Straight-Through Estimator as Projected Wasserstein Gradient Flow

Pengyu Cheng[1], **Chang Liu**[2], **Chunyuan Li**[3],
Dinghan Shen[1], **Ricardo Henao**[1] **and Lawrence Carin**[1]
[1]Duke University, [2]Tsinghua University, [3]Microsoft Research
pengyu.cheng@duke.edu

Abstract

The Straight-Through (ST) estimator is a widely used technique for back-propagating gradients through discrete random variables. However, this effective method lacks theoretical justification. In this paper, we show that ST can be interpreted as the simulation of the projected Wasserstein gradient flow (pWGF). Based upon this understanding, a theoretical foundation is established to justify the convergence properties of ST. Further, another pWGF estimator variant is proposed, which exhibits superior performance on distributions with infinite support, *e.g.*, Poisson distributions. Empirically, we show that ST and our proposed estimator, while applied to different types of discrete structures (including both Bernoulli and Poisson latent variables), exhibit comparable or even better performances relative to other state-of-the-art methods. Our results uncover the origin of the widespread adoption of ST estimator, and represent a helpful step towards exploring alternative gradient estimators for discrete variables.

1 Introduction

Learning distributions in discrete domains is a fundamental problem in machine learning. This problem can be formulated in general as minimizing the following expected cost

$$L(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{\theta}}}[f(\boldsymbol{z})], \tag{1}$$

where $f(\boldsymbol{z})$ is the cost function, $\boldsymbol{z}$ is a discrete (latent) random variable whose distribution $p_{\boldsymbol{\theta}}$ is parameterized by $\boldsymbol{\theta}$. Typically, $\boldsymbol{\theta}$ is obtained as the output of a Neural Network (NN), whose weights are learned by backpropagating the gradients through discrete random variables $\boldsymbol{z}$.

In practice, direct gradient computations through the discrete random variables, $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \sum_{\boldsymbol{z}} \nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\boldsymbol{z}) f(\boldsymbol{z})$ suffers from the curse of dimensionality, since it requires traversing through all possible joint configurations of the latent variable, whose number is exponentially large w.r.t. the latent dimension. Due to this limitation, existing approaches resort to estimating the gradient $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$ by approximating its expectation, where Monte Carlo sampling methods are typically employed.

The Straight-Through (ST) estimator [14, 3] is a widely applied method due to its simplicity and effectiveness. The idea of ST is directly using the gradients of discrete samples as the gradients of the distribution parameters. Since discrete samples can be generated as the output of hard threshold functions with distribution parameters as input, Bengio et al [3] explain the ST estimator by set the gradients of hard threshold functions to 1. However, this explanation lacks theoretical justification for the gradients of hard threshold functions.

In this paper, we show that ST can be interpreted as simulating the projected Wasserstein gradient flow (pWGF) of a functional $F[\mu] := \mathbb{E}_{\boldsymbol{z}\sim\mu}[f(\boldsymbol{z})]$, where μ is a distribution in the target discrete distribution family with density $p_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$. Further, a more general optimizing scheme

Third workshop on Bayesian Deep Learning (NeurIPS 2018), Montréal, Canada.

for (1) is introduced. Instead of directly updating μ in the discrete distribution family, μ is first updated to $\tilde{\mu}$ on a larger Wasserstein distribution space where gradients are easier to compute. Then, we project $\tilde{\mu}$ back to the discrete distribution family $\mathcal{M}$ as the updated distribution. Moreover, the projection follows the descending direction of $F[\cdot]$ in $\mathcal{M}$, which justifies the effectiveness of ST. This pWGF based updating scheme also motivates another variant that achieves faster convergence when the desired family of distributions has infinite support, *e.g.*, Poisson.

2 Proposed Algorithm

Denote $\mathcal{M} = \{\mu : \text{density of } \mu \text{ has the form of } p_{\boldsymbol{\theta}}\}$ as the d-dimensional discrete distributions family parameterized by $\boldsymbol{\theta}$. With $F[\mu] := \mathbb{E}_{\boldsymbol{z}\sim\mu}[f(\boldsymbol{z})]$, the task (1) can be rewritten as

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{\theta}}}[f(\boldsymbol{z})] = \min_{\mu\in\mathcal{M}} \mathbb{E}_{\boldsymbol{z}\sim\mu}[f(\boldsymbol{z})] = \min_{\mu\in\mathcal{M}} F[\mu], \tag{2}$$

where $f(\cdot)$ is assumed to be differentiable. To solve (2), directly calculating the gradient $\nabla_{\mathcal{M}} F$ is challenging, because the discrete distribution family $\mathcal{M}$ is very restrictive on the gradients. Alternatively, if we relax the discrete constraint and perform updates in an appropriate larger space $\tilde{\mathcal{M}}$, the calculation of the gradient $\nabla_{\tilde{\mathcal{M}}} F$ can be much easier. Therefore, as showed in Fig. 1, in k-th updating iteration, we consider first updating the current distribution μ_k to $\tilde{\mu}_k$ with stepsize ε in a larger 2-Wasserstein space $\tilde{\mathcal{M}}$ [24], then projecting $\tilde{\mu}_k$ back to $\mathcal{M}$ as updated discrete distribution μ_{k+1}. Theorem C.2 in supplement guarantees that our updating scheme converges with a small enough step size ε.

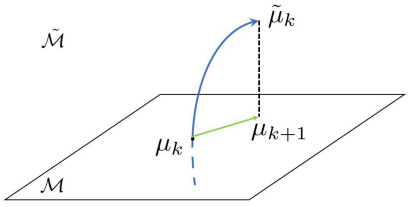

Figure 1: Updating scheme

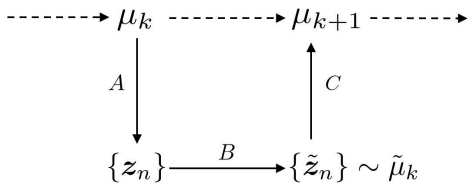

Figure 2: Algorithm outline

With Wasserstein gradient flow (WGF) [24], we show (in Appendix) that, the gradient in larger space $\tilde{\mathcal{M}}$ as $\nabla_{\tilde{\mathcal{M}}} F = \nabla f$, which means, if μ_k is represented by a group of its samples $\{\boldsymbol{z}_n\}_{n=1}^N$, then $\{\tilde{\boldsymbol{z}}_n\} = \{\boldsymbol{z}_n + \varepsilon \nabla f(\boldsymbol{z}_n)\}$ can be treated as a group of sample from $\tilde{\mu}_k$. Therefore, we can update μ_k to $\tilde{\mu}_k$ along the WGF simply by updating its samples. To project $\tilde{\mu}_k$ back to $\mathcal{M}$ as μ_{k+1}, we need to solve $\mu_{k+1} = \arg\min_{\mu\in\mathcal{M}} W(\mu, \tilde{\mu}_k)$, which is equivalent to solve $\arg\min_{\mu} W^2(\mu, \tilde{\mu}_k)$, where $W^2(\cdot,\cdot)$ is the square of the 2-Wasserstein distance [24]. Consequently, our pWGF algorithm proceeds in 3 steps shown in Fig. 2: (A) draw samples $\{\boldsymbol{z}_n\}$ from current distribution μ_k ; (B) update $\{\boldsymbol{z}_n\}$ to $\{\tilde{\boldsymbol{z}}_n\}$ as samples from $\tilde{\mu}_k$; (C) project $\tilde{\mu}_k$ back to μ_{k+1} by minimizing Wasserstein distance.

Since distributions in $\mathcal{M}$ are multidimensional, the exact Wasserstein distance is difficult to derive. We make a standard assumption [8] that μ and $\tilde{\mu}_k$ are factorized distributions. With the assumption, we prove in Theorem 2.1 that minimizing Wasserstein distance between factorized distributions is equivalent to minimizing the marginal distance on every dimension. Therefore, for simplicity, we describe our projection step using one-dimensional distributions. As the updated distribution $\tilde{\mu}_k$ is implicit, we can not obtain the closed form of Wasserstein distance $W^2(\mu_k,\tilde{\mu}_k)$. Therefore, we consider two approximations of $W(\mu_k, \tilde{\mu}_k)$.

Theorem 2.1. *If d-dimensional distributions μ and ν are factorized, then $W^2(\mu,\nu) = \sum_{i=1}^d W^2(\mu^{(i)}, \nu^{(i)})$, where $\mu^{(i)}$ and $\nu^{(i)}$ are the marginal distributions of μ and ν respectively.*

2.1 ST estimator: Absolute Difference of Expectation

We find that the Straight-Through (ST) estimator [3] is a special case of pWGF, when the Wasserstein distance is approximated via its lower bound, absolute difference of expectations.

Theorem 2.2. *For two one-dimensional distributions $\mu, \nu \in \tilde{\mathcal{M}}$, the absolute difference between $\mathbb{E}_\mu = \mathbb{E}_{x\sim\mu}[x]$ and $\mathbb{E}_\nu = \mathbb{E}_{y\sim\nu}[y]$ is a lower bound of $W(\mu,\nu)$, i.e. $|\mathbb{E}_\mu - \mathbb{E}_\nu| \le W(\mu,\nu)$.*

Remark. *If μ and ν are Bernoulli, then $W^2(\mu,\nu) = |\mathbb{E}_\mu - \mathbb{E}_\nu|$, which means minimizing the expectation difference is equivalent to minimizing the 2-Wasserstein distance under Bernoulli cases.*

For one-dimensional Bernoulli distribution, $\mu_k \sim \text{Bern}(p)$, noting that $p = \mathbb{E}_{\mu_k} \approx \frac{1}{N}\sum_{n=1}^{N} z_n$ and $\mathbb{E}_{\tilde{\mu}_k} \approx \frac{1}{N}\sum_{n=1}^{N} \tilde{z}_n$, we approximate the parameter gradient by: $\nabla_p W^2(\mu_k, \tilde{\mu}_k) \approx \nabla_p(\mathbb{E}_{z_k\sim\mu_k}[z_k] - \mathbb{E}_{\tilde{z}_k\sim\tilde{\mu}_k}[\tilde{z}_k])^2 \approx \nabla_p \left(p - \frac{1}{N}\sum_{n=1}^{N} \tilde{z}_n\right)^2 = 2(p - \frac{1}{N}\sum_{n=1}^{N} \tilde{z}_n)$. To reduce the variance caused by the sample mean, we use the control variate method [4] and write $\nabla_p W^2 \approx 2(p - \frac{1}{N}\sum_{n=1}^{N} \tilde{z}_n) = 2\left[(p - \mathbb{E}_{z_k\sim\mu_k}[z_k]) + (\mathbb{E}_{z_k\sim\mu_k}[z_k] - \frac{1}{N}\sum_{n=1}^{N} \tilde{z}_n)\right] \approx \frac{2}{N}\sum_{n=1}^{N}(z_n - \tilde{z}_n) = \frac{2\varepsilon}{N}\sum_{n=1}^{N} \nabla_z f(z_n)$. Thus, we have derived the pWGF estimator with expectation difference approximation, which has the same form as a multi-sample version ST estimator [3]. Parameter gradients for Poisson and Categorical distributions can be derived in a similar way.

2.2 Proposed estimator: Maximum Mean Discrepancy

A more principled way to approximate the Wasserstein distance is to use Maximum Mean Discrepancy (MMD) [12]: $\Delta^2(\mu, \nu) = \mathbb{E}_{\boldsymbol{x}_1,\boldsymbol{x}_2\sim\mu}[K(\boldsymbol{x}_1, \boldsymbol{x}_2)] + \mathbb{E}_{\boldsymbol{y}_1,\boldsymbol{y}_2\sim\nu}[K(\boldsymbol{y}_1, \boldsymbol{y}_2)] - 2\mathbb{E}_{\boldsymbol{x}\sim\mu,\boldsymbol{y}\sim\nu}[K(\boldsymbol{x}, \boldsymbol{y})]$, where $K(\cdot, \cdot)$ is a selected kernel. In practice, instead of minimizing $W(\mu, \tilde{\mu}_k)$, we can minimize the empirical expectation $\Delta^2(\mu, \tilde{\mu}) \approx \mathbb{E}_{z_1,z_2\sim\mu}[K(z_1, z_2)] + \frac{1}{N^2}\sum_{n,n'=1}^{N} K(\tilde{z}_n, \tilde{z}_{n'}) - 2\frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{z\sim\mu} K(z, \tilde{z}_n)$. Details on parameter gradients $\nabla_{\boldsymbol{\theta}}[\Delta^2]$ are shown in the supplement.

3 Experiments

We demonstrate the advantage of pWGF on updating Poisson distributions, and show the benchmark performance with a binary latent model in the supplement. Since the only difference between our pWGF version ST and the original ST is the learning rate scalar, if not specifically mentioned, we call pWGF-ST or the original ST together as ST, and call our MMD version method as pWGF.

3.1 Poisson Parameter Estimation

We apply pWGF to infer the parameter of a one-dimensional Poisson distribution. We use the true distribution $p(z) = \text{Pois}(\lambda_0 = 5)$ to generate data samples $\{z_i\}_{i=1}^{N}$, and use a Generative Adversarial learing framework to learn model parameters. A *generator* $q_\lambda(z)$ is constructed as $z \sim \text{Pois}(\lambda)$. A *discriminator* $w(z)$ is a network used to distinguish true/fake samples, which outputs the probability that the data comes from the true distribution. During the adversarial training, the generator aims to increase $\mathbb{E}_{z\sim q_\lambda}[w(z)]$, while the discriminator tries to decrease $\mathbb{E}_{z\sim q_\lambda}[w(z)]$ and increase $\mathbb{E}_{z\sim p}[w(z)]$. We can rewrite the training process as a min-max game with objective function: $\max_\lambda \min_w \{\mathbb{E}_{z\sim q_\lambda}[w(z)] - \mathbb{E}_{z\sim p}[w(z)]\}$. Similar to the observation in [10], the training process should finally converges to $\lambda^* = \lambda_0 = 5$. Therefore, for the generator, learning λ becomes optimizing $\mathbb{E}_{z\sim q_\lambda}[w(z)]$. We compare our pWGF against ST, Reinforce and Muprop [13] and show the learning curves on estimation in Figure 3. pWGF converges faster than others and exhibits much smaller oscillation. In Table 1, We report the mean and the standard derivation of the inferred parameter λ after 100 training epochs, where our pWGF exhibits higher inference accuracy and lower variance.

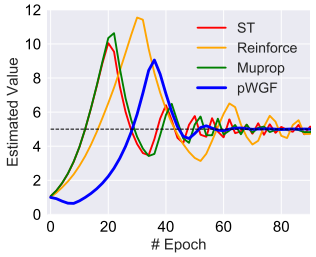

Figure 3: Learning curves of Poisson parameter.

Table 1: Mean and Standard Derivation of Inference

	Mean	Std
pWGF	5.0076	0.013
ST	5.1049	0.161
Muprop	5.0196	0.159
Reinforce	4.9452	0.173

4 Conclusion

We presented a theoretical foundation to justify the superior empirical performance of Straight-Through (ST) estimator for backpropagating gradients through discrete latent variables. Specifically, we show that ST can be interpreted as the simulation of the projected gradient flow on Wasserstein space. Based upon this theoretical framework, we further propose another gradient estimator for learning discrete variables, which exhibits even better performance while applied to distributions with infinite support, *e.g.*, Poisson.

References

[1] Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2008.

[2] Jean-David Benamou and Yann Brenier. A computational fluid mechanics solution to the monge-kantorovich mass transfer problem. *Numerische Mathematik*, 84(3):375–393, 2000.

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[4] Phelim P Boyle. Options: A monte carlo approach. *Journal of financial economics*, 4(3):323–338, 1977.

[5] Changyou Chen, Chunyuan Li, Liqun Chen, Wenlin Wang, Yunchen Pu, and Lawrence Carin. Continuous-time flows for deep generative models. *arXiv preprint arXiv:1709.01179*, 2017.

[6] Changyou Chen and Ruiyi Zhang. Particle optimization in stochastic gradient mcmc. *arXiv preprint arXiv:1711.10927*, 2017.

[7] Changyou Chen, Ruiyi Zhang, Wenlin Wang, Bai Li, and Liqun Chen. A unified particle-optimization framework for scalable bayesian sampling. *UAI submission*, 2018.

[8] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180, 2016.

[9] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[11] Will Grathwohl, Dami Choi, Yuhuai Wu, Geoff Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *arXiv preprint arXiv:1711.00123*, 2017.

[12] Arthur Gretton, Karsten M Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola. A kernel method for the two-sample-problem. In *Advances in neural information processing systems*, pages 513–520, 2007.

[13] Shixiang Gu, Sergey Levine, Ilya Sutskever, and Andriy Mnih. Muprop: Unbiased backpropagation for stochastic neural networks. *arXiv preprint arXiv:1511.05176*, 2015.

[14] G. Hinton. *Neural networks for machine learning, video lectures*. Coursera, 2012.

[15] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparametrization with gumble-softmax. In *International Conference on Learning Representations 2017*. OpenReviews. net, 2017.

[16] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *stat*, 1050:1, 2014.

[17] Chang Liu, Jingwei Zhuo, Pengyu Cheng, Ruiyi Zhang, Jun Zhu, and Lawrence Carin. Accelerated first-order methods on the wasserstein space for bayesian inference. *arXiv preprint arXiv:1807.01750*, 2018.

[18] Qiang Liu. Stein variational gradient descent as gradient flow. In *Advances in neural information processing systems*, pages 3118–3126, 2017.

[19] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances In Neural Information Processing Systems*, pages 2378–2386, 2016.

[20] Sayan Mukherjee, Qiang Wu, Ding-Xuan Zhou, et al. Learning gradients on manifolds. *Bernoulli*, 16(1):181–207, 2010.

[21] Felix Otto. The geometry of dissipative evolution equations: the porous medium equation. 2001.

[22] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286, 2014.

[23] George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In *Advances in Neural Information Processing Systems*, pages 2627–2636, 2017.

[24] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[25] Mingzhang Yin and Mingyuan Zhou. Arm: Augment-reinforce-merge gradient for discrete latent variable models. *arXiv preprint arXiv:1807.11143*, 2018.

[26] Ruiyi Zhang, Changyou Chen, Chunyuan Li, and Lawrence Carin. Policy optimization as wasserstein gradient flows. *arXiv preprint arXiv:1808.03030*, 2018.

A Background

To minimize the expected cost $\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{\theta}}}[f(\boldsymbol{z})]$ in (1), we assume that $\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}} f(\boldsymbol{z})] = 0$, if the cost function $f(\boldsymbol{z})$ depends of $\boldsymbol{\theta}$. For instance, in the Variational Autoencoder (VAE) [9], we seek to maximize the Evidence Lower Bound (ELBO) as $\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}[f(\boldsymbol{z})]$, where $f(\boldsymbol{z}) = \log[p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})/q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})]$ depends on parameter $\boldsymbol{\theta}$ through the variational posterior approximation $q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$. Since $\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}[\nabla_{\boldsymbol{\theta}} \log q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})] = 0$, we have $\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}[\nabla_{\boldsymbol{\theta}} f(\boldsymbol{z})] = 0$.

As described above, there are two types of updating methods for $\boldsymbol{\theta}$ under (1), namely, estimation of the parameter gradient $\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{\theta}}}[f(\boldsymbol{z})]$, and continuous relaxation of the discrete variable $\boldsymbol{z}$.

A.1 Continuous relaxation

Another approach used to obtain updates for $\boldsymbol{\theta}$ in (1) is to approximate samples of $\boldsymbol{z}$ from a deterministic function, $h(\cdot)$, of $\boldsymbol{\theta}$ and an independent random variable $\boldsymbol{\epsilon}$ with simple distribution $p_{\boldsymbol{\epsilon}}$, *e.g.*, uniform or normal, so $\boldsymbol{z} = h(\boldsymbol{\theta}, \boldsymbol{\epsilon})$. Then we can use the chain rule to derive the gradient of (1) as

$$\nabla_{\boldsymbol{\theta}}\mathbb{E}_{p_{\boldsymbol{\theta}}}[f(\boldsymbol{z})] = \nabla_{\boldsymbol{\theta}}\mathbb{E}_{p_{\boldsymbol{\epsilon}}}[f(h(\boldsymbol{\theta}, \boldsymbol{\epsilon}))] = \mathbb{E}_{p_{\boldsymbol{\epsilon}}}[\nabla_{\boldsymbol{\theta}} f(h(\boldsymbol{\theta}, \boldsymbol{\epsilon}))].$$

We can take expectation of the gradients, which is very convenient because $\nabla_{\boldsymbol{\theta}}$ can be computed by chain rule, noting that $f(\cdot)$ does not directly depend of $\boldsymbol{\theta}$. This reparameterization trick works quiet well when $\boldsymbol{z}$ originates from a continuous distribution. For example, given a normal distribution, $\boldsymbol{z} \sim N(\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$, we can rewrite $\boldsymbol{z} = \boldsymbol{\mu} + \mathrm{diag}(\boldsymbol{\sigma})N(0, \boldsymbol{I})$ and directly obtain $\nabla_{\boldsymbol{\mu}}\boldsymbol{z}$ and $\nabla_{\boldsymbol{\sigma}}\boldsymbol{z}$. This reparameterization has been widely used in the training of variational autoencoder with latent Gaussian priors [16, 22].

In the discrete case, it becomes very difficult to find a differentiable deterministic function to generate samples from $\boldsymbol{z}$. For the categorical distribution, [15] introduced the Gumbel-Softmax distribution to relax the *one-hot* vector encoding commonly used for categorical variables. For the multidimensional (factorized) Bernoulli distribution with parameter $\boldsymbol{\theta} = \boldsymbol{p}$, the Straight Through (ST) estimator [14, 3], which considers the gradient of N samples of $\boldsymbol{z}$ directly, as the gradient of parameter $\nabla_{\boldsymbol{\theta}} f$, can be also explained by setting the derivative $\nabla_{\boldsymbol{p}} h$ of the discrete function $\boldsymbol{z} = h(\boldsymbol{p}, \boldsymbol{\epsilon}) = \mathbf{1}_{\boldsymbol{\epsilon}>\boldsymbol{p}}$ (coordinate-wise) directly to the identity matrix $\boldsymbol{I}$ [3].

A.2 Wasserstein gradient flow

Wasserstein gradient flows (WGF) [7, 1, 24] have become popular in machine learning, due to its generality over parametric distribution families, and tractable computational efficiency. The Wasserstein space is a metric space of distributions. The WGF defines a family of steepest descending functions. It has been Bayesian inference, where the KL divergence of an approximating distribution to a target one is minimized by simulating its gradient flow. [7] developed a unfnied framework to simulate the WGF, including Stein Variational Gradient Descent (SVGD) [19, 18] and Stochastic Gradient MCMC as its special cases. [6] and [17] proposed an acceleration framework for these methods. WFGs have also been applied to deep generative models [5] and policy optimization in reinforcement learning [26]. However, all previous methods focus on simulating WGFs to approximate distributions in continuous domains. There has been little if any research reported for WGFs for discrete domains.

B Updating via Wasserstein gradient flow

Gradient computation and Wasserstein Gradient Flow (WGF) simulation are made possible by the Riemannian structure of $\tilde{\mathcal{M}}$, which consists of a proper inner product in the tangent space that is consistent with the Wasserstein distance [2, 21]. The tangent space of $\tilde{\mathcal{M}}$ at μ can be represented by a subspace of vector fields on $\mathbb{R}^d$ ([24], Thm 13.8; [1], Thm 8.3.1, Prop 8.4.5):

$$T_\mu\tilde{\mathcal{M}} := \overline{\{\nabla\varphi : \varphi \in C_c^\infty(\mathbb{R}^d)\}}^{L^2(\mu;\mathbb{R}^d)},$$

where $\mathcal{C}_c^\infty(\mathbb{R}^d)$ is the set of compactly supported smooth functions on $\mathbb{R}^d$, $L^2(\mu;\mathbb{R}^d) := \{v : \int_{\mathbb{R}^d} v(\boldsymbol{z})^\top v(\boldsymbol{z})\mu(\mathrm{d}\boldsymbol{z}) < +\infty\}$ is a Hilbert space with inner product $\langle v, u\rangle_{L^2(\mu;\mathbb{R}^d)} := \int v(\boldsymbol{z})^\top u(\boldsymbol{z})\mu(\mathrm{d}\boldsymbol{z})$, and the overline represents taking the closure in $L^2(\mu;\mathbb{R}^d)$.

With the inner product inherited from $L^2(\mu;\mathbb{R}^d)$, $\tilde{\mathcal{M}}$ being a Riemannian manifold is consistent with the Wasserstein distance due to the Benamou-Brenier formula [2]. We can then express the gradient of a function on $\tilde{\mathcal{M}}$ in the Riemannian sense. The explicit expression is intuitively proposed as Otto's calculus ([21]; [24], Chapter 15) and rigorously verified by subsequent work, *e.g.*, [24], Thm 23.18; [1], Lem 10.4.1. Specifically, they showed that given a functional $F[\mu] = \mathbb{E}_{\boldsymbol{z}\sim\mu}[f(\boldsymbol{z})]$ with $f(\cdot) \in \mathcal{C}_c^\infty(\mathbb{R}^d)$, its gradient is $\nabla_{\tilde{\mathcal{M}}} F[\mu] = \nabla f \in T_\mu\tilde{\mathcal{M}}$, a vector field on $\mathbb{R}^d$. This means that we can, in principle, compute the desired gradient $\nabla_{\tilde{\mathcal{M}}} F[\mu]$ using ∇f.

Another convenient property of $\tilde{\mathcal{M}}$ based on the physical interpretation of tangent vectors on $\tilde{\mathcal{M}}$ makes the gradient flow simulation possible. Consider a smooth curve of absolutely continuous measures, μ_t, with corresponding tangent vector $\boldsymbol{v}_t$, where $t \in \mathbb{R}$, and for which the gradient flow is simulated (iteratively) at discrete values $k = 1, \ldots, k, k+1, \ldots$, to estimate $\mu_1, \ldots, \mu_k, \mu_{k+1}, \ldots$ (the target distribution). For any $s \in \mathbb{R}$ and $\varepsilon \to 0$, Proposition 8.4.6 of [1] guarantees that $W(\mu_{s+\varepsilon}, (\mathrm{id} + \varepsilon \boldsymbol{v}_s)_\# \mu_s) = o(|\varepsilon|)$, where $(\mathrm{id} + \varepsilon \boldsymbol{v}_s)$ is a transformation on $\mathbb{R}^d$ (id is the identity map and $\boldsymbol{v}_s$ is a vector field on $\mathbb{R}^d$), and $(\mathrm{id} + \varepsilon \boldsymbol{v}_s)_\# \mu_s$ is the pushed-forward measure of μ_s that moves μ_s along the tangent vector $\boldsymbol{v}_s$ by distance ε, see Figure 1. When μ_t is a gradient flow (steepest descending curve) of $F[\cdot]$ defined in the form above, $\boldsymbol{v}_t = -\nabla_{\tilde{\mathcal{M}}} F[\mu_t] = -\nabla f$, as described before, then for $\mu_k := \mu_s$ having a set of samples $\{\boldsymbol{z}_n\}_{n=1}^N$ and the definition of pushed-forward measure [1], $\{\tilde{\boldsymbol{z}}_n := \boldsymbol{z}_n - \varepsilon \nabla f(\boldsymbol{z}_n)\}_{n=1}^N$ is a set of samples of $\tilde{\mu}_k := (\mathrm{id} + \varepsilon \boldsymbol{v}_s)_\# \mu_s$, which conform a first-order approximation of $\tilde{\mu}_{s+\varepsilon}$. Since $\tilde{\mu}_k \in \tilde{\mathcal{M}}$ is a good approximation of $\mu_{s+\varepsilon} \in \tilde{\mathcal{M}}$ (the optimal measure along the WGF) as discussed above, thus we can use $\tilde{\mu}_k \in \tilde{\mathcal{M}}$ to approximate $\mu_{k+1} \in \mathcal{M}$. This is done by projecting $\tilde{\mu}_k \in \tilde{\mathcal{M}}$ onto $\mu_{k+1} \in \mathcal{M}$. Then, per Theorem C.2, with small enough positive ε, we can always get a set of samples whose distribution improves $F[\cdot]$, the functional of the cost in (1).

C Proofs

Theorem C.1. *Let $F[\cdot]$ be a differentiable function on a manifold $\tilde{\mathcal{M}}$ and $\mathcal{M}$ a submanifold of $\tilde{\mathcal{M}}$, $\mathcal{M} \subset \tilde{\mathcal{M}}$, then at any $\mu \in \mathcal{M}$,*

$$\nabla_{\mathcal{M}} F = (\nabla_{\tilde{\mathcal{M}}} F)^\perp,$$

where $(\nabla_{\tilde{\mathcal{M}}} F)^\perp$ is the projection of $\nabla_{\tilde{\mathcal{M}}} F$ onto $T_\mu\mathcal{M}$.

Proof of Theorem C.1. By the definition of $\nabla_{\mathcal{M}} F$ [20], for any vector $\boldsymbol{v} \in T_\mu\mathcal{M}$,

$$\langle \nabla_{\mathcal{M}} F, \boldsymbol{v} \rangle = \boldsymbol{v}(F)[\mu]. \tag{3}$$

By the definition of $\nabla_{\tilde{\mathcal{M}}} F$, for any vector $\boldsymbol{u} \in T_\mu\tilde{\mathcal{M}}$,

$$\langle \nabla_{\tilde{\mathcal{M}}} F, \boldsymbol{u} \rangle = \boldsymbol{u}(F)[\mu]. \tag{4}$$

Since $T_\mu\mathcal{M}$ is the subspace of $T_\mu\tilde{\mathcal{M}}$, by definition of $(\nabla_{\tilde{\mathcal{M}}} F)^\perp$ we have

$$\langle \nabla_{\tilde{\mathcal{M}}} F, \boldsymbol{v} \rangle = \langle (\nabla_{\tilde{\mathcal{M}}} F)^\perp, \boldsymbol{v} \rangle. \tag{5}$$

By (3), (4), (5), for any $\boldsymbol{v} \in T_\mu\mathcal{M}$

$$\langle \nabla_{\mathcal{M}} F, \boldsymbol{v} \rangle = \boldsymbol{v}(F)[\mu] = \langle \nabla_{\tilde{\mathcal{M}}} F, \boldsymbol{v} \rangle = \langle (\nabla_{\tilde{\mathcal{M}}} F)^\perp, \boldsymbol{v} \rangle.$$

Therefore, $\nabla_{\mathcal{M}} F = (\nabla_{\tilde{\mathcal{M}}} F)^\perp$. □

Theorem C.2. *Let $\boldsymbol{v} = -\varepsilon \nabla_{\tilde{\mathcal{M}}} F[\mu_k]$ and $W(\cdot,\cdot)$ be the 2-Wasserstein distance in $\tilde{\mathcal{M}}$. Update μ_k in $\tilde{\mathcal{M}}$ along direction $\boldsymbol{v}$ to $\tilde{\mu}_k = \exp_{\mu_k}(\boldsymbol{v})$ (exponential map [20]), then project $\tilde{\mu}_k$ back to $\mathcal{M}$ as $\mu_{k+1} = \arg\min_{\mu\in\mathcal{M}} W(\mu, \tilde{\mu}_k)$. If $\nabla_{\tilde{\mathcal{M}}} F$ is Lipschitz continuous, then there exists $r > 0$, such that for any $\varepsilon < r$, $F[\mu_k] \geq F[\mu_{k+1}] + O(\varepsilon^2)$.*

Proof of Theorem 2.1. (1) First, we show that $W^2(\mu,\nu) \le \sum_{i=1}^d W^2(\mu_i,\nu_i)$.

Arbitrarily selecting $\gamma_i \in \Gamma(\mu_i,\nu_i)$, $i=1,\dots,d$, we define $\gamma^* = \prod_{i=1}^d \gamma_i$. Since $\mu_i(x) = \int \gamma_i(x,\mathrm{d}y)$, we have

$$\begin{aligned}
&\int_{\mathbb{R}^d} \gamma^*(x_1,\dots,x_d,\mathrm{d}y_1,\dots,\mathrm{d}y_d)\\
=&\int_{\mathbb{R}^d} \gamma_1(x_1,\mathrm{d}y_1)\gamma_2(x_2,\mathrm{d}y_2)\dots\gamma_D(x_d,\mathrm{d}y_d)\\
=&\prod_{i=1}^d \int_{\mathbb{R}} \gamma_i(x_i,\mathrm{d}y_i) = \prod_{i=1}^d \mu_i(x_i) = \mu(x_1,x_2,\dots,x_d),
\end{aligned} \tag{6}$$

which means the marginal distribution of γ^* on $\boldsymbol{x}$ is μ. Similarly, the marginal distribution of γ^* on $\boldsymbol{y}$ is ν. Therefore, $\gamma^* \in \Gamma(\mu,\nu)$. Then

$$\inf_{\gamma\in\Gamma(\mu,\nu)} \int \|\boldsymbol{x}-\boldsymbol{y}\|^2 \gamma(\mathrm{d}\boldsymbol{x},\mathrm{d}\boldsymbol{y}) \le \int \|\boldsymbol{x}-\boldsymbol{y}\|^2 \gamma^*(\mathrm{d}\boldsymbol{x},\mathrm{d}\boldsymbol{y}). \tag{7}$$

On the other hand,

$$\begin{aligned}
&\int \|\boldsymbol{x}-\boldsymbol{y}\|^2 \gamma^*(\mathrm{d}\boldsymbol{x},\mathrm{d}\boldsymbol{y})\\
=&\int \sum_{i=1}^d (x_i-y_i)^2 \gamma_1(\mathrm{d}x_1,\mathrm{d}y_1)\gamma_2(\mathrm{d}x_2,\mathrm{d}y_2)\dots\gamma_D(\mathrm{d}x_d,\mathrm{d}y_d)\\
=&\sum_{i=1}^d \int (x_i-y_i)^2 \gamma_i(\mathrm{d}x_i,\mathrm{d}y_i).
\end{aligned} \tag{8}$$

By (7) and (8), we have

$$W^2(\mu,\nu) \le \sum_{i=1}^d \int (x_i-y_i)^2 \gamma_i(\mathrm{d}x_i,\mathrm{d}y_i). \tag{9}$$

Take the infimum over both sides of the equation (9),

$$W^2(\mu,\nu) \le \sum_{i=1}^d \inf_{\gamma_i\in\Gamma(\mu_i,\nu_i)} \int (x_i-y_i)^2 \gamma_i(\mathrm{d}x_i,\mathrm{d}y_i) = \sum_{i=1}^d W^2(\mu_i,\nu_i). \tag{10}$$

(2) Then we show $W^2(\mu,\nu) \ge \sum_{i=1}^d W^2(\mu_i,\nu_i)$.

Note that

$$\begin{aligned}
&\int \|\boldsymbol{x}-\boldsymbol{y}\|^2 \gamma(\mathrm{d}\boldsymbol{x},\mathrm{d}\boldsymbol{y})\\
=&\int \sum_{i=1}^d (x_i-y_i)^2 \gamma(\mathrm{d}x_1,\dots,\mathrm{d}x_d,\mathrm{d}y_1,\dots,\mathrm{d}y_d)\\
=&\sum_{i=1}^d \int (x_i-y_i)^2 \hat{\gamma}_i(\mathrm{d}x_i,\mathrm{d}y_i),
\end{aligned} \tag{11}$$

where $\hat{\gamma}_i(x_i,y_i) = \int \gamma(\mathrm{d}x_1,\dots,\mathrm{d}x_{i-1},x_i,\mathrm{d}x_{i+1},\dots,\mathrm{d}y_{i-1},y_i,\mathrm{d}y_{i+1},\dots,\mathrm{d}y_d)$ is the marginal distribution of γ over (x_i,y_i).

By Fubini's Theorem,

$$\begin{aligned}
&\int \hat{\gamma}_i(x_i,\mathrm{d}y_i)\\
=&\int \gamma(\mathrm{d}x_1,\dots,\mathrm{d}x_{i-1},x_i,\mathrm{d}x_{i+1},\dots,\mathrm{d}y_{i-1},\mathrm{d}y_i,\mathrm{d}y_{i+1},\dots,\mathrm{d}y_d)\\
=&\int \mu(\mathrm{d}x_1,\dots,\mathrm{d}x_{i-1},x_i,\mathrm{d}x_{i+1},\dots\mathrm{d}x_d) \quad (\text{by } \mu(\boldsymbol{x}) = \int \gamma(\boldsymbol{x},\mathrm{d}\boldsymbol{y}))\\
=&\mu_i(x_i).
\end{aligned} \tag{12}$$

Similarly, $\int \hat{\gamma}_i(\mathrm{d}x_i, y_i) = \nu_i(y_i)$. Therefore, $\hat{\gamma}_i \in \Gamma(\mu_i, \nu_i)$. Then

$$\sum_{i=1}^{d} \int (x_i - y_i)^2 \hat{\gamma}_i(\mathrm{d}x_i, \mathrm{d}y_i) \geq \sum_{i=1}^{d} \inf_{\gamma_i \in \Gamma(\mu_i, \nu_i)} \int (x_i - y_i)^2 \gamma_i(\mathrm{d}x_i, \mathrm{d}y_i). \tag{13}$$

By (11) and (13),

$$\int \|\boldsymbol{x} - \boldsymbol{y}\|^2 \gamma(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{y}) \geq \sum_{i=1}^{d} W^2(\mu_i, \nu_i). \tag{14}$$

Take infimum over both sides, $W^2(\mu, \nu) \geq \sum_{i=1}^{d} W^2(\mu_i, \nu_i)$.

Therefore, $W^2(\mu, \nu) = \sum_{i=1}^{d} W^2(\mu_i, \nu_i)$. □

Proof of Remark 2.2. For $\mu = \text{Bern}(p)$ and $\nu = \text{Bern}(q)$,

$$\begin{aligned} W^2(\mu, \nu) &= \inf_{\{a_{i,j}\}} \sum_{i,j \in \{0,1\}} a_{i,j}(i-j)^2 \\ &= \min_{\{a_{i,j}\}} a_{1,0} + a_{0,1}, \end{aligned} \tag{15}$$

where $\sum_i a_{i,1} = q$, $\sum_j a_{1,j} = p$, and $a_{i,j} \geq 0$, $\sum_{i,j} a_{i,j} = 1$.

Problem in (15) is a linear programming. It can be shown easily that the minimum value of (15) is $W^2(\mu, \nu) = |p - q|$. □

Lemma C.3. *Let ν be an arbitrary distribution and $\mu = Bern(p)$ be a Bernoulli distribution. Then*

$$W^2(\mu, \nu) = \int_{-\infty}^{t^*} y^2 \nu(\mathrm{d}y) + \int_{t^*}^{\infty} (y-1)^2 \nu(\mathrm{d}y), \tag{16}$$

where $t^* = \inf\{t : \int_t^{\infty} \nu(\mathrm{d}y) = p\}$.

D Gradient For MMD Projection

We take the radial basis function kernel $K(x, y) = \exp(-\frac{(x-y)^2}{2h^2})$ for instance.

For Bernoulli distribution, $\mu = \text{Bern}(p)$, $\frac{\partial \Delta^2}{\partial p} = 2[(1-2p)(1-K(1,0)) - \frac{1}{n}\sum_{i=1}^{n}(K(1, \tilde{z}_i) - K(1, \tilde{z}_j))]$

E Binary Latent Models

As most of previous proposed algorithms are specifically designed for the discrete variables with finite support, we consider using a binary latent model as the benchmark. We use variational autoencoder (VAE) [16] with the Bernoulli latent variable (Bernoulli VAE). We compare pWGF with the baseline methods ST and Gumbel-Softmax [15], as well as three state-of-the-art algorithms: Rebar [23], Relax [11] and ARM [25]. Following the settings in [25], we build the model with different network architectures. We apply all methods and architectures to the MNIST dataset, and show the results in Table 2. From the results, pWGF is comparable with ST, and both pWGF/ST outperform other competing methods except ARM in all tested network architecture.

Table 2: Testing ELBO for Bernoulli VAE on MNIST

	pWGF	ST	ARM	RELAX	REBAR	Gumbel-Softmax
Linear	119.8	119.1	110.3	122.1	123.2	129.2
Two Layers	108.3	107.6	98.2	114	113.7	NA
Nonlinear	104.6	104.2	101.3	110.9	111.6	112.5