Python Optimal Transport: Fused Gromov-Wasserstein Conditional Gradient solver

Cédric Vincent-Cuaz

October 2023

We detail the computations involved in the Conditional Gradient solvers for the Gromov-Wasserstein (GW) and Fused Gromov-Wasserstein (FGW) distances introduced in [1]. These solvers available in the Python Optimal Transport (POT)[1] library [2] circumvent to certain limitations of the original implementations : i) support symmetric and asymmetric matrices incorporating recent theoretical findings from [3]; ii) correct certain typing errors present in [4, Proposition 2] and [1, Algorithm 2].

Contents

1 The Gromov-Wasserstein discrepancy

1.1 Objective function.

In the OT context, a graph $\mathcal{G}$ is modeled as a tuple $(\boldsymbol{C}, \boldsymbol{p})$. Where $\boldsymbol{C} \in \mathbb{R}^{n \times n}$ is any pairwise similarity matrix between the nodes of the graph. And $\boldsymbol{p} \in \Sigma_n$ is a probability vector encoding nodes relative importance within the graph. Considering two graphs $\mathcal{G} = (\boldsymbol{C}, \boldsymbol{p})$ and $\overline{\mathcal{G}} = (\overline{\boldsymbol{C}}, \boldsymbol{q})$ with respectively n and m nodes, the Gromov-Wasserstein discrepancy with inner loss $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ between both graphs reads as [4]:

$$\mathrm{GW}(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{h}, \overline{\boldsymbol{h}}) = \min_{\boldsymbol{T} \in \mathcal{U}(\boldsymbol{p}, \boldsymbol{q})} \mathcal{E}_L^{GW}(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{T}) := \sum_{ijkl} L(C_{ik}, \overline{C}_{jl}) T_{ij} T_{kl} \tag{1}$$

where $\mathcal{U}(\boldsymbol{p}, \boldsymbol{q}) = \left\{ \boldsymbol{T} \in \mathbb{R}_+^{n \times m} | \boldsymbol{T}\boldsymbol{1}_m = \boldsymbol{p} \ , \ \boldsymbol{T}^\top \boldsymbol{1}_n = \boldsymbol{q} \right\}$. The objective function $\mathcal{E}_L^{GW}$ can be conveniently factored using a 4-way tensor $\mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) = \left(L(C_{ik}, \overline{C}_{jl}) \right)_{ijkl}$ such that for any $\boldsymbol{T} \in \mathcal{U}(\boldsymbol{p}, \boldsymbol{q})$,

$$\mathcal{E}_L^{GW}(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{T}) = \langle \mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}, \boldsymbol{T} \rangle_F \tag{2}$$

where $\otimes$ is the tensor-matrix multiplication satisfying $\mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T} = \left(\sum_{kl} L(C_{ik}, \overline{C}_{jl}) T_{kl} \right)_{ij}$. [4] investigates a specific type of loss functions which can be decomposed as follows

$$L(a, b) = f_1(a) + f_2(b) - h_1(a) h_2(b) \tag{3}$$

for any $a, b \in \mathbb{R}$. Two specific inner losses that match this decomposition are

$$L_2(a, b) = (a - b)^2 \implies f_1(a) = a^2, \quad f_2(b) = b^2, \quad h_1(a) = a, \quad h_2(b) = 2b \tag{L2}$$

and

$$L_{KL}(a, b) = a \log \frac{a}{b} - a + b \implies f_1(a) = a \log a - a, \quad f_2(b) = b, \quad h_1(a) = a, \quad h_2(b) = \log b \tag{KL}$$

Proposition 1 in [4] then provides the following factorization for inner losses satisfying equation 3,

$$\mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes T = c_{\boldsymbol{C}, \overline{\boldsymbol{C}}} - h_1(\boldsymbol{C}) \boldsymbol{T} h_2(\overline{\boldsymbol{C}})^\top \tag{4}$$

where $c_{\boldsymbol{C}, \overline{\boldsymbol{C}}} = f_1(\boldsymbol{C}) \boldsymbol{p} \boldsymbol{1}_m^\top + \boldsymbol{1}_n \boldsymbol{q}^\top f_2(\overline{\boldsymbol{C}})^\top$. Then when we consider the quadratic distance we have

$$f_1(a) = a^2, \quad f_2(b) = b^2, \quad h_1(a) = a, \quad h_2(b) = 2b \tag{5}$$

Remark 1. The factorization in equation 4 holds true for any matrices $\boldsymbol{C}$ and $\overline{\boldsymbol{C}}$.
Remark 2. Relations with the POT implementations that can be found in the ot.gromov repository:

[1] Special features of the POT implementations will be highlighted in blue.

- ot.gromov.init_matrix: outputs $c_{\boldsymbol{C},\overline{\boldsymbol{C}}}$, $h_1(\boldsymbol{C})$ and $h_2(\overline{\boldsymbol{C}})$ that correspond to the desired inner loss functions L2 or KL.
- ot.gromov.tensor_product: outputs the tensor product $\mathcal{L}(\boldsymbol{C},\overline{\boldsymbol{C}}) \otimes \boldsymbol{T}$ following equation 4, given $c_{\boldsymbol{C},\overline{\boldsymbol{C}}}$, $h_1(\boldsymbol{C})$, $h_2(\overline{\boldsymbol{C}})$ and $\boldsymbol{T}$.
- ot.gromov.gwloss: outputs the GW loss using the factorization in equation 2, given $c_{\boldsymbol{C},\overline{\boldsymbol{C}}}$, $h_1(\boldsymbol{C})$, $h_2(\overline{\boldsymbol{C}})$ and $\boldsymbol{T}$.

1.2 Gradient computation.

The operations detailed above exactly coincide with those reported in [4]. However, when it comes down to the gradient computation authors considered the case where $\boldsymbol{C}$ and $\overline{\boldsymbol{C}}$ are symmetric. And they also forget a factor 2 in the formula present in [4, Proposition 2]. Therefore we took into considerations these two aspects in the POT implementation.

For any $\boldsymbol{T} \in \mathcal{U}(\boldsymbol{p},\boldsymbol{q})$, we have

$$
\begin{aligned}
\frac{\partial \mathcal{E}_L^{GW}}{\partial T_{pq}}(\boldsymbol{C},\overline{\boldsymbol{C}},\boldsymbol{T}) &= \frac{\partial}{\partial T_{pq}} \sum_{ijkl} \{f_1(C_{ik}) + f_2(\overline{C}_{jl}) - h_1(C_{ik})h_2(\overline{C}_{jl})\} T_{ij} T_{kl} \\
&= \sum_{kl} \{f_1(C_{pk}) + f_2(\overline{C}_{ql}) - h_1(C_{pk})h_2(\overline{C}_{ql})\} T_{kl} + \sum_{ij} \{f_1(C_{ip}) + f_2(\overline{C}_{jq}) - h_1(C_{ip})h_2(\overline{C}_{jq})\} T_{ij} \\
&= \sum_k f_1(C_{pk}) p_k + \sum_l f_2(\overline{C}_{ql}) q_l - \sum_{kl} h_1(C_{pk}) h_2(\overline{C}_{ql}) T_{kl} \\
&+ \sum_i f_1(C_{ip}) p_i + \sum_j f_2(\overline{C}_{jq}) q_j - \sum_{ij} h_1(C_{ip}) h_2(\overline{C}_{jq}) T_{ij}
\end{aligned}
\tag{6}
$$

Notice that following equation 4, we have

$$
\begin{aligned}
\left(\mathcal{L}(\boldsymbol{C},\overline{\boldsymbol{C}}) \otimes \boldsymbol{T}\right)_{ij} &= \sum_{kl} L(C_{ik},\overline{C}_{jl}) T_{kl} \\
&= \sum_{kl} \{f_1(C_{ik}) + f_2(\overline{C}_{jl}) - h_1(C_{ik}) h_2(\overline{C}_{jl})\} T_{kl} \\
&= \sum_k f_1(C_{ik}) p_k + \sum_l f_2(\overline{C}_{jl}) q_l - \sum_{kl} h_1(C_{ik}) h_2(\overline{C}_{jl}) T_{kl}
\end{aligned}
\tag{7}
$$

and that

$$
\begin{aligned}
\left(\mathcal{L}(\boldsymbol{C}^\top,\overline{\boldsymbol{C}}^\top) \otimes \boldsymbol{T}\right)_{ij} &= \sum_{kl} L(C_{ki},\overline{C}_{lj}) T_{kl} \\
&= \sum_{kl} \{f_1(C_{ki}) + f_2(\overline{C}_{lj}) - h_1(C_{ki}) h_2(\overline{C}_{lj})\} T_{kl} \\
&= \sum_k f_1(C_{ki}) p_k + \sum_l f_2(\overline{C}_{lj}) q_l - \sum_{kl} h_1(C_{ki}) h_2(\overline{C}_{lj}) T_{kl}
\end{aligned}
\tag{8}
$$

So we can conclude that

$$
\frac{\partial \mathcal{E}_L^{GW}}{\partial T_{pq}}(\boldsymbol{C},\overline{\boldsymbol{C}},\boldsymbol{T}) = \left(\mathcal{L}(\boldsymbol{C},\overline{\boldsymbol{C}}) \otimes \boldsymbol{T}\right)_{pq} + \left(\mathcal{L}(\boldsymbol{C}^\top,\overline{\boldsymbol{C}}^\top) \otimes \boldsymbol{T}\right)_{pq} \tag{9}
$$

which comes down to

$$
\nabla_{\boldsymbol{T}} \mathcal{E}_L^{GW}(\boldsymbol{C},\overline{\boldsymbol{C}},\boldsymbol{T}) = \mathcal{L}(\boldsymbol{C},\overline{\boldsymbol{C}}) \otimes \boldsymbol{T} + \mathcal{L}(\boldsymbol{C}^\top,\overline{\boldsymbol{C}}^\top) \otimes \boldsymbol{T} \tag{10}
$$

Obviously if $\boldsymbol{C}$ and $\overline{\boldsymbol{C}}$ are symmetric, both terms on the r.h.s are equal i.e

$$
\boldsymbol{C} = \boldsymbol{C}^\top \text{ and } \overline{\boldsymbol{C}} = \overline{\boldsymbol{C}}^\top \implies \nabla_{\boldsymbol{T}} \mathcal{E}_L^{GW}(\boldsymbol{C},\overline{\boldsymbol{C}},\boldsymbol{T}) = 2\mathcal{L}(\boldsymbol{C},\overline{\boldsymbol{C}}) \otimes \boldsymbol{T} \tag{11}
$$

Remark 3. we currently implemented these two settings to not change the API as follows

- ot.gromov.gwggrad: outputs the gradient of the GW loss assuming that $\boldsymbol{C}$ and $\overline{\boldsymbol{C}}$ are symmetric *i.e* according to equation 11, given $c_{\boldsymbol{C},\overline{\boldsymbol{C}}}$, $h_1(\boldsymbol{C})$, $h_2(\overline{\boldsymbol{C}})$ and $\boldsymbol{T}$.
- ot.gromov.gromov_wasserstein: which solves for the GW problem using Conditional Gradient handles both symmetric and asymmetric cases by defining a custom gradient function, which respectively coincide with equation 11 and equation 10.
- The gradient is handled in the same way within different solvers for GW *e.g* ot.gromov.entropic_gromov_wasserstein.

1.3 Exact line-search for Gromov-Wasserstein.

Following [1], POT allows to perform an exact linear-search step within the CG solver for GW. The latter involves two steps:

Step 1. Let us consider the gradient of $\mathcal{E}_L^{GW}(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{T})$ w.r.t $\boldsymbol{T}$ denoted here $\boldsymbol{G}(\boldsymbol{T})$ that satisfies equation 10. We compute the conditional direction

$$\boldsymbol{X} = arg \min_{\boldsymbol{X} \in \mathcal{U}(\boldsymbol{p}, \boldsymbol{q})} \langle \boldsymbol{X}, \boldsymbol{G}(\boldsymbol{T}) \rangle \tag{12}$$

which comes down to a linear OT problem solved using the network flow algorithm implemented in ot.emd.

Step 2. Then we seek for an optimal γ, such that

$$\gamma = arg \min_{\gamma \in [0,1]} f(\gamma) := \langle \mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \{\boldsymbol{T} + \gamma(\boldsymbol{X} - \boldsymbol{T})\}, \boldsymbol{T} + \gamma(\boldsymbol{X} - \boldsymbol{T}) \rangle \tag{13}$$

This objective function can be developed as a second order polynom: $f(\gamma) = a\gamma^2 + b\gamma + c$, where

$$c = f(0) = \langle \mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}, \boldsymbol{T} \rangle \tag{14}$$

Then writing $\mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) = \mathcal{L}$ for better readability, we have

$$a = \langle \mathcal{L} \otimes (\boldsymbol{X} - \boldsymbol{T}), \boldsymbol{X} - \boldsymbol{T} \rangle \tag{15}$$

Let us recall the tensor factorization of equation 4:

$$\mathcal{L} \otimes \boldsymbol{T} = c_{\boldsymbol{C}, \overline{\boldsymbol{C}}} - h_1(\boldsymbol{C}) \boldsymbol{T} h_2(\overline{\boldsymbol{C}})^\top \tag{16}$$

where $c_{\boldsymbol{C}, \overline{\boldsymbol{C}}} = f_1(\boldsymbol{C}) \boldsymbol{p} \mathbf{1}_m^\top + \mathbf{1}_n \boldsymbol{q}^\top f_2(\overline{\boldsymbol{C}})^\top$. Then we have

$$\begin{aligned} a &= \underbrace{\langle c_{\boldsymbol{C}, \overline{\boldsymbol{C}}}, \boldsymbol{X} - \boldsymbol{T} \rangle}_{=0} - \langle h_1(\boldsymbol{C})(\boldsymbol{X} - \boldsymbol{T}) h_2(\overline{\boldsymbol{C}})^\top, \boldsymbol{X} - \boldsymbol{T} \rangle \\ &= -\langle h_1(\boldsymbol{C})(\boldsymbol{X} - \boldsymbol{T}) h_2(\overline{\boldsymbol{C}})^\top, \boldsymbol{X} - \boldsymbol{T} \rangle \end{aligned} \tag{17}$$

knowing that the first term on the r.h.s is 0 because $\boldsymbol{X}$ and $\boldsymbol{T}$ have the same marginals $\boldsymbol{p}$ and $\boldsymbol{q}$.

Finally the coefficient b of the linear term is

$$\begin{aligned} b &= \langle \mathcal{L} \otimes \boldsymbol{T}, \boldsymbol{X} - \boldsymbol{T} \rangle + \langle \mathcal{L} \otimes (\boldsymbol{X} - \boldsymbol{T}), \boldsymbol{T} \rangle \\ &= \langle c_{\boldsymbol{C}, \overline{\boldsymbol{C}}}, \boldsymbol{X} - \boldsymbol{T} \rangle - \langle h_1(\boldsymbol{C}) \boldsymbol{T} h_2(\overline{\boldsymbol{C}})^\top, \boldsymbol{X} - \boldsymbol{T} \rangle \\ &+ \langle c_{\boldsymbol{C}, \overline{\boldsymbol{C}}}, \boldsymbol{T} \rangle - \langle h_1(\boldsymbol{C}) \boldsymbol{X} h_2(\overline{\boldsymbol{C}})^\top, \boldsymbol{T} \rangle - \langle c_{\boldsymbol{C}, \overline{\boldsymbol{C}}}, \boldsymbol{T} \rangle + \langle h_1(\boldsymbol{C}) \boldsymbol{T} h_2(\overline{\boldsymbol{C}})^\top, \boldsymbol{T} \rangle \\ &= -\langle h_1(\boldsymbol{C}) \boldsymbol{T} h_2(\overline{\boldsymbol{C}})^\top, \boldsymbol{X} - \boldsymbol{T} \rangle - \langle h_1(\boldsymbol{C})(\boldsymbol{X} - \boldsymbol{T}) h_2(\overline{\boldsymbol{C}})^\top, \boldsymbol{T} \rangle \end{aligned} \tag{18}$$

as terms depending on the constant $c_{\boldsymbol{C}, \overline{\boldsymbol{C}}}$ cancel each other.

Remark 4 . For now the exact line-search is only available in POT for the L2 loss, so equation L2 leads to

$$a = -2\langle \boldsymbol{C}(\boldsymbol{X} - \boldsymbol{T}) \overline{\boldsymbol{C}}^\top, \boldsymbol{X} - \boldsymbol{T} \rangle \tag{19}$$

and

$$b = -2\langle \boldsymbol{C} \boldsymbol{T} \overline{\boldsymbol{C}}^\top, \boldsymbol{X} - \boldsymbol{T} \rangle - 2\langle \boldsymbol{C}(\boldsymbol{X} - \boldsymbol{T}) \overline{\boldsymbol{C}}^\top, \boldsymbol{T} \rangle \tag{20}$$

References

[1] Vayer Titouan, Nicolas Courty, Romain Tavenard, and Rémi Flamary. Optimal transport for structured data with application on graphs. In *International Conference on Machine Learning*, pages 6275–6284. PMLR, 2019.

[2] Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, et al. Pot: Python optimal transport. *The Journal of Machine Learning Research*, 22(1):3571–3578, 2021.

[3] Samir Chowdhury and Facundo Mémoli. The gromov–wasserstein distance between networks and stable network invariants. *Information and Inference: A Journal of the IMA*, 8(4):757–787, 2019.

[4] Gabriel Peyré, Marco Cuturi, and Justin Solomon. Gromov-wasserstein averaging of kernel and distance matrices. In *International conference on machine learning*, pages 2664–2672. PMLR, 2016.