EFFICIENT, STABLE, AND ANALYTIC DIFFERENTIATION OF THE SINKHORN LOSS

Anonymous authors
Paper under double-blind review

ABSTRACT

Optimal transport and the Wasserstein distance have become indispensable building blocks of modern deep generative models, but their computional costs greatly prohibit their applications in statistical machine learning models. Recently, the Sinkhorn loss, as an approximation to the Wasserstein distance, has gained massive popularity, and much work has been done for its theoretical properties. To embed the Sinkhorn loss into gradient-based learning frameworks, efficient algorithms for both the forward and backward passes of the Sinkhorn loss are required. In this article, we first demonstrate issues of the widely-used Sinkhorn's algorithm, and show that the L-BFGS algorithm is a potentially better candidate for the forward pass. Then we derive an analytic form of the derivative of the Sinkhorn loss with respect to the input cost matrix, which results in a very efficient backward algorithm. We rigorously analyze the convergence and stability properties of the advocated algorithms, and use various numerical experiments to validate the superior performance of the proposed methods.

1 INTRODUCTION

Optimal transport (OT, Villani, 2009) is a powerful tool to characterize the transformation of probability distributions, and has become an indispensable building block of generative modeling. At the core of OT is the Wasserstein distance, which measures the difference between two distributions. For example, the Wasserstein generative adversarial network (WGAN, Arjovsky et al., 2017) uses the 1-Wasserstein distance as the loss function to minimize the difference between the data distribution and the model distribution, and a huge number of related works emerge afterwards.

Despite the various appealing theoretical properties, one major barrier for the wide applications of OT is the difficuilty in computing the Wasserstein distance. For two discrete distributions, OT solves a linear programming problem of nm variables, where n and m are the number of Diracs that define the two distributions. As a result, the computational cost of OT quickly becomes formidable, except for some special cases (Peyré et al., 2019).

To resolve this issue, many approximate solutions to OT have been proposed, among which the Sinkhorn distance, or Sinkhorn loss, has gained massive popularity (Cuturi, 2013). The Sinkhorn loss can be viewed as an entropic-regularized Wasserstein distance, which adds a smooth penalty term to the original objective function of OT. The Sinkhorn loss is attractive as its optimization problem can be efficiently solved, at least in exact arithmetics, via Sinkhorn's algorithm (Sinkhorn, 1964; Sinkhorn & Knopp, 1967), which merely involves matrix-vector multiplications and some other minor operations. Therefore, it is especially suited to modern computing hardware such as the graphics processing units (GPUs).

Many existing works on the Sinkhorn loss focus on its theoretical properties, for example Mena & Niles-Weed (2019) and Genevay et al. (2019). In this article, we are mostly concerned with the computational aspect. Since modern deep generative models mostly rely on the gradient-based learning framework, it is crucial to use the Sinkhorn loss with differentiation support. One simple and natural method to enable Sinkhorn loss in back-propagation is to unroll Sinkhorn's algorithm, adding every iteration to the auto-differentiation computing graph (Genevay et al., 2018; Cuturi et al., 2019). However, this approach is typically costly when the number of iterations are large. Instead, in this article we have derived an *analytic* expression for the derivative of Sinkhorn loss based on quantities

computed from the forward pass, which drastically simplifies the back-propagation of the Sinkhorn loss.

More importantly, one critical pain point of the Sinkhorn loss, though typically ignored in theoretical studies, is that Sinkhorn's algorithm is numerically unstable (Peyré et al., 2019). We show in numerical experiments that even for very simple settings, Sinkhorn's algorithm can quickly lose precision. Various stabilized versions of Sinkhorn's algorithm, though showing better numerical stability, still suffer from slow convergence in these cases. In this article, we have rigorously analyzed the solution to the Sinkhorn optimization problem, and have designed both forward and backward algorithms that are provably efficient and stable. The highlights of this article are summarized as follows:

- We have derived an analytic expression for the derivative of the Sinkhorn loss, which can be efficiently computed in back-propagation.
- We have rigorously analyzed the advocated forward and backward algorithms for the Sinkhorn loss, and show that they have superior efficiency and stability properties.
- We have implemented the Sinkhorn loss as an auto-differentiable function in the PyTorch and JAX frameworks, using the analytic derivative obtained in this article.

The code to reproduce the results in this article is available at `https://1drv.ms/u/s!ArsORq8a24WmoFjNQtZYE_BERzDQ`.

2 THE SINKHORN LOSS AS APPROXIMATE OT

Throughout this article we focus on discrete OT problems. Denote by $\Delta_n = \{w \in \mathbb{R}_+^n : w^{\mathrm{T}}\mathbf{1}_n = 1\}$ the n-dimensional probability simplex, and let $\mu = \sum_{i=1}^n a_i\delta_{x_i}$ and $\nu = \sum_{j=1}^m b_j\delta_{y_j}$ be two discrete probability measures supported on data points $\{x_i\}_{i=1}^n$ and $\{y_j\}_{j=1}^m$, respectively, where $a = (a_1, \ldots, a_n)^{\mathrm{T}} \in \Delta_n$, $b = (b_1, \ldots, b_m)^{\mathrm{T}} \in \Delta_m$, and δ_x is the Dirac at position x. Let $M \in \mathbb{R}^{n\times m}$ be a cost matrix with entries $M_{ij}, i = 1, \ldots, n, j = 1, \ldots, m$. Define the set $\Pi(a, b)$ as

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

and then OT can be characterized by the following optimization problem,

$$W(M, a, b) = \min_{P\in\Pi(a,b)} \langle P, M\rangle, \tag{1}$$

where $\langle A, B\rangle = \mathrm{tr}(A^{\mathrm{T}}B)$. The optimal solution to (1), denoted as P^*, is typically called the optimal transport plan, and can be viewed as a joint distribution whose marginals coincide with μ and ν. The optimal value $W(M, a, b) = \langle P^*, M\rangle$ is then called the Wasserstein distance between μ and ν if the cost matrix M satisfies some suitable conditions.

As is introduced in Section 1, solving the optimization problem (1) can be difficult even for moderate n and m. One approach to regularizing the optimization problem is to add an entropic penalty term to the objective function, leading to the entropic-regularzied OT problem (Cuturi, 2013):

$$\tilde{S}_\lambda(M, a, b) = \min_{T\in\Pi(a,b)} \mathcal{S}_\lambda(T) \coloneqq \min_{T\in\Pi(a,b)} \langle T, M\rangle - \lambda^{-1}h(T), \tag{2}$$

where $h(T) = \sum_{i=1}^n \sum_{j=1}^m T_{ij}(1 - \log T_{ij})$ is the entropy term. The new objective function $\mathcal{S}_\lambda(T)$ is λ^{-1}-strongly convex on $\Pi(a, b)$, so (2) has a unique global solution, denoted as T_λ^*. In this article, T_λ^* is referred to as the Sinkhorn transport plan. The entropic-regularized Wasserstein distance, also known as the Sinkhorn distance or Sinkhorn loss in the literature (Cuturi, 2013), is then defined as

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

To simplify the notation, in what follows we omit the subscript λ in T_λ^* when no confusion is caused. It is worth noting that in the literature, $S_\lambda(M, a, b)$ is sometimes referred to as the *sharp* Sinkhorn loss, whereas $\tilde{S}_\lambda(M, a, b)$ is called the *regularized* Sinkhorn loss. In this article, we focus on the sharp version, and simply call $S_\lambda(M, a, b)$ the Sinkhorn loss for brevity.

The Sinkhorn loss $S_\lambda(M, a, b)$ can be used as a proxy for the Wasserstein distance between μ and ν, and it has a benign approximation property:

Proposition 1 (Luise et al., 2018)**.** *There exists a constant $C > 0$ such that for any $\lambda > 0$, $|S_\lambda(M, a, b) - W(M, a, b)| \leq Ce^{-\lambda}$. The constant C is independent of λ, and depends on μ and ν.*

3 DIFFERENTIATION OF THE SINKHORN LOSS

To use the Sinkhorn loss in deep neural networks or other machine learning tasks, it is also crucial to obtain the derivative of $S_\lambda(M, a, b)$ with respect to its input parameters. Differentiating the Sinkhorn loss typically involves two stages, the forward and backward passes. In the forward pass, the Sinkhorn loss or the transport plan is computed using some optimization algorithm, and in the backward pass the derivative is computed, using either an analytic expression or the automatic differentiation technique.

3.1 ISSUES OF SINKHORN'S ALGORITHM

In the existing literature, one commonly-used method for the forward pass of the Sinkhorn loss is Sinkhorn's algorithm (Sinkhorn, 1964; Sinkhorn & Knopp, 1967). Unlike the original linear programming problem (1), the solution to the Sinkhorn problem has a special structure. Cuturi (2013) shows that the optimal solution T^* can be expressed as

$$T^* = \mathbf{diag}(u^*) M_e \mathbf{diag}(v^*) \tag{3}$$

for some vectors u^* and v^*, where $M_e = \left(e^{-\lambda M_{ij}}\right)$, and $\mathbf{diag}(v)$ stands for a diagonal matix whose diagonal is formed by a vector v. Sinkhorn's algorithm starts from an initial vector $v^{(0)} \in \mathbb{R}_+^m$, and generates iterates $u^{(k)} \in \mathbb{R}_+^n$ and $v^{(k)} \in \mathbb{R}_+^m$ as follows:

$$u^{(k+1)} \leftarrow a \odot [M_e v^{(k)}]^{-1}, \quad v^{(k+1)} \leftarrow b \odot [M_e^{\mathrm{T}} u^{(k+1)}]^{-1}, \tag{4}$$

where $\odot$ is the elementwise multiplication operator, and $\mu^- = (\mu_1^{-1}, \ldots, \mu_k^{-1})^{\mathrm{T}}$ for a vector $\mu = (\mu_1, \ldots, \mu_k)^{\mathrm{T}}$. It can be proved that $u^{(k)} \to u^*$ and $v^{(k)} \to v^*$, and then the Sinkhorn transport plan T^* can be recovered by (3).

Sinkhorn's algorithm is very efficient, as it only involves matrix-vector multiplication and other inexpensive operations. However, one major issue of Sinkhorn's algorithm is that the entries of $M_e = \left(e^{-\lambda M_{ij}}\right)$ may easily underflow when λ is large, making some elements of the vectors $M_e v^{(k)}$ and $M_e^{\mathrm{T}} u^{(k+1)}$ close to zero. As a result, some components of $u^{(k+1)}$ and $v^{(k+1)}$ would overflow. Therefore, Sinkhorn's algorithm in its original form is unstable, and in practice the iterations (4) are typically carried out in the logarithmic scale, which we call the Sinkhorn-log algorithm for simplicity. Besides, there are some other works also attempting to improve the numerical stability of Sinkhorn's algorithm (Schmitzer, 2019; Cuturi et al., 2022).

Despite the advancements of Sinkhorn's algorithm, one critical issue observed in practice is that Sinkhorn-type algorithms may be slow to converge, especially for small regularization parameters. This would severly slow down the computation, and may even give misleading results when the user sets a moderate limit on the total number of iterations. Below we show a motivating example to highlight this issue. Consider a triplet (M, a, b) for the Sinkhorn problem, and fix the regularization parameter λ^{-1} to be 0.001, where the detailed setting is provided in Section A.1. The true T^* matrix is visualized in Figure 1, along with the solutions given by various widely-used algorithms, including Sinkhorn's algorithm, Sinkhorn-log, and the stabilized scaling algorithm (Stabilized, Algorithm 2 of Schmitzer, 2019). The maximum number of iterations for all algorithms is 1000.

In Figure 1, it is clear that the plan given by Sinkhorn's algorithm is farthest to the true value, whereas the stable algorithms Sinkhorn-log and Stabilized greatly improve it. Sinkhorn's algorithm and Sinkhorn-log are equivalent in exact arithmetics, so their numerical differences highlight the need for numerically stable algorithms. However, Sinkhorn-log and Stabilized still have visible inconsistencies with the truth even after 1000 iterations.

3.2 THE ADVOCATED ALTERNATIVE FOR FORWARD PASS

To this end, we advocate a simple alternative scheme to solve the optimal plan T^*, and we show both theoretically and empirically that this method enjoys superior efficiency and stability. Consider

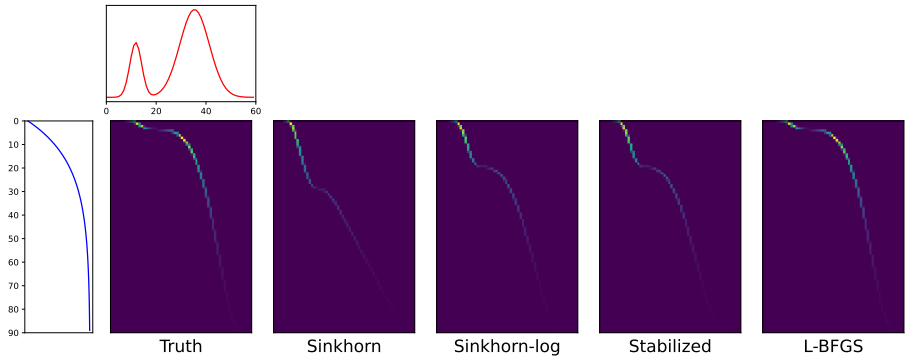


Figure 1: Visualization of Sinkhorn plans computed by different algorithms.

the dual problem of (2), which has the following form (Proposition 4.4 of Peyré et al., 2019):

$$\max_{\alpha,\beta} \mathcal{L}(\alpha,\beta) \coloneqq \max_{\alpha,\beta} \alpha^{\mathrm{T}} a + \beta^{\mathrm{T}} b - \lambda^{-1} \sum_{i=1}^{n} \sum_{j=1}^{m} e^{-\lambda(M_{ij}-\alpha_i-\beta_j)}, \quad \alpha \in \mathbb{R}^n, \beta \in \mathbb{R}^m. \tag{5}$$

Let $\alpha^* = (\alpha_1^*, \ldots, \alpha_n^*)^{\mathrm{T}}$ and $\beta^* = (\beta_1^*, \ldots, \beta_m^*)^{\mathrm{T}}$ be the optimal solutions to (5), and then the Sinkhorn transport plan T^* can be recovered as $T^* = \mathrm{e}_\lambda[\alpha^* \oplus \beta^* - M]$, where $\alpha \oplus \beta$ and $\mathrm{e}_\lambda[A]$ stand for matrices $(\alpha_i + \beta_j)$ and $(e^{\lambda A_{ij}})$, respectively. Remarkably, (5) is an *unconstrained* convex optimization problem, so a simple gradient ascent method suffices to find its optimal solution. But in practice, quasi-Newton methods such as the limited memory BFGS method (L-BFGS, Liu & Nocedal, 1989) can significantly accelerate the convergence. Using L-BFGS to solve (5) is a known practice (Cuturi & Peyré, 2018; Flamary et al., 2021), but little is known about its stability in solving the regularized OT problem. We first briefly describe the algorithm below, and in Section 4 we rigorously prove that L-BFGS converges fast and generates stable iterates for the Sinkhorn dual problem (5).

It is worth mentioning that one special step needs to be taken into account in applying L-BFGS to (5). As pointed out by Cuturi et al. (2019), the variables (α, β) in (5) has one redundant degree of freedom: if (α^*, β^*) is one solution to (5), then so is $(\alpha^* + c\mathbf{1}_n, \beta^* - c\mathbf{1}_m)$ for any c. Therefore, we take $\gamma = (\alpha, \tilde{\beta})$ as the free variables, where $\tilde{\beta} = (\beta_1, \ldots, \beta_{m-1})^{\mathrm{T}}$, and globally set $\beta_m = 0$.

The L-BFGS algorithm requires the access to both the objective function and its gradient with respect to the free variables. In this problem, the objective function is defined as

$$f(\gamma) = -\alpha^{\mathrm{T}} a - \beta^{\mathrm{T}} b + \lambda^{-1} \mathbf{1}_n^{\mathrm{T}} \mathrm{e}_\lambda[\alpha \oplus \beta - M] \mathbf{1}_m. \tag{6}$$

For convenience, let $T = \mathrm{e}_\lambda[\alpha \oplus \beta - M]$, and then it is easy to find that $\nabla_\alpha f = T\mathbf{1}_m - a$ and $\nabla_{\tilde{\beta}} f = [T^{\mathrm{T}}\mathbf{1}_n - b]_{1:(m-1)}$, where $[v]_{1:r}$ means the first r elements of a vector v. With $f(\gamma)$ and $\nabla_\gamma f = (\nabla_\alpha f, \nabla_{\tilde{\beta}} f)$, the L-BFGS algorithm can be readily used to minimize (6).

To demonstrate the advantage of L-BFGS, for the motivating example above we show its computed transport plan in the rightmost plot of Figure 1. Clearly, the solution given by L-BFGS is visually identical to the correct solution. To study the difference between L-BFGS and Sinkhorn's algorithm in more depth, we compute the objective function value of the dual problem (6) at each iteration for both Sinkhorn-log and L-BFGS. The results are visualized in Figure 2, with three different λ values, $\lambda^{-1} = 0.1, 0.01, 0.001$. Since each L-BFGS iteration may involve more than one gradient evaluation, for L-BFGS we plot the values against both the outer iteration and gradient evaluation counts.

Figure 2 gives a clear clue to the issue of Sinkhorn-log: it has a surprisingly slow convergence speed compared to L-BFGS when λ^{-1} is small. Theoretically, Sinkhorn algorithms will eventually converge with sufficient iterations, but in practice, a moderate limit on computational budget may prevent them from generating accurate results. To this end, L-BFGS appears to be a better candidate when one needs a small regularization parameter for better approximation to the OT problem.

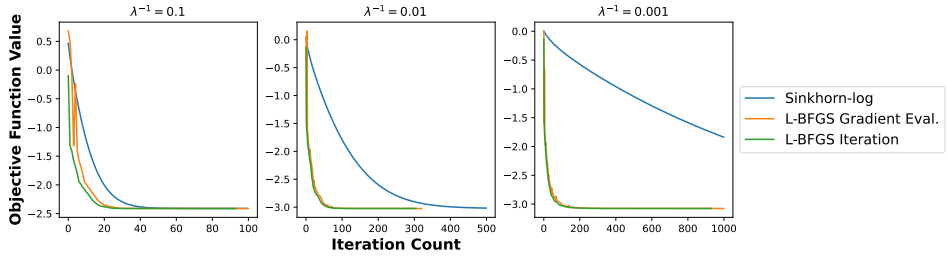


Figure 2: Comparing the convergence speed of Sinkhorn-log and L-BFGS.

3.3 THE ANALYTIC BACKWARD PASS

For the backward pass, one commonly used method is the *unrolled* Sinkhorn's algorithm, which is based on the fact that Sinkhorn's forward pass algorithm is differentiable with respect to a, b, and M. Therefore, one can use automatic differentiation software to compute the corresponding derivatives in the backward pass. This method is used in Genevay et al. (2018) for learning generative models with the Sinkhorn loss, but in practice it may be extremely slow if the forward pass takes a large number of iterations. To avoid the exessive cost of unrolled algorithms, various implicit differentiation methods have been developed for the Sinkhorn loss (Feydy et al., 2019; Campbell et al., 2020; Xie et al., 2020; Eisenberger et al., 2022), but they still do not provide the most straightforward way to compute the gradient.

To this end, we advocate the use of analytic derivatives of the Sinkhorn loss, which solves the optimization problem (2) in the forward pass, and use the optimal dual variables (α^*, β^*) for the backward pass. The analytic form for $\nabla_{a,b} S_\lambda(M, a, b)$ has been studied in Luise et al. (2018), and to our best knowledge, few result has been presented for $\nabla_M S_\lambda(M, a, b)$. Our first main theorem, given in Theorem 1, fills this gap and derives the analytic form for $\nabla_M S_\lambda(M, a, b)$.

Theorem 1. *For a vector $\mu = (\mu_1, \ldots, \mu_k)^{\mathrm{T}}$, denote by $\tilde{\mu} = (\mu_1, \ldots, \mu_{k-1})^{\mathrm{T}}$ the vector containing the first $(k-1)$ elements of μ. Similarly for a matrix A, let $\tilde{A}$ be the matrix formed by removing the last column of A. Then for a fixed $\lambda > 0$,*

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

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

4 CONVERGENCE AND STABILITY ANALYSIS

In Sections 3.2 and 3.3, we have introduced the advocated forward and backward algorithms for the Sinkhorn loss, respectively. In this section, we focus on the theoretical properties of these algorithms, and show that they enjoy superior efficiency and stability. As a first step, we consider the target of the optimization problem (6), and show that $f(\gamma)$ has a well-behaved minimizer, which does not underflow or overflow.

Theorem 2. *Denote by f^* the minimum value of $f(\gamma)$ and γ^* the optimal solution. Then $f^* > -\infty$, γ^* is unique, and $\|\gamma^*\| < \infty$. In particular, let $I = \arg\max_i T^*_{im}$, and then we have $L_{\alpha_j} \le \alpha^*_i \le U_{\alpha_j}$ and $L_{\beta_j} \le \beta^*_j \le U_{\beta_j}$ for all $i = 1, \ldots, n$ and $j = 1, \ldots, m$, where*

$$\begin{aligned}
U_{\alpha_i} &= M_{im} + \lambda^{-1}\min\{\log(a_i), \log(b_m)\}, \\
U_{\beta_j} &= M_{Ij} - M_{Im} + \lambda^{-1}\min\{\log(na_I/b_m), \log(nb_j/b_m)\}, \\
L_{\alpha_j} &= \lambda^{-1}\log(a_i/m) - \max_j (U_{\beta_j} - M_{ij}), \quad L_{\beta_j} = \lambda^{-1}\log(b_j/n) - \max_i (U_{\alpha_i} - M_{ij}).
\end{aligned}$$

Theorem 2 shows that the optimal dual variables γ^* are bounded, and more importantly, the bounds are roughly at the scale of the cost matrix entries M_{ij} and the log-weights $\log(a_i)$ and $\log(b_j)$. Therefore, at least the target of the optimization problem is well-behaved.

After verifying the properties of the optimal solution, a more interesting and critical problem is to seek a stable algorithm to approach the optimal solution. Indeed, in Theorem 3 we prove that the L-BFGS algorithm for minimizing (6) is one such method.

Theorem 3. *Let $\{\gamma^{(k)} := (\alpha^{(k)}, \tilde{\beta}^{(k)})\}$ be a sequence of iterates generated by the L-BFGS algorithm starting from a fixed initial value $\gamma^{(0)}$, and define $\beta^{(k)} = (\tilde{\beta}^{(k)}, 0)$, $T^{(k)} = \mathrm{e}_\lambda[\alpha^{(k)} \oplus \beta^{(k)} - M]$, and $f^{(k)} = f(\gamma^{(k)})$. Then there exist constants $0 \le r < 1$ and $C_1, C_2 > 0$ such that for each $k > 0$:*

(a) $f^{(k)} - f^* \le r^k (f^{(0)} - f^*) := \varepsilon^{(k)}$. *(Linear convergence for the objective value)*

(b) $\|\gamma^{(k)} - \gamma^*\|^2 \le C_1 \varepsilon^{(k)}$. *(Linear convergence for the iterates)*

(c) $\|\nabla_\gamma f(\gamma^{(k)})\|^2 \le C_2 \varepsilon^{(k)}$. *(Exponential decay of the gradient)*

(d) $0 < T_{ij}^{(k)} < \min\{a_i, b_j\} + \sqrt{C_2 \varepsilon^{(k)}}$. *($T^{(k)}$ does not overflow)*

(e) $\max_i T_{ij}^{(k)} > (b_j - \sqrt{C_2 \varepsilon^{(k)}})/n$, $\max_j T_{ij}^{(k)} > (a_i - \sqrt{C_2 \varepsilon^{(k)}})/m$. *($T^{(k)}$ does not underflow)*

The constants C_1 and C_2 implicitly depend on the initial value $\gamma^{(0)}$.

Theorem 3 reveals some important information. First, both the objective function value and the iterates have linear convergence speed, so the forward pass using L-BFGS takes $\mathcal{O}(\log(1/\varepsilon))$ iterations to obtain an ε-optimal solution. Second, the norm of the gradient is bounded at *any* iteration, and decays exponentially fast to zero. This validates the numerical stability of the L-BFGS algorithm. Third, the estimated transport plan at any iteration k, $T^{(k)}$, is also bounded and stable. This result can be compared with the formulation in (3): it is not hard to find that u^*, v^*, and M_e, when computed individually, can all be unstable due to the exponentiation operations, especially when λ is large. In contrast, T^* and $T^{(k)}$, thanks to the results in Theorem 3, never overflow or underflow.

The analysis above thus suggests a numerically stable method to evaluate $\nabla_\gamma f$, which is the key to applying the L-BFGS algorithm: (1) form the matrix $\alpha^{(k)} \oplus \beta^{(k)} - M$; (2) take the exponentiation operation $\mathrm{e}_\lambda[\cdot]$ to get $T^{(k)}$; (3) compute $\nabla_\alpha f(\gamma^{(k)}) = T^{(k)} \mathbf{1}_m - a$ and $\nabla_{\tilde{\beta}} f(\gamma^{(k)}) = [T^{(k)\mathrm{T}} \mathbf{1}_n - b]_{1:(m-1)}$. Although the exponentiation operations are more expensive than matrix multiplication as in Sinkhorn's algorithm, this extra cost can be greatly remedied by modern hardware such as GPUs. On the other hand, the huge gain in numerical stability is much more beneficial. Moreover, in our numerical experiments, we find that L-BFGS has an appealing convergene speed compared with other stable algorithms, so overall it is a good candidate for the forward pass of the Sinkhorn loss.

Likewise, we show that the derivative of $\nabla_M S_\lambda(M, a, b)$ as in Theorem 1 can also be computed in a numerically stable way. Let $\widehat{\nabla_M S}$ be the k-step approximation to $\nabla_M S := \nabla_M S_\lambda(M, a, b)$ using the L-BFGS algorithm, *i.e.*, replacing every T^* in $\nabla_M S$ by $T^{(k)}$. Then we have the following convergence result.

Theorem 4. *Using the symbols defined in Theorems 1 and 3, let $\sigma = 1/\lambda_{\min}(D)$, where $\lambda_{\min}(D)$ is the smallest eigenvalue of D. Assume that for some k_0,*

$$\varepsilon^{(k_0)} < C_1^{-1} \left[\frac{\min\{1, (6\sigma \|D\|_F)^{-1}\}}{4\lambda} \right]^2,$$

and then for every $k \ge k_0$,

$$\|\widehat{\nabla_M S} - \nabla_M S\|_F \le C_S \sqrt{\varepsilon^{(k)}},$$

where

$$\begin{aligned} C_S &= 4\lambda \sqrt{C_1} \left[\|\nabla_M S\|_F + 2\lambda \|T^*\|_F (C_v + C_u) \right], \\ C_v &= 2\sigma(\|\mu_c\| + 3\|T^{*\mathrm{T}}(a^- \odot \mu_r)\| + 3\|D\|_F \|s_v\|), \\ C_u &= \|\mu_r\| + 2C_v \|\mathbf{diag}(a^-) T^*\|_F + \|a^- \odot (T^* s_v)\|. \end{aligned}$$

5 APPLICATION: SINKHORN GENERATIVE MODELING

The Sinkhorn loss is useful in unsupervised learning tasks that attempt to match two distributions p_θ and p^*, where p^* stands for the data distribution, and p_θ is the model distribution. If p_θ and p^* can be represented or approximated by two discrete measures μ_θ and ν, respectively, then one would wish to minimize the Sinkhorn loss $S_\lambda(M_\theta, a_\theta, b_\theta)$ between μ_θ and ν, where the cost matrix M and weights a, b may depend on learnable parameters θ. In gradient-based learning frameworks, the key step of seeking the optimal parameter θ that minimizes $S_\lambda(M_\theta, a_\theta, b_\theta)$ is to compute the gradient $\nabla_\theta S_\lambda(M_\theta, a_\theta, b_\theta)$, which further reduces to evaluating $\nabla_{a,b} S_\lambda(M, a, b)$ and $\nabla_M S_\lambda(M, a, b)$.

Luise et al. (2018) assumes that M is fixed, and studies the gradients $\nabla_{a,b} S_\lambda(M, a, b)$. However, in many generative models it is more important to derive $\nabla_M S_\lambda(M, a, b)$, as the weights a and b are typically fixed, whereas the cost matrix M is computed from the output of some parameterized layers. Consider a data set $X_1, \ldots, X_n$ that follows the distribution p^*, $X_i \in \mathbb{R}^p$, and our target is to fit a deep neural network $g_\theta : \mathbb{R}^r \to \mathbb{R}^p$ such that $g_\theta(Z)$ approximately follows p^*, where $Z \sim N(0, I_r)$. To this end, we first generate random variates $Z_1, \ldots, Z_m \sim N(0, I_r)$, and let $Y_j = g_\theta(Z_j)$. The two weight vectors are simply taken to be $a = n^{-1}\mathbf{1}_n$, $b = m^{-1}\mathbf{1}_m$, and the cost matrix between X_i and Y_j is given by $M_\theta \in \mathbb{R}^{n\times m}$, where $(M_\theta)_{ij} = \|X_i - g_\theta(Z_j)\|^2$. Then we learn the network parameter θ by minimizing the Sinkhorn loss $\ell(\theta) = S_\lambda(M_\theta, n^{-1}\mathbf{1}_n, m^{-1}\mathbf{1}_m)$. We show such applications in Section 7.

6 RELATED WORK

In Table 1 we list some related work on the differentiation of the Sinkhorn loss, and provide a brief summary of the contribution and limitation of each work. The target of our work is closest to that of Genevay et al. (2018), *i.e.*, to differentiate the Sinkhorn loss with respect to the input cost matrix. However, they use unrolled Sinkhorn's algorithm, so there is no analytic form for $\nabla_M S_\lambda$. Our work is mostly motivated by Luise et al. (2018), but they consider the derivative with respect to the weights instead of the cost matrix. Similarly, Cuturi et al. (2019) and Cuturi et al. (2020) consider gradients on weights and data points, but not the general cost matrix. Campbell et al. (2020); Xie et al. (2020); Eisenberger et al. (2022) all consider the derivative of the transport plan T^* using implicit differentiation. Although this is a more general problem than computing $\nabla_M S_\lambda$, it loses the special structure of the Sinkhorn loss. As a result, the compact matrix form of the derivative presented in Theorem 1 is unique. Furthermore, the storage requirement for our result is $O(nm)$, whereas some existing works need to store much larger matrices. Finally, very few work has rigorously analyzed the stability of the algorithm, in both the forward and backward passes.

Table 1: A brief summary of existing literature on differentiation of Sinkhorn loss.

	Target	Analytic solution	Derivative w.r.t. the cost matrix	Compact matrix form	Minimum storage requirement	Stability analysis
Genevay et al. (2018)	$\nabla_M S_\lambda$	✗	✓	✗	✗	✗
Luise et al. (2018)	$\nabla_{a,b} S_\lambda$	✓	✗	—	✓	✗
Cuturi et al. (2019)	$\nabla_{a,x} T^*$	✗	✗	—	✗	✓
Cuturi et al. (2020)	$\nabla_{b,x} T^*$	✓	✗	—	✓	✗
Campbell et al. (2020)	$\nabla_M T^*$	✓	✓	✗	✓	✗
Xie et al. (2020)	$\nabla_M T^*$	✓	✓	✗	✓	✗
Eisenberger et al. (2022)	$\nabla_M T^*$	✓	✓	✗	✗	✗
This article	$\nabla_M S_\lambda$	✓	✓	✓	✓	✓

7 NUMERICAL EXPERIMENTS

7.1 RUNNING TIME OF FORWARD AND BACKWARD PASSES

In our subsequent experiments, we compare three algorithms for differentiating the Sinkhorn loss: the proposed method ("Analytic") using L-BFGS for the forward pass and the analytic derivative for backward pass, the implicit differentiation method ("Implicit"), and the unrolled Sinkhorn algorithm ("Unroll"). Both Implicit and Unroll use Sinkhorn-log for the forward pass, and are implemented in the OTT-JAX library (Cuturi et al., 2022).

We simulate cost matrices of different dimensions, and compare the forward time and total time of different algorithms with both $\lambda^{-1} = 0.1$ and $\lambda^{-1} = 0.01$. The detailed experiment setting and results are given in Section A.2 and Table 2. The main conclusion is that the forward pass of Analytic is roughly the same as and usually faster than those of the other two algorithms, and the backward time is significantly faster. Overall, the proposed analytic differentiation method enjoys superior computational efficiency.

7.2 GENERATIVE MODELS ON TOY DATA SETS

In this section we apply the Sinkhorn loss to generative modeling, and test the accuracy and efficiency of the proposed algorithm. Following the methods in Section 5, we consider a toy data set with $n = 1000$ and $p = 2$ (shown in Figure 3), and we attempt to learn a neural network g_θ such that g_θ approximately pushes forward $Z_1, \ldots, Z_m \sim N(0, I_r)$ to the data distribution. In our experiments, g_θ is a fully-connected ReLU neural network with input dimension $r = 5$ and hidden dimensions 64-128-64. The number of latent data points is $m = 1000$.

In the first setting, we intentionally keep both the observed data and the latent points $Z_1, \ldots, Z_m$ fixed, so that the optimization of g_θ is a deterministic process without randomness, and the optimization variable obtained from each forward pass is used as the warm start for the next training iteration. At every ten iterations, we compute the 2-Wasserstein distance between the observed data and the pushforward points $\{g_\theta(Z_i)\}$, and we train g_θ for 2000 iterations using the Adam optimizer with a learning rate of 0.001. This setting, though not common in generative modeling, helps us to monitor the computation of gradients without the impact of random sampling. Moreover, the Wasserstein distance has an achievable lower bound of zero if g_θ is sufficiently expressive, so we can study the accuracy and efficiency of gradient computation by plotting the metric against running time.

The comparison results for the three algorithms are shown in the second and third plots of Figure 3, from which we have the following findings. First, when $\lambda^{-1} = 0.5$, the 2-Wasserstein distance will significantly increase after certain number of iterations, indicating that the Sinkhorn loss is not an accurate approximation to the Wasserstein distance when λ^{-1} is large. Second, it is clear that the Unroll method has the slowest computation, as it needs to unroll a potentially large computational graph in automatic differentiation. Third, the proposed anlaytic differentiation is the most efficient method among the three, which highlights the huge advantage of a closed-form expression for the derivative.

Finally, to examin the performance of differentiation methods in a genuine generative modeling setting, we use a regular training scheme that randomly samples the observed data and latent points at each iteration. Due to the first finding above, we choose the smaller λ^{-1} value for the Sinkhorn loss, and use a mini-batch of size $n = m = 128$ for training. We run 5000 iterations in total, and the metric curves are shown in the last plot of Figure 3. It can be found that the performance of the three algorithms are similar to the fixed-Z case: all three methods properly reduce the Wasserstein distance over time, but the proposed algorithm is the most efficient one.

7.3 DEEP GENERATIVE MODELS

Finally, we use the Sinkhorn loss to train larger and deeper generative models on the MNIST (LeCun et al., 1998) and Fashion-MNIST (Xiao et al., 2017) data sets. In this experiment, we do not pursue training the best possible generative model; instead, we primarily highlight the efficiency of the proposed analytic differentiation method. The architecture of neural networks and details of the training process are given in Section A.3. During the training process, we compute the 2-Wasserstein dis-

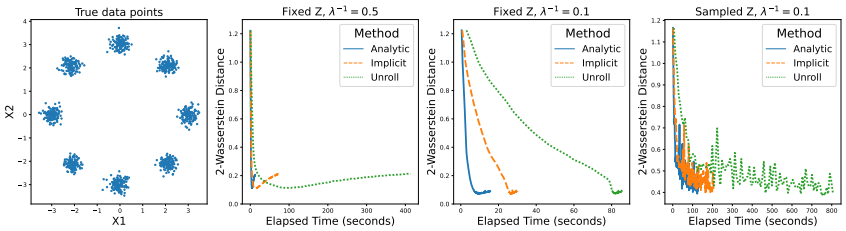
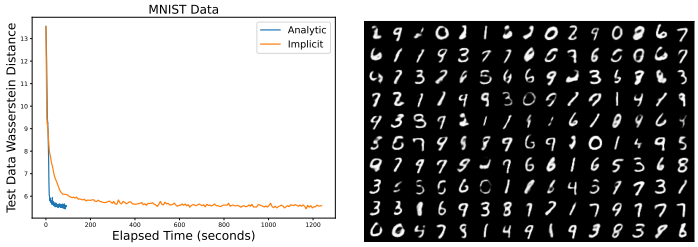
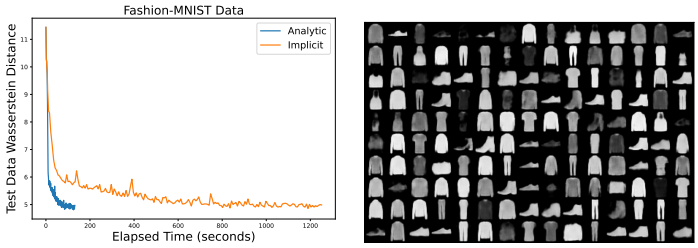


Figure 3: From left to right: the true observed data; the measured Wasserstein distance over time in the fixed Z setting, with Sinkhorn parameter $\lambda^{-1} = 0.1$; the similar plot for $\lambda^{-1} = 0.01$; the random Z setting with λ^{-1}.



(a) MNIST (b) MNIST



(c) Fashion-MNIST (d) Fashion-MNIST

Figure 4: (a)(c) 2-Wasserstein distances between randomly sampled test data and model-generated images over iterations. (b)(d) Randomly generated images from the models trained using the analytic differentiation.

tance between randomly sampled test data and model-generated images over iterations, and Figure 4 show the results of Analytic and Implicit algorithms. We do not include Unroll here since it is too time-consuming. The conclusion is similar: the proposed analytic differentiation method achives the same learning performance as existing methods, but significantly reduces the computing time. This once again highlights the value of the proposed method.

8 Conclusion

In this article we study the differentiation of the Sinkhorn loss with respect to its cost matrix, and have derived an analytic form of the derivative, which makes the backward pass of the differentiation very efficient. Moreover, we study the numerical stability of the forward pass, and rigorously prove that L-BFGS is a stable and efficient algorithm that complements the widely-used Sinkhorn's algorithm and its stablized versions. In particular, L-BFGS typically converges faster for Sinkhorn problems with small regularization parameters. It is worth noting that the proposed analytic differentiation method can be combined with different forward algorithms, and a reasonable scheme is to use L-BFGS for weakly regularized OT problems, and to choose Sinkhorn's algorithm otherwise. The differentiable Sinkhorn loss has many potential applications in generative modeling and permutation learning (Mena et al., 2018), and we anticipate that the technique developed in this article would boost future research on those directions.

REFERENCES

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, 2017.

Dylan Campbell, Liu Liu, and Stephen Gould. Solving the blind perspective-n-point problem end-to-end with robust differentiable geometric optimization. In *European Conference on Computer Vision*, pp. 244–261. Springer, 2020.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems 26*, 2013.

Marco Cuturi and Gabriel Peyré. Semi-dual regularized optimal transport. *arXiv preprint arXiv:1811.05527*, 2018.

Marco Cuturi, Olivier Teboul, and Jean-Philippe Vert. Differentiable ranking and sorting using optimal transport. In *Advances in Neural Information Processing Systems 32*, 2019.

Marco Cuturi, Olivier Teboul, Jonathan Niles-Weed, and Jean-Philippe Vert. Supervised quantile normalization for low rank matrix factorization. In *International Conference on Machine Learning*, 2020.

Marco Cuturi, Laetitia Meng-Papaxanthos, Yingtao Tian, Charlotte Bunne, Geoff Davis, and Olivier Teboul. Optimal transport tools (ott): A jax toolbox for all things wasserstein. *arXiv preprint arXiv:2201.12324*, 2022.

Marvin Eisenberger, Aysim Toker, Laura Leal-Taixé, Florian Bernard, and Daniel Cremers. Scalable sinkhorn backpropagation. *OpenReview preprint id:uR77O7SL55h*, 2022.

Jean Feydy, Thibault Séjourné, François-Xavier Vialard, Shun-ichi Amari, Alain Trouvé, and Gabriel Peyré. Interpolating between optimal transport and mmd using sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2681–2690. PMLR, 2019.

Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, et al. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021.

Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning generative models with sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, 2018.

Aude Genevay, Lénaic Chizat, Francis Bach, Marco Cuturi, and Gabriel Peyré. Sample complexity of sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, 2019.

Nicholas J Higham. *Accuracy and stability of numerical algorithms*. SIAM, 2002.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

Giulia Luise, Alessandro Rudi, Massimiliano Pontil, and Carlo Ciliberto. Differential properties of sinkhorn approximation for learning with wasserstein distance. In *Advances in Neural Information Processing Systems 31*, 2018.

Albert W Marshall, Ingram Olkin, and Barry C Arnold. *Inequalities: Theory of Majorization and Its Applications, Second Edition*. Springer Science & Business Media, 2011.

Gonzalo Mena and Jonathan Niles-Weed. Statistical bounds for entropic optimal transport: sample complexity and the central limit theorem. In *Advances in Neural Information Processing Systems 32*, 2019.

Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. Learning latent permutations with gumbel-sinkhorn networks. *arXiv preprint arXiv:1802.08665*, 2018.

Jorge Nocedal, Annick Sartenaer, and Ciyou Zhu. On the behavior of the gradient norm in the steepest descent method. *Computational Optimization and Applications*, 22(1):5–35, 2002.

Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

R Tyrrell Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

Bernhard Schmitzer. Stabilized sparse scaling algorithms for entropy regularized transport problems. *SIAM Journal on Scientific Computing*, 41(3):A1443–A1481, 2019.

Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics*, 35(2):876–879, 1964.

Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer, 2009.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Yujia Xie, Hanjun Dai, Minshuo Chen, Bo Dai, Tuo Zhao, Hongyuan Zha, Wei Wei, and Tomas Pfister. Differentiable top-k with optimal transport. In *Advances in Neural Information Processing Systems 33*, 2020.

A Additional experiment details

A.1 Settings of the motivating example

Consider a small problem of $n = 90$ and $m = 60$. Let $x_i = 5(i-1)/(n-1)$, $i = 1, \ldots, n$ be equally-spaced points on $[0, 5]$, and similarly define $y_j = 5(j-1)/(m-1)$, $j = 1, \ldots, m$. The cost matrix is set to $M_{ij} = (x_i - y_j)^2$, and the weights a and b are specified as follows. Let f_1 be the density function of an exponential distribution with mean 1, and f_2 be the density function of a mixture of two normal distributions, $0.2 \cdot N(1, 0.04) + 0.8 \cdot N(3, 0.25)$. And then we set $\tilde{a}_i = f_1(x_i)$, $\tilde{b}_j = f_2(y_j)$, $a_i = \tilde{a}_i / \sum_{k=1}^{n} \tilde{a_k}$, and $b_j = \tilde{b}_j / \sum_{k=1}^{m} \tilde{b}_k$.

We fix the regularization parameter λ^{-1} to be 0.001. This value is selected such that the resulting Sinkhorn plan T_λ^* is visually close to the OT plan P^*. In Figure 5, we show the Wasserstein and Sinkhorn transport plans under different values of λ^{-1}. It can be seen that when $\lambda^{-1} \leq 0.001$, T_λ^* is visually indistinguishable from P^*.

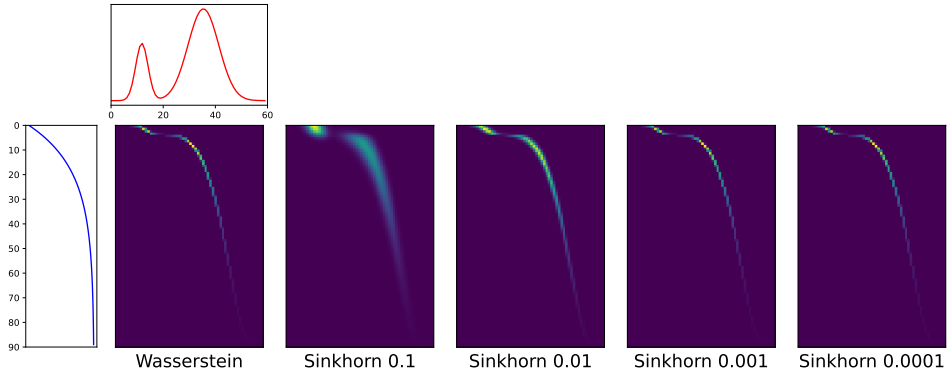


Figure 5: Visualization of Sinkhorn plans with different λ^{-1} values.

We compute the true T_λ^* using the ε-scaling algorithm (Algorithm 3 of Schmitzer, 2019). This algorithm is typically accurate, but it requires solving a sequence of Sinkhorn problems with increasing λ's, where the solution corresponding to the previous λ is used as a warm start for the next one. Therefore, its computational cost is typically large, and it does not compare fairly with other methods. Due to this reason, in this article we mainly use the ε-scaling algorithm to compute high-precision reference values, and do not include it for method comparison.

A.2 Running time of forward and backward passes

In this section we compare the running time of three algorithms on differenting the Sinkhorn loss.

(a) The proposed algorithm "Analytic": L-BFGS in the forward pass, and analytic differentiation in the backward pass.

(b) "Implicit" implemented in the OTT-JAX library (Cuturi et al., 2022): Sinkhorn-log in the forward pass, and implicit differentiation in the backward pass.

(c) "Unroll" implemented in the OTT-JAX library: Sinkhorn-log in the forward pass, and unrolled automatic differentiation in the backward pass.

We use the following data generation model to simulate data points. First, we simulate data $X_1, \ldots, X_n \sim f_1(x)$ and $Y_1, \ldots, Y_m \sim f_2(y)$ from some specific distributions $f_1(x)$ and $f_2(y)$, $x, y \in \mathbb{R}^p$, and construct the cost matrix as $M = (M_{ij})$, where $M_{ij} = \|X_i - Y_j\|^2$. The weights are fixed as $a = n^{-1}\mathbf{1}_n$ and $b = m^{-1}\mathbf{1}_m$. Both $f_1(x)$ and $f_2(y)$ have independent components. Each marginal distribution of f_1 is an exponential distribution with mean 1, and each marginal distribution of f_2 is a mixture of two normal distributions, $0.2 \cdot N(1, 0.04) + 0.8 \cdot N(3, 0.25)$.

Then we use the three algorithms above to compute the Sinkhorn loss and its derivative with respect to the cost matrix. For each configuration, we randomly generate the data 100 times, and compute their mean forward time and mean total time. Results for different data dimensions and regularization parameters are given in Table 2.

A.3 Architectures for deep generative models

For the deep generative models on MNIST and Fashion-MNIST data in Section 7.4, the architectures of the generators are given in Table 3, where FC_d stands for a fully-connected layer with d output dimensions, and $\mathrm{Conv}_{c,k,s,pin,pout}$ is the transposed convolutional layer with c output channels, kernel size k, stride s, input padding pin, and output padding $pout$.

Both models are trained using the Adam optimizer with learning rate 0.0001, on mini-batches of size 600. The Sinkhorn regularization parameter is set to $\lambda^{-1} = 0.1$, and the training process consists of two stages. In the first stage, we use the squared L^2 distance to construct the cost matrix, and in the second stage we switch to the L^1 distance. The intuition is that the squared L^2 distance has smoother derivatives, thereby making the training more stable in early steps; on the other hand, the L^1 distance makes the generated images sharper. The two stages are run for 20 and 30 epochs, respectively. The Wasserstein distances values in Figure 4 are computed in the first stage.

A.4 Computing environment

All of the experiments in this article were conducted on a personal computer with an Intel i9-11900K CPU and an Nvidia RTX 3090 GPU.

For the motivating example, results for Sinkhorn's algorithm, Sinkhorn-log, and Stabilized are computed using the Python Optimal Transport (POT) library (Flamary et al., 2021). For the experiments in Sections 7.2, 7.3, and A.2, the Implicit and Unroll algorithms are implemented in the OTT-JAX library (Cuturi et al., 2022).

Table 2: Running time of three algorithms for differentiating the Sinkhorn loss.

		Mean forward time (ms)	Std. of forward time	Mean total time (ms)	Std. of total time
$m = n = 64$ $p = 8$ $\lambda^{-1} = 0.1$	Analytic	17.98	2.00	**18.52**	1.90
	Implicit	26.35	0.35	96.12	3.38
	Unroll	26.41	0.12	154.41	0.43
$m = n = 64$ $p = 8$ $\lambda^{-1} = 0.01$	Analytic	21.83	3.30	**22.41**	3.32
	Implicit	26.56	0.09	96.71	2.95
	Unroll	26.55	0.13	154.77	0.10
$m = n = 128$ $p = 16$ $\lambda^{-1} = 0.1$	Analytic	22.78	2.95	**23.55**	2.91
	Implicit	28.84	0.36	189.99	0.27
	Unroll	28.92	0.15	161.45	0.17
$m = n = 128$ $p = 16$ $\lambda^{-1} = 0.01$	Analytic	30.91	4.54	**31.68**	4.54
	Implicit	28.96	0.11	183.65	1.81
	Unroll	28.91	0.14	161.44	0.18
$m = n = 256$ $p = 32$ $\lambda^{-1} = 0.1$	Analytic	26.86	3.33	**27.59**	3.27
	Implicit	32.94	0.23	372.70	6.03
	Unroll	33.03	0.38	174.49	0.14
$m = n = 256$ $p = 32$ $\lambda^{-1} = 0.01$	Analytic	41.29	5.64	**41.98**	5.61
	Implicit	33.30	0.20	365.77	7.00
	Unroll	33.39	0.13	175.24	0.63
$m = n = 512$ $p = 64$ $\lambda^{-1} = 0.1$	Analytic	39.54	4.99	**41.52**	5.04
	Implicit	40.91	0.04	674.25	10.39
	Unroll	40.90	0.03	211.14	0.43
$m = n = 512$ $p = 64$ $\lambda^{-1} = 0.01$	Analytic	64.75	9.19	**66.64**	9.22
	Implicit	40.83	0.02	857.25	12.42
	Unroll	40.89	0.03	210.00	0.42

Table 3: Architectures of the neural networks for MNIST and Fashion-MNIST data.

MNIST		Fashion-MNIST	
Layer	Output shape	Layer	Output shape
Input z	16	Input z	32
$\mathrm{FC}_{3136} + \mathrm{ReLU}$	3136	$\mathrm{FC}_{6272} + \mathrm{ReLU}$	6272
Reshape	$64 \times 7 \times 7$	Reshape	$128 \times 7 \times 7$
$\mathrm{Conv}_{32,4,2,1,0} + \mathrm{ReLU}$	$32 \times 14 \times 14$	$\mathrm{Conv}_{64,4,2,1,0} + \mathrm{ReLU}$	$64 \times 14 \times 14$
$\mathrm{Conv}_{32,4,2,1,1} + \mathrm{ReLU}$	$32 \times 29 \times 29$	$\mathrm{Conv}_{64,4,2,1,1} + \mathrm{ReLU}$	$64 \times 29 \times 29$
$\mathrm{Conv}_{1,4,1,2,0}$	$1 \times 28 \times 28$	$\mathrm{Conv}_{1,4,1,2,0}$	$1 \times 28 \times 28$

B PROOFS OF THEOREMS

B.1 TECHNICAL LEMMAS

In this section we state a few technical lemmas that are used to prove our main theorems. Lemma 1 to Lemma 3 below are standard conclusions in vector calculus, and Lemma 4 is derived from the eigenvalue theory.

We first introduce the following notation. For a matrix $A = (a_{ij}) \in \mathbb{R}^{n\times m}$, the vectorization operator, $\mathbf{vec}(A)$, creates a vector by stacking the column vectors of A together, *i.e.*,

$$\mathbf{vec}(A) = (a_{11}, \dots, a_{n1}, a_{12}, \dots, a_{n2}, \dots, a_{1m}, \dots, a_{nm})^{\mathrm{T}}.$$

For two matrices $A = (a_{ij}) \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^{p\times q}$, the Kronecker product of A and B is defined as

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1m}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \cdots & a_{nm}B \end{bmatrix}.$$

For a differentiable vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^m$, the partial derivative of f with respect to x is defined as

$$\frac{\partial f(x)}{\partial x^{\mathrm{T}}} = \begin{bmatrix} \frac{\partial f_1(x)}{\partial x_1} & \cdots & \frac{\partial f_1(x)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m(x)}{\partial x_1} & \cdots & \frac{\partial f_m(x)}{\partial x_n} \end{bmatrix}.$$

We use I_n to denote the $n \times n$ identity matrix.

Lemma 1. *Given two matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times r}$,*

$$\mathbf{vec}(AB) = (I_r \otimes A)\mathbf{vec}(B) = (B^{\mathrm{T}} \otimes I_m)\mathbf{vec}(A).$$

Lemma 2. *Let $f : \mathbb{R}^m \to \mathbb{R}^n$ and $g : \mathbb{R}^m \to \mathbb{R}^n$ be two vector-valued differentiable functions of $x \in \mathbb{R}^m$. Then*

$$\frac{\partial}{\partial x^{\mathrm{T}}} f(x)^{\mathrm{T}} g(x) = g(x)^{\mathrm{T}} \frac{\partial f(x)}{\partial x^{\mathrm{T}}} + f(x)^{\mathrm{T}} \frac{\partial g(x)}{\partial x^{\mathrm{T}}}.$$

Lemma 3. *Let $f : \mathbb{R}^m \to \mathbb{R}^l$ and $g : \mathbb{R}^m \to \mathbb{R}^r$ be two vector-valued differentiable functions of $x \in \mathbb{R}^m$. Then*

$$\frac{\partial}{\partial x^{\mathrm{T}}} \mathbf{vec}(f(x)g(x)^{\mathrm{T}}) = (g(x) \otimes I_l)\frac{\partial f(x)}{\partial x^{\mathrm{T}}} + (I_r \otimes f(x))\frac{\partial g(x)}{\partial x^{\mathrm{T}}}.$$

Lemma 4. *Let A and B be two $n \times n$ positive definite matrices, and let $\alpha_1 \geq \cdots \geq \alpha_n > 0$ and $\beta_1 \geq \cdots \geq \beta_n > 0$ be the ordered eigenvalues of A and B, respectively. Then for any $x \in \mathbb{R}^n$,*

$$x^{\mathrm{T}} A^{1/2} B A^{1/2} x \leq \alpha_1 \beta_1 \|x\|^2.$$

Proof. Let $U_{1\times n} = x^{\mathrm{T}} A^{1/2}$, and then $u \coloneqq UU^{\mathrm{T}} = x^{\mathrm{T}} A x \leq \alpha_1 \|x\|^2$, and $UBU^{\mathrm{T}} = x^{\mathrm{T}} A^{1/2} B A^{1/2} x$. By Theorem A.4 in page 788 of Marshall et al. (2011), we immediately get

$$UBU^{\mathrm{T}} = \mathrm{tr}(UBU^{\mathrm{T}}) \leq \beta_1 u \leq \alpha_1 \beta_1 \|x\|^2.$$

□

B.2 PROOF OF THEOREM 1

By definition we have

$$S_\lambda(M, a, b) = \langle T^*, M \rangle = \mathbf{vec}(T^*)^{\mathrm{T}} \mathbf{vec}(M),$$

so Lemma 2 gives

$$\frac{\partial S_\lambda(M, a, b)}{\partial \mathbf{vec}(M)^{\mathrm{T}}} = \mathbf{vec}(M)^{\mathrm{T}} \frac{\partial \mathbf{vec}(T^*)}{\partial \mathbf{vec}(M)^{\mathrm{T}}} + \mathbf{vec}(T^*)^{\mathrm{T}} \frac{\partial \mathbf{vec}(M)}{\partial \mathbf{vec}(M)^{\mathrm{T}}}. \tag{7}$$

Obviously, $\partial\mathbf{vec}(M)/\partial\mathbf{vec}(M)^{\mathrm{T}}$ is the $(nm)\times(nm)$ identity matrix $I_{(nm)}$, so the second term of (7) is essentially $\mathbf{vec}(T^*)^{\mathrm{T}}$, and the remaining task is to derive $\partial\mathbf{vec}(T^*)/\partial\mathbf{vec}(M)^{\mathrm{T}}$.

Let $R=\alpha^*\oplus\beta^*-M=\alpha^*\mathbf{1}_m^{\mathrm{T}}+\mathbf{1}_n\beta^{*\mathrm{T}}-M$, and then $T^*=\mathrm{e}_\lambda[R]$. Using the chain rule of derivatives, we have

$$\frac{\partial\mathbf{vec}(T^*)}{\partial\mathbf{vec}(M)^{\mathrm{T}}}=\frac{\partial\mathbf{vec}(T^*)}{\partial\mathbf{vec}(R)^{\mathrm{T}}}\cdot\frac{\partial\mathbf{vec}(R)}{\partial\mathbf{vec}(M)^{\mathrm{T}}}. \tag{8}$$

It is easy to find that $\partial\mathbf{vec}(T^*)/\partial\mathbf{vec}(R)^{\mathrm{T}}$ is an $(nm)\times(nm)$ diagonal matrix with diagonal elements $\mathbf{vec}(\lambda T^*)$, so

$$\mathbf{vec}(M)^{\mathrm{T}}\frac{\partial\mathbf{vec}(T^*)}{\partial\mathbf{vec}(R)^{\mathrm{T}}}=\lambda\mathbf{vec}(M\odot T^*)^{\mathrm{T}}. \tag{9}$$

Furthermore,

$$\begin{aligned}\frac{\partial\mathbf{vec}(R)}{\partial\mathbf{vec}(M)^{\mathrm{T}}}&=\frac{\partial\mathbf{vec}(\alpha^*\mathbf{1}_m^{\mathrm{T}}+\mathbf{1}_n\beta^{*\mathrm{T}}-M)}{\partial\mathbf{vec}(M)^{\mathrm{T}}}\\&=(\mathbf{1}_m\otimes I_n)\frac{\partial\alpha^*}{\partial\mathbf{vec}(M)^{\mathrm{T}}}+(I_m\otimes\mathbf{1}_n)\frac{\partial\beta^*}{\partial\mathbf{vec}(M)^{\mathrm{T}}}-I_{(nm)},\end{aligned} \tag{10}$$

where the second identity is an application of Lemma 3. Combine (8), (9) and (10), and then we get

$$\begin{aligned}\mathbf{vec}(M)^{\mathrm{T}}\frac{\partial\mathbf{vec}(T^*)}{\partial\mathbf{vec}(M)^{\mathrm{T}}}&=\lambda\mathbf{vec}(M\odot T^*)^{\mathrm{T}}\frac{\partial\mathbf{vec}(R)}{\partial\mathbf{vec}(M)^{\mathrm{T}}}\\&=\lambda\mathbf{vec}(M\odot T^*)^{\mathrm{T}}(\mathbf{1}_m\otimes I_n)\frac{\partial\alpha^*}{\partial\mathbf{vec}(M)^{\mathrm{T}}}\\&\quad+\lambda\mathbf{vec}(M\odot T^*)^{\mathrm{T}}(I_m\otimes\mathbf{1}_n)\frac{\partial\beta^*}{\partial\mathbf{vec}(M)^{\mathrm{T}}}\\&\quad-\lambda\mathbf{vec}(M\odot T^*)^{\mathrm{T}},\end{aligned}$$

which is the first term of (7).

Using the identities in Lemma 1, we have

$$\begin{aligned}\mathbf{vec}(M\odot T^*)^{\mathrm{T}}(\mathbf{1}_m\otimes I_n)&=\left[(\mathbf{1}_m\otimes I_n)^{\mathrm{T}}\mathbf{vec}(M\odot T^*)\right]^{\mathrm{T}}=\left[(\mathbf{1}_m^{\mathrm{T}}\otimes I_n)\mathbf{vec}(M\odot T^*)\right]^{\mathrm{T}}\\&=\left[\mathbf{vec}((M\odot T^*)\mathbf{1}_m)\right]^{\mathrm{T}}\coloneqq\mu_r^{\mathrm{T}},\\\mathbf{vec}(M\odot T^*)^{\mathrm{T}}(I_m\otimes\mathbf{1}_n)&=\left[(I_m\otimes\mathbf{1}_n)^{\mathrm{T}}\mathbf{vec}(M\odot T^*)\right]^{\mathrm{T}}=\left[(I_m\otimes\mathbf{1}_n^{\mathrm{T}})\mathbf{vec}(M\odot T^*)\right]^{\mathrm{T}}\\&=\left[\mathbf{vec}(\mathbf{1}_n^{\mathrm{T}}(M\odot T^*))\right]^{\mathrm{T}}\coloneqq\mu_c^{\mathrm{T}}.\end{aligned}$$

Note that one of the elements in β^* is redundant. If (α^*,β^*) is an optimal solution, then so is (α^*+c,β^*-c) for any $c\in\mathbb{R}$. Therefore, without loss of generality we can fix the last element of β^* to be zero, *i.e.*, $\beta^*=(\tilde{\beta}^{*\mathrm{T}},0)^{\mathrm{T}}$. Then (7) simplifies to

$$\frac{\partial S_\lambda(M,a,b)}{\partial\mathbf{vec}(M)^{\mathrm{T}}}=\lambda\left[\mu_r^{\mathrm{T}}\frac{\partial\alpha^*}{\partial\mathbf{vec}(M)^{\mathrm{T}}}+\tilde{\mu}_c^{\mathrm{T}}\frac{\partial\tilde{\beta}^*}{\partial\mathbf{vec}(M)^{\mathrm{T}}}-\mathbf{vec}(M\odot T^*)^{\mathrm{T}}\right]+\mathbf{vec}(T^*)^{\mathrm{T}}. \tag{11}$$

Let $w^*=(\alpha^{*\mathrm{T}},\tilde{\beta}^{*\mathrm{T}})^{\mathrm{T}}$, and then the main challenge is to calculate $\partial w^*/\partial\mathbf{vec}(M)^{\mathrm{T}}$.

First, note that the optimality condition for $(\alpha^*,\beta^*)=\arg\max_{\alpha,\beta}\ \mathcal{L}(\alpha,\beta)$ is

$$\nabla_\alpha\mathcal{L}(\alpha,\beta)|_{(\alpha,\beta)=(\alpha^*,\beta^*)}=\mathbf{0},\quad\nabla_\beta\mathcal{L}(\alpha,\beta)|_{(\alpha,\beta)=(\alpha^*,\beta^*)}=\mathbf{0}. \tag{12}$$

For general vectors $\alpha=(\alpha_1,\ldots,\alpha_n)^{\mathrm{T}}$ and $\beta=(\tilde{\beta}^{\mathrm{T}},0)^{\mathrm{T}}=(\beta_1,\ldots,\beta_{m-1},0)^{\mathrm{T}}$, define $T=\mathrm{e}_\lambda[\alpha\oplus\beta-M]$, and then it is easy to find that $\nabla_\alpha\mathcal{L}(\alpha,\beta)=a-T\mathbf{1}_m$ and $\nabla_{\tilde{\beta}}\mathcal{L}(\alpha,\beta)=\tilde{b}-\tilde{T}^{\mathrm{T}}\mathbf{1}_n$. Moreover,

$$\begin{aligned}\nabla_\alpha^2\mathcal{L}(\alpha,\beta)&=-\lambda\mathbf{diag}(T\mathbf{1}_m)\\\nabla_{\tilde{\beta}}^2\mathcal{L}(\alpha,\beta)&=-\lambda\mathbf{diag}(\tilde{T}^{\mathrm{T}}\mathbf{1}_n)\\\nabla_{\tilde{\beta}}\left(\nabla_\alpha\mathcal{L}(\alpha,\beta)\right)&=-\lambda\tilde{T}.\end{aligned}$$

Define the function

$$F(w, M) = \begin{pmatrix} \nabla_\alpha \mathcal{L}(\alpha, \beta) \\ \nabla_{\tilde{\beta}} \mathcal{L}(\alpha, \beta) \end{pmatrix} = \begin{pmatrix} a - T\mathbf{1}_m \\ \tilde{b} - \tilde{T}^{\mathrm{T}}\mathbf{1}_n \end{pmatrix}, \tag{13}$$

where $w = (\alpha^{\mathrm{T}}, \tilde{\beta}^{\mathrm{T}})^{\mathrm{T}}$, and then $\tilde{w}^*$ satisfies the equation $F(w^*, M) = \mathbf{0}$, indicating that w^* is implicitly a function of M, written as $w^* = w(M)$. By the implicit function theorem,

$$\frac{\partial w(M)}{\partial \mathbf{vec}(M)^{\mathrm{T}}} = -\left[\frac{\partial F(w, M)}{\partial w^{\mathrm{T}}}\bigg|_{w=w^*}\right]^{-1} \frac{\partial F(w, M)}{\partial \mathbf{vec}(M)^{\mathrm{T}}}\bigg|_{w=w^*} := -F_w^{-1} F_M.$$

Note that

$$F_w = F_w^{\mathrm{T}} = \begin{pmatrix} \nabla_\alpha^2 \mathcal{L}(\alpha, \beta) & \nabla_{\tilde{\beta}} \left(\nabla_\alpha \mathcal{L}(\alpha, \beta)\right) \\ \left[\nabla_{\tilde{\beta}} \left(\nabla_\alpha \mathcal{L}(\alpha, \beta)\right)\right]^{\mathrm{T}} & \nabla_{\tilde{\beta}}^2 \mathcal{L}(\alpha, \beta) \end{pmatrix} := -\lambda \begin{pmatrix} A & \tilde{B} \\ \tilde{B}^{\mathrm{T}} & \tilde{D} \end{pmatrix}.$$

Then by the inversion formula for block matrices, we have

$$F_w^{-1} = -\lambda^{-1} \begin{pmatrix} A & \tilde{B} \\ \tilde{B}^{\mathrm{T}} & \tilde{D} \end{pmatrix}^{-1} = -\lambda^{-1} \begin{pmatrix} A^{-1} + A^{-1}\tilde{B}\tilde{\Delta}^{-1}\tilde{B}^{\mathrm{T}}A^{-1} & -A^{-1}\tilde{B}\tilde{\Delta}^{-1} \\ -\tilde{\Delta}^{-1}\tilde{B}^{\mathrm{T}}A^{-1} & \tilde{\Delta}^{-1} \end{pmatrix},$$

where $\tilde{\Delta} = \tilde{D} - \tilde{B}^{\mathrm{T}}A^{-1}\tilde{B}$. For $g = (\mu_r^{\mathrm{T}}, \tilde{\mu}_c^{\mathrm{T}})^{\mathrm{T}}$, the vector $\tilde{s} = (s_u^{\mathrm{T}}, \tilde{s}_v^{\mathrm{T}})^{\mathrm{T}} = -\lambda F_w^{-1}\tilde{g}$ has the following expression:

$$\begin{aligned} \tilde{s}_v &= -\tilde{\Delta}^{-1}\tilde{B}^{\mathrm{T}}A^{-1}\mu_r + \tilde{\Delta}^{-1}\tilde{\mu}_c \\ s_u &= A^{-1}\mu_r + A^{-1}\tilde{B}\tilde{\Delta}^{-1}\tilde{B}^{\mathrm{T}}A^{-1}\mu_r - A^{-1}\tilde{B}\tilde{\Delta}^{-1}\tilde{\mu}_c, \\ &= A^{-1}\mu_r - A^{-1}\tilde{B}\tilde{s}_v. \end{aligned}$$

After some simplification, we obtain

$$\begin{aligned} \tilde{\Delta} &= \mathbf{diag}(\tilde{T}^{\mathrm{T}}\mathbf{1}_n) - \tilde{T}^{\mathrm{T}}\mathbf{diag}((T\mathbf{1}_m)^-)\tilde{T} \\ \tilde{s}_v &= \tilde{\Delta}^{-1}\tilde{\mu}_c - \tilde{\Delta}^{-1}\tilde{T}^{\mathrm{T}}((T\mathbf{1}_m)^- \odot \mu_r) \\ s_u &= (T\mathbf{1}_m)^- \odot \mu_r - (T\mathbf{1}_m)^- \odot (\tilde{T}\tilde{s}_v). \end{aligned}$$

Next, partition F_M as $F_M = \begin{pmatrix} G_M \\ \tilde{H}_M \end{pmatrix}$, where $G_M \in \mathbb{R}^{n\times(nm)}$ and $\tilde{H}_M \in \mathbb{R}^{(m-1)\times(nm)}$. By definition,

$$G_M = \begin{pmatrix} \frac{\partial G_1}{\partial \mathbf{vec}(M)^{\mathrm{T}}} \\ \vdots \\ \frac{\partial G_n}{\partial \mathbf{vec}(M)^{\mathrm{T}}} \end{pmatrix}, \quad G_i = -\sum_{j=1}^m T_{ij} = -\sum_{j=1}^m e^{\lambda(\alpha_i + \beta_j - M_{ij})},$$

so

$$\frac{\partial G_i}{\partial M_{kl}} = \begin{cases} 0, & i \neq k \\ \lambda T_{kl}, & i = k \end{cases}.$$

This indicates that $G_M = \lambda\left(\mathbf{diag}(T_1), \ldots, \mathbf{diag}(T_m)\right)$, where $T_1, \ldots, T_m$ are the column vectors of T. Similarly, for $H_j = -\sum_{i=1}^n T_{ij}$,

$$\tilde{H}_M = \begin{pmatrix} \frac{\partial H_1}{\partial \mathbf{vec}(M)^{\mathrm{T}}} \\ \vdots \\ \frac{\partial H_{m-1}}{\partial \mathbf{vec}(M)^{\mathrm{T}}} \end{pmatrix}, \quad \frac{\partial H_j}{\partial M_{kl}} = \begin{cases} 0, & j \neq l \\ \lambda T_{kl}, & j = l \end{cases},$$

implying that

$$\tilde{H}_M = \lambda \begin{pmatrix} T_1^{\mathrm{T}} & & & \\ & \ddots & & \\ & & T_{m-1}^{\mathrm{T}} & \mathbf{0}_n^{\mathrm{T}} \end{pmatrix}.$$

As a result,

$$
\begin{aligned}
&\mu_r^{\mathrm{T}} \frac{\partial \alpha^*}{\partial \mathbf{vec}(M)^{\mathrm{T}}} + \tilde{\mu}_c^{\mathrm{T}} \frac{\partial \tilde{\beta}^*}{\partial \mathbf{vec}(M)^{\mathrm{T}}} \\
&= (\mu_r^{\mathrm{T}}, \tilde{\mu}_c^{\mathrm{T}}) \frac{\partial w^*}{\partial \mathbf{vec}(M)^{\mathrm{T}}} = -(\mu_r^{\mathrm{T}}, \tilde{\mu}_c^{\mathrm{T}}) F_w^{-1} F_M \\
&= \left[-\lambda F_w^{-1} \begin{pmatrix} \mu_r \\ \tilde{\mu}_c \end{pmatrix} \right]^{\mathrm{T}} \begin{pmatrix} \lambda^{-1} G_M \\ \lambda^{-1} \tilde{H}_M \end{pmatrix} = (s_u^{\mathrm{T}}, \tilde{s}_v^{\mathrm{T}}) \begin{pmatrix} \lambda^{-1} G_M \\ \lambda^{-1} \tilde{H}_M \end{pmatrix} \\
&= \left((s_u \odot T_1)^{\mathrm{T}}, \ldots, (s_u \odot T_m)^{\mathrm{T}} \right) + \left(\tilde{s}_{v,1} T_1^{\mathrm{T}}, \ldots, \tilde{s}_{v,m-1} T_{m-1}^{\mathrm{T}}, \mathbf{0}_n^{\mathrm{T}} \right) \\
&= \left[\mathbf{vec}\left(\mathbf{diag}(s_u) T + T \mathbf{diag}(s_v) \right) \right]^{\mathrm{T}}. \qquad (14)
\end{aligned}
$$

Finally, substitute (14) into (11), and we have

$$
\frac{\partial S_\lambda(M, a, b)}{\partial \mathbf{vec}(M)^{\mathrm{T}}} = \lambda \left[\mathbf{vec}\left(\mathbf{diag}(s_u) T^* + T^* \mathbf{diag}(s_v) \right) - \mathbf{vec}(M \odot T) \right]^{\mathrm{T}} + \mathbf{vec}(T)^{\mathrm{T}}
$$

Transforming back to the matrix form, and we obtain

$$
\frac{\partial S_\lambda(M, a, b)}{\partial M} = \lambda (s_u \oplus s_v - M) \odot T + T.
$$

Replacing T with T^* and noting that $a = T^* \mathbf{1}_m$, $\tilde{b} = \tilde{T}^{*\mathrm{T}} \mathbf{1}_n$, we get the stated result.

B.3 Proof of Theorem 2

First of all, it is easy to see that $f(\gamma)$ is strictly convex, so the optimal value $\gamma^* = (\alpha^*, \tilde{\beta}^*)$ is unqiue.

Let $\beta^* = (\tilde{\beta}^*, 0)$, and then the optimality conditions for (α^*, β^*) are $T^* \mathbf{1}_m = a$ and $T^{*\mathrm{T}} \mathbf{1}_n = b$, where $T^* = \mathrm{e}_\lambda[\alpha^* \oplus \beta^* - M]$. Since $T_{ij}^* = \exp\{\lambda(\alpha_i^* + \beta_j^* - M_{ij})\} \geq 0$ and $a_i = \sum_{j=1}^m T_{ij}^*$, $b_j = \sum_{i=1}^n T_{ij}^*$, we have $T_{ij}^* \leq \min\{a_i, b_j\}$ for all i and j, implying that

$$
\alpha_i^* + \beta_j^* \leq U_{ij} \coloneqq M_{ij} + \lambda^{-1} \min\{\log(a_i), \log(b_j)\}.
$$

Since $\beta_m^* = 0$ by design, we have $\alpha_i^* \leq U_{\alpha_i} < +\infty, i = 1, \ldots, n$, where $U_{\alpha_i} = U_{im} = M_{im} + \lambda^{-1} \min\{\log(a_i), \log(b_m)\}$. This indicates that α_i^* is upper bounded.

Next, let $I = \arg\max_i T_{im}^*$. Since $T^{*\mathrm{T}} \mathbf{1}_n = b$ implies that $b_m = \sum_{i=1}^n T_{im}^* \leq n T_{Im}^*$, we have

$$
T_{Im}^* = \exp\{\lambda(\alpha_I^* - M_{Im})\} \geq b_m / n,
$$

and hence $\alpha_I^* \geq M_{Im} + \lambda^{-1} \log(b_m/n)$. Again, since $\alpha_i^* + \beta_j^* \leq U_{ij}$ for all i and j, it holds that

$$
\begin{aligned}
\beta_j^* \leq U_{Ij} - \alpha_I^* &\leq U_{Ij} - M_{Im} - \lambda^{-1} \log(b_m/n) \\
&= M_{Ij} + \lambda^{-1} \min\{\log(a_I), \log(b_j)\} - M_{Im} - \lambda^{-1} \log(b_m/n) \\
&= M_{Ij} - M_{Im} + \lambda^{-1} \min\{\log(n a_I / b_m), \log(n b_j / b_m)\} \\
&\coloneqq U_{\beta_j} < +\infty, \quad j = 1, \ldots, m. \qquad (15)
\end{aligned}
$$

On the other hand, $T^{*\mathrm{T}} \mathbf{1}_n = b$ implies that $b_j = \sum_{i=1}^n T_{ij}^* = \exp\{\lambda \beta_j^*\} \cdot \sum_{i=1}^n \exp\{\lambda(\alpha_i^* - M_{ij})\}$ for any j, so

$$
\log b_j = \lambda \beta_j^* + \log \left[\sum_{i=1}^n e^{\lambda(\alpha_i^* - M_{ij})} \right] \leq \lambda \beta_j^* + \log \left[\sum_{i=1}^n e^{\lambda(U_{\alpha_i} - M_{ij})} \right].
$$

It is well-known that

$$
\max\{x_1, \ldots, x_n\} \leq \log \left(\sum_{i=1}^n e^{x_i} \right) \leq \max\{x_1, \ldots, x_n\} + \log n
$$

for any $x_1, \ldots, x_n \in \mathbb{R}$, so

$$\begin{aligned}\beta_j^* &\geq \lambda^{-1} \log b_j - \lambda^{-1} \log \left[\sum_{i=1}^n e^{\lambda(U_{\alpha_i} - M_{ij})} \right] \\ &\geq \lambda^{-1} \log b_j - \max_i (U_{\alpha_i} - M_{ij}) - \lambda^{-1} \log n \\ &\geq \lambda^{-1} \log(b_j/n) - \max_i (U_{\alpha_i} - M_{ij}) \coloneqq L_{\beta_j} > -\infty, \quad j = 1, \ldots, m. \end{aligned} \tag{16}$$

Then (15) and (16) together show that $|\beta_j^*| < \infty$.

Similarly, $T^* \mathbf{1}_m = a$, so

$$\begin{aligned}\log a_i &\leq \lambda \alpha_i^* + \log \left[\sum_{j=1}^m e^{\lambda(U_{\beta_j} - M_{ij})} \right], \\ \alpha_i^* &\geq \lambda^{-1} \log(a_i/m) - \max_j (U_{\beta_j} - M_{ij}) \coloneqq L_{\alpha_j} > -\infty, \quad i = 1, \ldots, n.\end{aligned}$$

The results above verify that $|\alpha_i^*| < \infty$ and $|\beta_j^*| < \infty$, and hence $\|\gamma^*\| < \infty$. Finally, plugging in γ^* to the objective function, and we immediately get $f^* > -\infty$.

B.4 Proof of Theorem 3

Claims (a) and (b) are direct consequences of the convergence property of the L-BFGS algorithm (Theorem 7.1, Liu & Nocedal, 1989), and we only need to verify its three assumptions. First, f is twice continuously differentiable, so Assumption 7.1(1) of Liu & Nocedal (1989) is verified. Second, f is a closed convex function, and we define the level set of f as $L_c = \{\gamma \in \mathbb{R}^{n+m-1} : f(\gamma) \leq c\}$. Theorem 2 has shown that $f^* > -\infty$, and when $c = f^*$, obviously $L_c = \{\gamma^*\}$ is non-empty and bounded. Then Corollary 8.7.1 of Rockafellar (1970) shows that L_c is bounded for every c. In particular, for a fixed initial value $\gamma^{(0)}$, define $L = \{\gamma : f(\gamma) \leq f(\gamma^{(0)})\}$, and then L is a bounded, closed, and convex set, which verifies Assumption 7.1(2) of Liu & Nocedal (1989). Third, since f is twice continuously differentiable and strictly convex, the second derivative of f, denoted by $H(\gamma)$, is positive definite. As L is bounded and closed, there must exist constants $M_1, M_2 > 0$ such that

$$M_1 \|x\|^2 \leq x^{\mathrm{T}} H(\gamma) x \leq M_2 \|x\|^2$$

for all $x \in \mathbb{R}^{n+m-1}$ and $\gamma \in L$. This verifies Assumption 7.1(3) of Liu & Nocedal (1989).

For (c), we follow the analysis in Nocedal et al. (2002). Denote by $g(\gamma)$ the gradient of $f(\gamma)$, and then $g(\gamma^*) = \mathbf{0}$, and by Taylor's theorem we have

$$f(\gamma) - f^* = \frac{1}{2} (\gamma - \gamma^*)^{\mathrm{T}} H_1 (\gamma - \gamma^*), \tag{17}$$

where $H_1 = H(\xi)$ for some ξ in the line segment connecting γ and γ^*. Also,

$$g(\gamma) - g(\gamma^*) = g(\gamma) = H_2(\gamma - \gamma^*), \quad H_2 = \int_0^1 H(\gamma + t(\gamma^* - \gamma)) \mathrm{d}t. \tag{18}$$

Combining (17) and (18), we get

$$\|g(\gamma)\|^2 = (\gamma - \gamma^*)^{\mathrm{T}} H_2^2 (\gamma - \gamma^*) = \frac{2(\gamma - \gamma^*)^{\mathrm{T}} H_2^2 (\gamma - \gamma^*)}{(\gamma - \gamma^*)^{\mathrm{T}} H_1 (\gamma - \gamma^*)} \cdot (f(\gamma) - f^*)$$

for any $\gamma \in L$. Let $x = H_1^{1/2}(\gamma - \gamma^*)$, and then

$$\|g(\gamma)\|^2 = \frac{2x^{\mathrm{T}} H_1^{-1/2} H_2^2 H_1^{-1/2} x}{\|x\|^2} \cdot (f(\gamma) - f^*).$$

Denote by $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ the largest and smallest eigenvalues of a positive definite matrix A, respectively. Then it is easy to find that

$$\sigma_{\max}(H_1^{-1}) = [\sigma_{\min}(H_1)]^{-1} \leq M_1^{-1}, \quad \sigma_{\max}(H_2^2) = [\sigma_{\max}(H_2)]^2 \leq M_2^2.$$

By Lemma 4, we have $x^{\mathrm{T}} H_1^{-1/2} H_2^2 H_1^{-1/2} x \leq M_1^{-1} M_2^2 \|x\|^2$ for any $x \in \mathbb{R}^{n+m-1}$, and hence $\|g(\gamma)\|^2 \leq 2M_1^{-1} M_2^2 (f(\gamma) - f^*)$ for all $\gamma \in L$. Since $f^{(k)} \leq f^{(0)}$ due to claim (a), we find that $\gamma^{(k)} \in L$ for all $k > 0$. Therefore,

$$\|g(\gamma^{(k)})\|^2 \leq 2M_1^{-1} M_2^2 (f^{(k)} - f^*) \leq 2M_1^{-1} M_2^2 r^k (f^{(0)} - f^*),$$

and claim (c) is proved.

Claims (d) and (e) can be verified as follows. For any $\gamma \in L$, let $T = \mathrm{e}_\lambda[\alpha \oplus \beta - M]$, and then

$$\|T\mathbf{1}_m - a\|_\infty \leq \|T\mathbf{1}_m - a\| = \|\nabla_\alpha f(\gamma)\| \leq \|\nabla_\gamma f(\gamma)\|,$$

indicating that

$$\left| \sum_{j=1}^m T_{ij} - a_i \right| \leq \|\nabla_\gamma f(\gamma)\|, \quad i = 1, \ldots, n. \tag{19}$$

Then $0 < T_{ij} \leq \sum_{j=1}^m T_{ij} \leq a_i + \|\nabla_\gamma f(\gamma)\|$. Similarly, we can obtain $0 < T_{ij} \leq \sum_{i=1}^n T_{ij} \leq b_j + \|\nabla_\gamma f(\gamma)\|$. The gradient $\|\nabla_\gamma f(\gamma)\|$ can be bounded using claim (c), so (d) is also proved. On the other hand, (19) shows that

$$a_i - \|\nabla_\gamma f(\gamma)\| \leq \sum_{j=1}^m T_{ij} \leq m \cdot \max_j T_{ij},$$

and similarly we have

$$b_j - \|\nabla_\gamma f(\gamma)\| \leq \sum_{i=1}^n T_{ij} \leq n \cdot \max_i T_{ij}.$$

Replacing $\|\nabla_\gamma f(\gamma)\|$ by its upper bound, and claim (e) is verified.

B.5 Proof of Theorem 3

For matrix $A_{n\times m} = (a_{ij})$, define $\|A\|_\infty = \max_{i,j} |a_{ij}|$, and the notation $A \geq 0$ means $a_{ij} \geq 0$ for all i and j. First, it is easy to show that $\|A \odot B\|_F \leq \|A\|_\infty \|B\|_F$, since

$$\|A \odot B\|_F = \sqrt{\sum_{i,j} (a_{ij} b_{ij})^2} \leq \sqrt{\sum_{i,j} \|A\|_\infty^2 b_{ij}^2} = \|A\|_\infty \|B\|_F.$$

Next, we show that if $B_{n\times m} \geq 0$, $C_{p\times n} \geq 0$, and $v \geq 0$, where v is a vector, then $\|C(A \odot B)v\| \leq \|A\|_\infty \|CBv\|$.

Proof. Let u_i be the i-th element of $(A \odot B)v$, and then

$$u_i = \sum_{j=1}^m a_{ij} b_{ij} v_j, \quad |u_i| \leq \|A\|_\infty \sum_{j=1}^m b_{ij} v_j.$$

Consequently,

$$\begin{aligned} \|C(A \odot B)v\| &= \sqrt{\sum_{k=1}^p \left| \sum_{i=1}^n c_{ki} u_i \right|^2} \leq \sqrt{\sum_{k=1}^p \left(\sum_{i=1}^n c_{ki} |u_i| \right)^2} \leq \|A\|_\infty \sqrt{\sum_{k=1}^p \left[\sum_{i=1}^n c_{ki} \left(\sum_{j=1}^m b_{ij} v_j \right) \right]^2} \\ &= \|A\|_\infty \|CBv\|. \end{aligned}$$

□

Similarly, if $A \geq 0$ and $v \geq 0$, then

$$\|A(u \odot v)\| = \sqrt{\sum_{i=1}^n \left| \sum_{j=1}^m a_{ij} u_j v_j \right|^2} \leq \|u\|_\infty \sqrt{\sum_{i=1}^n \left(\sum_{j=1}^m a_{ij} v_j \right)^2} = \|u\|_\infty \|Av\|.$$

Moreover, for matrices $B_{n\times m} \geq 0$, and $C_{m\times p} \geq 0$, let C_j be the j-th column of C, and then

$$\|(A \odot B)C\|_F = \sqrt{\sum_{j=1}^{p} \|(A \odot B)C_i\|^2} \leq \|A\|_\infty \sqrt{\sum_{j=1}^{p} \|BC_i\|^2} = \|A\|_\infty \|BC\|_F.$$

Let $\alpha = \alpha^* + f$ and $\beta = \beta^* + g$ for some perturbation vectors f and g, and define $T = \mathrm{e}_\lambda[\alpha \oplus \beta - M]$. Then it is easy to find that

$$T - T^* = \mathrm{e}_\lambda[\alpha^* \oplus \beta^* - M + (f \oplus g)] - T^* = \mathrm{e}_\lambda[f \oplus g] \odot T^* - T^*.$$

Let $E_T = \mathrm{e}_\lambda[f \oplus g] - \mathbf{1}_n \mathbf{1}_m'$, so $T - T^* = E_T \odot T^*$. Since $|e^x - 1| < 2|x|$ for $|x| < 1$, we have

$$|(E_T)_{ij}| = |e^{\lambda(f_i + g_j)} - 1| < 2\lambda|f_i + g_j|$$

as long as $\lambda|f_i + g_j| < 1$. This can be achieved by assuming $\varepsilon := 2\lambda(\|f\|_\infty + \|g\|_\infty) < 1$, since in this case $\lambda|f_i + g_j| \leq \lambda(\|f\|_\infty + \|g\|_\infty) < 1/2$. Then clearly $\|E_T\|_\infty < \varepsilon$.

Consider $\hat{s}_u = s_u + \delta_u$ and $\hat{s}_v = s_v + \delta_v$ for some perturbation vectors δ_u and δ_v, and let

$$\begin{aligned}
\widehat{\nabla_M S} &= T + \lambda(\hat{s}_u \oplus \hat{s}_v - M) \odot T \\
&= T + \lambda(s_u \oplus s_v - M) \odot T + \lambda(\delta_u \oplus \delta_v) \odot T \\
&= \nabla_M S + E_T \odot \nabla_M S + \lambda(\delta_u \oplus \delta_v) \odot (T^* + E_T \odot T^*).
\end{aligned}$$

Then we have

$$\begin{aligned}
\|\widehat{\nabla_M S} - \nabla_M S\|_F &\leq \|E_T \odot \nabla_M S\|_F + \|\lambda(\delta_u \oplus \delta_v) \odot (T^* + E_T \odot T^*)\|_F \\
&\leq \|E_T\|_\infty \|\nabla_M S\|_F + \lambda\|\delta_u \oplus \delta_v\|_\infty \|T^* + E_T \odot T^*\|_F \\
&< \varepsilon\|\nabla_M S\|_F + \lambda\|\delta_u \oplus \delta_v\|_\infty \|\mathbf{1}_n \mathbf{1}_m' + E_T\|_\infty \|T^*\|_F \\
&\leq \varepsilon\|\nabla_M S\|_F + \lambda\|\delta_u \oplus \delta_v\|_\infty (1 + \|E_T\|_\infty)\|T^*\|_F \\
&< \varepsilon\|\nabla_M S\|_F + \lambda(1 + \varepsilon)\|\delta_u \oplus \delta_v\|_\infty \|T^*\|_F \\
&< \varepsilon\|\nabla_M S\|_F + 2\lambda\|T^*\|_F(\|\delta_u\|_\infty + \|\delta_v\|_\infty).
\end{aligned}$$

Therefore, we just need to show proper bounds for $\|\delta_u\|_\infty$ and $\|\delta_v\|_\infty$.

Consider $\hat{\mu}_r = (M \odot T)\mathbf{1}_m$ and $\hat{\mu}_c = (M \odot T)^{\mathrm{T}}\mathbf{1}_n$, and then

$$\begin{aligned}
\hat{\mu}_r &= (M \odot (T^* + E_T \odot T^*))\mathbf{1}_m = \mu_r + (E_T \odot M \odot T^*)\mathbf{1}_m := \mu_r + \delta_r, \\
\hat{\mu}_c &= (M \odot (T^* + E_T \odot T^*))^{\mathrm{T}}\mathbf{1}_n = \mu_c + (E_T \odot M \odot T^*)^{\mathrm{T}}\mathbf{1}_m := \mu_c + \delta_c.
\end{aligned}$$

It can be easily verified that

$$\begin{aligned}
\|a^- \odot \delta_r\| &= \|\mathbf{diag}(a^-)(E_T \odot M \odot T^*)\mathbf{1}_m\| \leq \|E_T\|_\infty \|\mathbf{diag}(a^-)(M \odot T^*)\mathbf{1}_m\| < \varepsilon\|\mu_r\|, \\
\|T^{*\mathrm{T}}(a^- \odot \delta_r)\| &= \|T^{*\mathrm{T}}\mathbf{diag}(a^-)(E_T \odot M \odot T^*)\mathbf{1}_m\| \leq \|E_T\|_\infty \|T^{*\mathrm{T}}\mathbf{diag}(a^-)(M \odot T^*)\mathbf{1}_m\| \\
&< \varepsilon\|T^{*\mathrm{T}}(a^- \odot \mu_r)\|, \\
\|\delta_c\| &= \|(E_T \odot M \odot T^*)^{\mathrm{T}}\mathbf{1}_m\| \leq \|E_T\|_\infty \|(M \odot T^*)^{\mathrm{T}}\mathbf{1}_m\| < \varepsilon\|\mu_c\|.
\end{aligned}$$

Define $b_v = \mu_c - T^{*\mathrm{T}}(a^- \odot \mu_r)$ and $\hat{b}_v = \hat{\mu}_c - T^{\mathrm{T}}(a^- \odot \hat{\mu}_r)$, and we have

$$\begin{aligned}
\hat{b}_v &= \mu_c + \delta_c - (T^* + E_T \odot T^*)^{\mathrm{T}}(a^- \odot (\mu_r + \delta_r)) \\
&= \mu_c + \delta_c - T^{*\mathrm{T}}(a^- \odot (\mu_r + \delta_r)) - (E_T \odot T^*)^{\mathrm{T}}(a^- \odot (\mu_r + \delta_r)) \\
&= \mu_c - T^{*\mathrm{T}}(a^- \odot \mu_r) + \delta_c - T^{*\mathrm{T}}(a^- \odot \delta_r) - (E_T \odot T^*)^{\mathrm{T}}(a^- \odot (\mu_r + \delta_r))
\end{aligned}$$

As a result,

$$\begin{aligned}
\|\hat{b}_v - b_v\| &= \|\delta_c - T^{*\mathrm{T}}(a^- \odot \delta_r) - (E_T \odot T^*)^{\mathrm{T}}(a^- \odot (\mu_r + \delta_r))\| \\
&\leq \|\delta_c\| + \|T^{*\mathrm{T}}(a^- \odot \delta_r)\| + \|E_T\|_\infty \|T^{*\mathrm{T}}(a^- \odot (\mu_r + \delta_r))\| \\
&< \varepsilon\|\mu_c\| + \varepsilon\|T^{*\mathrm{T}}(a^- \odot \mu_r)\| + \varepsilon\|T^{*\mathrm{T}}(a^- \odot \mu_r)\| + \varepsilon\|T^{*\mathrm{T}}(a^- \odot \delta_r)\| \\
&< \varepsilon\|\mu_c\| + \varepsilon\|T^{*\mathrm{T}}(a^- \odot \mu_r)\| + \varepsilon\|T^{*\mathrm{T}}(a^- \odot \mu_r)\| + \varepsilon^2\|T^{*\mathrm{T}}(a^- \odot \mu_r)\| \\
&< \varepsilon\|\mu_c\| + 3\varepsilon\|T^{*\mathrm{T}}(a^- \odot \mu_r)\|.
\end{aligned}$$

On the other hand, let $\hat{t}_{ij}$ be the (i,j) element of the matrix $\tilde{T}^{\mathrm{T}}\mathbf{diag}(a^-)\tilde{T}$, and t_{ij} be the (i,j) element of $\tilde{T}^{*\mathrm{T}}\mathbf{diag}(a^-)\tilde{T}^*$. Then

$$\begin{aligned}
\hat{t}_{ij} &= \sum_{k=1}^{n} T_{ki}T_{kj}/a_k = \sum_{k=1}^{n}[1+(E_T)_{ki}][1+(E_T)_{kj}]T^*_{ki}T^*_{kj}/a_k, \\
|\hat{t}_{ij} - t_{ij}| &= \left|\sum_{k=1}^{n}[(E_T)_{ki}+(E_T)_{kj}+(E_T)_{ki}(E_T)_{kj}]T^*_{ki}T^*_{kj}/a_k\right| \\
&\le (2\varepsilon+\varepsilon^2)\sum_{k=1}^{n}\left|T^*_{ki}T^*_{kj}/a_k\right| < 3\varepsilon t_{ij}.
\end{aligned}$$

This implies that

$$\|\hat{D}-D\|_F = \|\tilde{T}^{*\mathrm{T}}\mathbf{diag}(a^-)\tilde{T}^* - \tilde{T}^{\mathrm{T}}\mathbf{diag}(a^-)\tilde{T}\|_F = \sqrt{\sum_{i,j}|\hat{t}_{ij}-t_{ij}|^2} \le 3\varepsilon\|D\|_F.$$

Then by Theorem 7.2 of Higham (2002), if $3\varepsilon\sigma\|D\|_F < 1$, we have

$$\frac{\|\delta_v\|}{\|s_v\|} \le \frac{\varepsilon\sigma}{1-3\varepsilon\sigma\|D\|_F}\left(\frac{\|\mu_c\|+3\|T^{*\mathrm{T}}(a^-\odot\mu_r)\|}{\|s_v\|}+3\|D\|_F\right).$$

where $\sigma = \|D^{-1}\|_{\mathrm{op}} = 1/\lambda_{\min}(D)$. Assume that $\varepsilon < \min\{1, 1/(6\sigma\|D\|_F)\}$, then with slight simplification, we have $\|\delta_v\|_\infty \le \|\delta_v\| \le C_v\varepsilon$, where

$$C_v = 2\sigma(\|\mu_c\|+3\|T^{*\mathrm{T}}(a^-\odot\mu_r)\|+3\|D\|_F\|s_v\|).$$

On the other hand,

$$\begin{aligned}
\|\delta_u\|_\infty \le \|\delta_u\| &\le \|a^-\odot\delta_r\| + \|a^-\odot(T\hat{s}_v - T^*s_v)\| \\
&\le \varepsilon\|\mu_r\| + \|a^-\odot(T^*\hat{s}_v + (E_T\odot T^*)\hat{s}_v - T^*s_v)\| \\
&\le \varepsilon\|\mu_r\| + \|a^-\odot(T^*\delta_v)\| + \|a^-\odot((E_T\odot T^*)\hat{s}_v)\| \\
&\le \varepsilon\|\mu_r\| + \|a^-\odot(T^*\delta_v)\| + \|\mathbf{diag}(a^-)(E_T\odot T^*)\hat{s}_v\| \\
&\le \varepsilon\|\mu_r\| + \|a^-\odot(T^*\delta_v)\| + \varepsilon\|\mathbf{diag}(a^-)T^*\hat{s}_v\| \\
&= \varepsilon\|\mu_r\| + \|a^-\odot(T^*\delta_v)\| + \varepsilon\|\mathbf{diag}(a^-)T^*(s_v+\delta_v)\| \\
&\le \varepsilon\|\mu_r\| + (1+\varepsilon)\|a^-\odot(T^*\delta_v)\| + \varepsilon\|\mathbf{diag}(a^-)T^*s_v\| \\
&\le \varepsilon\|\mu_r\| + (1+\varepsilon)\|\mathbf{diag}(a^-)T^*\|_F\|\delta_v\| + \varepsilon\|a^-\odot(T^*s_v)\| \\
&\le \varepsilon(\|\mu_r\| + 2C_v\|\mathbf{diag}(a^-)T^*\|_F + \|a^-\odot(T^*s_v)\|).
\end{aligned}$$

Combining the results together, we get

$$\|\widehat{\nabla_M S} - \nabla_M S\|_F \le \varepsilon\left[\|\nabla_M S\|_F + 2\lambda\|T^*\|_F(C_v+C_u)\right],$$

where $C_u = \|\mu_r\| + 2C_v\|\mathbf{diag}(a^-)T^*\|_F + \|a^-\odot(T^*s_v)\|$, and $\varepsilon = 2\lambda(\|f\|_\infty + \|g\|_\infty) \le 4\lambda\|\gamma-\gamma^*\| \le 4\lambda\sqrt{C_1\varepsilon^{(k)}}$.