GF-SVGD on p_c to get a set of samples $\{\boldsymbol{x}_i\}_{i=1}^n$ from p_c and apply transform $\boldsymbol{z}_i = \Gamma(\boldsymbol{x}_i)$ to get samples $\{\boldsymbol{z}_i\}$ from p_*.

Definition 1. *A piecewise continuous distribution p_c on $\mathbb{R}^d$ and map $\Gamma : \mathbb{R}^d \to \mathcal{Z}$ is called to form a **continuous parameterization** of p_*, if $\boldsymbol{z} = \Gamma(\boldsymbol{x})$ follows p_* when $\boldsymbol{x} \sim p_c$.*

This definition immediately implies the following result.

Proposition 2. *The continuous distribution p_c and Γ form a continuous parameterization of discrete distribution p_* on $\mathcal{Z} = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_K\}$, iff*

$$p_*(\boldsymbol{a}_i) = \int_{\mathbb{R}^d} p_c(\boldsymbol{x})\mathbb{I}[\boldsymbol{a}_i = \Gamma(\boldsymbol{x})]d\boldsymbol{x}, \quad (11)$$

for all $i = 1, \ldots, K$. Here $\mathbb{I}(\cdot)$ is the 0/1 indicator function, $\mathbb{I}(t) = 0$ iff $t = 0$ and $\mathbb{I}(t) = 1$ if otherwise.

Constructing Continuous Parameterizations Given a discrete distribution p_*, there are many different continuous parameterizations. Because *exact* samples of p_c yield *exact* samples of p_* following the

Algorithm 1 GF-SVGD on Discrete Distributions

Goal: Approximate a given distribution $p_*(\boldsymbol{z})$ (input) on a finite discrete set $\mathcal{Z}$.

1) Decide a base distribution $p_0(\boldsymbol{x})$ on $\mathbb{R}^d$ (such as Gaussian distribution), and a map $\Gamma\colon \mathbb{R}^d \to \mathcal{Z}$ which partitions p_0 evenly. Construct a piecewise continuous distribution p_c by (15):

$$p_c(\boldsymbol{x}) \propto p_0(\boldsymbol{x}) p_*(\Gamma(\boldsymbol{x})).$$

2) Construct a differentiable surrogate of $p_c(\boldsymbol{x})$, for example, by $\rho(\boldsymbol{x}) \propto p_0(\boldsymbol{x})$ or $\rho(\boldsymbol{x}) \propto p_0(\boldsymbol{x})\tilde{p}_*(\tilde{\Gamma}(\boldsymbol{x}))$, where $\tilde{p}_*$ and $\tilde{\Gamma}$ are smooth approximations of p_* and Γ, respectively.

3) Run gradient-free SVGD on p_c with differentiable surrogate ρ: starting from an initial $\{\boldsymbol{x}_i\}_{i=1}^n$ and repeat

$$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \frac{\epsilon}{\sum_i w_i} \sum_{j=1}^n w_j (\nabla \rho(\boldsymbol{x}_j) k(\boldsymbol{x}_j, \boldsymbol{x}_i) + \nabla_{\boldsymbol{x}_j} k(\boldsymbol{x}_j, \boldsymbol{x}_i)).$$

where $w_j = \rho(\boldsymbol{x}_j)/p_c(\boldsymbol{x}_j)$, and $k(\boldsymbol{x}, \boldsymbol{x}')$ is a positive definite kernel.

4) Calculate $\boldsymbol{z}_i = \Gamma(\boldsymbol{x}_i)$ and **output** sample $\{\boldsymbol{z}_i\}_{i=1}^n$ for approximating discrete target distribution $p_*(\boldsymbol{z})$.

definition, we should prefer to choose continuous parameterizations whose p_c is easy to sample using continuous inference method, GF-SVGD in particular in our method. However, it is generally difficult to find a theoretically optimal continuous parameterization, because it is difficult to quantitatively the notation of difficulty of approximate inference by particular algorithms, and deriving the mathematically optimal continuous parameterization may be computationally demanding and requires analysis in a case by case basis.

In this work, we introduce a simple yet general framework for constructing continuous parameterizations. Our goal is not to search for the best possible continuous parameterization for individual discrete distribution, but rather to develop a general-purpose framework that works for a wide range of discrete distributions and can be implemented in an automatic fashion. Our method also naturally comes with effective differentiable surrogate distributions with which GF-SVGD can perform efficiently.

Even Partition Our method starts with choosing a simple base distribution p_0, which can be the standard Gaussian distribution. We then construct a map Γ that *evenly partition* p_0 into several regions with equal probabilities.

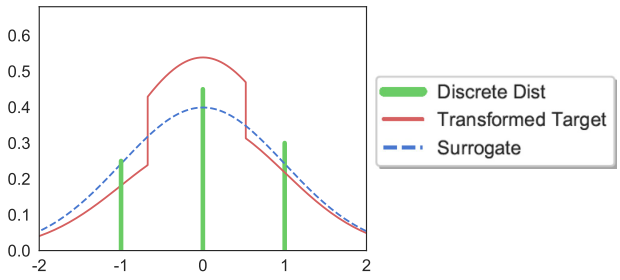


Figure 1: Illustrating the construction of $p_c(\boldsymbol{x})$ (red line) of a three-state discrete distribution p_* (green bars). The blue dash line represents the base distribution we use, which is a standard Gaussian distribution.

Definition 3. *A map $\Gamma\colon \mathcal{Z} \to \mathbb{R}^d$ is said to* ***evenly partition*** *p_0 if we have*

$$\int_{\mathbb{R}^d} p_0(\boldsymbol{x}) \mathbb{I}[\boldsymbol{a}_i = \Gamma(\boldsymbol{x})] d\boldsymbol{x} = \frac{1}{K}, \tag{12}$$

for $i = 1, \dots K$. Following (11), this is equivalent to saying that p_0 and Γ forms a continuous relaxation of the uniform distribution $q_*(\boldsymbol{a}_i) = 1/K$.

For simple p_0 such as standard Gaussian distributions, it is straightforward to construct even partitions using the quantiles of $p_0(\boldsymbol{x})$. For example, in the one dimensional case ($d = 1$), we can evenly partition any continuous $p_0(\boldsymbol{x})$, $\boldsymbol{x} \in \mathbb{R}$ by

$$\Gamma(\boldsymbol{x}) = \boldsymbol{a}_i \quad \text{if} \quad \boldsymbol{x} \in [\eta_{i-1}, \ \eta_i), \tag{13}$$

where η_i denotes the i/K-th quantile of distribution p_0. In multi-dimensional cases ($d > 1$) and when p_0 is a product distribution:

$$p_0(\boldsymbol{x}) = \prod_{i=1}^d p_{0,i}(x_i). \tag{14}$$

One can easily show that an even partition can be constructed by concatenating one-dimensional even partition: $\Gamma(\boldsymbol{x}) = (\Gamma_1(x_1), \cdots, \Gamma_d(x_d))$, where $\boldsymbol{x} = (x_1, \cdots, x_d)$ and $\Gamma_i(\cdot)$ an even partition of $p_{0,i}$.

A particularly simple case is when $\boldsymbol{z}$ is a binary vector, i.e., $\mathcal{Z} = \{\pm 1\}^d$, in which case $\Gamma(\boldsymbol{x}) = \text{sign}(\boldsymbol{x})$ evenly partitions any distribution p_0 that is symmetric around the origin.

Weighting the Partitions Given an even partition of p_0, we can conveniently construct a continuous parameterization of an arbitrary discrete distribution p_* by *weighting each bin of the partition with corresponding probability in p_**, that is, we may construct $p_c(\boldsymbol{x})$ by

$$p_c(\boldsymbol{x}) \propto p_0(\boldsymbol{x}) p_*(\Gamma(\boldsymbol{x})), \tag{15}$$

where $p_0(\boldsymbol{x})$ is weighted by $p_*(\Gamma(\boldsymbol{x}))$, the probability of the discrete value $\boldsymbol{z} = \Gamma(\boldsymbol{x})$ that $\boldsymbol{x}$ maps.

Proposition 4. *Assume Γ is an even partition of $p_0(\boldsymbol{x})$, and $p_c(\boldsymbol{x}) \propto p_0(\boldsymbol{x})p_*(\Gamma(\boldsymbol{x}))$, then $(p_c,\ \Gamma)$ is a continuous parameterization of p_*.*

Constructing Differentiable Surrogate Given such a transformation, it is also convenient to construct differentiable surrogate ρ of p_c in (15) for GF-SVGD. by simply removing $p_*(\Gamma(\boldsymbol{x}))$ (so that $\rho = p_0$), or approximate it with some smooth approximation, based on properties of p_* and Γ, that is,

$$\rho(\boldsymbol{x}) = p_0(\boldsymbol{x})\tilde{p}_*(\tilde{\Gamma}(\boldsymbol{x})), \tag{16}$$

where $\tilde{\Gamma}(\cdot)$ denotes a smooth approximation of $\Gamma(\boldsymbol{x})$, and $\tilde{p}_*$ is a continuous extension of $p_*(\boldsymbol{x})$ to the continuous domain $\mathbb{R}^d$. See Algorithm 1 for the summary of our main procedure.

Illustration Using 1D Categorical Distribution Consider the 1D categorical distribution p_* shown in Fig. 1, which takes $\{-1, 0, 1\}$ with probabilities $\{0.25, 0.45, 0.3\}$, respectively. We use the standard Gaussian base p_0 (blue dash), and obtain a continuous parameterization p_c using (15), in which $p_0(x)$ is weighted by the probabilities of p_* in each bin. Note that p_c is a piecewise continuous distribution. In this case, we may naturally choose the base distribution p_0 as the differentiable surrogate function to draw samples from p_c when using GF-SVGD.

Algorithm 2 Goodness-of-fit testing (GF-KSD)

Input: Sample $\{\boldsymbol{z}_i\}_{i=1}^n \sim q_*$ and its corresponding continuous-valued $\{\boldsymbol{x}_i\}_{i=1}^n \sim q_c$, and null distribution p_c. Base function $p_0(\boldsymbol{x})$ and bootstrap sample size m.
Goal: Test $H_0 : q_c = p_c$ vs. $H_1 : q_c \neq p_c$.
-Compute test statistics $\hat{\mathbb{S}}$ by (18).
-Compute m bootstrap sample $\hat{\mathbb{S}}^*$ by (19).
-Reject H_0 with significance level α if the percentage of $\{\hat{\mathbb{S}}^*\}_{i=1}^m$ that satisfies $\hat{\mathbb{S}}^* > \hat{\mathbb{S}}$ is less than α.

3.1 Goodness-of-fit Test on Discrete Distribution

Our approach implies a new method for goodness of fit test of discrete distributions, which we now explore. Given i.i.d. samples $\{\boldsymbol{z}_i\}_{i=1}^n$ from an *unknown* distribution q_*, and a candidate discrete distribution p_*, we are interested in testing $H_0 : q_* = p_*$ vs. $H_1 : q_* \neq p_*$.

Our idea is to transform the testing of discrete distributions $q_* = p_*$ to their continuous parameterizations. Let Γ be a even partition of a base distribution p_0, and p_c and q_c are the continuous parameterizations of p_* and q_* following our construction, respectively, that is,

$$p_c(\boldsymbol{x}) \propto p_0(\boldsymbol{x})p_*(\Gamma(\boldsymbol{x})), \quad q_c(\boldsymbol{x}) \propto p_0(\boldsymbol{x})q_*(\Gamma(\boldsymbol{x})).$$

Obviously, $p_c = q_c$ implies that $p_* = q_*$ (following the definition of continuous parameterization). This allows us to transform the problem to a goodness-of-fit test of continuous distributions, which we is achieved by testing if the gradient-free KSD (9) equals zero, $H_0 : q_c = p_c$ vs. $H_1 : q_c \neq p_c$.

In order to implement our idea, we need to convert the discrete sample $\{\boldsymbol{z}_i\}_{i=1}^n$ from q_* to a continuous sample $\{\boldsymbol{x}_i\}_{i=1}^n$ from the corresponding (unknown) continuous distribution q_c. To achieve, note that when $\boldsymbol{x} \sim q_c$ and $\boldsymbol{z} = \Gamma(\boldsymbol{x})$, the posterior distribution $\boldsymbol{x}$ of giving $\boldsymbol{z} = \boldsymbol{a}_i$ equals

$$q(\boldsymbol{x} \mid \boldsymbol{z} = \boldsymbol{a}_i) \propto p_0(\boldsymbol{x})\mathbb{I}(\Gamma(\boldsymbol{x}) = \boldsymbol{a}_i),$$

which corresponds to sampling a truncated version of p_0 inside the region defined $\{\boldsymbol{x}\colon\ \Gamma(\boldsymbol{x}) = \boldsymbol{a}_i\}$. This can be implemented easily for the simple choices of p_0 and Γ. For example, in the case when p_0 is the product distribution in (14) and Γ is the concatenation of the quantile-based partition in (13), we can sample $\boldsymbol{x} \mid \boldsymbol{z} = \boldsymbol{a}_i$ by sample $\boldsymbol{y}$ from Uniform$([\eta_{i-1}, \eta_i)^d)$ and obtain $\boldsymbol{x}$ by $\boldsymbol{x} = F^{-1}(\boldsymbol{y})$ where F^{-1} is the inverse CDF of p_0. To better understand how to transform the discrete data $\{\boldsymbol{z}_i\}_{i=1}^n$ to continuous samples $\{\boldsymbol{x}_i\}_{i=1}^n$, please refer to Appendix C for detail.

With the continuous data, the problem is reduced to testing if $\{\boldsymbol{x}_i\}_{i=1}^n \sim q_c$ is drawn from p_c. We achieve this using gradient-free KSD, similar to [18, 3]. In particular, using the surrogate $\rho(\boldsymbol{x})$ in (16), the GF-KSD between the transformed distributions q_c and p_c is

$$\mathbb{S}(q_c, p_c) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim q_c}[w(\boldsymbol{x})\kappa_\rho(\boldsymbol{x}, \boldsymbol{x}')w(\boldsymbol{x}')], \tag{17}$$

where κ_ρ is defined in (10). Under mild conditions [18], it can similarly derived that $\mathbb{S}(q_c, p_c) = 0$ iff $q_c = p_c$.

With $\{\boldsymbol{x}_i\}_{i=1}^n$ from q_c, the GF-KSD between q_c and p_c can be estimated by the U-statistics,

$$\hat{\mathbb{S}}(q_c, p_c) = \frac{1}{(n-1)n} \sum_{1\le i\neq j\le n} w(\boldsymbol{x}_i)\kappa_\rho(\boldsymbol{x}_i, \boldsymbol{x}_j)w(\boldsymbol{x}_j). \tag{18}$$

In practice, we can employ the U-statistics $\hat{\mathbb{S}}(q_c, p_c)$ to perform the goodness-of-fit test based on the similar result from [18, 3], which replaces their KSD with gradient-free KSD in (17) and follow other procedure.

Bootstrap Sample The asymptotic distribution of $\hat{\mathbb{S}}(q_c, p_c)$ under the hypothesis cannot be evaluated. In order to perform goodness-of-fit test,

we draw random multinomial weights $u_1, \cdots, u_n \sim \text{Multi}(n; 1/n, \cdots, 1/n)$, and calculate

$$\hat{\mathbb{S}}^*(q_c, p_c) = \sum_{i \neq j} (u_i - \frac{1}{n}) w(\boldsymbol{x}_i) \kappa_\rho(\boldsymbol{x}_i, \boldsymbol{x}_j) w(\boldsymbol{x}_j) (u_j - \frac{1}{n}). \tag{19}$$

We repeat this process by m times and calculate the critical values of the test by taking the $(1-\alpha)$-th quantile of the bootstrapped statistics $\{\hat{\mathbb{S}}^*(q_c, p_c)\}$. The whole procedure is summarized in Alg. 2.

4 EXPERIMENTS

We apply our algorithm to a number of large scale discrete distributions to demonstrate its empirical effectiveness. We start with illustrating our algorithm on sampling from a simple one-dimensional categorical distribution. We then apply our algorithm to sample from discrete Markov random field, Bernoulli restricted Boltzman machine. Then we apply our method to learn ensemble models of binarized neural networks (BNN). Finally, we perform experiments on goodness-of-fit test.

4.1 Statistical Models

Ising Model The Ising model [13] is widely used in Markov random field. Consider an (undirected) graph $G = (V, E)$, where each vertex $i \in V$ is associated with a binary spin, which consists of $\boldsymbol{z} = (z_1, \cdots, z_d)$. The probability mass function is $p(\boldsymbol{z}) = \frac{1}{Z} \sum_{(i,j) \in E} \sigma_{ij} z_i z_j$, $z_i \in \{-1, 1\}$, σ_{ij} is edge potential and Z is normalization constant, which is infeasible to calculate when d is high.

Bernoulli restricted Boltzmann Machine (RBM) Bernoulli RBM[11] is an undirected graphical model consisting of a bipartite graph between visible variables z and hidden variables h. In a Bernoulli RBM, the joint distribution of visible units $\boldsymbol{z} \in \{-1, 1\}^d$ and hidden units $h \in \{-1, 1\}^M$ is given by

$$p(\boldsymbol{z}, \boldsymbol{h}) \propto \exp(-E(\boldsymbol{z}, \boldsymbol{h})) \tag{20}$$

where $E(\boldsymbol{z}, \boldsymbol{h}) = -(\boldsymbol{z}^\top W \boldsymbol{h} + \boldsymbol{z}^\top b + \boldsymbol{h}^\top c)$, $W \in \mathbb{R}^{d \times M}$ is the weight, $b \in \mathbb{R}^d$ and $c \in \mathbb{R}^M$ are the bias. Marginalizing out the hidden variables $\boldsymbol{h}$, the probability mass function of $\boldsymbol{z}$ is given by $p(\boldsymbol{z}) \propto \exp(-E(\boldsymbol{z}))$, with free energy $E(\boldsymbol{z}) = -\boldsymbol{z}^\top b - \sum_k \log(1 + \varphi_k)$, where $\varphi_k = \exp(W_{k*}^\top \boldsymbol{z} + c_k)$ and W_{k*} is the k-th row of W.

4.2 Investigation of the Choice of Transform

There are many choices of the base function p_0 and the transform. We investigate the optimal choice of the transform on categorical distribution in Fig. 2. In Fig. 2(b, c, d), the base is chosen as $p_0(x) = \sum_{i=1}^5 p_i \mathcal{N}(x; \mu_i, 1.0)$ for different $\boldsymbol{\mu}$. The base $p_0(x)$ in Fig. 2(a) can be seen as $\boldsymbol{\mu} = (0., 0., 0., 0., 0.)$. We observe that with simple Gaussian base in Fig. 2(a), the transformed target is easier to draw samples, compared with the multi-modal target in Fig. 2(c, d). This suggests that Gaussian base p_0 is a simple but powerful choice as its induced transformed target is easy to sample by GF-SVGD.

4.3 Experiments on Sampling

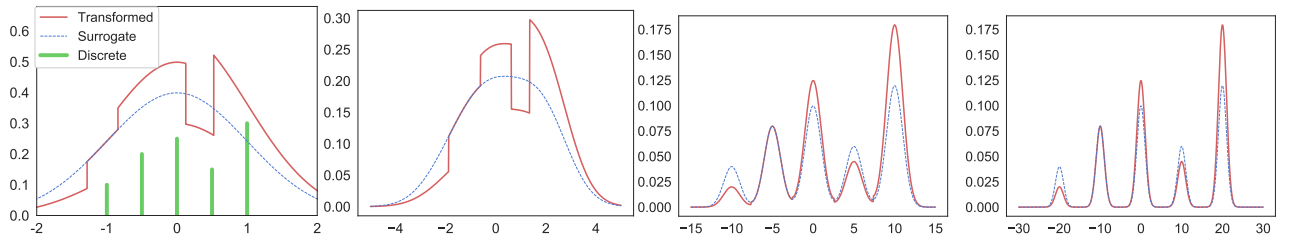
Ising Model We evaluate the mean square error (MSE) for estimating the mean value $\mathbb{E}_{p_*}[\boldsymbol{z}]$ in each dimension. As shown in Section 3, it is easy to map $\boldsymbol{z}$ to the piecewise continuous distribution of $\boldsymbol{x}$ in each dimension. We take $\Gamma(\boldsymbol{x}) = \text{sign}(\boldsymbol{x})$, with the transformed target $p_c(\boldsymbol{x}) \propto p_0(\boldsymbol{x}) p_*(\text{sign}(\boldsymbol{x}))$. The base function $p_0(\boldsymbol{x})$ is taken to be the standard Gaussian distribution on $\mathbb{R}^d$. We apply GF-SVGD to sample from $p_c(\boldsymbol{x})$ with the surrogate $\rho(\boldsymbol{x}) = p_0(\boldsymbol{x})$. The initial particles $\{\boldsymbol{x}_i\}$ is sampled from $\mathcal{N}(-2, 1)$ and update $\{\boldsymbol{x}_i\}$ by 500 iterations. We obtain $\{\boldsymbol{z}_i\}_{i=1}^n$ by $\boldsymbol{z}_i = \Gamma(\boldsymbol{x}_i)$, which approximates the target model $p_*(\boldsymbol{z})$. We compared our algorithm with both exact Monte Carlo (MC) and Gibbs sampling which is iteratively sampled over each coordinate and use same initialization (in terms of $\boldsymbol{z} = \Gamma(\boldsymbol{x})$) and number of iterations as ours.

Fig. 3(a) shows the log MSE over the log sample size. With fixed σ_s and σ_p, our method has the smallest MSE and the MSE has the convergence rate $\mathcal{O}(1/n)$. The correlation σ_p indicates the difficulty of inference. As $|\sigma_p|$ increases, the difficulty increases. As shown in Fig. 3(b), our method can lead to relatively less MSE in the chosen range of correlation. It is interesting to observe that as $\sigma_p \to 0$, our method significantly outperforms MC and Gibbs sampling.

Bernoulli Restricted Boltzmann Machine The base function $p_0(\boldsymbol{x})$ is the product of the p.d.f. of the standard Gaussian distribution over the dimension d. Applying the map $\boldsymbol{z} = \Gamma(\boldsymbol{x}) = \text{sign}(\boldsymbol{x})$, the transformed piecewise continuous target is $p_c(\boldsymbol{x}) \propto p_0(\boldsymbol{x}) p_*(\text{sign}(\boldsymbol{x}))$. Different from previous example, we construct a simple and more powerful surrogate distribution $\rho(\boldsymbol{x}) \propto \widetilde{p}(\sigma(\boldsymbol{y})) p_0(\boldsymbol{x})$ where $\widetilde{p}(\sigma(\boldsymbol{y}))$ is differentiable approximation of p_* and σ is defined as

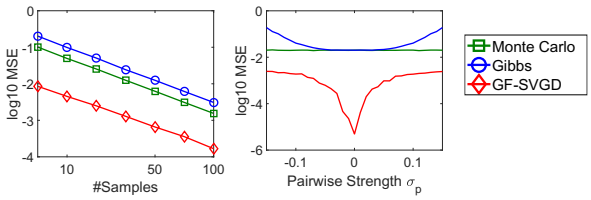
$$\sigma(\boldsymbol{x}) = \frac{2}{1 + \exp(-\boldsymbol{x})} - 1, \tag{21}$$

and $\sigma(\boldsymbol{x})$ approximates $\text{sign}(\boldsymbol{x})$. Intuitively, it relaxes p_c to a differentiable surrogate with tight approximation.



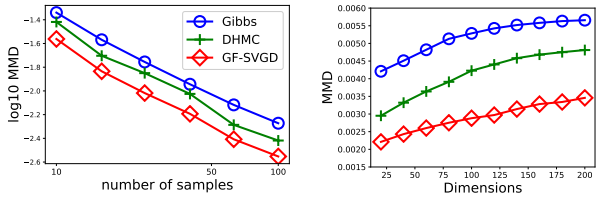
(a), Base $p_0(x) = \mathcal{N}(x; 0, 1)$ (b), $\boldsymbol{\mu} = (-2, -1, 0, 1, 2)$ (c), $\boldsymbol{\mu} = (-10, -5, 0, 5, 10)$ (d), $\boldsymbol{\mu} = (-20, -10, 0, 10, 20)$

Figure 2: Illustrating the construction of $p_c(x)$ (red line) of a five-state discrete distribution p_* (green bars) and the choice of transform. $p_*(z)$ takes values $[-2, -1, 0, 1, 2]$ with probabilities $[p_1, p_2, p_3, p_4, p_5] = [0.1, 0.2, 0.25, 0.15, 0.3]$ respectively. $K = 5$. The dash blue is the surrogate using base p_0. Let $p(y)$, $y \in [0, 1)$ be the stepwise density, $p(y \in [\frac{i-1}{K}, \frac{i}{K})) = p_i$, for $i = 1, \cdots, K$. In (b, c, d), the base is chosen as $p_0(x) = \sum_{i=1}^5 p_i \mathcal{N}(x; \mu_i, 1.)$ and $\boldsymbol{\mu} = (\mu_1, \mu_2, \mu_3, \mu_4, \mu_5)$. The base $p_0(x)$ in (a) can be seen as $\boldsymbol{\mu} = (0., 0., 0., 0., 0.)$. Let $F(x)$ be the c.d.f. of $p_0(x)$. With variable transform $x = F^{-1}(y)$, the transformed target is $p_c(x) = p(F(x))p_0(x)$.



(a) Fixed σ_p (b) Fixed n

Figure 3: Performance of different methods on the Ising model with 10×10 grid. We compute the MSE for estimating $\mathbb{E}[\boldsymbol{z}]$ in each dimension. Let $\sigma_{ij} = \sigma_p$. In (a), we fix $\sigma_p = 0.1$ and vary the sample size n. In (b), we fix the sample size $n = 20$ and vary σ_p from -0.15 to 0.15. In both (a) and (b) we evaluate the log MSE based on 200 trails. DHMC has similar performance as Monte Carlo and is omitted for clear figure.



(a) Fix dimension (b) Fix sample size

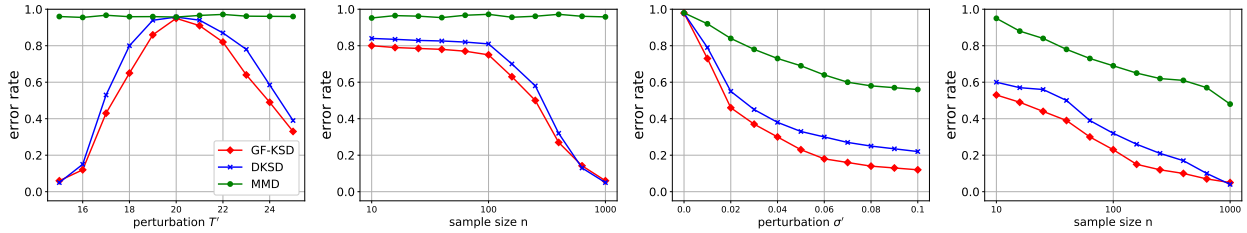
Figure 4: Bernoulli RBM with number of visible units $M = 25$. In (a), we fix the dimension of visible variables $d = 100$ and vary the number of samples $\{\boldsymbol{z}^j\}_{j=1}^n$. In (b), we fix the number of samples $n = 100$ and vary the dimension of visible variables d. We calculate the MMD between the sample of different methods and the ground-truth sample. MSE is provided on Appendix.

We compare our algorithm with Gibbs sampling and discontinuous HMC(DMHC, [22]). In Fig. 4, W is drawn from $N(0, 0.05)$, both b and c are drawn from $N(0, 1)$. With 10^5 iterations of Gibbs sampling, we draw 500 parallel chains to take the last sample of each chain to get 500 ground-truth samples. We run Gibbs, DHMC and GF-SVGD at 500 iterations for fair comparison. In Gibbs sampling, $p(\boldsymbol{z} \mid \boldsymbol{h})$ and $p(\boldsymbol{h} \mid \boldsymbol{z})$ are iteratively sampled. In DHMC, a coordinate-wise integrator with Laplace momentum is applied to update the discontinuous states. We calculate MMD [8] between the ground truth sample and the sample drawn by different methods. The kernel in MMD is the exponentiated Hamming kernel from [27], defined as, $k(\boldsymbol{z}, \boldsymbol{z}') = \exp(-H(\boldsymbol{z}, \boldsymbol{z}'))$, where $H(\boldsymbol{z}, \boldsymbol{z}') := \frac{1}{d} \sum_{i=1}^d \mathbb{I}_{\{z_i \neq z_i'\}}$ is normalized Hamming distance. We perform experiments by fixing $d = 100$ and varying sample size in Fig. 4(a) and fixing $n = 100$ and varying d. Fig. 4(a) indicates that the samples from our method match the ground truth samples better in terms of MMD. Fig. 4(b) shows that the performance of our method is least sensitive to the dimension of the model than that of Gibss and DHMC. Both Fig. 4(a) and Fig. 4(b) show that our algorithm converges fastest.

4.4 Learning Binarized Neural Network

We slightly modify our algorithm to train binarized neural network (BNN), where both the weights and activation functions are binary ± 1. BNN has been studied extensively because of its fast computation, energy efficiency and low memory cost [24, 12, 4, 29]. The challenging problem in training BNN is that the gradients of the weights cannot be backpropagated through the binary activation functions because the gradients are zero almost everywhere.

We train an ensemble of n neural networks (NN) with the same architecture ($n \geq 2$). Let $\boldsymbol{w}_i^b$ be the binary weight of model i, for $i = 1, \cdots, n$, and $p_*(\boldsymbol{w}_i^b; D)$



(a) $T = 20$ (n=1000) (b) $T = 20$ and $T' = 15$ (c) $\sigma = 0$ (n=100) (d) $\sigma = 0$ and $\sigma' = 15$

Figure 5: Goodness-of-fit test on Ising model (a, b) and Bernoulli RBM (c, d) with significant level $\alpha = 0.05$. In (a, b), p_* and q_* has temperature T and T' respectively. In (c, d), p_* has $W \sim \mathcal{N}(0, 1/M)$ and q_* has $W + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma')$. b and c in p_* and q_* are the same. In (a, c) we vary the parameters of q_*. We fix the models and vary the sample size n in (b, d). We test $H_0 : q_* = p_*$ vs. $H_1 : q_* \neq p_*$.

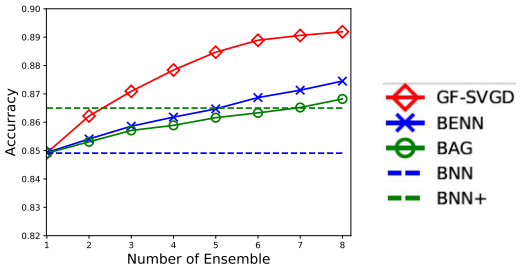


Figure 6: Comparison of different methods using AlexNet with binarized weights and activation on CIFAR10 dataset. We compare our GF-SVGD with BNN [12], BNN+[4] and BENN [29]. "BAG" denote models are independently trained and linearly averaged the softmax output for prediction. Performance is based on the accuracy of different models w.r.t. ensemble size n on test data.

be the target probability model with softmax layer as last layer given the data D. Learning the target probability model is framed as drawing n samples $\{\boldsymbol{w}_i^b\}_{i=1}^n$ to approximate the posterior distribution $p_*(\boldsymbol{w}^b; D)$. We train an ensemble of n neural networks (NN) with the same architecture ($n \geq 2$). Let $\boldsymbol{w}_i^b$ be the binary weight of model i, for $i = 1, \cdots, n$, and $p_*(\boldsymbol{w}_i^b; D)$ be the target probability model with softmax layer as last layer given the data D. Learning the target probability model is framed as drawing n samples $\{\boldsymbol{w}_i^b\}_{i=1}^n$ to approximate the posterior distribution $p_*(\boldsymbol{w}^b; D)$. This involves sampling $\{\boldsymbol{w}_i^b\}_{i=1}^n$ from discrete distributions $p_*(\boldsymbol{w}^b; D)$, where our proposed sampling algorithm can be applied. Please refer to Appendix B for the detail.

We test our ensemble algorithm by using binarized AlexNet [15] on CIFAR-10 dataset. We use the same setting for AlexNet as that in [29], which can be found in Appendix E. We compare our ensemble algorithm with typical ensemble method using bagging and AdaBoost (BENN, [29]), BNN [12] and BNN+[4]. Both BNN and BNN+ are trained on a single model with same network. From Fig. 6, we can see that all three ensemble methods (GF-SVGD, BAG and BENN) improve test accuracy over one single model (BNN and BNN+). To use the same setting for all methods, we don't use data augmentation or pre-training. Our ensemble method has the highest accuracy among all three ensemble methods. This is because our ensemble model are sufficiently interactive during training and our ensemble models $\{\boldsymbol{w}_i\}$ in principle are approximating the posterior distribution $p(\boldsymbol{w}; D)$.

4.5 Experiments on Goodness-of-fit Testing

We perform goodness-of-fit tests on Ising model and Bernoulli RBM in Fig. 5, which shows type-II error rate (False negative error). The data $\{\boldsymbol{z}_i\}_{i=1}^n$ is transformed to its corresponding continuous-valued samples $\{\boldsymbol{y}_i\}_{i=1}^n$, $y_i^j \in [0, \frac{1}{2})$, if $z_i^j = -1$; $y_i^j \in [\frac{1}{2}, 1)$, if $z_i^j = 1$. Let F be the c.d.f. of Gaussian base p_0. By the same variable transform induced from F, we obtain data $\boldsymbol{x}^i = F^{-1}(\boldsymbol{y}^i)$ and the transformed $p_c(\boldsymbol{x})$. The surrogate ρ is chosen as that in sampling. Fig. 5 shows that our GF-KSD performs much better than DKSD [27] and MMD [8] when the sample size n is relatively small and the difference between q_* and p_* is within some range.

5 CONCLUSION

In this paper, we propose a simple yet general framework to perform approximate inference and goodness-of-fit test on discrete distributions. We demonstrate the effectiveness of our proposed algorithm on a number of discrete graphical models. Based on our sampling method, we propose a new promising approach for learning an ensemble model of binarized neural networks. Future research includes applying our ensemble method to train BNN with larger networks such as VGG net and larger dataset such as ImageNet dataset and extending our method to learn deep generative models with discrete distributions.

References

[1] H. M. Afshar and J. Domke. Reflection, refraction, and hamiltonian monte carlo. In *NIPS*, 2015.

[2] S.-S. Ahn, M. Chertkov, and J. Shin. Synthesis of mcmc and belief propagation. In *Advances in Neural Information Processing Systems*, 2016.

[3] K. Chwialkowski, H. Strathmann, and A. Gretton. A kernel test of goodness of fit. In *ICML*, 2016.

[4] S. Darabi, M. Belbahri, M. Courbariaux, and V. P. Nia. Bnn+: Improved binary network training. *arXiv:1812.11800*, 2018.

[5] C. Daskalakis, N. Dikkala, and G. Kamath. Testing ising models. *IEEE Transactions on Information Theory*, 2019.

[6] R. Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *Learning in graphical models*. Springer, 1998.

[7] V. Dinh, A. Bilge, C. Zhang, and F. A. Matsen IV. Probabilistic path hamiltonian monte carlo. In *ICML*, 2017.

[8] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar), 2012.

[9] J. Han and Q. Liu. Stein variational adaptive importance sampling. *arXiv preprint arXiv:1704.05201*, 2017.

[10] J. Han and Q. Liu. Stein variational gradient descent without gradient. *arXiv preprint arXiv:1806.02775*, 2018.

[11] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[12] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks. In *NIPS*, 2016.

[13] E. Ising. *Beitrag zur theorie des ferro-und paramagnetismus*. PhD thesis, Hamburg, 1924.

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[16] Q. Liu. Stein variational gradient descent as gradient flow. In *Advances in neural information processing systems*, 2017.

[17] Q. Liu, J. W. Fisher III, and A. T. Ihler. Probabilistic variational bounds for graphical models. In *Advances in Neural Information Processing Systems*, 2015.

[18] Q. Liu, J. Lee, and M. Jordan. A kernelized stein discrepancy for goodness-of-fit tests. In *International Conference on Machine Learning*, 2016.

[19] Q. Liu and D. Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *NIPS*, pages 2378–2386, 2016.

[20] Q. Lou, R. Dechter, and A. T. Ihler. Dynamic importance sampling for anytime bounds of the partition function. In *Advances in Neural Information Processing Systems*, 2017.

[21] A. Martín del Campo, S. Cepeda, and C. Uhler. Exact goodness-of-fit testing for the ising model. *Scandinavian Journal of Statistics*, 2017.

[22] A. Nishimura, D. Dunson, and J. Lu. Discontinuous hamiltonian monte carlo for discrete parameters and discontinuous likelihoods. *arXiv:1705.08510*, 2019.

[23] A. Pakman and L. Paninski. Auxiliary-variable exact hamiltonian monte carlo samplers for binary distributions. In *NIPS*, pages 2490–2498, 2013.

[24] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*. Springer, 2016.

[25] G. Valiant and P. Valiant. Instance optimal learning of discrete distributions. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*. ACM, 2016.

[26] M. J. Wainwright, M. I. Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 2008.

[27] J. Yang, Q. Liu, V. Rao, and J. Neville. Goodness-of-fit testing for discrete distributions via stein discrepancy. In *ICML*, 2018.

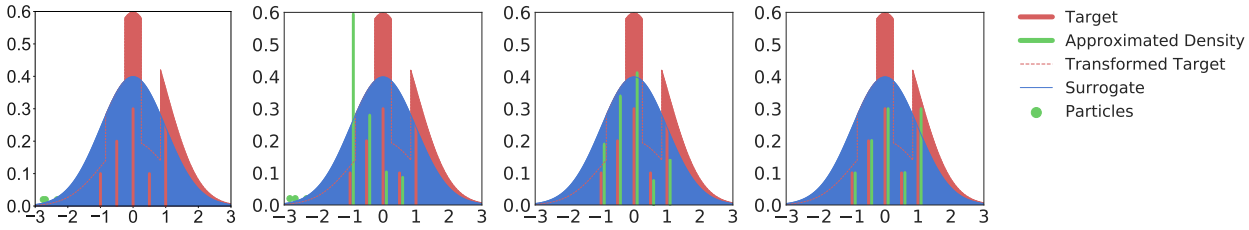
[28] Y. Zhang, Z. Ghahramani, A. J. Storkey, and C. A. Sutton. Continuous relaxations for discrete hamiltonian monte carlo. In *NIPS*, 2012.

[29] S. Zhu, X. Dong, and H. Su. Binary ensemble neural network: More bits per network or more networks per bit? *arXiv:1806.07550*, 2018.

Appendix

A Additional Experimental Result

Result on Categorical Distribution We apply our algorithm to sample from one-dimensional categorical distribution $p_*(z)$ shown in red bars in Fig. 7, defined on $\mathcal{Z} := \{-1, -0.5, 0, 0.5, 1\}$ with corresponding probabilities $\{0.1, 0.2, 0.3, 0.1, 0.3\}$. The blue dash line is the surrogate distribution $\rho(x) = p_0(x)$, where the base function $p_0(x)$ is the p.d.f. of standard Gaussian distribution. The red dash line is the transformed piecewise continuous density $p_c(x) \propto p_0(x)p_*(\Gamma(x))$, where $\Gamma(x) = a_i$ if $x \in [\eta_{i-1}, \eta_i)$ and η_i is $i/5$-th quantile of standard Gaussian distribution. We apply Algorithm 1 to draw a set of samples $\{x_i\}_{i=1}^n$ (shown in green dots) to approximate the transformed target distribution. Then we can obtain a set of samples $\{z_i\}_{i=1}^n$ by $z_i = \Gamma(x_i))$, to get an approximation of the original categorical distribution.

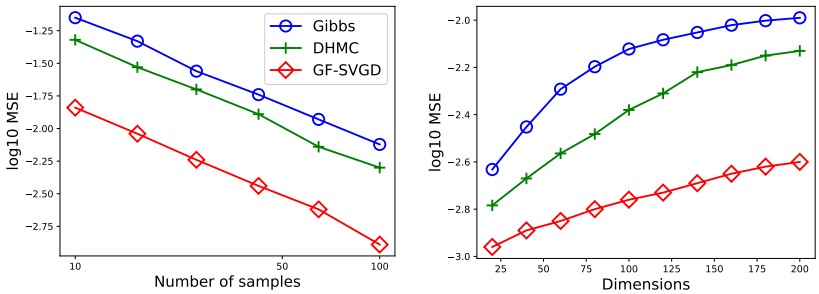


(a) 0th iteration (b) 25th iteration (c) 50th iteration (d) 100th iteration

Figure 7: Evolution of real-valued particles $\{x_i\}_{i=1}^n$ (in green dots) by our discrete sampler in Alg.1 on a one-dimensional categorical distribution. (a-d) shows particles $\{x^i\}$ at iteration 0, 10, 50 and 100 respectively. The categorical distribution is defined on states $z \in \{-1, -0.5, 0, 0.5, 1\}$ denoted by a_1, a_2, a_3, a_4, a_5, with probabilities $\{0.1, 0.2, 0.3, 0.1, 0.3\}$ denoted by c_1, c_2, c_3, c_4, c_5, respectively. $p_*(z = a_i) = c_i$. The base function is $p_0(x)$, shown in blue line. The transformed target to be sampled $p_c(x) \propto p_0(x)p_*(\Gamma(x))$, where $\Gamma(x) = a_i$ if $x \in [\eta_{i-1}, \eta_i)$ and η_i is $i/5$-th quantile of standard Gaussian distribution. The surrogate distribution $\rho(x)$ is chosen as $p_0(x)$. We obtain discrete samples $\{z_i\}_{i=1}^n$ by $z_i = \Gamma(x_i)$.

As shown in Fig 7, the empirical distribution of the discretized sample $\{\boldsymbol{z}_i\}_{i=1}^n$ (shown in green bars) aligns closely with the true distribution (the red bars) when the algorithm converges (e.g., at the 100-th iteration).

Results on Bernoulli RBM The probability model is given in (20) and the score function is derived in Section 5.3 [9]. We also evaluate the sample quality based on the mean square error (MSE) between the estimation and the ground truth value. From Fig. 8(a), we can see that when fixing the dimension of the distribution $p_*(\boldsymbol{z})$, our sampling method has much lower MSE than Gibbs and DHMC. In Fig. 8(b), as the dimension of the model increases, our sampling method has relatively better MSE than that of Gibbs and DHMC.



(a) Fix dimension (b) Fix sample size

Figure 8: Bernoulli RBM with number of visible units $M = 25$. In (a), we fix the dimension of visible variables $d = 100$ and vary the number of samples $\{\boldsymbol{z}^j\}_{j=1}^n$. In (b), we fix the number of samples $n = 100$ and vary the dimension of visible variables d. We calculate the MSE for estimating the mean $\mathbb{E}[z]$ (lower is better).

B Training BNN Algorithm

In this section, we provide the procedure of our principled ensemble algorithm to train binarized neural network. We train an ensemble of n neural networks (NN) with the same architecture ($n \geq 2$). Let $\boldsymbol{w}_i^b$ be the binary weight of model i, for $i = 1, \cdots, n$, and $p_*(\boldsymbol{w}_i^b; D)$ be the target probability model with softmax layer as last layer given the data D. Learning the target probability model is framed as drawing n samples $\{\boldsymbol{w}_i^b\}_{i=1}^n$ to approximate the posterior distribution $p_*(\boldsymbol{w}^b; D)$. We apply multi-dimensional transform $\boldsymbol{F}$ to transform the original discrete-valued target to the target distribution of real-valued $\boldsymbol{w} \in \mathbb{R}^d$. Let $p_0(w)$ be the base function, which is the product of the p.d.f. of the standard Gaussian distribution over the dimension d. Based on the derivation in Section 3, the distribution of $\boldsymbol{w}$ has the form $p_c(\boldsymbol{w}; D) \propto p_*(\text{sign}(\boldsymbol{w}); D)p_0(\boldsymbol{w})$ with weight $\boldsymbol{w}$ and the sign function is applied to each dimension of $\boldsymbol{w}$. To backpropagate the gradient to the non-differentiable target, we construct a surrogate probability model $\rho(\boldsymbol{w}; D)$ which approximates $\text{sign}(\boldsymbol{w})$ in the transformed target by $\sigma(\boldsymbol{x})$ and relax the binary activation function $\{-1, 1\}$ by σ, where σ is defined by (21), denoted by $\widetilde{p}(\sigma(\boldsymbol{w}); D)p_0(\boldsymbol{w})$. Here $\widetilde{p}(\sigma(\boldsymbol{w}); D)$ is a differentiable approximation of $p_*(\text{sign}(\boldsymbol{w}); D)$. Then we apply GF-SVGD to update $\{\boldsymbol{w}_i\}$ to approximate the transformed target distribution of $p_c(\boldsymbol{w}; D)$ of $\boldsymbol{w}$ as follows, $\boldsymbol{w}_i \leftarrow \boldsymbol{w}_i + \frac{\epsilon_i}{\Omega}\Delta\boldsymbol{w}_i$, $\forall i = 1, \cdots, n$,

$$\Delta\boldsymbol{w}_i \leftarrow \sum_{j=1}^{n} \gamma_j [\nabla_{\boldsymbol{w}} \log \rho(\boldsymbol{w}_j; D_i) k(\boldsymbol{w}_j, \boldsymbol{w}_i) + \nabla_{\boldsymbol{w}_j} k(\boldsymbol{w}_j, \boldsymbol{w}_i)] \tag{22}$$

where D_i is batch data i and $\mu_j = \rho(\boldsymbol{w}_j; D_i)/p_c(\boldsymbol{w}_j; D_i)$, $H(t) \stackrel{\text{def}}{=} \sum_{j=1}^n \mathbb{I}(\mu_j \geq t)/n$, $\gamma_j = (H(\boldsymbol{w}_j))^{-1}$ and $\Omega = \sum_{j=1}^n \gamma_j$. Note that we don't need to calculate the cumbersome term $p_0(\boldsymbol{w})$ as it can be canceled from the ratio between the surrogate distribution and the transformed distribution. In practice, we find a more effective way to estimate this density ratio denoted by γ_j. Intuitively, this corresponds to assigning each particle a weight according to the rank of its density ratio in the population. Algorithm 3 on Appendix B can be viewed as a new form of ensemble method for training NN models with discrete parameters.

Algorithm 3 GF-SVGD on training BNN

Inputs: training set D and testing set D_{test}
Outputs: classification accuracy on testing set.
Initialize full-precision models $\{\boldsymbol{w}^i\}_{i=1}^n$ and its binary form $\{\boldsymbol{w}_i^b\}_{i=1}^n$ where $\boldsymbol{w}_i^b = \text{sign}(\boldsymbol{w}^i)$.
while not converge **do**
-Sample n batch data $\{D_i\}_{i=1}^n$.
-Calculate the true likelihood $p_c(\boldsymbol{w}_i; D_i) \propto p_*(\text{sign}(\boldsymbol{w}_i); D_i)p_0(x)$
-Relax $\boldsymbol{w}_b^i$ with $\sigma(\boldsymbol{w}_i)$
-Relax each sign activation function to the smooth function defined in (21) to get $\widetilde{p}$
-Calculate the surrogate likelihood $\rho(\boldsymbol{w}^i; D_i) \propto \widetilde{p}(\sigma(\boldsymbol{w}_i); D_i)p_0(\boldsymbol{x})$
-$\boldsymbol{w}_i \leftarrow \boldsymbol{w}_i + \Delta\boldsymbol{w}_i$, $\forall i = 1, \cdots, n$, where $\Delta\boldsymbol{w}_i$ is defined in (22).
-Clip $\{\boldsymbol{w}_i\}$ to interval $(-1, 1)$ for stability.
end while
-Calculate the probability output by softmax layer $p(\boldsymbol{w}_i^b; D_{\text{test}})$
-Calculate the average probability $f(\boldsymbol{w}_b; D_{\text{test}}) \leftarrow \sum_{i=1}^n p(\boldsymbol{w}_i^b; D_{\text{test}})$
Output test accuracy from $f(\boldsymbol{w}_b; D_{\text{test}})$.

C Transform Discrete Samples to Continuous Samples for Goodness-of-fit Test

Let F be the c.d.f. of Gaussian base density p_0. Let us first illustrate how to transform one-dimensional samples $\{z_i\}_{i=1}^n$ to continuous samples.

1. Given discrete data $\{z_i\}_{i=1}^n$. Let $\{a_j\}_{j=1}^K$ are possible discrete states. Assume K is large so that for any z_i, we have $z_i = a_j$ for one j.
2. For any z_i such as $z_i = a_j$, randomly sample $y_i \in [\frac{j-1}{K}, \frac{j}{K})$. We obtain data $\{y_i\}_{i=1}^n$.
3. Apply $x = F^{-1}(y)$, we obtain data $\{x_i\}_{i=1}^n$.

For $\boldsymbol{x} = (x^1, \cdots, x^d)$, let $F(\boldsymbol{x}) = (F_1(x^1), \cdots, F_d(x^d)$, where F_i is the c.d.f. of Gaussian density $p_{0,i}(x^i)$. We apply the above one-dimensional transform to each dimension of $\{\boldsymbol{z}_i\}_{i=1}^n$, $\boldsymbol{z}_i = (z_i^1, \cdots, z_i^d)$. We can easily obtain the continuous data $\{\boldsymbol{x}_i\}_{i=1}^n$.

D Proofs

In the following, we prove proposition 4.

Proposition 4 Assume Γ is an even partition of $p_0(\boldsymbol{x})$, and $p_c(\boldsymbol{x}) = Kp_0(\boldsymbol{x})p_*(\Gamma(\boldsymbol{x}))$, where K severs as a normalization constant, then $(p_c,\ \Gamma)$ is a continuous parameterisation of p_*.

Proof. We just need to verify that (11) holds.

$$\begin{aligned}
&\int p_c(\boldsymbol{x})\mathbb{I}[\boldsymbol{a}_i = \Gamma(\boldsymbol{x})]d\boldsymbol{x} \\
&= K\int p_0(\boldsymbol{x})p_*(\Gamma(\boldsymbol{x}))\mathbb{I}[a_i = \Gamma(\boldsymbol{x})]d\boldsymbol{x} \\
&= K\int p_0(\boldsymbol{x})p_*(\boldsymbol{a}_i)\mathbb{I}[\boldsymbol{a}_i = \Gamma(\boldsymbol{x})]d\boldsymbol{x} \\
&= Kp_*(\boldsymbol{a}_i)\int p_0(\boldsymbol{x})\mathbb{I}[\boldsymbol{a}_i = \Gamma(\boldsymbol{x})]d\boldsymbol{x} \\
&= p_*(\boldsymbol{a}_i),
\end{aligned}$$

where the last step follows (12). □

E Detail of Experiments and Network Architecture

In all experiments, we use RBF kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\|\boldsymbol{x} - \boldsymbol{x}'\|^2/h)$ for the updates of our proposed algorithms; the bandwidth h is taken to be h=med$^2/(2\log(n+1))$ where med is the median of the current n particles. Adam optimizer [14] is applied to our proposed algorithms for accelerating convergence. $\epsilon = 0.0001$ works for all the experiments.

We use the same AlexNet as [29], which is illustrated in the following.

Layer	Type	Parameters
1	Conv	Depth: 96, K: 11×11, S: 4, P:0
2	Relu	-
3	MaxPool	K: 3×3, S: 2
4	BatchNorm	-
5	Conv	Depth: 256, K: 5×5, S: 1, P:1
6	Relu	-
7	MaxPool	K: 3×3, S: 2
8	BatchNorm	-
9	Conv	Depth: 384, K: 3×3, S: 1, P:1
10	Relu	-
11	Conv	Depth: 384, K: 3×3, S: 1, P:1
12	Relu	-
13	Conv	Depth: 256, K: 3×3, S: 1, P:1
14	Relu	-
15	MaxPool	K: 3×3, S: 2
16	Dropout	$p = 0.5$
17	FC	Width=4096
18	Relu	-
19	Dropout	$p = 0.5$
20	FC	Width=4096
21	Relu	-
22	FC	Width=10

Table 1: Architecture of AlexNet. "K" denotes kernel size; "S" denotes stride; "P" denotes padding.