Efficient, Stable, and Analytic Differentiation of the Sinkhorn Loss

Yixuan Qiu

Shanghai University of Finance and Economics

The 9th RUC International Forum on Statistics

Joint work with Haoyun Yin and Xiao Wang

Outline

Motivation and Problem Setting

Statistical Divergence

- Given two distributions p and q, define a function $D(p, q)$ such that:
  - $D(p, q) \geq 0$ for all p and q
  - $D(p, q) = 0 \Leftrightarrow p = q$

Statistical Divergence

- Given two distributions p and q, define a function $D(p, q)$ such that:
  - $D(p, q) \geq 0$ for all p and q
  - $D(p, q) = 0 \Leftrightarrow p = q$
- Examples:
  - Kullback–Leibler divergence
$$D_{\mathrm{KL}}(p \| q) = \int p(x) \log \left(\frac{p(x)}{q(x)} \right) \mathrm{d}x$$
  - Squared Hellinger distance
$$H^2(p, q) = 2 \int \left(\sqrt{p(x)} - \sqrt{q(x)} \right)^2 \mathrm{d}x$$
  - Wasserstein distance
$$W_r(p, q) = \left(\inf_{\gamma \in \Pi(p,q)} \mathbb{E}_{(X,Y)\sim\gamma} \|X - Y\|^r \right)^{1/r}$$
  - ...

Problem Setting

- Given data $X_1, \dots, X_n \sim p^*$ and a model p_θ
- Task 1: Evaluate or estimate the divergence $D(p^*, p_\theta)$
  - A measure of goodness-of-fit
  - Testing distributional difference
- Task 2: Evaluate or estimate the gradient $\nabla_\theta D(p^*, p_\theta)$
  - This gradient makes p_θ learnable
  - Enables us to find a good model p_θ that minimizes $D(p^*, p_\theta)$

Wasserstein Distance

- Wasserstein distance is a popular metric to quantify the difference between distributions
- Inspired many breakthroughs in deep learning such as the Wasserstein generative adversarial networks (WGAN)
- However, its computation is notoriously difficult

Wasserstein Distance

- Given two discrete distributions $X \sim p$ and $Y \sim q$

$$\mathbb{P}(X = x_i) = a_i, \quad \mathbb{P}(Y = y_j) = b_i, \quad \sum_{i=1}^{n} a_i = \sum_{j=1}^{m} b_j = 1$$

- Define the cost matrix M with $M_{ij} = \|x_i - y_j\|^r$
- The r-Wasserstein distance between p and q is

$$W_r(p, q) = \left[\min_{P \in \Pi(a,b)} \langle P, M \rangle \right]^{1/r}$$

$$\Pi(a, b) = \{ T \in \mathbb{R}_+^{n \times m} : T\mathbf{1}_m = a, T^{\mathrm{T}}\mathbf{1}_n = b \}$$

- Solving P is also called the optimal transport (OT) problem

Sinkhorn Loss as Approximate OT

- Assuming $m = n$, standard linear programming solvers cost $\mathcal{O}(n^3 \log n)$
- Cuturi (2013) proposes the entropic-regularized OT

$$\min_{T \in \Pi(a,b)} \langle T, M \rangle - \lambda^{-1} h(T), \quad h(T) = \sum_{i=1}^{n} \sum_{j=1}^{m} T_{ij}(1 - \log T_{ij})$$

- Let T_λ^* be the unique global solution, and then the Sinkhorn loss is defined as

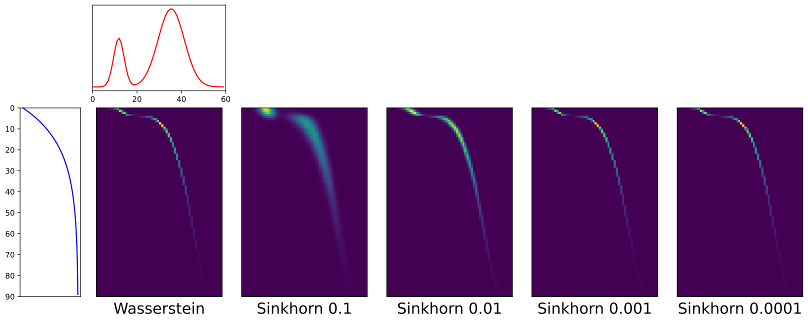
$$S_\lambda(M, a, b) = \langle T_\lambda^*, M \rangle$$

Sinkhorn Loss as Approximate OT

- Luise et al. (2018) shows that

$$|S_\lambda(M, a, b) - W_r^r(p, q)| \leq Ce^{-\lambda}$$

- For smaller and smaller λ^{-1}, $S_\lambda(M, a, b)$ is closer to the Wasserstein distance



Forward and Backward Pass

- Corresponding to the two tasks in the problem setting
- Forward pass of Sinkhorn loss
  - Compute $S_\lambda(M, a, b)$
  - Existing method: Sinkhorn's algorithm
- Backward pass of Sinkhorn loss
  - Compute $\nabla_M S_\lambda(M, a, b)$
  - Existing method: automatic differentiation

Sinkhorn's Algorithm

- Cuturi (2013) shows that T_λ^* can be expressed as

$$T_\lambda^* = \textbf{diag}(u^*) M_e \textbf{diag}(v^*)$$

for some vectors u^* and v^*, where $M_e = \left(e^{-\lambda M_{ij}}\right)$

- Interestingly, u^* and v^* can be solved using the iterations

$$u^{(k+1)} \leftarrow a \odot [M_e v^{(k)}]^{-1}, \quad v^{(k+1)} \leftarrow b \odot [M_e^{\mathrm{T}} u^{(k+1)}]^{-1}$$

- Notation: for vectors $u = (u_1, \ldots, u_k)^{\mathrm{T}}$, $v = (v_1, \ldots, v_k)^{\mathrm{T}}$,

$$u^{-1} = (u_1^{-1}, \ldots, u_k^{-1})^{\mathrm{T}}, \quad u \odot v = (u_1 v_1, \ldots, u_k v_k)^{\mathrm{T}}$$

Automatic Differentiation

- Note that the iterations in Sinkhorn's algorithm are all differentiable
- So one can use the automatic differentiation technique to compute $\nabla_M S_\lambda(M, a, b)$
- This requires "unrolling" the iterations
- Genevay et al. (2018) uses this method to learn generative models with the Sinkhorn loss

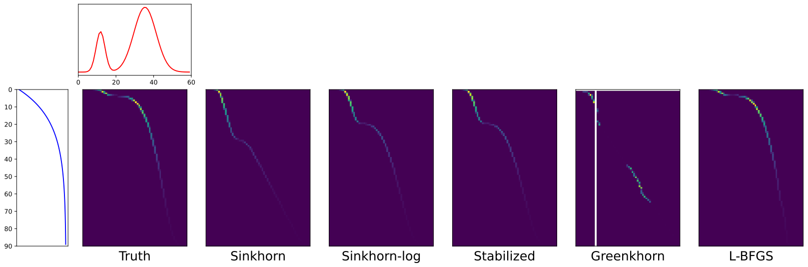
Our Contribution

Issues of Sinkhorn's Algorithm

- Numerical instability
  - M_e may underflow when λ is large
  - Making $M_e v^{(k)}$ and $M_e^{\mathrm{T}} u^{(k+1)}$ close to zero
  - $u^{(k+1)}$ and $v^{(k+1)}$ overflow
- Many works to improve
  - Log-domain iterations
  - Stabilized sparse scaling (Schmitzer, 2019)

Issues of Sinkhorn's Algorithm

- May be slow to converge
- Especially for large λ



Dual Problem

- The dual problem of the Sinkhorn optimization is $\max_{\alpha,\beta} \mathcal{L}(\alpha,\beta)$, where $\alpha \in \mathbb{R}^n$, $\beta \in \mathbb{R}^m$,

$$\mathcal{L}(\alpha,\beta) = \alpha^{\mathrm{T}} a + \beta^{\mathrm{T}} b - \lambda^{-1} \sum_{i=1}^{n} \sum_{j=1}^{m} e^{-\lambda(M_{ij}-\alpha_i-\beta_j)}$$

- This dual maximization problem is concave and unconstrained
- Once we get the optimal point (α^*, β^*), T^*_λ is recovered as

$$T^*_\lambda = \mathrm{e}_\lambda[\alpha^* \oplus \beta^* - M], \quad \mathrm{e}_\lambda[A] = (e^{\lambda A_{ij}})$$

- Notation: for vectors $u = (u_1, \dots, u_l)^{\mathrm{T}}$, $v = (v_1, \dots, v_k)^{\mathrm{T}}$,

$$u \oplus v = (u_i + v_j) \in \mathbb{R}^{l \times k}$$

Forward Pass

- (α, β) has one redundant degree of freedom, so globally set $\beta_m = 0$
- For a fixed β, the optimal α is

$$\alpha^*(\beta)_i = \lambda^{-1} \log a_i - \lambda^{-1} \log \left[\sum_{j=1}^{m} e^{\lambda(\beta_j - M_{ij})} \right]$$

- Then the optimization problem becomes

$$f(\beta) = -\alpha^*(\beta)^{\mathrm{T}} a - \beta^{\mathrm{T}} b + \lambda^{-1}$$

- We can also show that

$$\nabla_{\tilde{\beta}} f = \tilde{T}(\beta)^{\mathrm{T}} \mathbf{1}_n - \tilde{b}, \quad T(\beta) = \mathrm{e}_\lambda[\alpha^*(\beta) \oplus \beta - M]$$

- For a vector v or a matrix A, $\tilde{v}$ or $\tilde{A}$ means removing the last element or column

Stability of Solution

Theorem (informal)
Let f^* be the minimum value of $f(\beta)$, β^* an optimal solution, and $\alpha^* = \alpha^*(\beta^*)$. Then $f^* > -\infty$, β^* is unique, and α^*, β^* have computable bounds.

L-BFGS Algorithm

- The L-BFGS algorithm (Liu and Nocedal, 1989) is a well-known quasi-Newton method for smooth optimization problems
- It only requires evaluating $f(\beta)$ and $\nabla_{\tilde{\beta}} f$
- We provide theoretical guarantees for the L-BFGS algorithm on our problem

Convergence Analysis

Theorem (informal)
Let $\beta^{(k)}$ be the k-th iterate of β, and $f^{(k)} = f(\beta^{(k)})$, then

- *Objective function and iterates converge exponentially fast*

$$f^{(k)} - f^* \leq r^k (f^{(0)} - f^*) := \varepsilon^{(k)}, \quad \|\beta^{(k)} - \beta^*\|^2 \leq C_1 \varepsilon^{(k)}$$

- *Exponential decay of gradient*

$$\|\nabla_{\tilde{\beta}} f(\beta^{(k)})\|^2 = \|\tilde{T}^{(k)\mathrm{T}} \mathbf{1}_n - \tilde{b}\|^2 \leq C_2 \varepsilon^{(k)}$$

- *Stability of iterates*

$$0 < T_{ij}^{(k)} < \min\{a_i, b_j + \sqrt{C_2 \varepsilon^{(k)}}\}, \quad 1 \leq j \leq m-1$$

Backward Pass

- Instead of relying on automatic differentiation
- We develop an analytic expression for $\nabla_M S_\lambda(M, a, b)$
- Based on the implicit function theorem

Analytic Differentiation

Theorem
For a fixed $\lambda > 0$,

$$\nabla_M S_\lambda(M, a, b) = T^*_\lambda + \lambda(s_u \oplus s_v - M) \odot T^*_\lambda,$$

where

$$\begin{aligned} T^*_\lambda &= \mathrm{e}_\lambda[\alpha^* \oplus \beta^* - M] & D &= \mathbf{diag}(\tilde{b}) - \tilde{T}^{*\mathrm{T}}_\lambda \mathbf{diag}(a^{-1})\,\tilde{T}^*_\lambda \\ s_u &= a^{-1} \odot (\mu_r - \tilde{T}^* \tilde{s}_v) & \tilde{s}_v &= D^{-1}\left[\tilde{\mu}_c - \tilde{T}^{*\mathrm{T}}(a^{-1} \odot \mu_r)\right] \\ \mu_r &= (M \odot T^*)\mathbf{1}_m & \tilde{\mu}_c &= (\tilde{M} \odot \tilde{T}^*)^{\mathrm{T}}\mathbf{1}_n \end{aligned}$$

Error Bounds

Theorem
There exists a k_0 such that for every $k \geq k_0$,

$$\|\nabla_M S^{(k)} - \nabla_M S\|_F \leq C_S \sqrt{\varepsilon^{(k)}} = C_S \sqrt{f^{(0)} - f^*} \cdot r^{k/2}$$

Application and Experiments

Application: Deep Generative Models

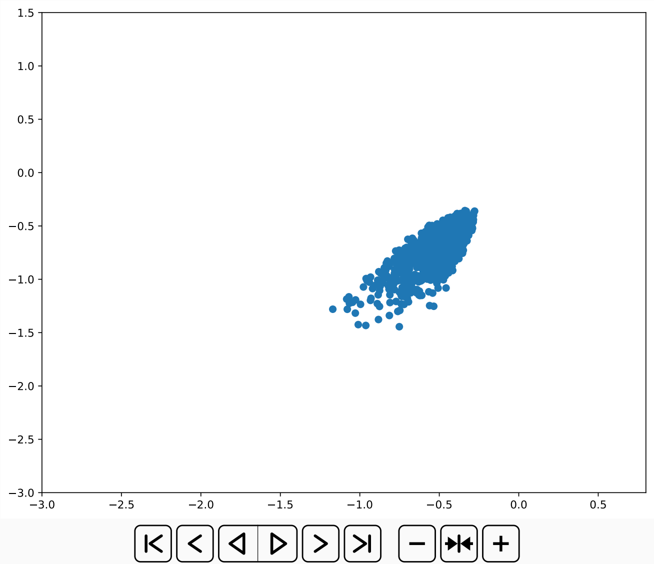
- Given a data set $X_1, \ldots, X_n \sim p^*$, find a deep neural network g_θ such that $g_\theta(Z) \sim p_\theta$ and $p_\theta \approx p^*$, where $Z \sim N(0, I_r)$
- This can be viewed as the implicit distribution estimation

Application: Deep Generative Models

- Given a data set $X_1, \ldots, X_n \sim p^*$, find a deep neural network g_θ such that $g_\theta(Z) \sim p_\theta$ and $p_\theta \approx p^*$, where $Z \sim N(0, I_r)$
- This can be viewed as the implicit distribution estimation

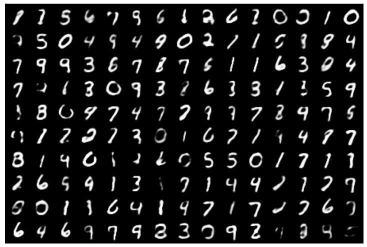
- Generate $Z_1, \ldots, Z_m \sim N(0, I_r)$, and let $Y_j = g_\theta(Z_j)$
- Cost matrix $(M_\theta)_{ij} = \|X_i - g_\theta(Z_j)\|^2$
- Find θ such that $\ell(\theta) = S_\lambda(M_\theta, n^{-1}\mathbf{1}_n, m^{-1}\mathbf{1}_m)$ is minimized

2D Example

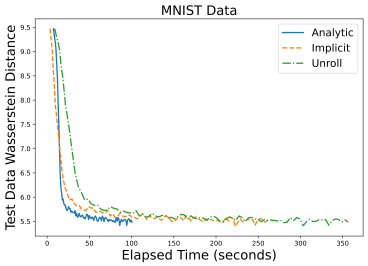


MNIST Dataset

- Compare the computing time of different methods
- Unroll: automatic differentiation; Implicit: existing software package; Analytic: proposed method

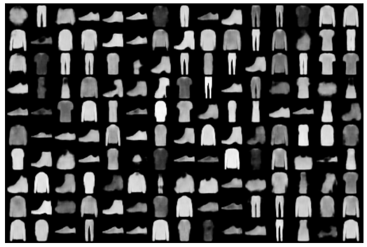


(a) Generated images

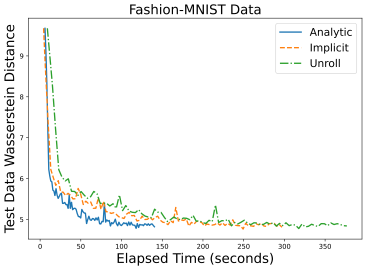


(b) Training process

Fasion-MNIST Dataset



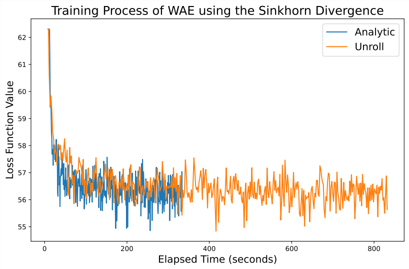
(c) Generated images



(d) Training process

CelebA Dataset

(e) Generated images



(f) Training process

Summary

- The Sinkhorn loss is an approximation to the popular Wasserstein distance
- We advocate the L-BFGS algorithm for forward pass, and analytic differentiation for the backward pass
- We rigorously prove that L-BFGS is stable and efficient for Sinkhorn loss
- Numerical results show the effiency of the advocated algorithms

THANK YOU!