

wig: An R Package for Wasserstein Index Generation (WIG) Model

Fangzhou Xie

Department of Economics, Rutgers University

August 18, 2024

Abstract

In this paper, I review the basics of Computation Optimal Transport, especially the entropic regularized OT problem, and the Sinkhorn algorithm

TODO:

1. edit all “Sinkhorn”

Contents

1	Introduction	3
2	Monge-Kantorovich Problem and Its Entropic Regularization	5
3	Sinkhorn Algorithms	7
3.1	Vanilla Sinkhorn Algorithm	7
3.2	Parallel Sinkhorn Algorithm	8
3.3	Log-stabilized Sinkhorn	10
4	Sinkhorn Algorithms with Gradients	13
4.1	Vanilla Sinkhorn	13
4.2	Parallel Sinkhorn	15
4.3	Log-Stabilized Sinkhorn	17
5	Wasserstein Barycenter	20
5.1	Parallel Algorithm	21
5.2	Log-stabilized Algorithm	22
6	Wasserstein Barycenter with Gradients	26
7	Wasserstein Dictionary Learning	26
7.1	Numeric Optimization Methods	26
7.2	Wasserstein Dictionary Learning	28
8	Wasserstein Index Generation Model	31
9	<i>wig</i> Package in R for a WIG Implementation	31
	References	32
	Appendix	35
	Mathematical Notations	35

1 Introduction

intro, literature review, etc

The remainder of this paper will be organized as follows. Section 2 briefly reviews the Monge-Kantorovich problem, its entropic regularized Sinkhorn problem and solution. Section 3 introduces the Sinkhorn algorithm and its variants to solve the entropic regularized problem. Section 4 derives the gradients and Jacobians of the family of Sinkhorn algorithms. Section 5 discusses the Wasserstein Barycenter and Wasserstein Dictionary Learning problem. Section 6 shows the gradient of Sinkhorn-like algorithms to solve the Wasserstein Barycenter problem. Section 7 considers the Wasserstein Dictionary Learning problem, which is built on top of the Barycenter problem and “learns” the latent topics. Section 8 finally presents the Wasserstein Index Generation model for automatically time-series index generation, based on Wasserstein Dictionary Learning. Section 9 provides some code snippets of the *wig* package to demonstrate how to carry out the computations in practice. Section 9 lists all the mathematical notations used throughout this note.

This note can be seen as a review of some selected topics on Computational Optimal Transport and a relatively complete reference to understand the Wasserstein Index Generation model. Moreover, this is a companion paper for the R package *wig*¹ to illustrate the usage of the package and how to solve the Optimal Transport problems as discussed in Sections 2 to 8.

Figure 1 provides a diagram of the structure of the paper. For readers interested in entropic regularized OT problems, they can refer to Sections 2 and 3 with optional Section 4; for readers interested in Wasserstein Barycenter problems, they can refer to Sections 2, 3 and 5 with optional Section 6; for readers interested in Wasserstein Dictionary Learning, they can refer to Sections 2, 3 and 5 to 7; for readers interested in Wasserstein Index Generation model, they can refer to everything from the WDL sections, plus Section 8, namely Sections 2, 3 and 5 to 8; Section 9 is required if one needs to use the *wig* package to carry out any of the aforementioned computation.

¹TODO: CRAN url here:

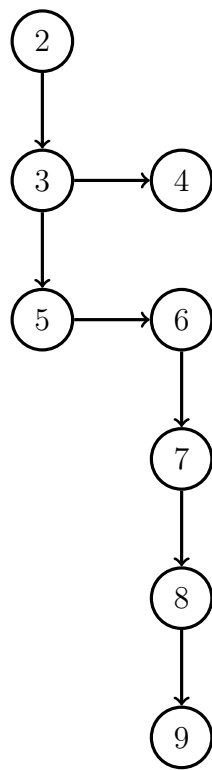


Figure 1: Diagram of the structure of the paper.

2 Monge-Kantorovich Problem and Its Entropic Regularization

The Monge-Kantorovich problem² states that: given two probability vectors³ $\mathbf{a} \in \Sigma_M$ and $\mathbf{b} \in \Sigma_N$, how to find a *coupling matrix* $\mathbf{P} \in \mathbb{R}_+^{M \times N}$ where each \mathbf{P}_{ij} describes the flow of masses from bin i to bin j , such that the total cost of moving masses are optimal (minimal). The cost of moving one unit of mass from bin i to bin j is \mathbf{C}_{ij} and $\mathbf{C} \in \mathbb{R}_+^{M \times N}$.

Problem 1 (Monge-Kantorovich). *Let $\mathbf{a} \in \Sigma_M$ and $\mathbf{b} \in \Sigma_N$. The set of all coupling matrices is:*

$$\mathbf{U}(\mathbf{a}, \mathbf{b}) \equiv \{ \mathbf{P} \in \mathbb{R}_+^{M \times N} : \mathbf{P} \mathbf{1}_M = \mathbf{a} \text{ and } \mathbf{P}^\top \mathbf{1}_N = \mathbf{b} \}. \quad (1)$$

Given a cost matrix $\mathbf{C} \in \mathbb{R}^{M \times N}$, the Kantorovich optimal transport problem tries to find the solution of the following

$$L_{\mathbf{C}}(\mathbf{a}, \mathbf{b}) \equiv \min_{\mathbf{P} \in \mathbf{U}(\mathbf{a}, \mathbf{b})} \langle \mathbf{C}, \mathbf{P} \rangle = \sum_{i,j} \mathbf{C}_{ij} \mathbf{P}_{ij}. \quad (2)$$

To solve this problem in practice, one needs to resort to linear programming⁴, which can be very challenging when the problem becomes large enough. Instead of finding the exact solution to the above problem, one can actually add an entropy regularization term to make this problem convex and hence greatly ease the computation. This was first proposed in the seminal work of Cuturi (2013) which leverages Sinkhorn-Knopp double scaling algorithm (Sinkhorn, 1964; Sinkhorn and Knopp, 1967; Knight, 2008).⁵

Problem 2 (Cuturi, 2013; Peyré and Cuturi, 2019, Entropic Regularized OT Problem). *Given the coupling matrix $\mathbf{P} \in \mathbb{R}_+^{M \times N}$, its discrete entropy is defined as*

$$\mathbf{H}(\mathbf{P}) \equiv -\langle \mathbf{P}, \log(\mathbf{P}) - 1 \rangle = -\sum_{i,j} \mathbf{P}_{ij} (\log(\mathbf{P}_{ij}) - 1).$$

²This is also called Kantorovich relaxation of the original Monge problem, as the problem was first proposed by Monge (1781) and then relaxed by Kantorovich (1942). For mathematical foundation of Optimal Transport theory, the classic references are Villani (2003) and Villani (2008); for its application in Economics, see Galichon (2016); for an introduction for applied mathematicians, see Santambrogio (2015); for the computational aspects (algorithms and their properties) and its applications in data science and machine learning, see Peyré and Cuturi (2019).

³Sometimes also called “histograms” in the Computational Optimal Transport community, for example in Peyré and Cuturi (2019).

⁴See Peyré and Cuturi (2019, Chapter 3) for a historical overview and related algorithms.

⁵Therefore, in Optimal Transport literature, people usually refer to the numeric algorithm to solve the entropic regularized Kantorovich problem as the “Sinkhorn algorithm”. Hence, its extensions are also named as “X-khorn algorithms”, for example “Greenkhorn algorithm” (Altschuler, Weed, and Rigollet, 2017) or “Screenkhorn” (Alaya et al., 2019). Because of its simplicity, it has been re-discovered multiple times in history and renamed accordingly, for example, it can be linked to Schrödinger bridge problem (Schrödinger, 1931) or the RAS method (Bacharach, 1970). See Knight (2008), Léonard (2013), and Modin (2024) for historical accounts.

With the cost matrix \mathbf{C} , the entropic regularized loss function is

$$L_{\mathbf{C}}^{\varepsilon}(\mathbf{a}, \mathbf{b}) \equiv \langle \mathbf{P}, \mathbf{C} \rangle - \varepsilon \mathbf{H}(\mathbf{P}), \quad (3)$$

and the entropic regularized OT problem is thus

$$\min_{\mathbf{P} \in \mathbf{U}(\mathbf{a}, \mathbf{b})} L_{\mathbf{C}}^{\varepsilon}(\mathbf{a}, \mathbf{b}) \equiv \min_{\mathbf{P} \in \mathbf{U}(\mathbf{a}, \mathbf{b})} \langle \mathbf{P}, \mathbf{C} \rangle - \varepsilon \mathbf{H}(\mathbf{P}). \quad (4)$$

The Sinkhorn problem has a unique optimal solution, since it is ε -strongly convex (Peyré and Cuturi, 2019).

Proposition 3 (Peyré and Cuturi, 2019, Proposition 4.3). *The solution to Equation (4) is unique and has the form*

$$\mathbf{P}_{ij} = \mathbf{u}_i \mathbf{K}_{ij} \mathbf{v}_j, \quad (5)$$

or in matrix notation

$$\mathbf{P} = \text{diag}(\mathbf{u}) \mathbf{K} \text{diag}(\mathbf{v}) \quad (6)$$

where $i \in \{1, \dots, M\}$ and $j \in \{1, \dots, N\}$ with two (unknown) scaling variables $\mathbf{u} \in \mathbb{R}^M$ and $\mathbf{v} \in \mathbb{R}^N$.

Proof. Let $\mathbf{f} \in \mathbb{R}^M$ and $\mathbf{g} \in \mathbb{R}^N$ be two dual variables for the two constraints $\mathbf{P} \mathbf{1}_N = \mathbf{a}$ and $\mathbf{P}^\top \mathbf{1}_M = \mathbf{b}$, respectively. The Lagrangian of Equation (4) becomes

$$\mathcal{L}(\mathbf{P}, \mathbf{f}, \mathbf{g}) = \langle \mathbf{P}, \mathbf{C} \rangle - \varepsilon \mathbf{H}(\mathbf{P}) - \langle \mathbf{f}, \mathbf{P} \mathbf{1}_m - \mathbf{a} \rangle - \langle \mathbf{g}, \mathbf{P}^\top \mathbf{1}_n - \mathbf{b} \rangle. \quad (7)$$

The first-order-condition gives us

$$\frac{\partial \mathcal{L}(\mathbf{P}, \mathbf{f}, \mathbf{g})}{\partial \mathbf{P}_{ij}} = \mathbf{C}_{ij} + \varepsilon \log(\mathbf{P}_{ij}) - \mathbf{f}_i - \mathbf{g}_j = 0$$

Therefore we can solve for the optimal coupling matrix \mathbf{P} as:

$$\log(\mathbf{P}_{ij}) = \frac{\mathbf{f}_i}{\varepsilon} \left(-\frac{\mathbf{C}_{ij}}{\varepsilon} \right) \frac{\mathbf{g}_j}{\varepsilon}$$

or

$$\mathbf{P}_{ij} = \exp\left(\frac{\mathbf{f}_i}{\varepsilon}\right) \exp\left(-\frac{\mathbf{C}_{ij}}{\varepsilon}\right) \exp\left(\frac{\mathbf{g}_j}{\varepsilon}\right) = \mathbf{u}_i \mathbf{C}_{ij} \mathbf{v}_j, \quad (8)$$

where $\mathbf{u}_i = \exp(\frac{\mathbf{f}_i}{\varepsilon})$, $\mathbf{v}_j = \exp(\frac{\mathbf{g}_j}{\varepsilon})$, and $\mathbf{K}_{ij} = \exp(-\frac{\mathbf{C}_{ij}}{\varepsilon})$, respectively. \square

Remark. Note that computation of \mathbf{P}_{ij} are effectively in the exponential domain, if we choose to find optimal solution for \mathbf{P} by updating \mathbf{f} and \mathbf{g} . This is indeed the case for the vanilla Sinkhorn Algorithm, and hence it has numeric instability issue which need to be overcome by the “log-stabilization” technique, which moves all computation in the log-domain (Section 3.3).

3 Sinkhorn Algorithms

In this section, I will discuss three different versions of Sinkhorn algorithms.

3.1 Vanilla Sinkhorn Algorithm

Given the optimal solution to the Sinkhorn problem in Equation (6), with the constraints in Equation (1), we have

$$\text{diag}(\mathbf{u}) \mathbf{K} \text{diag}(\mathbf{v}) \mathbf{1}_N = \mathbf{a} \quad \text{and} \quad \text{diag}(\mathbf{v}) \mathbf{K}^\top \text{diag}(\mathbf{u}) \mathbf{1}_M = \mathbf{b}. \quad (9)$$

Since we also have $\text{diag}(\mathbf{v}) \mathbf{1}_N = \mathbf{v}$ and $\text{diag}(\mathbf{u}) \mathbf{1}_M = \mathbf{u}$, we have that

$$\mathbf{u} \odot (\mathbf{K} \mathbf{v}) = \mathbf{a} \quad \text{and} \quad \mathbf{v} \odot (\mathbf{K}^\top \mathbf{u}) = \mathbf{b}. \quad (10)$$

Equivalently, we can also write the solution as

$$\mathbf{u} = \mathbf{a} \oslash (\mathbf{K} \mathbf{v}) \quad \text{and} \quad \mathbf{v} = \mathbf{b} \oslash (\mathbf{K}^\top \mathbf{u}). \quad (11)$$

In the above equations, \odot and \oslash refer to element-wise multiplication and division, respectively. Therefore, we can leverage Equation (11) to update \mathbf{u} and \mathbf{v} till convergence and conclude with optimal coupling \mathbf{P} by Equation (6).

Update 3.1 (Vanilla Sinkhorn Updates). *At any iteration $\ell = 1, \dots, L$ we can update the scaling variables \mathbf{u} and \mathbf{v} by the following equations:*

$$\mathbf{u}^{(\ell)} \equiv \mathbf{a} \oslash (\mathbf{K} \mathbf{v}^{(\ell-1)}) \quad \text{and} \quad \mathbf{v}^{(\ell)} \equiv \mathbf{b} \oslash (\mathbf{K}^\top \mathbf{u}^{(\ell)}). \quad (12)$$

Remark. *In Update 3.1, the choice of $\mathbf{v}^{(0)}$ is rather arbitrary, as long as it is not $\mathbf{0}$. The most common choice is to consider the vectors of ones, i.e. $\mathbf{1}_N$. Changing the initialization vector will result in different solutions to \mathbf{u} and \mathbf{v} , since they are only defined up to a multiplicative constant. But \mathbf{u} and \mathbf{v} will converge regardless of the initialization, and result in the same optimal coupling matrix \mathbf{P} upon convergence. See Peyré and Cuturi (2019, p.82) for discussion.*

After L iterations, the scaling variables $\mathbf{u}^{(L)}$ and $\mathbf{v}^{(L)}$ can be used to compute the optimal coupling matrix $\mathbf{P} = \text{diag} \mathbf{u}^{(L)} \mathbf{K} \text{diag} \mathbf{v}^{(L)}$.

From Update 3.1, we can arrive at the vanilla Sinkhorn algorithm (Algorithm 3.1).

Algorithm 3.1 Vanilla Sinkhorn**Input:** $\mathbf{a} \in \Sigma_M$, $\mathbf{b} \in \Sigma_N$, $\mathbf{C} \in \mathbb{R}^{M \times N}$, $\varepsilon > 0$.**Initialize:** $\mathbf{u} = \mathbf{1}_M$, $\mathbf{v} = \mathbf{1}_N$.

- 1: $\mathbf{K} = \exp(-\frac{\mathbf{C}}{\varepsilon})$
- 2: **while** Not Convergence **do**
- 3: $\mathbf{u} \leftarrow \mathbf{a} \oslash (\mathbf{K}\mathbf{v})$
- 4: $\mathbf{v} \leftarrow \mathbf{b} \oslash (\mathbf{K}^\top \mathbf{u})$
- 5: **end while**
- 6: $\mathbf{P} = \text{diag}(\mathbf{u}) \mathbf{K} \text{diag}(\mathbf{v})$

Output: Optimal Coupling \mathbf{P}

Remark (Termination/Convergence Condition). *Since we have Equation (10), we can use those equations to check if the scaling variables \mathbf{u} and \mathbf{v} at the current iteration could approximate \mathbf{a} and \mathbf{b} well. That is, for some small $\rho > 0$,*

$$\|\mathbf{u}^{(\ell)} \odot (\mathbf{K}\mathbf{v}^{(\ell)}) - \mathbf{a}\|_2 \leq \rho \quad \text{and} \quad \|\mathbf{v}^{(\ell)} \odot (\mathbf{K}^\top \mathbf{u}^{(\ell)}) - \mathbf{b}\|_2 \leq \rho$$

If the norms are smaller than some threshold ρ , we can then terminate the program.

3.2 Parallel Sinkhorn Algorithm

Note that in the Vanilla Sinkhorn Algorithm, the computation are carried out by vector-vector element-wise computation, i.e., \mathbf{u}, \mathbf{a} , and $\mathbf{K}\mathbf{v} \in \mathbb{R}^M$ and \mathbf{v}, \mathbf{b} , and $\mathbf{K}^\top \mathbf{u} \in \mathbb{R}^N$. That is, if we have a single source \mathbf{a} and a single target \mathbf{b} , we are computing the “one-to-one” transportation. However, sometimes we need to calculate one-to-many, many-to-one, or many-to-many transport and this can be easily computed in a parallelized fashion. All we need to do is to define the matrices \mathbf{A} and \mathbf{B} instead of the vectors \mathbf{a} and \mathbf{b} , and all that remains is to carry out the matrix computation instead of the vector ones.

Definition 4. *For some $S \in \mathbb{N}_+$:*

- *Suppose we have $\mathbf{a} \in \Sigma_M$ and $\mathbf{b}_1, \dots, \mathbf{b}_S \in \Sigma_N$. Then we can define*

$$\mathbf{A} = [\mathbf{a}, \dots, \mathbf{a}] \in \mathbb{R}^{M \times S} \quad \text{and} \quad \mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_S] \in \mathbb{R}^{N \times S}.$$

- *Suppose we have $\mathbf{a}_1, \dots, \mathbf{a}_S \in \Sigma_M$ and $\mathbf{b} \in \Sigma_N$. Then we can define*

$$\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_S] \in \mathbb{R}^{M \times S} \quad \text{and} \quad \mathbf{B} = [\mathbf{b}, \dots, \mathbf{b}] \in \mathbb{R}^{N \times S}.$$

- *Suppose we have $\mathbf{a}_1, \dots, \mathbf{a}_S \in \Sigma_M$ and $\mathbf{b}_1, \dots, \mathbf{b}_S \in \Sigma_N$. Then we can define*

$$\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_S] \in \mathbb{R}^{M \times S} \quad \text{and} \quad \mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_S] \in \mathbb{R}^{N \times S}.$$

Remark. In the many-to-many case, the number of columns in \mathbf{A} should be equal to that of \mathbf{B} ; whereas in one-to-many or many-to-one cases, we need to duplicate the single vector S times, as if we had a S -column matrix to begin with. In many modern linear algebra packages, this can be achieved automatically by “broadcasting,”⁶ without the need of actually doing the duplication of the column vectors. Or in terms of matrix computation, for example, $\mathbf{B} = [\mathbf{b}, \dots, \mathbf{b}] = \mathbb{1}_S^\top \otimes \mathbf{b}$.

Therefore, we can have the matrix version⁷ of Equation (11) with scaling variable $\mathbf{U} \in \mathbb{R}^{M \times S}$ and $\mathbf{V} \in \mathbb{R}^{N \times S}$,

$$\mathbf{U} = \mathbf{A} \oslash (\mathbf{K}\mathbf{V}) \quad \text{and} \quad \mathbf{V} = \mathbf{B} \oslash (\mathbf{K}^\top \mathbf{U}). \quad (13)$$

Hence we could have the matrix version of the updating equations of Sinkhorn algorithm.

Update 3.2. At any iteration $\ell = 1, \dots, L$ and initialized $\mathbf{V}^{(0)} \in \mathbb{R}^{N \times S}$, we have

$$\mathbf{U}^{(\ell)} \equiv \mathbf{A} \oslash (\mathbf{K}\mathbf{V}^{(\ell-1)}) \quad \text{and} \quad \mathbf{V}^{(\ell)} \equiv \mathbf{B} \oslash (\mathbf{K}^\top \mathbf{U}^{(\ell)}). \quad (14)$$

Remark. Since Update 3.2 only involves matrix operations, the Sinkhorn algorithm are intrinsically GPU friendly, as matrix operations are even more efficient on GPU than on CPU.

Algorithm 3.2 Parallel Sinkhorn Algorithm

Input: $\mathbf{A} \in \Sigma_{M \times S}$, $\mathbf{B} \in \Sigma_{N \times S}$, $\mathbf{C} \in \mathbb{R}^{M \times N}$, $\varepsilon > 0$.

Initialize: $\mathbf{U} = \mathbb{1}_{M \times S}$, $\mathbf{V} = \mathbb{1}_{N \times S}$.

- 1: $\mathbf{K} = \exp(-\frac{\mathbf{C}}{\varepsilon})$
- 2: **while** Not Convergence **do**
- 3: $\mathbf{U} \leftarrow \mathbf{A} \oslash (\mathbf{K}\mathbf{V})$
- 4: $\mathbf{V} \leftarrow \mathbf{B} \oslash (\mathbf{K}^\top \mathbf{U})$
- 5: **end while**
- 6: $\forall s, \mathbf{P}_s = \text{diag}(\mathbf{U}_s) \mathbf{K} \text{diag}(\mathbf{V}_s)$

Output: Optimal Coupling $\mathbf{P}_s, \forall s$

Remark (Convergence Condition). Similar to the termination condition in Section 3.1, we now have the matrix norms to check convergence. For some $\rho > 0$,

$$\|\mathbf{U}^{(\ell)} \odot (\mathbf{K}\mathbf{V}^{(\ell)}) - \mathbf{A}\|_2 \leq \rho, \quad \text{and} \quad \|\mathbf{V}^{(\ell)} \odot (\mathbf{K}^\top \mathbf{U}^{(\ell)}) - \mathbf{B}\|_2 \leq \rho.$$

⁶For example, Eigen (C++): https://eigen.tuxfamily.org/dox/group_TutorialReductionsVisitorsBroadcasting.html, NumPy (Python): <https://numpy.org/doc/stable/user/basics.broadcasting.html>.

⁷Instead of computing the vector-vector computation in Equation (11) S times, we are computing a matrix-matrix product/division once. They are mathematically equivalent, but in practice, the latter would be much faster due to the implementation of linear algebra routines (e.g. BLAS). Hence, this simple extension is considered parallel since we process S columns of data in one batch.

3.3 Log-stabilized Sinkhorn

As can be seen in Equations (8) and (11), the updates on \mathbf{u} and \mathbf{v} are in the exponential domain. Therefore, if either \mathbf{C}_{ij} is large or ε is small, the corresponding \mathbf{K}_{ij} would be very large, and the subsequent updates will fail because of the Inf/NaN values introduced by the numeric overflow. This is commonly known as the numeric instability issue of the Sinkhorn algorithm (Peyré and Cuturi, 2019, Chapter 4.4).

To circumvent this instability issue, we could move the computation into the log-domain, therefore without the need for computation of the exponential function. To this end, instead of updating the \mathbf{u} and \mathbf{v} as in Update 3.1, we can consider the dual variables \mathbf{f} and \mathbf{g} in the Lagrangian in Equation (7). Since $\mathbf{u} = \exp\left(\frac{\mathbf{f}}{\varepsilon}\right)$ and $\mathbf{v} = \exp\left(\frac{\mathbf{g}}{\varepsilon}\right)$, we have

$$\mathbf{f} = \varepsilon \log(\mathbf{u}) \quad \text{and} \quad \mathbf{g} = \varepsilon \log(\mathbf{v}). \quad (15)$$

Then, we can also rewrite the elements in the coupling matrix \mathbf{P} as

$$\mathbf{P}_{ij}^{(\ell)} = \mathbf{u}_i^{(\ell)} \mathbf{K}_{ij} \mathbf{v}_j^{(\ell)} = \exp\left(-\frac{\mathbf{C}_{ij} - \mathbf{f}_i^{(\ell)} - \mathbf{g}_j^{(\ell)}}{\varepsilon}\right)$$

and the matrix form of \mathbf{P} expressed in \mathbf{f} and \mathbf{g} is

$$\mathbf{P}^{(\ell)} = \exp\left(-\frac{\mathbf{C} - \mathbf{f}^{(\ell)} \cdot \mathbf{1}_N^\top - \mathbf{1}_M \cdot \mathbf{g}^{(\ell)\top}}{\varepsilon}\right). \quad (16)$$

Let us also denote a function $\mathcal{P}(\mathbf{f}, \mathbf{g})$ with $\mathbf{f} \in \mathbb{R}^M$ and $\mathbf{g} \in \mathbb{R}^N$,

$$\mathcal{P}(\mathbf{f}, \mathbf{g}) = \exp\left(-\frac{\mathbf{C} - \mathbf{f} \cdot \mathbf{1}_N^\top - \mathbf{1}_M \cdot \mathbf{g}^\top}{\varepsilon}\right), \quad (17)$$

and rewrite $\mathbf{P}^{(\ell)} = \mathcal{P}(\mathbf{f}^{(\ell)}, \mathbf{g}^{(\ell)})$. Note that \mathbf{f} and \mathbf{g} are dual variables in the Lagrangian in Equation (7). Then we can plug in \mathbf{u} and \mathbf{v} to get the following updating equations

$$\begin{aligned} \mathbf{f}^{(\ell)} &= \varepsilon \log \mathbf{u}^{(\ell)} = \varepsilon \log \mathbf{a} - \varepsilon \log(\mathbf{K} \mathbf{v}^{(\ell-1)}), \\ \mathbf{g}^{(\ell)} &= \varepsilon \log \mathbf{v}^{(\ell)} = \varepsilon \log \mathbf{b} - \varepsilon \log(\mathbf{K}^\top \mathbf{u}^{(\ell)}) \end{aligned} \quad (18)$$

Since $\mathbf{K} \mathbf{v}^{(\ell-1)} \in \mathbb{R}^m$, $\forall i$, the i -th element is

$$\begin{aligned} (\mathbf{K} \mathbf{v}^{(\ell-1)})_i &= \sum_j \mathbf{K}_{ij} \mathbf{v}_j^{(\ell-1)} \\ &= \sum_j \exp\left(-\frac{\mathbf{C}_{ij}}{\varepsilon}\right) \exp\left(\frac{\mathbf{g}_j^{(\ell-1)}}{\varepsilon}\right) \\ &= \left[\sum_j \exp\left(-\frac{\mathbf{C}_{ij} - \mathbf{f}_i^{(\ell-1)} - \mathbf{g}_j^{(\ell-1)}}{\varepsilon}\right) \right] \exp\left(-\frac{\mathbf{f}_i^{(\ell-1)}}{\varepsilon}\right), \end{aligned}$$

and

$$\begin{aligned}
-\varepsilon \log (\mathbf{K} \mathbf{v}^{(\ell-1)})_i &= -\varepsilon \log \sum_j \exp \left(-\frac{\mathbf{C}_{ij} - \mathbf{f}_i^{(\ell-1)} - \mathbf{g}_j^{(\ell-1)}}{\varepsilon} \right) + \mathbf{f}_i^{(\ell-1)} \\
&= -\varepsilon \log \left[\exp \left(-\frac{\mathbf{C} - \mathbf{f}^{(\ell-1)} \mathbf{1}_N^\top - \mathbf{1}_M (\mathbf{g}^{(\ell-1)})^\top}{\varepsilon} \right) \cdot \mathbf{1}_N \right]_i + \mathbf{f}_i^{(\ell-1)} \\
&= -\varepsilon \log [\mathcal{P}(\mathbf{f}^{(\ell-1)}, \mathbf{g}^{(\ell-1)}) \cdot \mathbf{1}_N]_i + \mathbf{f}_i^{(\ell-1)}.
\end{aligned}$$

Therefore, we have

$$-\varepsilon \log (\mathbf{K} \mathbf{v}^{(\ell)}) = -\varepsilon \log [\mathcal{P}(\mathbf{f}^{(\ell)}, \mathbf{g}^{(\ell)}) \cdot \mathbf{1}_N] + \mathbf{f}^{(\ell)}. \quad (19)$$

And we have the updating equation for $\mathbf{f}^{(\ell)}$. If we follow similar steps, we can also get it for $\mathbf{g}^{(\ell)}$. That is,

$$\begin{aligned}
\mathbf{f}^{(\ell)} &= \mathbf{f}^{(\ell-1)} + \varepsilon \log \mathbf{a} - \varepsilon \log [\mathcal{P}(\mathbf{f}^{(\ell-1)}, \mathbf{g}^{(\ell-1)}) \cdot \mathbf{1}_N], \\
\mathbf{g}^{(\ell)} &= \mathbf{g}^{(\ell-1)} + \varepsilon \log \mathbf{b} - \varepsilon \log [\mathcal{P}^\top(\mathbf{f}^{(\ell)}, \mathbf{g}^{(\ell-1)}) \cdot \mathbf{1}_M].
\end{aligned} \quad (20)$$

Note that $\log [\mathcal{P}(\mathbf{f}, \mathbf{g}) \cdot \mathbf{1}_N]$ is essentially a “log-sum-exp” form, and we can apply the “log-sum-exp” trick⁸ to make this equation numeric stable while computing it in practice. Therefore, let us define the following two functions,

$$\begin{aligned}
\mathcal{Q}(\mathbf{f}, \mathbf{g}) &= \exp \left(-\frac{\mathbf{C} - \mathbf{f} \cdot \mathbf{1}_N^\top - \mathbf{1}_M \cdot \mathbf{g}^\top - c(\mathbf{f}, \mathbf{g})}{\varepsilon} \right), \text{ where} \\
c(\mathbf{f}, \mathbf{g}) &= \min \{ \mathbf{C} - \mathbf{f} \cdot \mathbf{1}_N^\top - \mathbf{1}_M \cdot \mathbf{g}^\top \},
\end{aligned} \quad (21)$$

we can again rewrite Equation (20) as

$$\begin{aligned}
\mathbf{f}^{(\ell)} &= \mathbf{f}^{(\ell-1)} + \varepsilon \log \mathbf{a} - \varepsilon \log [\mathcal{P}(\mathbf{f}^{(\ell-1)}, \mathbf{g}^{(\ell-1)}) \cdot \mathbf{1}_N], \\
&= \mathbf{f}^{(\ell-1)} + \varepsilon \log \mathbf{a} - \varepsilon \log \left[\exp \left(-\frac{\mathbf{C} - \mathbf{f}^{(\ell-1)} \cdot \mathbf{1}_N^\top - \mathbf{1}_M \cdot \mathbf{g}^{(\ell-1)\top}}{\varepsilon} \right) \cdot \mathbf{1}_N \right] \\
&= \mathbf{f}^{(\ell-1)} + \varepsilon \log \mathbf{a} - \varepsilon \log \left[\exp \left(-\frac{c(\mathbf{f}^{(\ell-1)}, \mathbf{g}^{(\ell-1)})}{\varepsilon} \right) \mathcal{Q}(\mathbf{f}^{(\ell-1)}, \mathbf{g}^{(\ell-1)}) \cdot \mathbf{1}_N \right] \\
&= \mathbf{f}^{(\ell-1)} + \varepsilon \log \mathbf{a} + c(\mathbf{f}^{(\ell-1)}, \mathbf{g}^{(\ell-1)}) - \varepsilon \log [\mathcal{Q}(\mathbf{f}^{(\ell-1)}, \mathbf{g}^{(\ell-1)}) \cdot \mathbf{1}_N], \\
\mathbf{g}^{(\ell)} &= \mathbf{g}^{(\ell-1)} + \varepsilon \log \mathbf{b} + c(\mathbf{f}^{(\ell)}, \mathbf{g}^{(\ell-1)}) - \varepsilon \log [\mathcal{Q}^\top(\mathbf{f}^{(\ell)}, \mathbf{g}^{(\ell-1)}) \cdot \mathbf{1}_M].
\end{aligned}$$

Thus, we have obtained the numerically stable version of the updating equations for \mathbf{f} and \mathbf{g} .

⁸<https://en.wikipedia.org/wiki/LogSumExp>

Update 3.3. At any iteration $\ell = 1, \dots, L$ and initialized $\mathbf{f}^{(0)}$ and $\mathbf{g}^{(0)}$, we have

$$\begin{aligned}\mathbf{f}^{(\ell)} &= \mathbf{f}^{(\ell-1)} + \varepsilon \log \mathbf{a} + c(\mathbf{f}^{(\ell-1)}, \mathbf{g}^{(\ell-1)}) - \varepsilon \log [\mathcal{Q}(\mathbf{f}^{(\ell-1)}, \mathbf{g}^{(\ell-1)}) \cdot \mathbf{1}_N], \\ \mathbf{g}^{(\ell)} &= \mathbf{g}^{(\ell-1)} + \varepsilon \log \mathbf{b} + c(\mathbf{f}^{(\ell)}, \mathbf{g}^{(\ell-1)}) - \varepsilon \log [\mathcal{Q}^\top(\mathbf{f}^{(\ell)}, \mathbf{g}^{(\ell-1)}) \cdot \mathbf{1}_M].\end{aligned}\tag{22}$$

Remark. As mentioned in Section 3.1, the initialization for \mathbf{u} and \mathbf{v} are vectors of ones. This is to say that, the initializations of \mathbf{f} and \mathbf{g} are vectors of zeros, i.e. $\mathbf{f}^{(0)} = \mathbf{0}_M$ and $\mathbf{g}^{(0)} = \mathbf{0}_N$.

Therefore, we can have the Log-stabilized Sinkhorn algorithm.

Algorithm 3.3 Log-Stabilized Sinkhorn Algorithm

Input: $\mathbf{a} \in \Sigma_M$, $\mathbf{b} \in \Sigma_N$, $\mathbf{C} \in \mathbb{R}^{M \times N}$, $\varepsilon > 0$.

Initialize: $\mathbf{f} = \mathbf{0}_M$, $\mathbf{g} = \mathbf{0}_N$.

```

1: while Not Convergence do
2:   # update  $f$ 
3:    $\mathbf{R} = \mathbf{C} - \mathbf{f} \cdot \mathbf{1}_N^\top - \mathbf{1}_M \cdot \mathbf{g}^\top$ 
4:    $c = \min \mathbf{R}$ 
5:    $\mathbf{Q} = \exp\left(-\frac{\mathbf{R}-c}{\varepsilon}\right)$ 
6:    $\mathbf{f} = \mathbf{f} + \varepsilon \log \mathbf{a} + c - \varepsilon \log [\mathbf{Q} \cdot \mathbf{1}_N]$ 
7:   # update  $g$ 
8:    $\mathbf{R} = \mathbf{C} - \mathbf{f} \cdot \mathbf{1}_N^\top - \mathbf{1}_M \cdot \mathbf{g}^\top$ 
9:    $c = \min \mathbf{R}$ 
10:   $\mathbf{Q} = \exp\left(-\frac{\mathbf{R}-c}{\varepsilon}\right)$ 
11:   $\mathbf{g} = \mathbf{g} + \varepsilon \log \mathbf{b} + c - \varepsilon \log [\mathbf{Q}^\top \cdot \mathbf{1}_M]$ 
12: end while
13:  $\mathbf{P} = \exp\left(-\frac{\mathbf{C} - \mathbf{f} \mathbf{1}_N^\top - \mathbf{1}_M \mathbf{g}^\top}{\varepsilon}\right)$ 
```

Output: \mathbf{P}

Remark. The advantage of using log-stabilized Sinkhorn is to make sure the computation is stable for any arbitrary ε . The disadvantage, however, is that we cannot easily parallel its computation over several margins as in Section 3.2. Moreover, the computation of matrix \mathbf{Q} involves exp and log at every iteration twice for both updating \mathbf{f} and \mathbf{g} . Thus, the log-stabilized Sinkhorn is less efficient than Vanilla and Parallel Sinkhorn algorithms. This is the tradeoff we need to make if we need to make sure the computation is stable, especially if we want to use a very small regularization constant ε . As in Peyré and Cuturi (2019, Chapter 4.1), the Sinkhorn problem converges to original Kantorovich problem when ε is small, i.e. $L_C^\varepsilon(\mathbf{a}, \mathbf{b}) \rightarrow L_C(\mathbf{a}, \mathbf{b})$ as $\varepsilon \rightarrow 0$.

Remark (Convergence Condition). Recall that in Equation (17), we can recover the optimal coupling matrix from \mathbf{f} and \mathbf{g} , then for some $\rho > 0$,

$$\|\mathcal{P}(\mathbf{f}^{(\ell)}, \mathbf{g}^{(\ell)}) - \mathbf{a}\|_2 \leq \rho \quad \text{and} \quad \|\mathcal{P}^\top(\mathbf{f}^{(\ell)}, \mathbf{g}^{(\ell)}) - \mathbf{b}\|_2 \leq \rho.$$

4 Sinkhorn Algorithms with Gradients

In this section, I will derive the gradients of Sinkhorn loss with respect to its arguments for the algorithms described in Section 3. This is useful when one wishes to use regularized loss in Equation (3) as a loss function for training purpose in machine learning, for example, Wasserstein Dictionary Learning in Sections 5 to 7.

4.1 Vanilla Sinkhorn

The optimal coupling matrix \mathbf{P} from the Vanilla Sinkhorn algorithm is

$$\mathbf{P} = \text{diag}(\mathbf{u}^{(L)})\mathbf{K}\text{diag}(\mathbf{v}^{(L)}), \quad (23)$$

and the loss function evaluated at the solution is $L_{\mathbf{C}}^{\varepsilon}(\mathbf{P}) = \langle \mathbf{P}, \mathbf{C} \rangle + \varepsilon \langle \mathbf{P}, \log \mathbf{P} - \mathbf{1} \rangle$.

Therefore, by Chain's rule, the Jacobian of Sinkhorn loss w.r.t. first argument \mathbf{a} is

$$DL_{\mathbf{C}}^{\varepsilon}(\mathbf{a}) = DL_{\mathbf{C}}^{\varepsilon}(\mathbf{P}) \cdot D\mathbf{P}(\mathbf{a}). \quad (24)$$

Note that $\langle \mathbf{A}, \mathbf{B} \rangle = \mathbf{1}_M^{\top} (\mathbf{A} \odot \mathbf{B}) \mathbf{1}_N$, for matrices \mathbf{A} and \mathbf{B} in $\mathbb{R}^{M \times N}$. To get the first term $DL_{\mathbf{C}}^{\varepsilon}(\mathbf{P})$, we need the differential as the following

$$\begin{aligned} d \text{vec } L_{\mathbf{C}}^{\varepsilon}(\mathbf{P}) &= d \text{vec } [\mathbf{1}_M^{\top} (\mathbf{P} \odot \mathbf{C}) \mathbf{1}_N] + \varepsilon d \text{vec } \{ \mathbf{1}_M^{\top} [\mathbf{P} \odot [\log \mathbf{P} - \mathbf{1}]] \mathbf{1}_N \} \\ &= d \left[(\mathbf{1}_N \otimes \mathbf{1}_M)^{\top} (\text{vec } \mathbf{C} \otimes \text{vec } \mathbf{P}) \right] + \varepsilon d \left\{ (\mathbf{1}_N \otimes \mathbf{1}_M)^{\top} [\text{vec } \mathbf{P} \otimes \text{vec } (\log \mathbf{P} - \mathbf{1})] \right\} \\ &= \mathbf{1}_{MN}^{\top} (\text{vec } \mathbf{C} \otimes d \text{vec } \mathbf{P}) + \varepsilon \mathbf{1}_{MN}^{\top} \cdot d [\text{vec } \mathbf{P} \otimes \text{vec } (\log \mathbf{P} - \mathbf{1})] \\ &= \mathbf{1}_{MN}^{\top} \cdot \text{diag } \text{vec } \mathbf{C} \cdot d \text{vec } \mathbf{P} + \varepsilon \mathbf{1}_{MN}^{\top} \cdot \text{diag } \text{vec } \log \mathbf{P} \cdot d \text{vec } \mathbf{P} \\ &= \mathbf{1}_{MN}^{\top} \cdot \text{diag } \text{vec } (\mathbf{C} + \varepsilon \log \mathbf{P}) \cdot d \text{vec } \mathbf{P}. \end{aligned}$$

and the Jacobian is

$$DL_{\mathbf{C}}^{\varepsilon}(\mathbf{P}) = \mathbf{1}_{MN}^{\top} \text{diag } \text{vec } (\mathbf{C} + \varepsilon \log \mathbf{P}). \quad (25)$$

Note that this Jacobian matrix has only one row, since $(\mathbf{1}_N \otimes \mathbf{1}_M) = \mathbf{1}_{MN}$ has only one column. Thus, the gradient⁹ $\nabla_{\mathbf{a}} L_{\mathbf{C}}^{\varepsilon}$, would be column vector of size M .

To get the second term, i.e. $D\mathbf{P}(\mathbf{a})$, we also need the differential of \mathbf{P} in Equation (16) instead of Equation (23), as Equation (16) involves the dual variables \mathbf{f} and \mathbf{g} , which will be useful when we derive the Jacobian matrices for the log-stabilized Sinkhorn algorithm in Section 3.3.

The differential is

$${}^9\nabla_{\mathbf{a}} L_{\mathbf{C}}^{\varepsilon} = (DL_{\mathbf{C}}^{\varepsilon}(\mathbf{a}))^{\top}.$$

$$\begin{aligned}
d \operatorname{vec} \mathbf{P} &= -\frac{1}{\varepsilon} \operatorname{diag} \operatorname{vec} \mathbf{P} d \operatorname{vec} \left(\mathbf{C} - \mathbf{f}^{(L)} \mathbb{1}_N^\top - \mathbb{1}_M (\mathbf{g}^{(L)})^\top \right) \\
&= \frac{1}{\varepsilon} \operatorname{diag} \operatorname{vec} \mathbf{P} \left[d \operatorname{vec} (\mathbf{f}^{(L)} \mathbb{1}_N^\top) + d \operatorname{vec} (\mathbb{1}_m (\mathbf{g}^{(L)})^\top) \right] \\
&= \frac{1}{\varepsilon} \operatorname{diag} \operatorname{vec} \mathbf{P} \left[(\mathbb{1}_N \otimes I_M) d\mathbf{f}^{(L)} + (I_N \otimes \mathbb{1}_M) d\mathbf{g}^{(L)} \right] \\
&= \operatorname{diag} \operatorname{vec} \mathbf{P} \left[(\mathbb{1}_N \otimes I_M) \cdot \operatorname{diag} \frac{1}{\mathbf{u}^{(L)}} \cdot d\mathbf{u}^{(L)} + (I_N \otimes \mathbb{1}_M) \cdot \operatorname{diag} \frac{1}{\mathbf{v}^{(L)}} \cdot d\mathbf{v}^{(L)} \right],
\end{aligned} \tag{26}$$

since we also have $\mathbf{f} = \varepsilon \log \mathbf{u}$ and $\mathbf{g} = \varepsilon \log \mathbf{v}$, $d\mathbf{f} = \varepsilon \operatorname{diag} \frac{1}{\mathbf{u}} \cdot d\mathbf{u}$ and $d\mathbf{g} = \varepsilon \operatorname{diag} \frac{1}{\mathbf{v}} \cdot d\mathbf{v}$, respectively.

Therefore,

$$D\mathbf{P}(\mathbf{a}) = \operatorname{diag} \operatorname{vec} \mathbf{P} \left[(\mathbb{1}_N \otimes I_M) \cdot \operatorname{diag} \frac{1}{\mathbf{u}^{(L)}} \cdot D\mathbf{u}^{(L)}(\mathbf{a}) + (I_N \otimes \mathbb{1}_M) \cdot \operatorname{diag} \frac{1}{\mathbf{v}^{(L)}} \cdot D\mathbf{v}^{(L)}(\mathbf{a}) \right]. \tag{27}$$

From Equation (27), all that remains to be shown are the formulae of $D\mathbf{u}^{(L)}(\mathbf{a})$ and $D\mathbf{v}^{(L)}(\mathbf{a})$, which can be obtained iteratively differentiating Update 3.1. Again, we need first to calculate the differentials

$$\begin{aligned}
d\mathbf{u}^{(\ell)} &= d \left[\mathbf{a} \odot (\mathbf{K}\mathbf{v}^{(\ell-1)}) \right] = d \left(\mathbf{a} \odot \frac{1}{\mathbf{K}\mathbf{v}^{(\ell-1)}} \right) \\
&= d\mathbf{a} \odot \frac{1}{\mathbf{K}\mathbf{v}^{(\ell-1)}} - \mathbf{a} \odot d \left(\frac{1}{\mathbf{K}\mathbf{v}^{(\ell-1)}} \right) \\
&= \operatorname{diag} \left(\frac{1}{\mathbf{K}\mathbf{v}^{(\ell-1)}} \right) \cdot d\mathbf{a} - \operatorname{diag} \frac{\mathbf{a}}{(\mathbf{K}\mathbf{v}^{(\ell-1)})^2} \cdot \mathbf{K} \cdot d\mathbf{v}^{(\ell-1)} \\
d\mathbf{v}^{(\ell)} &= d \left(\mathbf{b} \odot \frac{1}{\mathbf{K}^\top \mathbf{u}^{(\ell)}} \right) \\
&= d\mathbf{b} \odot \frac{1}{\mathbf{K}^\top \mathbf{u}^{(\ell)}} + \mathbf{b} \odot d \left(\frac{1}{\mathbf{K}^\top \mathbf{u}^{(\ell)}} \right) \\
&= -\operatorname{diag} \frac{\mathbf{b}}{(\mathbf{K}^\top \mathbf{u}^{(\ell)})^2} \cdot \mathbf{K}^\top \cdot d\mathbf{u}^{(\ell)}
\end{aligned}$$

Therefore, the Jacobian updating equations are

Update 4.1 (Updating Equations for Jacobians of Vanilla Sinkhorn).

$$\begin{aligned}
D\mathbf{u}^{(\ell)}(\mathbf{a}) &= \operatorname{diag} \frac{1}{\mathbf{K}\mathbf{v}^{(\ell-1)}} - \operatorname{diag} \frac{\mathbf{a}}{(\mathbf{K}\mathbf{v}^{(\ell-1)})^2} \cdot \mathbf{K} \cdot D\mathbf{v}^{(\ell-1)}(\mathbf{a}), \\
D\mathbf{v}^{(\ell)}(\mathbf{a}) &= -\operatorname{diag} \frac{\mathbf{b}}{(\mathbf{K}^\top \mathbf{u}^{(\ell)})^2} \cdot \mathbf{K}^\top \cdot D\mathbf{u}^{(\ell)}(\mathbf{a}).
\end{aligned} \tag{28}$$

Therefore, for the initialized $D\mathbf{v}^{(0)}(\mathbf{u}^{(0)}) = \mathbf{0}_{N \times N}$, we can compute $D\mathbf{u}^{(\ell)}$ and $D\mathbf{v}^{(\ell)}$ for any ℓ , including L . Then we can finally arrive at the formula for the Jacobian

$$\begin{aligned} DL_{\mathbf{C}}^{\varepsilon}(\mathbf{a}) &= DL_{\mathbf{C}}^{\varepsilon}(\mathbf{P}) \cdot D\mathbf{P}(\mathbf{a}) \\ &= \mathbf{1}_{MN}^{\top} \text{diag vec}(\mathbf{C} + \varepsilon \log \mathbf{P}) \text{diag vec } \mathbf{P} \left[(\mathbf{1}_N \otimes I_M) \text{diag} \frac{1}{\mathbf{u}^{(L)}} \cdot D\mathbf{u}^{(L)}(\mathbf{a}) \right. \\ &\quad \left. + (I_N \otimes \mathbf{1}_M) \text{diag} \frac{1}{\mathbf{v}^{(L)}} \cdot D\mathbf{v}^{(L)}(\mathbf{a}) \right], \end{aligned} \quad (29)$$

and the gradient $\nabla_{\mathbf{a}} L_{\mathbf{C}}^{\varepsilon} = \left[DL_{\mathbf{C}}^{\varepsilon}(\mathbf{a}) \right]^{\top}$.

We can combine the Jacobian updates with the Sinkhorn updates in one loop, hence without the need to save intermediate Jacobian matrices.

Algorithm 4.1 Vanilla Sinkhorn with Gradient

Input: $\mathbf{a} \in \Sigma_M$, $\mathbf{b} \in \Sigma_N$, $\mathbf{C} \in \mathbb{R}^{M \times N}$, $\varepsilon > 0$.

Initialize: $\mathbf{u} = \mathbf{1}_M$, $\mathbf{v} = \mathbf{1}_N$, $\mathbf{J}_{\mathbf{u}} = \mathbf{0}_{M \times M}$, $\mathbf{J}_{\mathbf{v}} = \mathbf{0}_{N \times M}$.

```

1:  $\mathbf{K} = \exp(-\frac{\mathbf{C}}{\varepsilon})$ 
2: while Not Convergence do
3:   # update  $\mathbf{J}_{\mathbf{u}}$  and  $\mathbf{u}$ 
4:    $\mathbf{J}_{\mathbf{u}} = \text{diag} \frac{1}{\mathbf{K}\mathbf{v}} - \text{diag} \frac{\mathbf{a}}{(\mathbf{K}\mathbf{v})^2} \cdot \mathbf{K} \cdot \mathbf{J}_{\mathbf{v}}$ 
5:    $\mathbf{u} \leftarrow \mathbf{a} \odot (\mathbf{K}\mathbf{v})$ 
6:   # update  $\mathbf{J}_{\mathbf{v}}$  and  $\mathbf{v}$ 
7:    $\mathbf{J}_{\mathbf{v}} = -\text{diag} \frac{\mathbf{b}}{(\mathbf{K}^{\top}\mathbf{u}^{(\ell)})^2} \cdot \mathbf{K}^{\top} \cdot \mathbf{J}_{\mathbf{u}}$ 
8:    $\mathbf{v} \leftarrow \mathbf{b} \odot (\mathbf{K}^{\top}\mathbf{u})$ 
9: end while
10: # compute Optimal Coupling
11:  $\mathbf{P} = \text{diag}(\mathbf{u}) \mathbf{K} \text{diag}(\mathbf{v})$ 
12: # compute gradient
13:  $\mathbf{J}_{\mathbf{P}} = \mathbf{1}_{MN}^{\top} \cdot \text{diag vec}(\mathbf{C} + \varepsilon \log \mathbf{P})$ 
14:  $\mathbf{J}_{\mathbf{a}} = \text{diag vec } \mathbf{P} \cdot [(\mathbf{1}_N \otimes I_M) \text{diag} \frac{1}{\mathbf{u}} \cdot \mathbf{J}_{\mathbf{u}} + (I_N \times \mathbf{1}_m) \text{diag} \frac{1}{\mathbf{v}} \cdot \mathbf{J}_{\mathbf{v}}]$ 
15:  $\nabla_{\mathbf{a}} = (\mathbf{J}_{\mathbf{P}} \cdot \mathbf{J}_{\mathbf{a}})^{\top}$ 
Output:  $\mathbf{P}$ ,  $\nabla_{\mathbf{a}}$ 

```

4.2 Parallel Sinkhorn

For the Parallel Sinkhorn algorithm, since we have S transport maps being computed at the same time, as supposed to the Vanilla Sinkhorn where we only compute one transport map at a time, we would have a loss vector of length S instead of a scalar. Therefore, without a proper way to aggregate the loss vector into a scalar, we cannot derive the gradient of the loss. We can, however, derive the Jacobian updating equations for the Parallel Sinkhorn

algorithm, which is the focus of this subsection. With the Jacobians defined recursively, one can easily apply it to gradient computation, once the (scalar) loss function is properly defined.

A natural way to introduce an aggregation is to define the Wasserstein barycenter problem, where we can take the weighted sum of the Sinkhorn losses as the loss function for the minimization problem. But this discussion is deferred to Section 5.

Here, to derive the Jacobian updating equations, we can take the differentials

$$\begin{aligned}
d \operatorname{vec} \mathbf{U}^{(\ell)} &= d \left[\operatorname{vec} \mathbf{A} \odot \frac{1}{\operatorname{vec}(\mathbf{K}\mathbf{V}^{(\ell-1)})} \right] \\
&= d \operatorname{vec} \mathbf{A} \odot \frac{1}{\operatorname{vec}(\mathbf{K}\mathbf{V}^{(\ell-1)})} + \operatorname{vec} \mathbf{A} \odot d \frac{1}{\operatorname{vec}(\mathbf{K}\mathbf{V}^{(\ell-1)})} \\
&= \operatorname{diag} \operatorname{vec} \frac{1}{\mathbf{K}\mathbf{V}^{(\ell-1)}} \cdot d \operatorname{vec} \mathbf{A} - \operatorname{diag} \operatorname{vec} \mathbf{A} \cdot \operatorname{diag} \operatorname{vec} \frac{1}{(\mathbf{K}\mathbf{V}^{(\ell-1)})^2} \cdot d \operatorname{vec}(\mathbf{K}\mathbf{V}^{(\ell-1)}) \\
&= \operatorname{diag} \operatorname{vec} \frac{1}{\mathbf{K}\mathbf{V}^{(\ell-1)}} \cdot d \operatorname{vec} \mathbf{A} - \operatorname{diag} \operatorname{vec} \frac{\mathbf{A}}{(\mathbf{K}\mathbf{V}^{(\ell-1)})^2} \cdot (I_S \otimes \mathbf{K}) \cdot d \operatorname{vec} \mathbf{V}^{(\ell-1)},
\end{aligned} \tag{30}$$

and

$$\begin{aligned}
d \operatorname{vec} \mathbf{V}^{(\ell)} &= d \left[\operatorname{vec} \mathbf{B} \odot \frac{1}{\operatorname{vec}(\mathbf{K}^\top \mathbf{U}^{(\ell)})} \right] \\
&= \operatorname{vec} \mathbf{B} \odot d \frac{1}{\operatorname{vec}(\mathbf{K}^\top \mathbf{U}^{(\ell)})} \\
&= - \operatorname{diag} \operatorname{vec} \mathbf{B} \cdot \operatorname{diag} \operatorname{vec} \frac{1}{(\mathbf{K}^\top \mathbf{U}^{(\ell)})^2} \cdot d \operatorname{vec}(\mathbf{K}^\top \mathbf{U}^{(\ell)}) \\
&= - \operatorname{diag} \operatorname{vec} \frac{\mathbf{B}}{(\mathbf{K}^\top \mathbf{U}^{(\ell)})^2} \cdot (I_S \otimes \mathbf{K})^\top \cdot d \operatorname{vec} \mathbf{U}^{(\ell)}.
\end{aligned} \tag{31}$$

Therefore, we obtain the Jacobian updating equations.

Update 4.2 (Updating Equations for Jacobians of Parallel Sinkhorn).

$$\begin{aligned}
D \mathbf{U}^{(\ell)}(\mathbf{A}) &= \operatorname{diag} \operatorname{vec} \frac{1}{\mathbf{K}\mathbf{V}^{(\ell-1)}} - \operatorname{diag} \operatorname{vec} \frac{\mathbf{A}}{(\mathbf{K}\mathbf{V}^{(\ell-1)})^2} \cdot (I_S \otimes \mathbf{K}) \cdot D \mathbf{V}^{(\ell-1)}(\mathbf{A}), \\
D \mathbf{V}^{(\ell)}(\mathbf{A}) &= - \operatorname{diag} \operatorname{vec} \frac{\mathbf{B}}{(\mathbf{K}^\top \mathbf{U}^{(\ell)})^2} \cdot (I_S \otimes \mathbf{K})^\top \cdot D \mathbf{U}^{(\ell)}(\mathbf{A}).
\end{aligned} \tag{32}$$

Algorithm 4.2 Parallel Sinkhorn Algorithm with Jacobians**Input:** $\mathbf{A} \in \Sigma_{M \times S}$, $\mathbf{B} \in \Sigma_{N \times S}$, $\mathbf{C} \in \mathbb{R}^{M \times N}$, $\varepsilon > 0$.**Initialize:** $\mathbf{U} = \mathbf{1}_{M \times S}$, $\mathbf{V} = \mathbf{1}_{N \times S}$, $\mathbf{J}_\mathbf{U} = \mathbf{0}_{MS \times MS}$, $\mathbf{J}_\mathbf{V} = \mathbf{0}_{NS \times MS}$.1: $\mathbf{K} = \exp(-\frac{\mathbf{C}}{\varepsilon})$ 2: **while** Not Convergence **do**3: $\mathbf{J}_\mathbf{U} = \text{diag vec } \frac{1}{\mathbf{K}\mathbf{V}} - \text{diag vec } \frac{\mathbf{A}}{(\mathbf{K}\mathbf{V})^2} \cdot (I_S \otimes \mathbf{K}) \cdot \mathbf{J}_\mathbf{V}$ # update $\mathbf{J}_\mathbf{U}$ 4: $\mathbf{U} \leftarrow \mathbf{A} \oslash (\mathbf{K}\mathbf{V})$ 5: $\mathbf{J}_\mathbf{V} = -\text{diag vec } \frac{\mathbf{B}}{(\mathbf{K}^\top \mathbf{U})^2} \cdot (I_S \otimes \mathbf{K})^\top \cdot \mathbf{J}_\mathbf{U}$ # update $\mathbf{J}_\mathbf{V}$ 6: $\mathbf{V} \leftarrow \mathbf{B} \oslash (\mathbf{K}^\top \mathbf{U})$ 7: **end while**8: $\forall s, \mathbf{P}_s = \text{diag}(\mathbf{U}_s) \mathbf{K} \text{diag}(\mathbf{V}_s)$ **Output:** \mathbf{P}_s for all s , $\mathbf{J}_\mathbf{U}$, $\mathbf{J}_\mathbf{V}$

4.3 Log-Stabilized Sinkhorn

To derive the gradient of the log-stabilized Sinkhorn algorithm, we again use the Chain's rule to compute the Jacobian first as in Equation (24). We would thus have the same $DL_{\mathbf{C}}^\varepsilon(\mathbf{P}) = \mathbf{1}_{MN}^\top \cdot \text{diag vec } (\mathbf{C} + \varepsilon \log \mathbf{P})$, where $\mathbf{P} = \mathcal{P}(\mathbf{f}^{(L)}, \mathbf{g}^{(L)})$.

Then all we need is $D\mathbf{P}(\mathbf{a})$. To get this term, we will take the differential \mathcal{P} defined as Equation (17) in Section 3.3, and $\mathbf{f} = \mathbf{f}(\mathbf{a})$ and $\mathbf{g} = \mathbf{g}(\mathbf{a})$,

$$\begin{aligned}
d \text{ vec } \mathcal{P}(\mathbf{f}, \mathbf{g}) &= d \text{ vec } \exp \left(-\frac{\mathbf{C} - \mathbf{f} \cdot \mathbf{1}_N^\top - \mathbf{1}_M \cdot \mathbf{g}^\top}{\varepsilon} \right) \\
&= \text{diag vec } \mathcal{P}(\mathbf{f}, \mathbf{g}) \cdot d \left[-\frac{\mathbf{C} - \mathbf{f} \cdot \mathbf{1}_N^\top - \mathbf{1}_M \cdot \mathbf{g}^\top}{\varepsilon} \right] \\
&= \frac{1}{\varepsilon} \text{diag vec } \mathcal{P}(\mathbf{f}, \mathbf{g}) \cdot d \text{ vec } (\mathbf{f} \cdot \mathbf{1}_N^\top + \mathbf{1}_M \cdot \mathbf{g}^\top) \\
&= \frac{1}{\varepsilon} \text{diag vec } \mathcal{P}(\mathbf{f}, \mathbf{g}) \cdot \left[(\mathbf{1}_N \otimes I_M) d\mathbf{f} + (I_N \otimes \mathbf{1}_M) d\mathbf{g} \right].
\end{aligned} \tag{33}$$

Therefore, the Jacobian of $\mathbf{P} = \mathcal{P}(\mathbf{f}^{(L)}, \mathbf{g}^{(L)})$ is

$$\begin{aligned}
D\mathbf{P}(\mathbf{a}) &= D\mathcal{P}(\mathbf{f}^{(L)}(\mathbf{a}), \mathbf{g}^{(L)}(\mathbf{a})) \\
&= \frac{1}{\varepsilon} \text{diag vec } \mathbf{P} \cdot \left[(\mathbf{1}_N \otimes I_M) D\mathbf{f}^{(L)}(\mathbf{a}) + (I_N \otimes \mathbf{1}_M) D\mathbf{g}^{(L)}(\mathbf{a}) \right].
\end{aligned} \tag{34}$$

Note that for \mathcal{P} , \mathcal{Q} , and c defined in Equations (17) and (21),

$$\mathcal{P}(\mathbf{f}, \mathbf{g}) = \exp \left(-\frac{\mathbf{C} - \mathbf{f} \cdot \mathbf{1}_N^\top - \mathbf{1}_M \cdot \mathbf{g}^\top}{\varepsilon} \right) = \exp \left(-\frac{c(\mathbf{f}, \mathbf{g})}{\varepsilon} \right) \cdot \mathcal{Q}(\mathbf{f}, \mathbf{g}), \tag{35}$$

and thus

$$\begin{aligned} \text{diag vec } \frac{1}{\mathcal{P}(\mathbf{f}, \mathbf{g}) \cdot \mathbb{1}_N} &= \frac{1}{\exp\left(-\frac{c(\mathbf{f}, \mathbf{g})}{\varepsilon}\right)} \cdot \text{diag vec } \frac{1}{\mathcal{Q}(\mathbf{f}, \mathbf{g}) \cdot \mathbb{1}_N}, \\ \text{diag vec } \mathcal{P}(\mathbf{f}, \mathbf{g}) &= \exp\left(-\frac{c(\mathbf{f}, \mathbf{g})}{\varepsilon}\right) \cdot \text{diag vec } \mathcal{Q}(\mathbf{f}, \mathbf{g}). \end{aligned}$$

Then we need to differentiate $\mathbf{f}^{(L)}$ and $\mathbf{g}^{(L)}$ to obtain $D\mathbf{f}^{(L)}(\mathbf{a})$ and $D\mathbf{g}^{(L)}(\mathbf{a})$ to obtain the formulae for the computation of $D\mathbf{P}(\mathbf{a})$. Instead of directly differentiating from Equation (22), we will take the differentials from Equation (20), as this is equivalent to the form in Equation (22), but without the need for differentiation on the min function:

$$\begin{aligned} d \text{ vec } \mathbf{f}^{(\ell)} &= d \mathbf{f}^{(\ell-1)} + \varepsilon d \log \mathbf{a} - \varepsilon d \text{ vec } \log [\mathcal{P}(\mathbf{f}^{(\ell-1)}, \mathbf{g}^{(\ell-1)}) \cdot \mathbb{1}_N] \\ &= d \mathbf{f}^{(\ell-1)} + \varepsilon \text{diag } \frac{1}{\mathbf{a}} \cdot d \mathbf{a} - \varepsilon \text{diag vec } \frac{1}{\mathcal{P}(\mathbf{f}^{(\ell-1)}, \mathbf{g}^{(\ell-1)}) \cdot \mathbb{1}_N} \\ &\quad \cdot d \text{ vec } [\mathcal{P}(\mathbf{f}^{(\ell-1)}, \mathbf{g}^{(\ell-1)}) \cdot \mathbb{1}_N] \\ &= d \mathbf{f}^{(\ell-1)} + \varepsilon \text{diag } \frac{1}{\mathbf{a}} \cdot d \mathbf{a} - \frac{\varepsilon}{\exp\left(-\frac{c(\mathbf{f}^{(\ell-1)}, \mathbf{g}^{(\ell-1)})}{\varepsilon}\right)} \\ &\quad \cdot \text{diag vec } \frac{1}{\mathcal{Q}(\mathbf{f}^{(\ell-1)}, \mathbf{g}^{(\ell-1)}) \cdot \mathbb{1}_N} \cdot (\mathbb{1}_N^\top \otimes I_M) \cdot d \text{ vec } \mathcal{P}(\mathbf{f}^{(\ell-1)}, \mathbf{g}^{(\ell-1)}) \\ &= d \mathbf{f}^{(\ell-1)} + \varepsilon \text{diag } \frac{1}{\mathbf{a}} \cdot d \mathbf{a} - \text{diag vec } \frac{1}{\mathcal{Q}(\mathbf{f}^{(\ell-1)}, \mathbf{g}^{(\ell-1)}) \cdot \mathbb{1}_N} \cdot (\mathbb{1}_N \otimes I_M)^\top \\ &\quad \cdot \text{diag vec } \mathcal{Q}(\mathbf{f}^{(\ell-1)}, \mathbf{g}^{(\ell-1)}) \cdot [(\mathbb{1}_N \otimes I_M) \cdot d \mathbf{f}^{(\ell-1)} + (I_N \otimes \mathbb{1}_M) \cdot d \mathbf{g}^{(\ell-1)}]. \end{aligned} \tag{36}$$

Similarly, we can have the differential for \mathbf{g}^ℓ . Note that a commutation matrix¹⁰ $\mathcal{K}^{(M,N)}$ is one such that $\mathcal{K}^{(M,N)} \cdot \text{vec } \mathbf{A} = \text{vec } (\mathbf{A}^\top)$ for any matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$. Then,

¹⁰https://en.wikipedia.org/wiki/Commutation_matrix

$$\begin{aligned}
d \operatorname{vec} \mathbf{g}^{(\ell)} &= d \mathbf{g}^{(\ell-1)} - \varepsilon \operatorname{diag} \operatorname{vec} \frac{1}{\mathcal{P}^\top(\mathbf{f}^{(\ell)}, \mathbf{g}^{(\ell-1)}) \cdot \mathbb{1}_M} \cdot d \operatorname{vec} [\mathcal{P}^\top(\mathbf{f}^{(\ell)}, \mathbf{g}^{(\ell-1)}) \cdot \mathbb{1}_M] \\
&= d \mathbf{g}^{(\ell-1)} - \varepsilon \operatorname{diag} \operatorname{vec} \frac{1}{\mathcal{P}^\top(\mathbf{f}^{(\ell)}, \mathbf{g}^{(\ell-1)}) \cdot \mathbb{1}_M} \cdot d [(\mathbb{1}_M^\top \otimes I_N) \cdot \mathcal{K}^{(M,N)} \cdot \mathcal{P}(\mathbf{f}^{(\ell)}, \mathbf{g}^{(\ell-1)})] \\
&= d \mathbf{g}^{(\ell-1)} - \varepsilon \operatorname{diag} \operatorname{vec} \frac{1}{\mathcal{P}^\top(\mathbf{f}^{(\ell)}, \mathbf{g}^{(\ell-1)}) \cdot \mathbb{1}_M} \cdot (\mathbb{1}_M^\top \otimes I_N) \cdot \mathcal{K}^{(M,N)} \cdot d \operatorname{vec} \mathcal{P}(\mathbf{f}^{(\ell)}, \mathbf{g}^{(\ell-1)}) \\
&= d \mathbf{g}^{(\ell-1)} - \frac{\varepsilon}{\exp\left(-\frac{c(\mathbf{f}^{(\ell)}, \mathbf{g}^{(\ell-1)})}{\varepsilon}\right)} \cdot \operatorname{diag} \operatorname{vec} \frac{1}{\mathcal{Q}^\top(\mathbf{f}^{(\ell)}, \mathbf{g}^{(\ell-1)}) \cdot \mathbb{1}_M} \\
&\quad \cdot (\mathbb{1}_M \otimes I_N)^\top \cdot \mathcal{K}^{(M,N)} \cdot \frac{\exp\left(-\frac{c(\mathbf{f}^{(\ell)}, \mathbf{g}^{(\ell-1)})}{\varepsilon}\right)}{\varepsilon} \\
&\quad \cdot \operatorname{diag} \operatorname{vec} \mathcal{Q}(\mathbf{f}^{(\ell)}, \mathbf{g}^{(\ell-1)}) \cdot [(\mathbb{1}_N \otimes I_M) \cdot d \mathbf{f}^{(\ell)} + (I_N \otimes \mathbb{1}_M) \cdot d \mathbf{g}^{(\ell-1)}] \\
&= d \mathbf{g}^{(\ell-1)} - \operatorname{diag} \operatorname{vec} \frac{1}{\mathcal{Q}^\top(\mathbf{f}^{(\ell)}, \mathbf{g}^{(\ell-1)}) \cdot \mathbb{1}_M} \cdot (\mathbb{1}_M \otimes I_N)^\top \cdot \mathcal{K}^{(M,N)} \\
&\quad \cdot \operatorname{diag} \operatorname{vec} \mathcal{Q}(\mathbf{f}^{(\ell)}, \mathbf{g}^{(\ell-1)}) \cdot [(\mathbb{1}_N \otimes I_M) \cdot d \mathbf{f}^{(\ell)} + (I_N \otimes \mathbb{1}_M) \cdot d \mathbf{g}^{(\ell-1)}].
\end{aligned} \tag{37}$$

Therefore, from Equations (36) and (37), we have obtained the updating equations for the Jacobians with the numerically stable \mathcal{Q} instead of \mathcal{P} .

Update 4.3 (Updating Equations for Jacobians of Log-Stabilized Sinkhorn).

$$\begin{aligned}
D \mathbf{f}^{(\ell)}(\mathbf{a}) &= D \mathbf{f}^{(\ell-1)}(\mathbf{a}) + \varepsilon \operatorname{diag} \frac{1}{\mathbf{a}} - \operatorname{diag} \operatorname{vec} \frac{1}{\mathcal{Q}(\mathbf{f}^{(\ell-1)}, \mathbf{g}^{(\ell-1)}) \cdot \mathbb{1}_N} \cdot (\mathbb{1}_N \otimes I_M)^\top \\
&\quad \cdot \operatorname{diag} \operatorname{vec} \mathcal{Q}(\mathbf{f}^{(\ell-1)}, \mathbf{g}^{(\ell-1)}) \cdot [(\mathbb{1}_N \otimes I_M) \cdot D \mathbf{f}^{(\ell-1)}(\mathbf{a}) + (I_N \otimes \mathbb{1}_M) \cdot D \mathbf{g}^{(\ell-1)}(\mathbf{a})],
\end{aligned} \tag{38}$$

$$\begin{aligned}
D \mathbf{g}^{(\ell)}(\mathbf{a}) &= D \mathbf{g}^{(\ell-1)}(\mathbf{a}) - \operatorname{diag} \operatorname{vec} \frac{1}{\mathcal{Q}^\top(\mathbf{f}^{(\ell)}, \mathbf{g}^{(\ell-1)}) \cdot \mathbb{1}_M} \cdot (\mathbb{1}_M \otimes I_N)^\top \cdot \mathcal{K}^{(M,N)} \\
&\quad \cdot \operatorname{diag} \operatorname{vec} \mathcal{Q}(\mathbf{f}^{(\ell)}, \mathbf{g}^{(\ell-1)}) \cdot [(\mathbb{1}_N \otimes I_M) \cdot D \mathbf{f}^{(\ell)}(\mathbf{a}) + (I_N \otimes \mathbb{1}_M) \cdot D \mathbf{g}^{(\ell-1)}(\mathbf{a})].
\end{aligned} \tag{39}$$

Therefore, from Update 4.3, we could iteratively obtain $D \mathbf{f}^{(L)}(\mathbf{a})$ and $D \mathbf{g}^{(L)}(\mathbf{a})$. And thus obtain $D \mathbf{P}(\mathbf{a})$ from Equation (34). Then with the Jacobian in Equation (25), we finally have the Jacobian of the loss with respect to argument \mathbf{a} , and its transpose would be the gradient.

Algorithm 4.3 Log-Stabilized Sinkhorn with Gradient**Input:** $\mathbf{a} \in \Sigma_M$, $\mathbf{b} \in \Sigma_N$, $\mathbf{C} \in \mathbb{R}^{M \times N}$, $\varepsilon > 0$.**Initialize:** $\mathbf{f} = \mathbf{0}_M$, $\mathbf{g} = \mathbf{0}_N$, $\mathbf{J}_\mathbf{f} = \mathbf{0}_{M \times M}$, $\mathbf{J}_\mathbf{g} = \mathbf{0}_{N \times M}$.

```

1:  $\mathbf{K} = \text{Commutation}(M, N)$  # commutation matrix K
2: while Not Convergence do
3:   # update f and Jf
4:    $\mathbf{R} = \mathbf{C} - \mathbf{f} \cdot \mathbf{1}_N^\top - \mathbf{1}_M \cdot \mathbf{g}^\top$ 
5:    $c = \min \mathbf{R}$ 
6:    $\mathbf{Q} = \exp\left(-\frac{\mathbf{R}-c}{\varepsilon}\right)$ 
7:    $\mathbf{J} = (\mathbf{1}_N \otimes I_M) \cdot \mathbf{J}_\mathbf{f} + (I_N \otimes \mathbf{1}_M) \cdot \mathbf{J}_\mathbf{g}$ 
8:    $\mathbf{J}_\mathbf{f} = \mathbf{J}_\mathbf{f} + \varepsilon \text{diag} \frac{1}{\mathbf{a}} - \text{diag} \text{vec} \frac{1}{\mathbf{Q} \cdot \mathbf{1}_N} \cdot (\mathbf{1}_N \otimes I_M)^\top \cdot \text{diag} \text{vec} \mathbf{Q} \cdot \mathbf{J}$ 
9:    $\mathbf{f} = \mathbf{f} + \varepsilon \log \mathbf{a} + c - \varepsilon \log [\mathbf{Q} \cdot \mathbf{1}_N]$ 
10:  # update g and Jg
11:   $\mathbf{R} = \mathbf{C} - \mathbf{f} \cdot \mathbf{1}_N^\top - \mathbf{1}_M \cdot \mathbf{g}^\top$ 
12:   $c = \min \mathbf{R}$ 
13:   $\mathbf{Q} = \exp\left(-\frac{\mathbf{R}-c}{\varepsilon}\right)$ 
14:   $\mathbf{J} = (\mathbf{1}_N \otimes I_M) \cdot \mathbf{J}_\mathbf{f} + (I_N \otimes \mathbf{1}_M) \cdot \mathbf{J}_\mathbf{g}$ 
15:   $\mathbf{J}_\mathbf{g} = \mathbf{J}_\mathbf{g} - \text{diag} \text{vec} \frac{1}{\mathbf{Q}^\top \cdot \mathbf{1}_M} \cdot (\mathbf{1}_M \otimes I_N)^\top \cdot \mathbf{K} \cdot \text{diag} \text{vec} \mathbf{Q} \cdot \mathbf{J}$ 
16:   $\mathbf{g} = \mathbf{g} + \varepsilon \log \mathbf{b} + c - \varepsilon \log [\mathbf{Q}^\top \cdot \mathbf{1}_M]$ 
17: end while
18:  $\mathbf{P} = \exp\left(-\frac{\mathbf{C}-\mathbf{f} \cdot \mathbf{1}_N^\top - \mathbf{1}_M \cdot \mathbf{g}^\top}{\varepsilon}\right)$  # optimal coupling
19:  $\mathbf{J}_\mathbf{P} = \mathbf{1}_{MN}^\top \cdot \text{diag} \text{vec} (\mathbf{C} + \varepsilon \log \mathbf{P})$ 
20:  $\mathbf{J}_\mathbf{a} = \frac{1}{\varepsilon} \text{diag} \text{vec} \mathbf{P} \cdot [(\mathbf{1}_N \otimes I_M) \cdot \mathbf{J}_\mathbf{f} + (I_N \otimes \mathbf{1}_M) \cdot \mathbf{J}_\mathbf{g}]$ 
21:  $\nabla_\mathbf{a} = (\mathbf{J}_\mathbf{P} \cdot \mathbf{J}_\mathbf{a})^\top$ 
Output:  $\mathbf{P}, \nabla_\mathbf{a}$ .
```

5 Wasserstein Barycenter

As is often in the case in practice, one wants to compute the “mean” or “center” given a sequence of points. This is useful when we want to find a unknown center as an estimate to the group of points (usually data). Suppose we have $(x_s)_{s=1}^S \in \mathcal{X}^S$ in a metric space (\mathcal{X}, d) , where $d(\cdot)$ is a metric defined on \mathcal{X} , then we have the following problem

$$\min_{x \in \mathcal{X}} \sum_s^S \lambda_s d(x, x_s)^p,$$

for a weight vector $\boldsymbol{\lambda} \in \Sigma_S$ and some power p but is often set to 2. This is often called “Fréchet” or “Karcher” mean¹¹.

¹¹Or Riemannian center of mass. See Karcher (2014) for a historical account on the naming.

5.1 Parallel Algorithm

We can then state the Wasserstein Barycenter problem, similar to the idea of “Fréchet mean” as a weighted sum of the individual distances/losses. Consider the loss function in Equation (2), a weight vector $\boldsymbol{\lambda} \in \Sigma_S$, and a sequence of discrete densities $\{\mathbf{a}_s\}_{s=1}^S$, where $\mathbf{a}_s \in \Sigma_{M_s}$ and M_s is the length of \mathbf{a}_s , a Wasserstein barycenter is computed by the following minimization problem

$$\arg \min_{\mathbf{b} \in \Sigma_N} \sum_{s=1}^S \lambda_s L_{\mathbf{C}_s}(\mathbf{a}_s, \mathbf{b}),$$

where the cost matrix $\mathbf{C}_s \in \mathbb{R}^{M_s \times N}$ specifies the cost for transporting between the probability vectors \mathbf{a}_s and \mathbf{b} . Of course, this problem can be approximated by the entropic regularized version by simply swapping the loss function by Equation (3). Thus, we have the entropic regularized Wasserstein barycenter problem

$$\arg \min_{\mathbf{b} \in \Sigma_N} \sum_{s=1}^S \lambda_s L_{\mathbf{C}_s}^\varepsilon(\mathbf{a}_s, \mathbf{b}). \quad (40)$$

\mathbf{b} is therefore the unknown computed barycenter as the “mean” of the atoms $\mathbf{a}_1, \dots, \mathbf{a}_S$. Again, we will arrive at the Sinkhorn-like updates (Benamou et al., 2015; Schmitz et al., 2018):

$$\begin{aligned} \mathbf{u}_s^{(\ell)} &= \frac{\mathbf{a}_s}{\mathbf{K}_s \mathbf{v}_s^{(\ell-1)}}, \\ \mathbf{b}^{(\ell)} &= \Pi_{s=1}^S \left(\mathbf{K}_s^\top \mathbf{u}_s^{(\ell)} \right)^{\lambda_s}, \\ \mathbf{v}_s^{(\ell)} &= \frac{\mathbf{b}^{(\ell)}}{\mathbf{K}_s^\top \mathbf{u}_s^{(\ell)}}, \end{aligned} \quad (41)$$

where $\mathbf{K}_s = \exp\left(-\frac{\mathbf{C}_s}{\varepsilon}\right)$, and $\mathbf{u}_s, \mathbf{v}_s$ are scaling vectors for $s = 1, \dots, S$. Also, $\dot{\cdot}$ and $(\cdot)^{(\cdot)}$ are to be understood as element-wise operations.

In practical applications, however, we usually have $M_1 = \dots = M_S = N$ and $\mathbf{K} = \mathbf{K}_1 = \mathbf{K}_2 = \dots = \mathbf{K}_S$. We can thus rewrite Equation (41) similar to Section 3.2 as the following updating equations

Update 5.1 (Updating Equations for Parallel Wasserstein Barycenter Algorithm).

$$\begin{aligned} \mathbf{U}^{(\ell)} &= \frac{\mathbf{A}}{\mathbf{K} \mathbf{V}^{(\ell-1)}}, \\ \mathbf{b}^{(\ell)} &= \Pi_{row} \left(\mathbf{K}^\top \mathbf{U}^{(\ell)} \right)^{\boldsymbol{\lambda}^\top \otimes \mathbf{1}_N}, \\ \mathbf{V}^{(\ell)} &= \frac{\mathbf{B}^{(\ell)}}{\mathbf{K}^\top \mathbf{U}^{(\ell)}}, \end{aligned} \quad (42)$$

where $\mathbf{B}^{(\ell)} = [\mathbf{b}^{(\ell)}, \dots, \mathbf{b}^{(\ell)}] \in \mathbb{R}^{N \times S}$, and the product operator Π_{row} refers to the row-wise product on the matrix.

The row-wise product will reduce the matrix into a length N vector, and the power operation in the second line refers to the element-wise power operation. The matrix $\boldsymbol{\lambda}^\top \otimes \mathbf{1}_N$ simply populates the length S vector $\boldsymbol{\lambda}$ into an $N \times S$ matrix, which can also be seen as applying each element of the $\boldsymbol{\lambda}$ vector to the base matrix column-wise.

Algorithm 5.1 Wasserstein Barycenter Algorithm (Parallel)

Input: $\mathbf{A} \in \Sigma_{N \times S}$, $\mathbf{C} \in \mathbb{R}^{N \times N}$, $\varepsilon > 0$.

Initialize: $\mathbf{U} = \mathbf{1}_{N \times S}$, $\mathbf{V}_{N \times S} = \mathbf{1}_{N \times S}$, $\mathbf{b} = \mathbf{0}_N$.

- 1: $\mathbf{K} = \exp(-\frac{\mathbf{C}}{\varepsilon})$
- 2: **while** Not Convergence **do**
- 3: $\mathbf{U} \leftarrow \mathbf{A} \odot (\mathbf{K}\mathbf{V})$
- 4: $\mathbf{b} = \Pi_{row}(\mathbf{K}^\top \mathbf{U})^{\boldsymbol{\lambda}^\top \otimes \mathbf{1}_N}$
- 5: $\mathbf{V} \leftarrow \mathbf{b} \odot (\mathbf{K}^\top \mathbf{U})$
- 6: **end while**

Output: \mathbf{b}

Remark. In Line 4 of Algorithm 5.1, the notation Π_s refers to row-wise product reduction as in Equation (42). In Line 5, the numerator \mathbf{b} can be broadcast to be the same numerator of all columns of the denominator matrix as discussed in Section 3.2.

Remark (Convergence Condition). Similar to the Convergence Condition in Equation (), we can check the following condition for convergence. For some $\rho > 0$,

$$\|\mathbf{U}^{(\ell)} \odot (\mathbf{K}\mathbf{V}^{(\ell)}) - \mathbf{A}\|_2 \leq \rho.$$

5.2 Log-stabilized Algorithm

Similar to the discussion of Section 3.3, numerical instability is still an issue for Algorithm 5.1 due the computation of the exponential function and its potential numeric overflow. Therefore, we would also need to stabilize the computation in the log-domain. In order to do so, we will also need to state the algorithms (updating equations) in terms of the dual variables \mathbf{f} and \mathbf{g} .

One issue arises if we try to adapt Section 3.3 to the problem in Equation (42), that is, we can only calculate the value of the function \mathcal{P} for each margin, i.e. each pair of \mathbf{f} and \mathbf{g} , without being able to process the data for all the margins at once as in Section 3.2. Therefore, we need to extend (abuse) the notation in Section 3.3 for the equations in Update 5.1.

Let us define

$$\begin{aligned} \mathbf{F}^{(\ell)} &\equiv [\mathbf{f}_1^{(\ell)}, \dots, \mathbf{f}_S^{(\ell)}] = [\varepsilon \log \mathbf{u}_1^{(\ell)}, \dots, \varepsilon \log \mathbf{u}_S^{(\ell)}] = \varepsilon \log \mathbf{U}^{(\ell)} \in \mathbb{R}^{M \times S}, \\ \mathbf{G}^{(\ell)} &\equiv [\mathbf{g}_1^{(\ell)}, \dots, \mathbf{g}_S^{(\ell)}] = [\varepsilon \log \mathbf{v}_1^{(\ell)}, \dots, \varepsilon \log \mathbf{v}_S^{(\ell)}] = \varepsilon \log \mathbf{V}^{(\ell)} \in \mathbb{R}^{N \times S}. \end{aligned}$$

Since we have Equation (19), then

$$\begin{aligned}
-\varepsilon \log \mathbf{KV}^{(\ell)} &= -\varepsilon \log [\mathbf{Kv}_1^{(\ell)}, \dots, \mathbf{Kv}_S^{(\ell)}] \\
&= -\varepsilon \log [\mathcal{P}(\mathbf{f}_1^{(\ell)}, \mathbf{g}_1^{(\ell)}) \cdot \mathbf{1}_N, \dots, \mathcal{P}(\mathbf{f}_S^{(\ell)}, \mathbf{g}_S^{(\ell)}) \cdot \mathbf{1}_N] + [\mathbf{f}_1^{(\ell)}, \dots, \mathbf{f}_S^{(\ell)}] \\
&= -\varepsilon \log \left\{ [\mathcal{P}(\mathbf{f}_1^{(\ell)}, \mathbf{g}_1^{(\ell)}), \dots, \mathcal{P}(\mathbf{f}_S^{(\ell)}, \mathbf{g}_S^{(\ell)})] \cdot (I_S \otimes \mathbf{1}_N) \right\} + \mathbf{F}^{(\ell)}.
\end{aligned}$$

By abusing the notation, we could also have

$$\begin{aligned}
\mathcal{P}(\mathbf{F}, \mathbf{G}) &\equiv [\mathcal{P}(\mathbf{f}_1, \mathbf{g}_1), \dots, \mathcal{P}(\mathbf{f}_S, \mathbf{g}_S)] \\
&= \left[\exp \left(-\frac{\mathbf{C} - \mathbf{f}_1 \mathbf{1}_N^\top - \mathbf{1}_M \mathbf{g}_1^\top}{\varepsilon} \right), \dots, \exp \left(-\frac{\mathbf{C} - \mathbf{f}_S \mathbf{1}_N^\top - \mathbf{1}_M \mathbf{g}_S^\top}{\varepsilon} \right) \right] \\
&= \exp \left(-\frac{1}{\varepsilon} [\mathbf{C} - \mathbf{f}_1 \mathbf{1}_N^\top - \mathbf{1}_M \mathbf{g}_1^\top, \dots, \mathbf{C} - \mathbf{f}_S \mathbf{1}_N^\top - \mathbf{1}_M \mathbf{g}_S^\top] \right) \\
&= \exp \left\{ -\frac{1}{\varepsilon} [\mathbf{1}_S^\top \otimes \mathbf{C} - \mathbf{F} \cdot (I_S \otimes \mathbf{1}_N^\top) - \mathbf{1}_M \otimes (\text{vec } \mathbf{G})^\top] \right\}.
\end{aligned} \tag{43}$$

Note that in the previous display, we have avoided matrix concatenation and transform the expression into basic matrix computations (multiplication and kronecker product). This will also make our life easier when we derive the Jacobians later in Section 6. Therefore, we could have the updating equation for

$$\begin{aligned}
\mathbf{F}^{(\ell)} &= \varepsilon \log \mathbf{A} - \varepsilon \log \mathbf{KV}^{(\ell-1)} \\
&= \mathbf{F}^{(\ell-1)} + \varepsilon \log \mathbf{A} - \varepsilon \log [\mathcal{P}(\mathbf{F}^{(\ell-1)}, \mathbf{G}^{(\ell-1)}) \cdot (I_S \otimes \mathbf{1}_N)].
\end{aligned}$$

To get the expression for $\mathbf{G}^{(\ell)}$, we need to follow similar steps. First, define

$$\begin{aligned}
\tilde{\mathcal{P}}(\mathbf{F}, \mathbf{G}) &\equiv [\mathcal{P}^\top(\mathbf{f}_1, \mathbf{g}_1), \dots, \mathcal{P}^\top(\mathbf{f}_S, \mathbf{g}_S)] \\
&= \exp \left\{ -\frac{1}{\varepsilon} [\mathbf{1}_S^\top \otimes \mathbf{C}^\top - \mathbf{1}_N \otimes (\text{vec } \mathbf{F})^\top - \mathbf{G} \cdot (I_S \otimes \mathbf{1}_M^\top)] \right\}.
\end{aligned}$$

Then by Equation (15), we have

$$\begin{aligned}
\mathbf{G}^{(\ell)} &= \varepsilon \log \mathbf{V}^{(\ell)} \\
&= \varepsilon \log \mathbf{B} - \varepsilon \log \mathbf{K}^\top \mathbf{V}^{(\ell)} \\
&= \mathbf{G}^{(\ell-1)} + \varepsilon \log \mathbf{B} - \varepsilon \log [\mathcal{P}^\top(\mathbf{f}_1^{(\ell)}, \mathbf{g}_1^{(\ell-1)}) \cdot \mathbf{1}_M, \dots, \mathcal{P}^\top(\mathbf{f}_S^{(\ell)}, \mathbf{g}_S^{(\ell-1)}) \cdot \mathbf{1}_M] \\
&= \mathbf{G}^{(\ell-1)} + \varepsilon \log \mathbf{B} - \varepsilon \log [\tilde{\mathcal{P}}(\mathbf{F}^{(\ell)}, \mathbf{G}^{(\ell-1)}) \cdot (I_S \otimes \mathbf{1}_M)].
\end{aligned} \tag{44}$$

Putting everything together, we have the following updating equations,

$$\begin{aligned}
\mathbf{F}^{(\ell)} &= \mathbf{F}^{(\ell-1)} + \varepsilon \log \mathbf{A} - \varepsilon \log [\mathcal{P}(\mathbf{F}^{(\ell-1)}, \mathbf{G}^{(\ell-1)}) \cdot (I_S \otimes \mathbb{1}_N)], \\
\mathbf{G}^{(\ell)} &= \mathbf{G}^{(\ell-1)} + \varepsilon \log \mathbf{B} - \varepsilon \log [\tilde{\mathcal{P}}(\mathbf{F}^{(\ell)}, \mathbf{G}^{(\ell-1)}) \cdot (I_S \otimes \mathbb{1}_M)],
\end{aligned} \tag{45}$$

where

$$\begin{aligned}
\mathcal{P}(\mathbf{F}^{(\ell-1)}, \mathbf{G}^{(\ell-1)}) &= \exp \left\{ -\frac{1}{\varepsilon} \left[\mathbb{1}_S^\top \otimes \mathbf{C} - \mathbf{F}^{(\ell-1)} \cdot (I_S \otimes \mathbb{1}_N^\top) - \mathbb{1}_M \otimes (\text{vec } \mathbf{G}^{(\ell-1)})^\top \right] \right\}, \\
\tilde{\mathcal{P}}(\mathbf{F}^{(\ell)}, \mathbf{G}^{(\ell-1)}) &= \exp \left\{ -\frac{1}{\varepsilon} \left[\mathbb{1}_S^\top \otimes \mathbf{C}^\top - \mathbb{1}_N \otimes (\text{vec } \mathbf{F}^{(\ell)})^\top - \mathbf{G}^{(\ell-1)} \cdot (I_S \otimes \mathbb{1}_M^\top) \right] \right\}.
\end{aligned}$$

Again, as discussed in Section 3.3, the above expressions are not necessarily numerically stable. Then, we also need to extend the definitions of functions c and \mathcal{Q} from Equation (21), again using the “log-sum-exp” trick as in Section 3.3,

$$\begin{aligned}
\mathcal{Q}(\mathbf{F}, \mathbf{G}) &= \exp \left\{ -\frac{1}{\varepsilon} \left[\mathbb{1}_S^\top \otimes \mathbf{C} - \mathbf{F} \cdot (I_S \otimes \mathbb{1}_N^\top) - \mathbb{1}_M \otimes (\text{vec } \mathbf{G})^\top - c_1(\mathbf{F}, \mathbf{G}) \right] \right\}, \\
\tilde{\mathcal{Q}}(\mathbf{F}, \mathbf{G}) &= \exp \left\{ -\frac{1}{\varepsilon} \left[\mathbb{1}_S^\top \otimes \mathbf{C}^\top - \mathbb{1}_N \otimes (\text{vec } \mathbf{F})^\top - \mathbf{G} \cdot (I_S \otimes \mathbb{1}_M^\top) - c_2(\mathbf{F}, \mathbf{G}) \right] \right\}, \\
c_1(\mathbf{F}, \mathbf{G}) &= \min \left\{ \mathbb{1}_S^\top \otimes \mathbf{C} - \mathbf{F} \cdot (I_S \otimes \mathbb{1}_N^\top) - \mathbb{1}_M \otimes (\text{vec } \mathbf{G})^\top \right\}, \\
c_2(\mathbf{F}, \mathbf{G}) &= \min \left\{ \mathbb{1}_S^\top \otimes \mathbf{C}^\top - \mathbb{1}_N \otimes (\text{vec } \mathbf{F})^\top - \mathbf{G} \cdot (I_S \otimes \mathbb{1}_M^\top) \right\}.
\end{aligned} \tag{46}$$

All that remains is to derive the updating formula for \mathbf{b} , or $\log \mathbf{b}$ in the log-domain. We then have

$$\begin{aligned}
\log \mathbf{b}^{(\ell)} &= \log \Pi_s (\mathbf{K}^\top \mathbf{u}_s^{(\ell)})^{\lambda_s} \\
&= \Sigma_s \log (\mathbf{K}^\top \mathbf{u}_s^{(\ell)})^{\lambda_s} \\
&= \Sigma_s \lambda_s \log (\mathbf{K}^\top \mathbf{u}_s^{(\ell)}) \\
&= \log (\mathbf{K}^\top \mathbf{U}^{(\ell)}) \cdot \boldsymbol{\lambda} \\
&= \log \left[\tilde{\mathcal{P}}(\mathbf{F}^{(\ell)}, \mathbf{G}^{(\ell-1)}) \cdot (I_S \otimes \mathbb{1}_M) \right] \cdot \boldsymbol{\lambda} - \frac{1}{\varepsilon} \cdot \mathbf{G}^{(\ell-1)} \cdot \boldsymbol{\lambda} \\
&= \log \left[\tilde{\mathcal{Q}}(\mathbf{F}^{(\ell)}, \mathbf{G}^{(\ell-1)}) \cdot (I_S \otimes \mathbb{1}_M) \right] \cdot \boldsymbol{\lambda} - \frac{1}{\varepsilon} \cdot \mathbf{G}^{(\ell-1)} \cdot \boldsymbol{\lambda} - \frac{c_2(\mathbf{F}^{(\ell)}, \mathbf{G}^{(\ell-1)})}{\varepsilon}
\end{aligned}$$

where in the last two lines, we used the formula when deriving Equation (44) and the numerically stable expression from Equation (46). Therefore, we can finally arrive at the following updating equations.

Update 5.2 (Updating Equations for Log-Stabilized Wasserstein Barycenter Algorithm).

$$\begin{aligned}\mathbf{F}^{(\ell)} &= \mathbf{F}^{(\ell-1)} + \varepsilon \log \mathbf{A} + c_1 (\mathbf{F}^{(\ell-1)}, \mathbf{G}^{(\ell-1)}) - \varepsilon \log [\mathcal{Q}(\mathbf{F}^{(\ell-1)}, \mathbf{G}^{(\ell-1)}) \cdot (I_S \otimes \mathbb{1}_N)], \\ \log \mathbf{b}^{(\ell)} &= \log \left[\tilde{\mathcal{Q}}(\mathbf{F}^{(\ell)}, \mathbf{G}^{(\ell-1)}) \cdot (I_S \otimes \mathbb{1}_M) \right] \cdot \boldsymbol{\lambda} - \frac{1}{\varepsilon} \cdot \mathbf{G}^{(\ell-1)} \cdot \boldsymbol{\lambda} - \frac{c_2(\mathbf{F}^{(\ell)}, \mathbf{G}^{(\ell-1)})}{\varepsilon}, \\ \mathbf{G}^{(\ell)} &= \mathbf{G}^{(\ell-1)} + \varepsilon \log \mathbf{B} + c_2 (\mathbf{F}^{(\ell)}, \mathbf{G}^{(\ell-1)}) - \varepsilon \log \left[\tilde{\mathcal{Q}}(\mathbf{F}^{(\ell)}, \mathbf{G}^{(\ell-1)}) \cdot (I_S \otimes \mathbb{1}_M) \right],\end{aligned}$$

where \mathcal{Q} , $\tilde{\mathcal{Q}}$, c_1 , and c_2 are defined in Equation (46).

Also, we can let $M = N$ as is common in applications to further simplify the notation.

Algorithm 5.2 Wasserstein Barycenter Algorithm (Log-Stabilized)

Input: $\mathbf{A} \in \Sigma_{N \times S}$, $\mathbf{C} \in \mathbb{R}^{N \times N}$, $\boldsymbol{\lambda} \in \Sigma_S$, $\varepsilon > 0$.

Initialize: $\mathbf{F} = \mathbf{0}_{N \times S}$, $\mathbf{G}_{N \times S} = \mathbb{1}_{N \times S}$, $\log \mathbf{b} = \mathbf{0}_N$.

```

1: while Not Convergence do
2:   # update F
3:    $\mathbf{R} = \mathbb{1}_S^\top \otimes \mathbf{C} - \mathbf{F} \cdot (I_S \otimes \mathbb{1}_N^\top) - \mathbb{1}_N \otimes (\text{vec } \mathbf{G})^\top$ 
4:    $c = \min \mathbf{R}$ 
5:    $\mathbf{Q} = \exp \left( -\frac{\mathbf{R} - c}{\varepsilon} \right)$ 
6:    $\mathbf{F} = \mathbf{F} + \varepsilon \cdot \log \mathbf{A} + c - \varepsilon \log [\mathbf{Q} \cdot (I_S \otimes \mathbb{1}_N)]$ 
7:
8:    $\mathbf{R} = \mathbb{1}_S^\top \otimes \mathbf{C}^\top - \mathbb{1}_N \otimes (\text{vec } \mathbf{F})^\top - \mathbf{G} \cdot (I_S \otimes \mathbb{1}_N^\top)$ 
9:    $c = \min \mathbf{R}$ 
10:   $\mathbf{Q} = \exp \left( -\frac{\mathbf{R} - c}{\varepsilon} \right)$ 
11:  # update log b
12:   $\log \mathbf{b} = \log [\mathbf{Q} \cdot (I_S \otimes \mathbb{1}_N)] \cdot \boldsymbol{\lambda} - \frac{1}{\varepsilon} \cdot \mathbf{G} \cdot \boldsymbol{\lambda} - \frac{c}{\varepsilon}$ 
13:  # update G
14:   $\log \mathbf{B} = \mathbb{1}_S^\top \otimes \log \mathbf{b}$ 
15:   $\mathbf{G} = \mathbf{G} + \varepsilon \cdot \log \mathbf{B} + c - \varepsilon \log [\mathbf{Q} \cdot (I_S \otimes \mathbb{1}_N)]$ 
16: end while
```

Output: $\mathbf{b} = \exp \log \mathbf{b}$

Remark (Convergence Condition). *Note that for the algorithm to converge, $\mathbf{U}^{(\ell)} \odot (\mathbf{KV}^{(\ell)})$ must approximate \mathbf{A} well. This leads us to the following Convergence Condition in the log-domain, for some $\rho > 0$,*

$$\left\| \left(\mathcal{Q}(\mathbf{F}^{(\ell)}, \mathbf{G}^{(\ell)}) - \frac{c_1(\mathbf{F}^{(\ell)}, \mathbf{G}^{(\ell)})}{\varepsilon} \right) \cdot (I_S \otimes \mathbb{1}_N) - \mathbf{A} \right\|_2 \leq \rho.$$

To see this, let us take the logarithm first

$$\begin{aligned}
\log [\mathbf{U}^{(\ell)} \odot (\mathbf{KV}^{(\ell)})] &= \log \mathbf{U}^{(\ell)} + \log (\mathbf{KV}^{(\ell)}) \\
&= \frac{1}{\varepsilon} \mathbf{F}^{(\ell)} + \log [\mathcal{P}(\mathbf{F}^{(\ell)}, \mathbf{G}^{(\ell)}) \cdot (I_S \otimes \mathbb{1}_N)] - \frac{1}{\varepsilon} \mathbf{F}^{(\ell)} \\
&= \log [\mathcal{P}(\mathbf{F}^{(\ell)}, \mathbf{G}^{(\ell)}) \cdot (I_S \otimes \mathbb{1}_N)] \\
&= \log \left[\left(\mathcal{Q}(\mathbf{F}^{(\ell)}, \mathbf{G}^{(\ell)}) - \frac{c_1(\mathbf{F}^{(\ell)}, \mathbf{G}^{(\ell)})}{\varepsilon} \right) \cdot (I_S \otimes \mathbb{1}_N) \right].
\end{aligned}$$

6 Wasserstein Barycenter with Gradients

7 Wasserstein Dictionary Learning

7.1 Numeric Optimization Methods

Before moving on to discuss the Wasserstein Dictionary Learning algorithm, let me briefly review some gradient-based optimization algorithms commonly employed in machine learning applications.

The most famous algorithm for solving an unconstrained optimization problem is probably gradient descent method. This celebrated method can be attributed to Cauchy (1847), even though he didn't believe it will actually find the minimum nor did he show its convergence (Lemaréchal, 2012). Hadamard (1908) also proposed this idea independently, following the work of Hilbert (1900), in the context of solving differential equations, as an alternative to Rayleigh-Ritz method (Rayleigh, 1896; Ritz, 1909; Courant, 1943).

Since gradient descent method only leverages gradient information, it is considered as first-order optimization method. Though there exist other optimization methods in numeric optimization, such as second-order methods¹² or global optimization methods¹³, it is the stochastic gradient descent (SGD) method that is *de facto* the only¹⁴ method being used in modern machine learning applications. Its history can be traced back to the stochastic approximation method (Robbins and Monro, 1951), its update (Kiefer and Wolfowitz, 1952), and was used in the perceptron model (Rosenblatt, 1958).

What is now commonly employed is the mini-batch SGD, where a batch of sample are evaluated and their gradients averaged as the current iteration's estimate for the gradient of

¹²The most famous example being BFGS algorithm, named after the authors of Broyden (1970), Fletcher (1970), Goldfarb (1970), and Shanno (1970).

¹³Usually using meta-heuristics like Differential Evolution (Storn and Price, 1997).

¹⁴For example, one of the most popular neural network library, PyTorch (Paszke et al., 2017), only includes first-order optimizers, with the only exception of L-BFGS (Liu and Nocedal, 1989) being a second-order optimizer, an algorithm that is based on the renowned BFGS algorithm. But it is not common to consider L-BFGS algorithm in very large neural network models, as it is not as efficient as first-order methods. The most commonly used first-order method is Adam optimizer (Kingma and Ba, 2015); in fact, all first-order methods in PyTorch are either derived from Adam, or its predecessors, simply because of its empirical performance.

the objective function. Then this average gradient is used in some first-order optimization algorithm updating rule to update the parameters for the model. The procedure repeats until convergence.

Here, I list the parameter updating rules for several commonly employed optimizers¹⁵, which will be implemented in the *wig* package.

Definition 5 (Dekel et al., 2012, Mini-Batch Gradient). *For some batch size B and objective function to be minimized $f(\mathbf{x}; \boldsymbol{\theta}_t)$ with data \mathbf{x} and parameters $\boldsymbol{\theta}_t$ at timestamp t , define the mini-batch gradient as*

$$\mathbf{g}_t = \frac{1}{B} \sum_{b=1}^B \nabla_{\boldsymbol{\theta}} f(\mathbf{x}_b; \boldsymbol{\theta}_t), \quad (47)$$

where $\{\mathbf{x}_b\}_{b=1}^B$ denote the sequence of data.

Update 7.1 (Stochastic Gradient Descent). *Given a function to be minimized $f(\mathbf{x}; \boldsymbol{\theta}_t)$ and a mini-batched gradient \mathbf{g}_{t-1} with current parameters $\boldsymbol{\theta}_{t-1}$ and learning rate η ,*

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta \cdot \mathbf{g}_{t-1}.$$

Update 7.2 (Kingma and Ba, 2015, Adam). *Given a function to be minimized $f(\mathbf{x}; \boldsymbol{\theta}_t)$ and a mini-batched gradient \mathbf{g}_{t-1} with current parameters $\boldsymbol{\theta}_{t-1}$ and learning rate η ,*

$$\begin{aligned} \mathbf{m}_t &= \beta_1 \cdot \mathbf{m}_{t-1} + (1 - \beta_1) \cdot \mathbf{g}_{t-1}, \\ \mathbf{v}_t &= \beta_2 \cdot \mathbf{v}_{t-1} + (1 - \beta_2) \cdot \mathbf{g}_{t-1}^2, \\ \hat{\mathbf{m}}_t &= \frac{\mathbf{m}_t}{1 - \beta_1^t}, \\ \hat{\mathbf{v}}_t &= \frac{\mathbf{v}_t}{1 - \beta_2^t}, \\ \boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} - \eta \cdot \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon}, \end{aligned}$$

where $(\cdot)^2$ denote element-wise power of 2.

Update 7.3 (Loshchilov and Hutter, 2019, AdamW). *Given a function to be minimized $f(\mathbf{x}; \boldsymbol{\theta}_t)$ and a mini-batched gradient \mathbf{g}_{t-1} with current parameters $\boldsymbol{\theta}_{t-1}$ decay parameter τ ,*

¹⁵Interested readers should refer to Nocedal and Wright (2006) for a general treatment on the field of numeric optimization.

and learning rate η ,

$$\begin{aligned}\hat{\mathbf{g}}_t &= \mathbf{g}_{t-1} + \tau \cdot \boldsymbol{\theta}_{t-1}, \\ \mathbf{m}_t &= \beta_1 \cdot \mathbf{m}_{t-1} + (1 - \beta_1) \cdot \hat{\mathbf{g}}_t, \\ \mathbf{v}_t &= \beta_2 \cdot \mathbf{v}_{t-1} + (1 - \beta_2) \cdot \hat{\mathbf{g}}_t^2, \\ \hat{\mathbf{m}}_t &= \frac{\mathbf{m}_t}{1 - \beta_1^t}, \\ \hat{\mathbf{v}}_t &= \frac{\mathbf{v}_t}{1 - \beta_2^t}, \\ \boldsymbol{\theta}_t &= \boldsymbol{\theta}_t - \eta_t \cdot \left(\eta \cdot \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon} + \tau \cdot \boldsymbol{\theta}_{t-1} \right),\end{aligned}$$

where η_t is some schedule multiplier that can be fixed or decreasing to change the step size dynamically.

7.2 Wasserstein Dictionary Learning

In the Wasserstein barycenter problem, we have the data as sequence of probability vectors $\{\mathbf{a}_s\}_{s=1}^S$, and would like to obtain a “centroid” of the data \mathbf{b} . Here, \mathbf{a}_s ’s are known as data, but \mathbf{b} is unknown and thus need to be estimated or computed by optimization. The so-called Wasserstein Dictionary Learning (Schmitz et al., 2018) is the almost converse problem: suppose we have the observed \mathbf{b} ’s, how to find the latent and thus unknown \mathbf{a}_s such that the barycenter from which would reconstruct the data \mathbf{b} . In this case, it is the data \mathbf{b} that is known, but not \mathbf{a}_s ’s.

The Wasserstein Dictionary Learning can be considered broadly as one of the (nonlinear) topic models (Blei and Lafferty, 2009), the most famous of which would probably be Latent Dirichlet Allocation model (Blei, Ng, and Jordan, 2003). As is the case with all other topic models, we have a sequence of data and we need to discover the common “factors” or “topics” among them, either in Natural Language Processing (Xu et al., 2018) or Image Processing (Schmitz et al., 2018) settings.

To state the problem, let $\mathbf{B} \in \mathbb{R}_+^{N \times M}$ be the matrix containing the data, i.e. M probability vectors each of size N , and thus for each column \mathbf{b}_m we have $\mathbf{b}_m \in \Sigma_N$ for $m = 1, \dots, M$. For the latent topic¹⁶ matrix $\mathbf{A} \in \mathbb{R}_+^{N \times S}$ where each column $\mathbf{a}_s \in \Sigma_N$, and the weight matrix $\mathbf{W} \in \mathbb{R}_+^{S \times M}$ where each column $\mathbf{w}_m \in \Sigma_S$, we have

¹⁶Sometimes also called atom or factor.

$$\begin{aligned}
& \min_{\mathbf{A}, \mathbf{W}} \sum_{m=1}^M \mathcal{L}(\hat{\mathbf{b}}_m, \mathbf{b}_m), \\
& \text{s.t. } \hat{\mathbf{b}}_m = \arg \min_{\mathbf{b} \in \Sigma_N} \sum_{s=1}^S \mathbf{W}_{sm} L_C^\varepsilon(\mathbf{a}_s, \mathbf{b}), \\
& \mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_S] \in \mathbb{R}_+^{N \times S}, \text{ where } \mathbf{a}_s \in \Sigma_N, \\
& \mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_M] \in \mathbb{R}_+^{S \times M}, \text{ where } \mathbf{w}_m \in \Sigma_S.
\end{aligned} \tag{48}$$

Essentially, we are considering the $\hat{\mathbf{b}}_m$, the computed barycenter from the current \mathbf{A} , as the reconstruction for the true \mathbf{b}_m that we observe in the data. Then we will need to optimize \mathbf{A} (and also implicitly \mathbf{W}) such that the reconstructions approximate the original data well. This problem, however, is not guaranteed to be convex (Schmitz et al., 2018), whether jointly in \mathbf{A} and \mathbf{W} or for each separately, unlike the original Sinkhorn problem discussed in Section 2. Thus, we aim to solve this problem by a gradient descent approach to find a local minimum, as is the usual case in machine learning literature. In order to do so, we will take the gradient of the loss with respect to the parameters (here \mathbf{A} and \mathbf{W}), and then optimize them by some learning rate, and then repeat until some stopping criterion. The gradient computation can be easily carried out by any modern Automatic Differentiation system¹⁷, but in this note, I manually derive the gradients and Jacobians of all algorithms (in Sections 4 and 6) to achieve memory efficient and faster code.

Note, however, we cannot directly optimize the parameters from Equation (48), since the problem is a constrained optimization problem with two constraints on the \mathbf{a}_s and \mathbf{w}_m being in their respective simplices. Therefore, we need to convert this constrained optimization problem into an unconstrained one (Schmitz et al., 2018; Xu et al., 2018): that is, to let $\mathbf{a}_s = \text{softmax}(\boldsymbol{\alpha}_s)$ and $\mathbf{w}_m = \text{softmax}(\boldsymbol{\lambda}_s)$, as the softmax function will ensure its output to be a probability vector, i.e. $\mathbf{a}_s \in \Sigma_N$ and $\mathbf{w}_m \in \Sigma_S$. Thus, we have transformed the problem as an optimization problem with parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$. Now we need to define the softmax function.

Definition 6 (Softmax function). *Let $\mathbf{x} \in \mathbb{R}^N$, and the softmax function is*

$$\text{softmax}(\mathbf{x}_i) = \frac{\exp \mathbf{x}_i}{\sum_j \exp \mathbf{x}_j}.$$

¹⁷They are usually embedded within neural network libraries, e.g. PyTorch (Paszke et al., 2017), TensorFlow (Abadi et al., 2016), etc., but can also be found as stand-alone libraries, e.g. autodiff in C++ (Leal, 2018) or Zygote in Julia (Innes, 2019). Unlike Symbolic or Numeric Differentiation, Automatic Differentiation (AD) essentially relies on chain's rule and could provide accurate gradient computation. There are usually two modes for AD systems: Forward mode or Reverse mode, and each has their own advantages. To see this, let a function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$, when $n \ll m$, then Forward mode is more efficient; if, however, $n \gg m$, then Reverse mode is more efficient. Neural Network libraries are usually implemented in Reverse mode, since there is usually a scalar loss but with many parameters of the neural network model. Interested readers could refer to Nocedal and Wright (2006, Chapter 8).

Then the softmax function for a vector \mathbf{x} is

$$\text{softmax}(\mathbf{x}) = \frac{\exp \mathbf{x}}{\mathbf{1}_N^\top \cdot \exp \mathbf{x}} = \frac{\exp \mathbf{x}}{\mathbf{1}_N^\top \cdot \exp \mathbf{x} \cdot \mathbf{1}_N}.$$

Then for a matrix $\mathbf{X} \in \mathbb{R}^{N \times S}$, we have softmax for a matrix

$$\begin{aligned} \text{softmax}(\mathbf{X}) &= [\text{softmax}(\mathbf{x}_1), \dots, \text{softmax}(\mathbf{x}_S)] \\ &= \left[\frac{\exp \mathbf{x}_1}{\mathbf{1}_N^\top \cdot \exp \mathbf{x}_1 \cdot \mathbf{1}_N}, \dots, \frac{\exp \mathbf{x}_S}{\mathbf{1}_N^\top \cdot \exp \mathbf{x}_S \cdot \mathbf{1}_N} \right], \\ &= \frac{\exp \mathbf{X}}{[\mathbf{1}_N^\top \cdot \exp \mathbf{x}_1, \dots, \mathbf{1}_N^\top \cdot \exp \mathbf{x}_S] \otimes \mathbf{1}_N} \\ &= \frac{\exp \mathbf{X}}{(\mathbf{1}_N^\top \cdot \exp \mathbf{X}) \otimes \mathbf{1}_N}. \end{aligned}$$

The softmax of a matrix is equivalent to softmax of its column vectors' separately.

Therefore, we can enforce the constraints $\mathbf{a}_s \in \Sigma_N$ and $\mathbf{w}_m \in \Sigma_S$ by the following change of variables

$$\mathbf{A} = \text{softmax}(\boldsymbol{\alpha}), \quad \text{and} \quad \mathbf{W} = \text{softmax}(\boldsymbol{\lambda}), \quad (49)$$

where $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_S] \in \mathbb{R}^{N \times S}$, and $\boldsymbol{\lambda} = [\boldsymbol{\lambda}_1, \dots, \boldsymbol{\lambda}_M] \in \mathbb{R}^{S \times M}$. Let us slightly abuse the notation to denote the vector of losses across the topics

$$L_{\mathbf{C}}^\varepsilon(\mathbf{A}, \mathbf{b}) = [L_{\mathbf{C}}^\varepsilon(\mathbf{a}_1, \mathbf{b}), \dots, L_{\mathbf{C}}^\varepsilon(\mathbf{a}_S, \mathbf{b})]^\top \in \mathbb{R}^N.$$

Note that $L_{\mathbf{C}}^\varepsilon(\mathbf{A}, \mathbf{b})$ is essentially the same as $L_{\mathbf{C}}^\varepsilon(\mathbf{A}, \mathbf{B})$, where $\mathbf{B} = [\mathbf{b}, \dots, \mathbf{b}] \in \mathbb{R}^{N \times S}$ as the ‘‘many-to-one’’ transport problem in Section 3.2. Therefore, we can rewrite the problem in Equation (48) as an unconstrained one with respect to parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$

$$\begin{aligned} \min_{\boldsymbol{\alpha}, \boldsymbol{\lambda}} \quad & \sum_{m=1}^M \mathcal{L}(\hat{\mathbf{b}}_m, \mathbf{b}_m), \\ \text{s.t.} \quad & \hat{\mathbf{b}}_m = \arg \min_{\mathbf{b} \in \Sigma_N} \mathbf{w}_m^\top \cdot L_{\mathbf{C}}^\varepsilon(\mathbf{A}, \mathbf{b}), \\ & \mathbf{A} = \text{softmax}(\boldsymbol{\alpha}), \\ & \mathbf{w}_m = \text{softmax}(\boldsymbol{\lambda}_m). \end{aligned} \quad (50)$$

Essentially, we will take some random initialized parameters \mathbf{A} and \mathbf{W} , calculate their optimal barycenter, calculate the loss between the barycenter and the original data, take the gradients of the loss with respect to the parameters, and then update the parameters based on the gradients. The computation of the optimal barycenter is based on the Wasserstein barycenter problem, detailed in Section 5, and the parameter-updating routine can be chosen to be any optimization technique, for example Schmitz et al. (2018) chooses L-BFGS (Liu and Nocedal, 1989), and Xie (2020b) uses Adam (Kingma and Ba, 2015) optimizer.

8 Wasserstein Index Generation Model

Xie (2020b)

9 *wig* Package in R for a WIG Implementation

Xie (2020a)

References

- Abadi, Martin et al. (2016). “TensorFlow: A System for Large-Scale Machine Learning.” In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283.
- Alaya, Mokhtar Z. et al. (2019). “Screening Sinkhorn Algorithm for Regularized Optimal Transport.”
- Altschuler, Jason, Jonathan Weed, and Philippe Rigollet (2017). “Near-Linear Time Approximation Algorithms for Optimal Transport via Sinkhorn Iteration.” In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., pp. 1964–1974.
- Bacharach, Michael (1970). *Biproportional Matrices and Input-Output Change*. CUP Archive. 192 pp.
- Benamou, Jean-David et al. (2015). “Iterative Bregman Projections for Regularized Transportation Problems.” In: *SIAM Journal on Scientific Computing* 37.2, A1111–A1138. DOI: 10.1137/141000439.
- Blei, David M. and John D. Lafferty (2009). “Topic Models.” In: *Text Mining*. Chapman and Hall/CRC.
- Blei, David M., Andrew Y. Ng, and Michael I. Jordan (2003). “Latent Dirichlet Allocation.” In: *Journal of Machine Learning Research* 3 (Jan), pp. 993–1022.
- Broyden, C. G. (1970). “The Convergence of a Class of Double-Rank Minimization Algorithms 1. General Considerations.” In: *IMA Journal of Applied Mathematics* 6.1, pp. 76–90. DOI: 10.1093/imamat/6.1.76.
- Cauchy, Augustin (1847). “Méthode Générale Pour La Résolution Des Systemes d’équations Simultanées.” In: *Comp. Rend. Sci. Paris* 25.1847, pp. 536–538.
- Courant, R. (1943). “Variational Methods for the Solution of Problems of Equilibrium and Vibrations.” In: *Bulletin of the American Mathematical Society* 49.1, pp. 1–23. DOI: 10.1090/S0002-9904-1943-07818-4.
- Cuturi, Marco (2013). “Sinkhorn Distances: Lightspeed Computation of Optimal Transport.” In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges et al. Curran Associates, Inc., pp. 2292–2300.
- Dekel, Ofer et al. (2012). “Optimal Distributed Online Prediction Using Mini-Batches.” In: *Journal of Machine Learning Research* 13.1.
- Fletcher, R. (1970). “A New Approach to Variable Metric Algorithms.” In: *The Computer Journal* 13.3, pp. 317–322. DOI: 10.1093/comjnl/13.3.317.
- Galichon, Alfred (2016). *Optimal Transport Methods in Economics*. Princeton University Press. 185 pp. DOI: 10.23943/princeton/9780691172767.001.0001.
- Goldfarb, Donald (1970). “A Family of Variable-Metric Methods Derived by Variational Means.” In: *Mathematics of Computation* 24.109, pp. 23–26. DOI: 10.2307/2004873.
- Hadamard, Jacques (1908). *Mémoire Sur Le Problème d’analyse Relatif à l’équilibre Des Plaques Élastiques Encastrées*. Vol. 33. Imprimerie nationale.

- Hilbert, David (1900). “Sur Le Principe de Dirichlet.” In: *Nouvelles annales de mathématiques: journal des candidats aux écoles polytechnique et normale* 19, pp. 337–344.
- Innes, Michael (2019). *Don’t Unroll Adjoint: Differentiating SSA-form Programs*. DOI: 10.48550/arXiv.1810.07951. URL: <http://arxiv.org/abs/1810.07951> (visited on 07/27/2022). Pre-published.
- Kantorovich, L. V. (1942). “On the Translocation of Masses.” In: *Dokl. Akad. Nauk. USSR (N.S.)* 37, pp. 199–201.
- Karcher, Hermann (2014). *Riemannian Center of Mass and so Called Karcher Mean*. DOI: 10.48550/arXiv.1407.2087. URL: <http://arxiv.org/abs/1407.2087> (visited on 07/28/2024). Pre-published.
- Kiefer, J. and J. Wolfowitz (1952). “Stochastic Estimation of the Maximum of a Regression Function.” In: *The Annals of Mathematical Statistics* 23.3, pp. 462–466. DOI: 10.1214/aoms/1177729392.
- Kingma, Diederick P and Jimmy Ba (2015). “Adam: A Method for Stochastic Optimization.” In: *International Conference on Learning Representations (ICLR)*.
- Knight, P. (2008). “The Sinkhorn-Knopp Algorithm: Convergence and Applications.” In: *SIAM Journal on Matrix Analysis and Applications* 30.1, pp. 261–275. DOI: 10.1137/060659624.
- Leal, Allan M. M. (2018). *Autodiff, a Modern, Fast and Expressive C++ Library for Automatic Differentiation*. <https://autodiff.github.io>.
- Lemaréchal, Claude (2012). “Cauchy and the Gradient Method.” In: *Doc Math Extra* 251.254, p. 10.
- Léonard, Christian (2013). “A Survey of the Schrödinger Problem and Some of Its Connections with Optimal Transport.”
- Liu, Dong C. and Jorge Nocedal (1989). “On the Limited Memory BFGS Method for Large Scale Optimization.” In: *Mathematical Programming* 45.1, pp. 503–528. DOI: 10.1007/BF01589116.
- Loshchilov, Ilya and Frank Hutter (2019). *Decoupled Weight Decay Regularization*. DOI: 10.48550/arXiv.1711.05101. URL: <http://arxiv.org/abs/1711.05101> (visited on 08/01/2024). Pre-published.
- Modin, Klas (2024). *On the Geometry and Dynamical Formulation of the Sinkhorn Algorithm for Optimal Transport*. DOI: 10.48550/arXiv.2309.09089. URL: <http://arxiv.org/abs/2309.09089> (visited on 08/03/2024). Pre-published.
- Monge, Gaspard (1781). “Memoire Sur La Theorie Des Deblais et Des Remblais.” In: *Histoire de l’Academie Royale des Sciences de Paris*.
- Nocedal, Jorge and Stephen Wright (2006). *Numerical Optimization*. 2nd edition. New York: Springer. 686 pp.
- Paszke, Adam et al. (2017). “Automatic Differentiation in PyTorch.” In: *NIPS-W*.
- Peyré, Gabriel and Marco Cuturi (2019). “Computational Optimal Transport: With Applications to Data Science.” In: *Foundations and Trends® in Machine Learning* 11.5-6, pp. 355–607. DOI: 10.1561/22000000073.
- Rayleigh, John William Strutt Baron (1896). *The Theory of Sound*. Macmillan. 526 pp.

- Ritz, Walter (1909). “Über eine neue Methode zur Lösung gewisser Variationsprobleme der mathematischen Physik.” In: *Journal für die reine und angewandte Mathematik* 135, pp. 1–61.
- Robbins, Herbert and Sutton Monro (1951). “A Stochastic Approximation Method.” In: *The Annals of Mathematical Statistics* 22.3, pp. 400–407. DOI: 10.1214/aoms/1177729586.
- Rosenblatt, F. (1958). “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain.” In: *Psychological Review* 65.6, pp. 386–408. DOI: 10.1037/h0042519.
- Santambrogio, Filippo (2015). *Optimal Transport for Applied Mathematicians: Calculus of Variations, PDEs, and Modeling*. Birkhäuser. 376 pp.
- Schmitz, Morgan A. et al. (2018). “Wasserstein Dictionary Learning: Optimal Transport-Based Unsupervised Nonlinear Dictionary Learning.” In: *SIAM Journal on Imaging Sciences* 11.1, pp. 643–678. DOI: 10.1137/17M1140431.
- Schrödinger, Erwin (1931). *Über Die Umkehrung Der Naturgesetze*. Verlag der Akademie der Wissenschaften in Kommission bei Walter De Gruyter u . . .
- Shanno, D. F. (1970). “Conditioning of Quasi-Newton Methods for Function Minimization.” In: *Mathematics of Computation* 24.111, pp. 647–656. DOI: 10.1090/S0025-5718-1970-0274029-X.
- Sinkhorn, Richard (1964). “A Relationship Between Arbitrary Positive Matrices and Doubly Stochastic Matrices.” In: *The Annals of Mathematical Statistics* 35.2, pp. 876–879. DOI: 10.1214/aoms/1177703591.
- Sinkhorn, Richard and Paul Knopp (1967). “Concerning Nonnegative Matrices and Doubly Stochastic Matrices.” In: *Pacific Journal of Mathematics* 21.2, pp. 343–348.
- Storn, Rainer and Kenneth Price (1997). “Differential Evolution – a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces.” In: *Journal of Global Optimization* 11.4, pp. 341–359. DOI: 10.1023/A:1008202821328.
- Villani, Cédric (2003). *Topics in Optimal Transportation*. Vol. 58. Graduate Studies in Mathematics. American Mathematical Society. DOI: 10.1090/gsm/058.
- Villani, Cédric (2008). *Optimal Transport: Old and New*. Springer Science & Business Media. 970 pp.
- Xie, Fangzhou (2020a). “Pruned Wasserstein Index Generation Model and Wigpy Package.” In: *CARMA 2020 - 3rd International Conference on Advanced Research Methods and Analytics*. DOI: 10.4995/CARMA2020.2020.11557.
- Xie, Fangzhou (2020b). “Wasserstein Index Generation Model: Automatic Generation of Time-Series Index with Application to Economic Policy Uncertainty.” In: *Economics Letters* 186, p. 108874. DOI: 10.1016/j.econlet.2019.108874.
- Xu, Hongteng et al. (2018). “Distilled Wasserstein Learning for Word Embedding and Topic Modeling.” In: *Advances in Neural Information Processing Systems* 31. Ed. by S. Bengio et al. Curran Associates, Inc., pp. 1723–1732.

Appendix

Mathematical Notations

Mathematical notations used in this note:

- \mathbf{a} : lower case letters in bold denote vectors
- \mathbf{A} : upper case letters in bold denote matrices
- $\mathbf{1}_n, \mathbf{0}_n$: a vector of ones (or zeros) of length n
- Σ_n : probability simplex $\Sigma_n \equiv \{\mathbf{a} \in \mathbb{R}_+^n : \sum_{i=1}^n \mathbf{a}_i = 1\}$.
- $\langle \cdot, \cdot \rangle$: inner (dot) product of two matrices of same size
- \odot : element-wise (Hadamard) multiplication
- \oslash or $\frac{1}{\mathbf{x}}$: element-wise (Hadamard) division
- $\log(\cdot), \exp(\cdot)$: element-wise logarithm and exponential functions
- $\text{diag}(\mathbf{x})$: create a diagonal matrix $n \times n$ from vector $\mathbf{x} \in \mathbb{R}^n$
- $\text{vec } \mathbf{AB} = (I_m \otimes \mathbf{A}) \text{vec } \mathbf{B} = (\mathbf{B}^\top \otimes I_k) \text{vec } \mathbf{A}$: for $\mathbf{A} : k \times l$ and $\mathbf{B} : l \times m$
- $\mathcal{K}^{(m,n)}$: Commutation matrix such that $\mathcal{K}^{(m,n)} \text{vec } \mathbf{A} = \text{vec } (\mathbf{A}^\top)$ for a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$