Progressive Entropic Optimal Transport Solvers

Parnian Kassraie[1*,3], Aram-Alexandre Pooladian[2], Michal Klein[1], James Thornton[1], Jonathan Niles-Weed[2], Marco Cuturi[1]
NeurIPS 2024 · [1]Apple, [2]NYU, [3]ETH, *internship

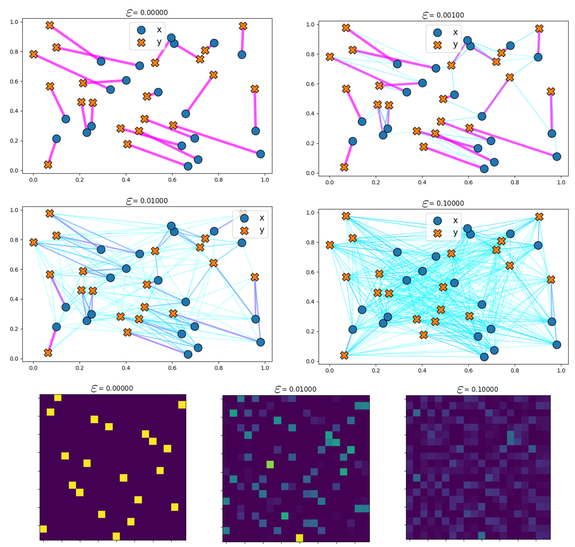
Abstract

- **Context:** Solving optimal transport (OT) with the Sinkhorn algorithm is the default approach when trying to match datasets, It's used in e.g. mini-batch flow matching and many scientific applications. It is also a cheap and promising OT map estimator.
- **Problem:** Sinkhorn is costly, $O(n^2)$ when matching n to n points, and has an important regularization parameter ε, that can be hard to tune, notably when using a Sinkhorn OT map estimator.
- **Goal:** Propose a "divide and conquer" approach to regularized OT that can reduce compute costs and alleviate this selection.
- **Contribution: ProgOT**, a progressive formulation of regularized OT that can be used with any translation invariant cost.

Baseline Approach: Sinkhorn

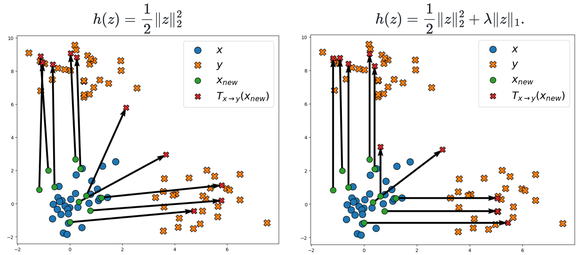
- **Sinkhorn** solves a regularized OT problem finding a regularized matching between two families of points

$$\min_{P \text{coupling}} \langle P, [c(x_i, y_j]_{ij}) \rangle - \varepsilon H(P)$$

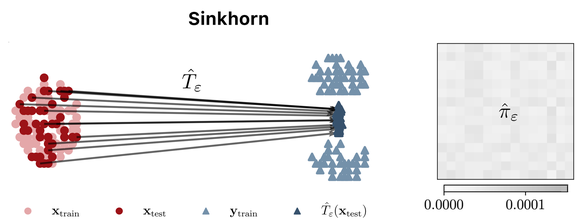


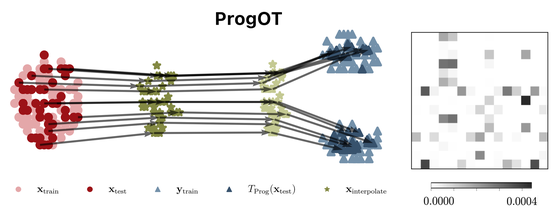
- Sinkhorn also provides an **out-of-sample estimator** for OT maps, which can work for any cost function $c(x, y) = h(x - y)$

$$h(z) = \frac{1}{2}\|z\|_2^2 \qquad h(z) = \frac{1}{2}\|z\|_2^2 + \lambda\|z\|_1.$$

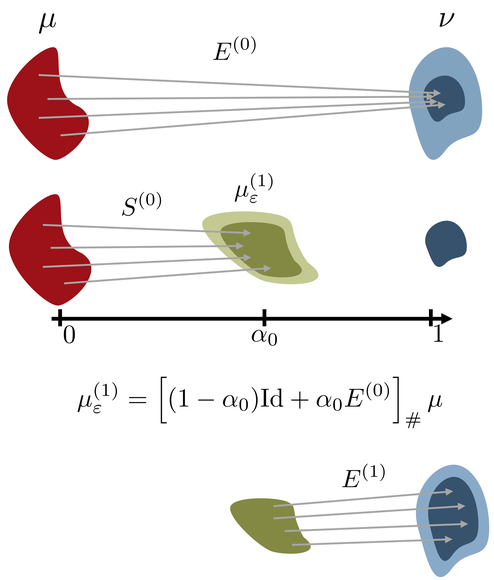


Progressive Optimal Transport for Coupling and Transport Solvers





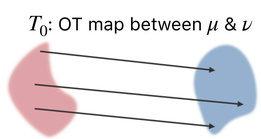
- **Proposal:** We propose to benefit from the fact that large regularization OT can be computed quickly, but only use it to create an interpolation between the source/target measures, used to compute another plan with a typically smaller regularization value.
- **Recursion and time schedule**: We apply this transformation repeatedly, taking special care in tuning the number of steps, and the intermediate values for epsilon regularization.

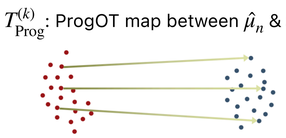


$$\mu_\varepsilon^{(1)} = \left[(1-\alpha_0)\mathrm{Id} + \alpha_0 E^{(0)}\right]_{\#} \mu$$

$$T_{\text{Prog}}^{(1)} = E^{(1)} \circ S^{(0)}$$

- **ProgOT divides and conquer OT problems**:
  - Overall compute consists in solving smaller OT problems, typically with larger inner regularization constants.
  - These regularization constants, stepsizes and convergence thresholds can be all adapted to apply faster/cheaper compute in early stages.





Theorem (Non-Asymptotic Consistency)

Given n i.i.d. samples from μ and ν, for an appropriate choice of $(\varepsilon_k)_k$ and $(\alpha_k)_k$, the K-step progressive map $T_{Prog}^{(K)}$ satisfies

$$\mathbb{E}\left\| T_{Prog}^{(k)} - T_0 \right\|_{L^2(\mu)}^2 \lesssim n^{-\frac{1}{d}},$$

under regularity assumptions on μ, ν, and the true map T_0.

- **Sinkhorn** algorithm (given here in dual form) vs. **ProgOT** recursive approach to compute couplings.

Algorithm 1 $\text{SINK}(\mathbf{a}, \mathbf{X}, \mathbf{b}, \mathbf{Y}, \varepsilon, \tau, \mathbf{f}_{\text{init}}, \mathbf{g}_{\text{init}})$.

$\mathbf{f}, \mathbf{g} \leftarrow \mathbf{f}_{\text{init}}, \mathbf{g}_{\text{init}}$ (zero by default)
$\mathbf{x}_1, \dots, \mathbf{x}_n = \mathbf{X}, \quad \mathbf{y}_1, \dots, \mathbf{y}_m = \mathbf{Y}$
$\mathbf{C} \leftarrow [h(\mathbf{x}_i - \mathbf{y}_j)]_{ij}$
while $\|\exp\left(\frac{\mathbf{C} - \mathbf{f} \oplus \mathbf{g}}{\varepsilon}\right)\mathbf{1}_m - \mathbf{a}\|_1 < \tau$ **do**
  $\mathbf{f} \leftarrow \varepsilon \log \mathbf{a} - \min_\varepsilon(\mathbf{C} - \mathbf{f} \oplus \mathbf{g}) + \mathbf{f}$
  $\mathbf{g} \leftarrow \varepsilon \log \mathbf{b} - \min_\varepsilon(\mathbf{C}^\top - \mathbf{g} \oplus \mathbf{f}) + \mathbf{g}$
end while
return $\mathbf{f}, \mathbf{g}, \mathbf{P} = \exp((\mathbf{C} - \mathbf{f} \oplus \mathbf{g})/\varepsilon)$

Algorithm 2 $\text{PROGOT}(\mathbf{a}, \mathbf{X}, \mathbf{b}, \mathbf{Y}, (\varepsilon_k, \alpha_k, \tau_k)_k)$

$\mathbf{f} = \mathbf{0}_n, \mathbf{g}^{(-1)} = \mathbf{0}_m$.
for $k = 0, \dots, K$ **do**
  $\mathbf{f}_{\text{init}}, \mathbf{g}_{\text{init}} \leftarrow (1-\alpha_k)\,\mathbf{f}, (1-\alpha_k)\,\mathbf{g}^{(k-1)}$
  $\mathbf{f}, \mathbf{g}^{(k)}, \mathbf{P} \leftarrow \text{Sink}(\mathbf{a}, \mathbf{X}, \mathbf{b}, \mathbf{Y}, \varepsilon_k, \tau_k, \mathbf{f}_{\text{init}}, \mathbf{g}_{\text{init}})$
  $\mathbf{Q} \leftarrow \text{diag}(1/\mathbf{P}\mathbf{1}_m)\mathbf{P}$
  $\mathbf{Z} \leftarrow [\nabla h^*(\sum_j \mathbf{Q}_{ij} \nabla h(\mathbf{x}_i - \mathbf{y}_j))]_i \in \mathbb{R}^{n \times d}$
  $\mathbf{X} \leftarrow \mathbf{X} - \alpha_k \mathbf{Z}$
end for
return: Coupling matrix $\mathbf{P}$,
Map estimator $\mathcal{T}_{\text{Prog}}[\mathbf{b}, \mathbf{Y}, (\mathbf{g}^{(k)}, \varepsilon_k, \alpha_k)_k](\cdot)$

- Two algorithms: the ProgOT OT/Monge map estimator, valid out of sample, and a heuristic designed to help schedule ε throughout iterations, adaptively through a small optimisation.

Algorithm 3 $\mathcal{T}_{\text{Prog}}[\mathbf{b}, \mathbf{Y}, (\mathbf{g}^{(k)}, \varepsilon_k, \alpha_k)_k]$

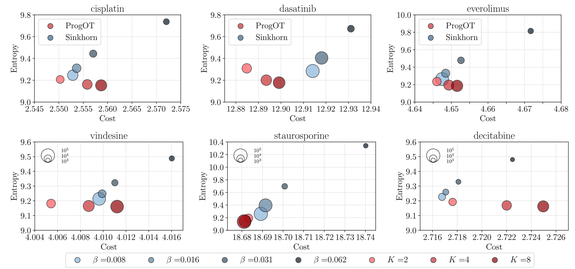
input: Source point $\mathbf{x} \in \mathbb{R}^d$
initialize: $\mathbf{y} = \mathbf{x}$, α_K reset to 1.
for $k = 0, \dots K$ **do**
  $\mathbf{p} \leftarrow [\mathbf{b}_j \exp(\frac{\mathbf{g}^{(k)} - h(\mathbf{y} - \mathbf{y}_j)}{\varepsilon_k})]_j$
  $\mathbf{p} \leftarrow \mathbf{p}/\mathbf{1}_m^T\mathbf{p} \in \mathbb{R}^{m \times_k}$
  $\mathbf{\Delta} \leftarrow [\nabla h(\mathbf{y} - \mathbf{y}_j)]_j \in \mathbb{R}^{m \times d}$
  $\mathbf{z} \leftarrow \nabla h^*(\mathbf{p}^T\mathbf{\Delta}) \in \mathbb{R}^d$
  $\mathbf{y} \leftarrow \mathbf{y} - \alpha_k \mathbf{z}$.
end for
return: $\mathbf{y}$

Algorithm 4 ε-scheduler$(\mathbf{Y}_{\text{test}}, (s_p)_p, \beta_0)$

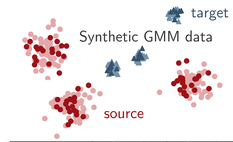
recover $\mathbf{a}, \mathbf{X}, \mathbf{b}, \mathbf{Y}, (\alpha_k)_k$.
set $\varepsilon_0 \leftarrow \frac{1}{20}\frac{1}{nm}\sum_{ij} h(\mathbf{x}_i - \mathbf{y}_j)$.
set $\sigma \leftarrow \frac{1}{20}\frac{1}{m^2}\sum_{lj} h(\mathbf{y}_l - \mathbf{y}_j)$.
for $p = 1, \dots, P$ **do**
  $\varepsilon \leftarrow s_p \times \sigma$.
  $_, \mathbf{g}, _ \leftarrow \text{SINK}(\mathbf{b}, \mathbf{Y}, \mathbf{b}, \mathbf{Y}, \varepsilon, \tau)$
  $T = \mathcal{T}_{\text{Prog}}[\mathbf{b}, \mathbf{Y}, (\mathbf{g}, \varepsilon, 1)]$
  $\text{error}_p \leftarrow \|\mathbf{Y}_{\text{test}} - T(\mathbf{Y}_{\text{test}})\|^2$
end for
$p^* = \arg\min_p \text{error}_p$.
$\varepsilon_1 \leftarrow s_{p^*} \times \sigma$.
$(t_k)_k = (1 - \prod_{\ell=1}^k (1 - \alpha_\ell))_k$.
return: $((1-t_k)\beta_0\varepsilon_0 + t_k\varepsilon_1)_k$.

Experiments

- ProgOT's cost/regularization frontier as coupling solver.

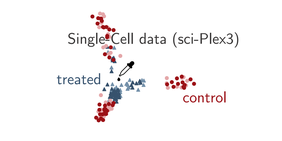


- ProgOT as OT map estimator vs. alternatives.



	$d = 128$	$d = 256$
PROGOT	**0.099**±0.009	**0.12**±0.01
EOT	0.12±0.01	0.16±0.02
Debiased EOT	0.11±0.01	0.128±0.002
Untuned EOT	0.250±0.023	0.276±0.006
Monge Gap	0.36±0.02	0.273±0.005
ICNN	0.177±0.023	0.117±0.005

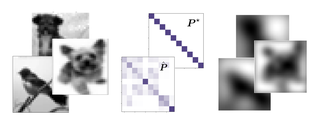
Known ground-truth, **MSE** between maps on test points



Drug	Hesperadin			5-drug
d_{PCA}	16	64	256	rank
PROGOT	**3.7**±0.4	**10.1**±0.4	**23.1**±0.4	**1**
EOT	4.1±0.4	10.4±0.5	26±1.3	2
Debiased EOT	4.0±0.5	15.2±0.6	41±1.1	4
Monge Gap	3.7±0.5	11.0±0.5	36±1.1	3
ICNN	3.9±0.4	14.3±0.5	46±2	5

Ground-truth unknown, **Wasserstein** distance

- Running ProgOT computations on a benchmark task of matching 60k images vs. their blurred versions (OT map is known!). Easy because natively shard-able (thanks to JAX).



σ		2	4
Sinkhorn	Tr	0.9999	0.9954
	KL	0.00008	0.02724
	# iters	10	2379
PROGOT	Tr	1.000	0.9989
	KL	0.00000	0.00219
	# iters	40	1590

Conclusion

ProgOT, a new family of regularized OT solvers

- Leverages **multiple opportunities** (regularization, thresholds) to schedule OT computations along the natural geodesic occurring between two source/target distributions.
- Adds a **dynamic flavor** into static reg-OT, supported by experiments, and can scale to 1e5~1e6 points using sharding.
- **Implemented** in **OTT-JAX**, see tutorial using QR code.

TM and © 2024 Apple. All rights reserved.