Large-scale Optimal Transport

Weijie Chen[*]
School of Physics
Peking University
1500011335
1500011335@pku.edu.cn

Dinghuai Zhang
School of Mathematics
Peking University
1600013525
1600013525@pku.edu.cn

1 Introduction to Optimal Transport

Optimal transport (OT) problems are an important series of problems considering the minimal cost of transportation, receiving increasing attention from the community of applied mathematics. Its history can be date back to 18th century, the time of the French engineer Gaspard Monge or 1920s when mathematicians were trying to figure out a way to move things efficiently during World War I. Sometimes optimal transport can be seen as some kind of network flow problem, aiming to "transport" one distribution μ to another distribution ν under some certain conditions [2], see figure 1. To put it in another way, given two distributions and a cost function, we aim to find a transport method that minimize a certain kind of cost. A specific definition can be seen in 1 and 4.

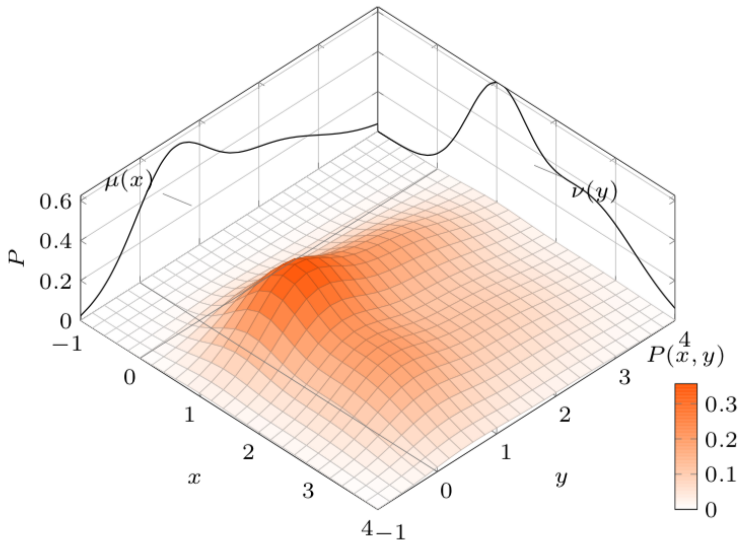

Figure 1: an example of optimal transport

[*]Pre-admission 2019 PKU AAIS
[2]figure is taken from Justin Solomon's tutorial

Over nearly 200 years, theory of optimal transport have not made much progress until some big mathematical breakthroughs in the 1980s and 1990s. Since then the field has flourished and optimal transport theory has found applications in PDEs, geometry, statistics, economics and image processing [5]. However, although it is such a prosperous research field, efficient algorithm to numerically solve continuous or discrete optimal transport problems is absent. While there are some existing algorithms [6] [7], there is still vast space for improvement.

In this report, we derive and implement some algorithms for solving a discrete optimal transport problem from a linear programming view. For some of them which is hard for tunning, we make some discussion about the reasons. Our codes can be reached in GitHub: https://github.com/zdhNarsil/OptimalTransport or https://github.com/BaozCWJ/OptimalTransport.

2 Problem Statement

The standard formulation of optimal transport are derived from couplings. [8] That is, let $(\mathcal{X}, \mu)$ and $(\mathcal{Y}, \nu)$ be two probability spaces, and a probability distribution π on $\mathcal{X} \times \mathcal{Y}$ is called *coupling* if $proj_{\mathcal{X}}(\pi) = \mu$ and $proj_{\mathcal{Y}}(\pi) = \nu$. An optimal transport between $(\mathcal{X}, \mu)$ and $(\mathcal{Y}, \nu)$, or an optimal coupling, is a coupling minimize

$$\int_{\mathcal{X}\times\mathcal{Y}} c(x,y)d\pi(x,y) \tag{1}$$

Optimal transport problems can be categorized according to the discreteness of μ and ν. In this report, we only consider discrete optimal tranport problems, where the two distributions are distributions of finite weighted points. A discrete optimal transport problem can be formulated into a linear program as

$$\begin{aligned} &\min_{\pi} \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}\pi_{ij} \\ &s.t. \sum_{j=1}^{n} \pi_{ij} = \mu_i, \forall i \\ &\qquad \sum_{i=1}^{m} \pi_{ij} = \nu_j, \forall j \\ &\qquad \pi_{ij} \geq 0, \end{aligned} \tag{2}$$

where c stands for the cost and s for the transportation plan, while μ and ν are restrictions. Note that we always suppose $c \geq 0$, $\mu \geq 0$, $\nu \geq 0$ and $\sum_{i=1}^{m} \mu_i = \sum_{j=1}^{n} \nu_j = 1$ implicitly. From realistic background, c is always valued the squared Euclidean distanced or some other norms. Note that there are mn variables in this formulation, and this leads to intensive computation.

In order to decrease the number of variables, we can derive the dual problem of discrete optimal transport.

$$\begin{aligned}
&\max_{\lambda,\eta} \sum_{i=1}^{m} \mu_i \lambda_i + \sum_{j=1}^{n} \nu_j \eta_j \\
&s.t. c_{ij} - \lambda_i - \eta_j \geq 0, \forall i, j
\end{aligned} \tag{3}$$

Although this formulation only has m + n variables, there are still challenges including the recovery of π from λ and η and the great number of constraints.

3 Algorithms

3.1 ADMM for Primal Problem

We first implement a first order algorithm called **alternative direction method of multipliers** (ADMM). According to a reformulation of primal problem,

$$\begin{aligned}
&\min_{\pi} \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} \pi_{ij} + \mathbb{I}_+(\hat{\pi}) \\
&s.t. \sum_{j=1}^{n} \pi_{ij} = \mu_i, \forall i \\
&\sum_{i=1}^{m} \pi_{ij} = \nu_j, \forall j \\
&\pi = \hat{\pi}
\end{aligned} \tag{4}$$

where $\mathbb{I}_+$ is indicator of $\mathbb{R}_+^{m \times n}$. The augmented Lagrangian can be written as

$$\begin{aligned}
\mathcal{L}_\rho(\pi, \hat{\pi}, \lambda, \eta, e) = &\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} \pi_{ij} + \mathbb{I}_+(\hat{\pi}) \\
&+ \sum_{i=1}^{m} \lambda_i \left(\mu_i - \sum_{j=1}^{n} \pi_{ij} \right) + \sum_{j=1}^{n} \eta_j \left(\nu_j - \sum_{i=1}^{m} \pi_{ij} \right) + \sum_{i=1}^{m} \sum_{j=1}^{n} e_{ij} \left(\pi_{ij} - \hat{\pi}_{ij} \right) \\
&+ \frac{\rho}{2} \sum_{i=1}^{m} \left(\mu_i - \sum_{j=1}^{n} \pi_{ij} \right)^2 + \frac{\rho}{2} \sum_{j=1}^{n} \left(\nu_j - \sum_{i=1}^{m} \pi_{ij} \right)^2 + \frac{\rho}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} \left(\pi_{ij} - \hat{\pi}_{ij} \right)^2
\end{aligned} \tag{5}$$

The minimizer of $\hat{\pi}$ can be written easily as

$$argmin_{\hat{\pi}} \mathcal{L}_\rho(\pi, \hat{\pi}, \lambda, \eta, e) = max \left(\pi + \frac{e}{\rho}, 0 \right) \tag{6}$$

For the minimizer of π, we can derive the following equation:

$$\sum_{k=1}^{n} \pi_{ik} + \sum_{k=1}^{m} \pi_{kj} + \pi_{ij} = \frac{1}{\rho} \left(-e_{ij} + \lambda_i + \eta_j - c_{ij} \right) + \mu_i + v_j + \hat{\pi}_{ij} \equiv r_{ij} \tag{7}$$

It's a linear equation of π_{ij} for the given r_{ij}, which can be solved directly.

$$\pi_{ij} = r_{ij} - \frac{1}{n+1}\sum_{k=1}^{n}\left(r_{ik} - \frac{1}{m+n+1}\sum_{l=1}^{m} r_{lk}\right) - \frac{1}{m+1}\sum_{k=1}^{m}\left(r_{kj} - \frac{1}{m+n+1}\sum_{l=1}^{n} r_{kl}\right) \tag{8}$$

Then, we can write the explicit form of ADMM algorithm. This algorithm is implemented in **ADMM_primal.py**.

Algorithm 1: Alternating direction method of multipliers for the primal problem

```
Input: input data $c$, $\mu$, $\nu$, step size$\alpha$, penalty scalar $\rho$ and maximum iteration $N$
Output: solution $\pi$
1  initializing $k = 0$
2  $\pi^{(k)}, \hat{\pi}^{(k)}, e^{(k)}, \lambda^{(k)}, \eta^{(k)} := 0$
3  while $k < N$ do
4      $\pi^{(k+1)} := argmin_{\pi}\mathcal{L}_\rho(\pi, \hat{\pi}^{(k)}, \lambda^{(k)}, \eta^{(k)}, e^{(k)})$
5      $\hat{\pi}^{(k+1)} := argmin_{\hat{\pi}}\mathcal{L}_\rho(\pi^{(k+1)}, \hat{\pi}, \lambda^{(k)}, \eta^{(k)}, e^{(k)})$
6      $\lambda^{(k+1)} := \lambda^{(k)} + \alpha\rho(\mu - \sum_{j=1}^{n} \pi_{ij})$
7      $\eta^{(k+1)} := \eta^{(k)} + \alpha\rho(\nu - \sum_{i=1}^{m} \pi_{ij})$
8      $e^{(k+1)} := e^{(k)} + \alpha\rho(\pi - \hat{\pi})$
9      $k := k + 1$
10 end
11 return $\hat{\pi}$
```

3.2 ADMM for Dual Problem

According the reformulation of dual problem,

$$\begin{aligned} &\min_{\lambda,\eta} - \sum_{i=1}^{m} \mu_i\lambda_i - \sum_{j=1}^{n} \nu_j\eta_j + \mathbb{I}_+(e) \\ &s.t. c_{ij} - \lambda_i - \eta_j - e_{ij} = 0, \forall i, j \end{aligned} \tag{9}$$

we can write down the augmented Lagrangian as

$$\begin{aligned} \mathcal{L}_\rho(\lambda, \eta, e, d) = &- \sum_{i=1}^{m} \mu_i\lambda_i - \sum_{j=1}^{n} \nu_j\eta_j + \mathbb{I}_+(e) \\ &+ \sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij}(c_{ij} - \lambda_i - \eta_j - e_{ij}) + \frac{\rho}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}(c_{ij} - \lambda_i - \eta_j - e_{ij})^2 \end{aligned} \tag{10}$$

The minimizer of e can be done directly by solving for zero gradient and projection, while the minimizer of λ and η can be done by solving for zero gradient.

$$\begin{aligned} argmin_{e_{ij}}\mathcal{L}_\rho(\lambda,\eta,e,d) =& max\left(c_{ij}+\frac{d_{ij}}{\rho}-\lambda_i-\eta_j,0\right) \\ argmin_{\lambda_i}\mathcal{L}_\rho(\lambda,\eta,e,d) =& \frac{1}{n}\left((\mu_i+\sum_{j=1}^{n}d_{ij})/\rho+\sum_{j=1}^{n}(c_{ij}-\eta_j-e_{ij})\right) \\ argmin_{\eta_j}\mathcal{L}_\rho(\lambda,\eta,e,d) =& \frac{1}{m}\left((\nu_j+\sum_{i=1}^{m}d_{ij})/\rho+\sum_{i=1}^{m}(c_{ij}-\lambda_i-e_{ij})\right) \end{aligned} \tag{11}$$

The algorithm is implemented in **ADMM_dual.py**. Solution π can be recovered by $\pi=-d$ from KKT conditions.

Algorithm 2: Alternating direction method of multipliers for the primal problem

Input: input data c, μ, ν, step sizeα, penalty scalar ρ and maximum iteration N
Output: solution π

```
1  initializing $k = 0$
2  $\lambda^{(k)}, \eta^{(k)}, e^{(k)}, d^{(k)} := 0$
3  while $k < N$ do
4      $\lambda_i^{(k+1)} := argmin_{\lambda_i}\mathcal{L}_\rho(\lambda, \eta^{(k)}, e^{(k)}, d^{(k)})$
5      $\eta_j^{(k+1)} := argmin_{\eta_j}\mathcal{L}_\rho(\lambda^{(k+1)}, \eta, e^{(k)}, d^{(k)})$
6      $e_{ij}^{(k+1)} := argmin_{e_{ij}}\mathcal{L}_\rho(\lambda^{(k+1)}, \eta^{(k+1)}, e, d^{(k)})$
7      $d_{ij}^{(k+1)} := d_{ij}^{(k)} + \alpha\rho(c_{ij} - \lambda_i - \eta_j - e_{ij})$
8      $k := k + 1$
9  end
10 return $\pi = -d$
```

3.3 Add Entropy Regularization: Sinkhorn-Knopp Algorithm

The discrete entropy of a coupling matrix is defined as [13]:

$$\mathbf{H}(\mathbf{P}) \overset{\text{def}}{=} -\sum_{i,j}\mathbf{P}_{i,j}\left(\log\left(\mathbf{P}_{i,j}\right)-1\right) \tag{12}$$

The function $\mathbf{H}$ is strongly concave, we can see it by computing its 2-order derivatives:

$$\frac{\partial^2\mathbf{H}(P)}{\partial\mathbf{P}_{ij}^2} = -\operatorname{diag}\left(1/\mathbf{P}_{i,j}\right) \tag{13}$$

Notice that $\mathbf{P}_{i,j}\leq 1$, it's obvious that $\mathbf{H}(\mathbf{P})$ has a good property of convexity.

The idea of the entropic regularization of optimal transport is to use $-\mathbf{H}$ as a regularizing function to obtain approximate solutions to the original transport problem:

$$\mathrm{L}_{\mathrm{C}}^{\varepsilon}(\mathbf{a},\mathbf{b}) \overset{\text{def}}{=} \min_{\mathbf{P}\in\mathbf{U}(\mathbf{a},\mathbf{b})}\langle\mathbf{P},\mathbf{C}\rangle-\varepsilon\mathbf{H}(\mathbf{P}) \tag{14}$$

With strong convexity of $\mathbf{H}(\mathbf{P})$ our new target has a global minima.

For readers who are interest in its asympotic property, we refer to [10]. Up to now, all we need to know is that we have

$$\lim_{\epsilon\to 0} \mathrm{L}_{\mathrm{C}}^{\varepsilon}(\mathbf{a},\mathbf{b}) = \mathrm{L}_{\mathrm{C}}(\mathbf{a},\mathbf{b}) = \min_{\mathbf{P}\in\mathbf{U}(\mathbf{a},\mathbf{b})} \langle \mathbf{P},\mathbf{C}\rangle \tag{15}$$

What's more, this new target can be interpreted as KL($\mathbf{P}|\mathbf{K}$):

$$\mathbf{KL}(\mathbf{P}|\mathbf{K}) \overset{\text{def}}{=} \sum_{i,j} \mathbf{P}_{i,j} \log\left(\frac{\mathbf{P}_{i,j}}{\mathbf{K}_{i,j}}\right) - \mathbf{P}_{i,j} + \mathbf{K}_{i,j} \tag{16}$$

where $\log$ is taken element-wise and $\mathbf{K}_{i,j} = e^{-\mathbf{C}_{i,j}/\epsilon}$, which will be interpreted later.

One can show that the solution to 14 has the form of

$$\mathbf{P}_{i,j} = \mathbf{u}_i \mathbf{K}_{i,j} \mathbf{v}_j \tag{17}$$

where $\mathbf{K}_{i,j} = e^{-\mathbf{C}_{i,j}/\epsilon}$ by calculating the KKT condition: Introducing two dual variables $\mathbf{f}\in\mathbb{R}^n, \mathbf{g}\in\mathbb{R}^n$ and calculate the lagrangian:

$$\mathcal{L}(\mathbf{P},\mathbf{f},\mathbf{g}) = \langle \mathbf{P},\mathbf{C}\rangle - \varepsilon\mathbf{H}(\mathbf{P}) - \langle \mathbf{f}, \mathbf{P}\mathbf{1}_n - \mathbf{a}\rangle - \left\langle \mathbf{g}, \mathbf{P}^{\mathrm{T}}\mathbf{1}_n - \mathbf{b}\right\rangle \tag{18}$$

take first order gradient and we get

$$\frac{\partial \mathcal{L}(\mathbf{P},\mathbf{f},\mathbf{g})}{\partial \mathbf{P}_{i,j}} = \mathbf{C}_{i,j} + \varepsilon \log\left(\mathbf{P}_{i,j}\right) - \mathbf{f}_i - \mathbf{g}_j = 0 \tag{19}$$

$$\Rightarrow \mathbf{P}_{i,j} = e^{\mathbf{f}_i/\varepsilon} e^{-\mathbf{C}_{i,j}/\varepsilon} e^{\mathbf{g}_j/\varepsilon} \tag{20}$$

Thus we also get

$$\mathbf{u} = e^{\mathbf{f}/\epsilon} \tag{21}$$

$$\mathbf{v} = e^{\mathbf{g}/\epsilon} \tag{22}$$

Based on the constrain that:

$$\operatorname{diag}(\mathbf{u})\mathbf{K}\operatorname{diag}(\mathbf{v})\mathbf{1}_m = \mathbf{a} \tag{23}$$

$$\operatorname{diag}(\mathbf{v})\mathbf{K}^{\top}\operatorname{diag}(\mathbf{u})\mathbf{1}_n = \mathbf{b} \tag{24}$$

or :

$$\mathbf{u}\odot(\mathbf{K}\mathbf{v}) = \mathbf{a} \quad \text{and} \quad \mathbf{v}\odot\left(\mathbf{K}^{\mathrm{T}}\mathbf{u}\right) = \mathbf{b} \tag{25}$$

(where $\odot$ means entry-wise multiplication of vectors) we can develop our algorithm as iteratively updating $\mathbf{u}$ and $\mathbf{v}$, which is famous Sinkhorn-Knopp algorithm[12]:

$$\mathbf{u}^{(\ell+1)} = \frac{\mathbf{a}}{\mathbf{K}\mathbf{v}^{(\ell)}} \text{ and } \mathbf{v}^{(\ell+1)} = \frac{\mathbf{b}}{\mathbf{K}^{\mathrm{T}}\mathbf{u}^{(\ell+1)}} \tag{26}$$

with $\mathbf{v}^{(0)} = \mathbf{1}_m$ and $\mathbf{K}_{i,j} = e^{-\mathbf{C}_{i,j}/\epsilon}$.

We remark that with different initial value for $\mathbf{v}^{(0)}$, the final solution for $\mathbf{v}$ and $\mathbf{u}$ would be different, since if we use $\lambda\mathbf{v}^{(0)}$ to replace $\mathbf{v}$, then we will get $\mathbf{u}/\lambda$ at every iteration. Even so, our final solution for $\mathbf{P}$ won't change based on the way we calculate it 20.

3.4 Sinkhorn-Newton Method [11]

If we take $-\mathbf{f}, -\mathbf{g}$ to take place with $\mathbf{f}, \mathbf{g}$ (notice that $\mathbf{f}, \mathbf{g}$ are just multiplier for equality constrain, so changing their sign as a whole won't make any difference), then from 20 and 25 we can conclude that our target could be reformulated as finding a zero point of

$$F(\mathbf{f}, \mathbf{g}) := \begin{pmatrix} a - \operatorname{diag}(e^{-\mathbf{f}/\epsilon})\mathbf{K}e^{-\mathbf{g}/\epsilon} \\ b - \operatorname{diag}(e^{-\mathbf{g}/\epsilon})\mathbf{K}^T e^{-\mathbf{f}/\epsilon} \end{pmatrix}$$

where a, b, ϵ and $\mathbf{K}$ are known. What we need to do is to use newton-raphson method to find its zero points [11]:

$$\begin{pmatrix} \mathbf{f}^{k+1} \\ \mathbf{g}^{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{f}^{k} \\ \mathbf{g}^{k} \end{pmatrix} - J_F\left(\mathbf{f}^k, \mathbf{g}^k\right)^{-1} F\left(\mathbf{f}^k, \mathbf{g}^k\right) \tag{27}$$

where the Jacobian of F is:

$$J_F(\mathbf{f}, \mathbf{g}) = \frac{1}{\varepsilon}\begin{bmatrix} \operatorname{Diag}\left(\mathbf{P}\mathbf{1}_m\right) & \mathbf{P} \\ \mathbf{P}^\top & \operatorname{Diag}\left(\mathbf{P}^\top \mathbf{1}_n\right) \end{bmatrix} \tag{28}$$

that is, we can use conjugate gradient to solve

$$J_F\left(\mathbf{f}^k, \mathbf{g}^k\right)\begin{pmatrix} \delta\mathbf{f} \\ \delta\mathbf{g} \end{pmatrix} = -F\left(\mathbf{f}^k, \mathbf{g}^k\right) \tag{29}$$

and then update variables by

$$\begin{aligned} \mathbf{f}^{k+1} &= \mathbf{f}^k + \delta\mathbf{f} \\ \mathbf{g}^{k+1} &= \mathbf{g}^k + \delta\mathbf{g} \end{aligned} \tag{30}$$

Because $\mathbf{P}^k := \operatorname{Diag}\left(\mathrm{e}^{-\mathbf{f}^k/\varepsilon}\right)\mathbf{K}\operatorname{Diag}\left(\mathrm{e}^{-\mathbf{g}^k/\varepsilon}\right)$, the update step can be rewrite as

$$\begin{aligned} \mathbf{P}^{k+1} &= \operatorname{Diag}\left(\mathrm{e}^{-\left[\mathbf{f}^k+\delta\mathbf{f}\right]/\varepsilon}\right)\mathbf{K}\operatorname{Diag}\left(\mathrm{e}^{-\left[\mathbf{g}^k+\delta\mathbf{g}\right]/\varepsilon}\right) \\ &= \operatorname{Diag}\left(\mathrm{e}^{-\delta\mathbf{f}/\varepsilon}\right)\mathbf{P}^k\operatorname{Diag}\left(\mathrm{e}^{-\delta\mathbf{g}/\varepsilon}\right) \end{aligned} \tag{31}$$

Algorithm 3: Sinkhorn-Newton method in primal variable

```
Input: $\mathbf{a} \in \Sigma_n, \mathbf{b} \in \Sigma_m, \mathbf{C} \in \mathbb{R}^{n\times m}$
1 initializing $\mathbf{P}^0 = \exp(-\mathbf{C}/\varepsilon)$, set $k = 0$
2 repeat
3     $\mathbf{a}^k \leftarrow \mathbf{P}^k\mathbf{1}_m$
4     $\mathbf{b}^k \leftarrow \left(\mathbf{P}^k\right)^\top \mathbf{1}_n$
5     compute $\delta\mathbf{f}, \delta\mathbf{g}$: $\frac{1}{\varepsilon}\begin{bmatrix} \operatorname{Diag}\left(\mathbf{a}^k\right) & \mathbf{P}^k \\ \left(\mathbf{P}^k\right)^\top & \operatorname{Diag}\left(\mathbf{b}^k\right) \end{bmatrix}\begin{bmatrix} \delta\mathbf{f} \\ \delta\mathbf{g} \end{bmatrix} = \begin{bmatrix} \mathbf{a}^k - \mathbf{a} \\ \mathbf{b}^k - \mathbf{b} \end{bmatrix}$
6     $\mathbf{P}^{k+1} \leftarrow \operatorname{Diag}\left(\mathrm{e}^{-\delta\mathbf{f}/\varepsilon}\right)\mathbf{P}^k\operatorname{Diag}\left(\mathrm{e}^{-\delta\mathbf{g}/\varepsilon}\right)$
7     $k \leftarrow k + 1$
8 until some stopping criteria fulfilled;
9 return $\mathbf{P}$
```

3.5 Sinkhorn-Newton for Dual Problem

For the dual problem

$$\mathcal{L}(\mathbf{P},\mathbf{f},\mathbf{g}) = \langle \mathbf{P},\mathbf{C}\rangle - \varepsilon\mathbf{H}(\mathbf{P}) - \langle \mathbf{f}, \mathbf{P}\mathbf{1}_n - \mathbf{a}\rangle - \left\langle \mathbf{g}, \mathbf{P}^{\mathrm{T}}\mathbf{1}_n - \mathbf{b}\right\rangle \tag{32}$$

$$\frac{\partial\mathcal{L}(\mathbf{P},\mathbf{f},\mathbf{g})}{\partial \mathbf{P}_{i,j}} = 0 \tag{33}$$

$$\Rightarrow \hat{\mathbf{P}} = e^{\mathbf{f}/\varepsilon}e^{-\mathbf{C}/\varepsilon}e^{\mathbf{g}/\varepsilon} \tag{34}$$

$$\Rightarrow \mathcal{L}(\hat{\mathbf{P}},\mathbf{f},\mathbf{g}) = \langle \mathbf{f},\mathbf{a}\rangle + \langle \mathbf{g},\mathbf{b}\rangle - \varepsilon\left\langle e^{\mathbf{f}/\varepsilon}, \mathbf{K}e^{\mathbf{g}/\varepsilon}\right\rangle \tag{35}$$

We only need to solve

$$\max_{\mathbf{f}\in\mathbb{R}^n,\mathbf{g}\in\mathbb{R}^m} \langle \mathbf{f},\mathbf{a}\rangle + \langle \mathbf{g},\mathbf{b}\rangle - \varepsilon\left\langle e^{\mathbf{f}/\varepsilon}, \mathbf{K}e^{\mathbf{g}/\varepsilon}\right\rangle = Q(\mathbf{f},\mathbf{g}) \tag{36}$$

(Notice that this is an approximation of the original Kantorovich dual

$$\mathrm{L}_{\mathbf{C}}(\mathbf{a},\mathbf{b}) = \max_{(\mathbf{f},\mathbf{g})\in\mathbf{R}(\mathbf{a},\mathbf{b})} \langle \mathbf{f},\mathbf{a}\rangle + \langle \mathbf{g},\mathbf{b}\rangle \tag{37}$$

$$\mathbf{R}(\mathbf{a},\mathbf{b}) \overset{\text{def}}{=} \{(\mathbf{f},\mathbf{g}) \in \mathbb{R}^n \times \mathbb{R}^m : \forall (i,j) \in \mathbb{Z}[n]\times[m], \mathbf{f}\oplus\mathbf{g} \le \mathbf{C}\} \tag{38}$$

)

We can calculate its gradient

$$\begin{aligned} \nabla|_{\mathbf{f}}\, Q(\mathbf{f},\mathbf{g}) &= \mathbf{a} - e^{\mathbf{f}/\varepsilon} \odot \left(\mathbf{K}e^{\mathbf{g}/\varepsilon}\right) \\ \nabla|_{\mathbf{g}}\, Q(\mathbf{f},\mathbf{g}) &= \mathbf{b} - e^{\mathbf{g}/\varepsilon} \odot \left(\mathbf{K}^{T}e^{\mathbf{f}/\varepsilon}\right) \end{aligned} \tag{39}$$

and its Hessian matrix respectively.

Then we derive the Newton-Raphson algorithm for minimizing $Q(\mathbf{f},\mathbf{g})$:

Algorithm 4: Sinkhorn-Newton method in dual variable

```
   Input: $\mathbf{a} \in \Sigma_n, \mathbf{b} \in \Sigma_m, \mathbf{K}$ and $\mathbf{K}^\top$
   Output: solution $\mathbf{P}$
 1 initializing $a^0 \in \mathbb{R}^n, b^0 \in \mathbb{R}^m$, set $k = 0$
 2 repeat
 3     $a^k \leftarrow \mathrm{e}^{-f^k/\varepsilon} \odot K\mathrm{e}^{-g^k/\varepsilon}$
 4     $b^k \leftarrow \mathrm{e}^{-g^k/\varepsilon} \odot K^\top\mathrm{e}^{-f^k/\varepsilon}$
 5     Compute updates $\delta f$ and $\delta g$ by solving $M\begin{bmatrix}\delta f\\ \delta g\end{bmatrix} = \begin{bmatrix} a^k - a \\ b^k - b\end{bmatrix}$
 6     where the application of $M$ is given by
       $M\begin{bmatrix}\delta f\\ \delta g\end{bmatrix} = \frac{1}{\varepsilon}\begin{bmatrix} a^k \odot \delta f + \mathrm{e}^{-f^k/\varepsilon} \odot K\left(\mathrm{e}^{-g^k/\varepsilon}\odot\delta g\right) \\ b^k \odot \delta g + e^{-g^k/\varepsilon} \odot K^\top\left(\mathrm{e}^{-f^k/\varepsilon}\odot\delta f\right)\end{bmatrix}$
 7     $f^{k+1} \leftarrow f^k + \delta f$
 8     $g^{k+1} \leftarrow g^k + \delta g$
 9     $k \leftarrow k + 1$
10 until some stopping criteria fulfilled;
11 return $\mathbf{P}$
```

Or, in 39 we can set the gradient to 0 straightly

$$\mathbf{f}^{(\ell+1)} = \varepsilon \log \mathbf{a} - \varepsilon \log \left(\mathbf{K} e^{\mathbf{g}^{(\ell)}/\varepsilon}\right) \tag{40}$$

$$\mathbf{g}^{(\ell+1)} = \varepsilon \log \mathbf{b} - \varepsilon \log \left(\mathbf{K}^{\mathrm{T}} e^{\mathbf{f}^{(\ell+1)}/\varepsilon}\right) \tag{41}$$

for $\ell \geq 0$. However, it's actually the same as Sinkhorn-Knopp algorithm based on $(\mathbf{u}, \mathbf{v}) = \left(e^{\mathbf{f}/\varepsilon}, e^{\mathbf{g}/\varepsilon}\right)$.

3.6 Log-domain Sinkhorn

[13] We can rewrite the above formula 40 and 41 as

$$\begin{aligned} \mathbf{f}^{(\ell+1)} &= \operatorname{Min}_{\varepsilon}^{\text{row}}\left(\mathbf{S}\left(\mathbf{f}^{(\ell)}, \mathbf{g}^{(\ell)}\right)\right) - \mathbf{f}^{(\ell)} + \varepsilon \log(\mathbf{a}) \\ \mathbf{g}^{(\ell+1)} &= \operatorname{Min}_{\varepsilon}^{\text{col}}\left(\mathbf{S}\left(\mathbf{f}^{(\ell+1)}, \mathbf{g}^{(\ell)}\right)\right) - \mathbf{g}^{(\ell)} + \varepsilon \log(\mathbf{b}) \end{aligned} \tag{42}$$

where $\mathbf{S}(\mathbf{f}, \mathbf{g}) = \left(\mathbf{C}_{i,j} - \mathbf{f}_i - \mathbf{g}_j\right)_{i,j}$ and

$$\begin{aligned} \operatorname{Min}_{\varepsilon}^{\text{row}}(\mathbf{A}) &\stackrel{\text{def}}{=} \left(\min_{\varepsilon}\left(\mathbf{A}_{i,j}\right)_j\right)_i \in \mathbb{R}^n \\ \operatorname{Min}_{\varepsilon}^{\text{col}}(\mathbf{A}) &\stackrel{\text{def}}{=} \left(\min_{\varepsilon}\left(\mathbf{A}_{i,j}\right)_i\right)_j \in \mathbb{R}^m \end{aligned} \tag{43}$$

and $\min_{\varepsilon} \mathbf{z} = -\varepsilon \log \sum_i e^{-\mathbf{z}_i/\varepsilon}$ for a vector $\mathbf{z}$.

Let's probe into this expression:

First, $\min_{\varepsilon} \mathbf{z}$ is nothing but a differentiable approximation of $\min$ function. When we take $\epsilon \to 0$, we have $\min_{\varepsilon} \mathbf{z} = \min \mathbf{z}$. Besides, the above formula 40 and 41 are just

$$\left(\mathbf{f}^{(\ell+1)}\right)_i = \min_{\varepsilon}\left(\mathbf{C}_{ij} - \mathbf{g}_j^{(\ell)}\right)_j + \varepsilon \log \mathbf{a}_i \tag{44}$$

$$\left(\mathbf{g}^{(\ell+1)}\right)_j = \min_{\varepsilon}\left(\mathbf{C}_{ij} - \mathbf{f}_i^{(\ell)}\right)_i + \varepsilon \log \mathbf{b}_j \tag{45}$$

where $\left(\mathbf{C}_{ij} - \mathbf{g}_j^{(\ell)}\right)_j$ denotes the soft-minimum of all values of the j-th column of matrix $\left(\mathbf{C} - \mathbf{1}_n \left(\mathbf{g}^{(\ell)}\right)^{\top}\right)$. If we define $\operatorname{Min}_{\varepsilon}^{\text{row}}(\mathbf{A})$ and $\operatorname{Min}_{\varepsilon}^{\text{col}}(\mathbf{A})$ as above, then we get

$$\mathbf{f}^{(\ell+1)} = \operatorname{Min}_{\varepsilon}^{\text{row}}\left(\mathbf{C} - \mathbf{1}_n \mathbf{g}^{(\ell)^{\mathrm{T}}}\right) + \varepsilon \log \mathbf{a} \tag{46}$$

$$\mathbf{g}^{(\ell+1)} = \operatorname{Min}_{\varepsilon}^{\text{col}}\left(\mathbf{C} - \mathbf{f}^{(\ell)} \mathbf{1}_m^{\mathrm{T}}\right) + \varepsilon \log \mathbf{b} \tag{47}$$

After that we use a little stable trick

$$\min_{\varepsilon} \mathbf{z} = \underline{z} - \varepsilon \log \sum_i e^{-(\mathbf{z}_i - \underline{z})/\varepsilon} \tag{48}$$

where $\underline{z} = \min \mathbf{z}$. Instead of $\underline{z}$, we use former iteration's value $\mathbf{f}^{(\ell)}$, then we get 42.

3.7 More on Sinkhorn

Following [1] [2] [3] [4], the 14 could be think of as a special case of the following convex optimization problem

$$\min_{\mathbf{P}} \sum_{i,j} \mathbf{C}_{i,j}\mathbf{P}_{i,j} - \varepsilon \mathbf{H}(\mathbf{P}) + \iota_{\{\mathbf{a}\}}\left(\mathbf{P}\mathbf{1}_m\right) + \iota_{\{\mathbf{b}\}}\left(\mathbf{P}^{\mathrm{T}}\mathbf{1}_n\right) \tag{49}$$

where ι_C means the indicator of set C, which is

$$\iota_{\mathcal{C}}(x) = \begin{cases} 0 & \text{if} \quad x \in \mathcal{C} \\ +\infty & \text{otherwise} \end{cases} \tag{50}$$

Under this situation, the Sinkhorn-Knopp iteration can be extended as

$$\mathbf{u} \leftarrow \frac{\operatorname{Prox}_F^{\mathrm{KL}}(\mathbf{K}\mathbf{v})}{\mathbf{K}\mathbf{v}} \tag{51}$$

$$\mathbf{v} \leftarrow \frac{\operatorname{Prox}_G^{\mathbf{KL}}\left(\mathbf{K}^{\mathrm{T}}\mathbf{u}\right)}{\mathbf{K}^{\mathrm{T}}\mathbf{u}} \tag{52}$$

where the proximal operator for KL divergence is defined as

$$\operatorname{Prox}_F^{\mathbf{KL}}(\mathbf{u}) = \underset{\mathbf{u}' \in \mathbb{R}_+^N}{\operatorname{argmin}} \mathbf{KL}\left(\mathbf{u}'|\mathbf{u}\right) + F\left(\mathbf{u}'\right), \forall \mathbf{u} \in \mathbb{R}_+^N \tag{53}$$

(an extention of normal proximal operator)

4 Numerical Result and Interpretation

4.1 Description of datasets

In order to compare the performance of differnet algorithms, we have to use some classic and challenging datasets.

In general, the i-th datapoint can be denoted as (x_i, μ_i) , where $x_i \in \mathbb{R}^d$ is the position of datapoint and μ_i is the probability at x_i.

For convenience, we assume that datapoints are followed 2D distribution (i.e. $x_i \in \mathbb{R}^2$). Besides, we use the squared Euclidean distance to define the cost matrix between two datasets $\{(x_i, \mu_i)\}_{i=1}^m$ and $\{(y_j, \nu_j)\}_{j=1}^n$ as following

$$c_{ij} = ||x_i - y_j||_2^2 \quad \forall i, j \tag{54}$$

We have tested our algorithms on four types of datasets

- Randomly generated dataset
  The position are uniformly sampled from $[0,1] \times [0,1]$. The weights μ and ν are randomly sampled from $[0,1]$ and scaled to $\sum_{i=1}^m \mu_i = \sum_{j=1}^n \nu_j = 1$.
- ellipses [**Gerber2017**]
  The ellipse example consists of two uniform samples (source and target data set) of size $m = n$ from the unit circle with normal distributed noise added with zero mean and standard deviation 0.1. The source data sample is then

scaled in the x-Axis by 0.5 and in the y-Axis by 2, while the target data set is scaled in the x-Axis by 2 and in the y-Axis by 0.5. Besides, the weights are both normalized uniform distributions.

- Caffarelli [**Gerber2017**]
  Caffarelli's example consists of two uniform samples (source and target data set) on $[-1, 1] \times [-1, 1]$ of size $m = n$. Any points outside the unit circle are then discarded. Additionally, the target data sample is split along the x-Axis at 0 and shifted by $+2$ and -2 for points with positive and negative x-Axis values, respectively. The weights are both normalized uniform distributions, too.

- DOTmark [**Schrieber2017**]
  In DOTmark, we always have $m = n = r^2$, and $(x_i)_{1\leq i\leq m} = (y_j)_{1\leq j\leq n}$ form a regular square grid of resolution $r \times r$ in $\mathbb{R}^2$, which are the natural position of source and target data set. The weights are the brightness distributions with normalization. Besides, DOTmark consists of 10 classes of 10 different images, each of which is available at the 5 different resolutions from 32×32 to 512×512 (in doubling steps per dimension).

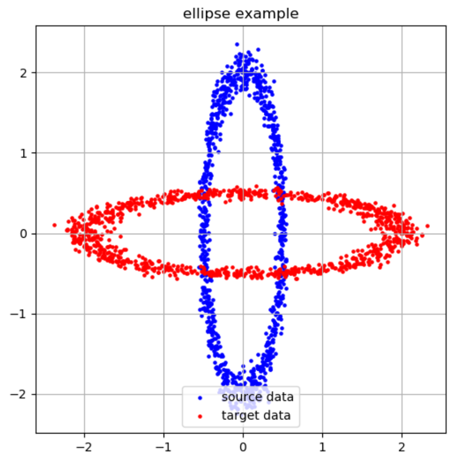

Figure 2: $m = n = 1000$, ellipse example

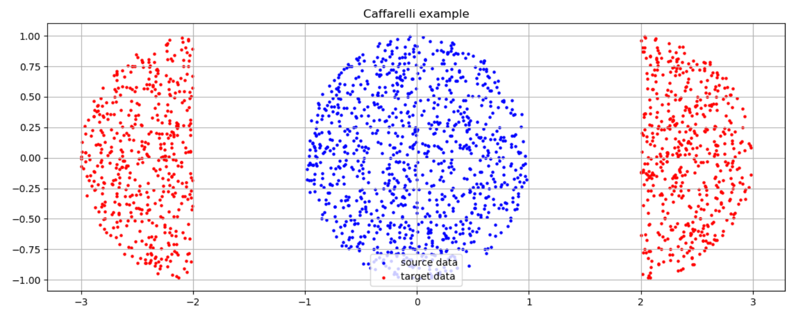

Figure 3: $m = n = 1000$, Caffarelli's example

#	Name
1	WhiteNoise
2	GRFrough
3	GRFmoderate
4	GRFsmooth
5	LogGRF
6	LogitGRF
7	CauchyDensity
8	Shapes
9	ClassicImages
10	Microscopy

Table 1: The 10 classes in the DOTmark

4.2 Numerical result

$m = n$		M prim	M dual	M int	G prim	G dual	G int	ADMM prim	Sinkhorn
256	dist	5.45e-3	7.25e-3	8.09e-3	5.94e-3	8.62e-3	5.13e-3	5.04e-3	8.64e-3
	time	3.43e-1	3.75e-1	5.28e-1	6.79e-1	7.07e-1	8.56e-1	26.82	4.19
	err μ	1.79e-5	2.04e-17	2.31e-12	1.34e-17	9.97e-18	4.34e-18	4.67e-5	7.01e-3
	err ν	1.79e-5	2.45e-16	2.79e-12	5.36e-16	3.87e-16	5.05e-16	4.74e-5	8.54e-17
512	dist	2.90e-3	2.88e-3	3.37e-3	3.79e-3	2.91e-3	5.52e-3	3.13e-3	3.61e-3
	time	1.39	1.70	2.93	2.83	3.40	4.27	1.03e+2	25.8
	err μ	4.35e-5	2.67e-17	1.30e-13	1.08e-17	1.09e-15	1.21e-17	8.40e-5	8.65e-3
	err ν	4.35e-5	1.09e-16	1.04e-13	1.72e-16	1.26e-17	7.46e-16	8.39e-5	1.20e-16
1024	dist	1.78e-3	1.94e-3	1.89e-3	1.91e-3	1.80e-3	1.84e-3	2.94e-3	2.79e-3
	time	8.11	15.3	15.2	14.1	14.0	18.7	4.52e+2	1.53e+2
	err μ	1.04e-4	1.09e-16	1.38e-12	1.01e-17	1.12e-17	1.91e-15	2.27e-4	1.07e-2
	err ν	1.04e-4	1.91e-15	1.27e-12	6.49e-16	3.42e-16	9.43e-18	2.20e-4	1.18e-16
2048	dist	1.35e-3	1.36e-3	1.31e-3	1.42e-3	1.46e-3	1.49e-3	1.48e-3	1.06e-3
	time	35.0	1.05e+2	68.7	3.45e+2	58.4	88.9	8.74e+3	1.99e+3
	err μ	2.71e-4	2.61e-16	1.81e-12	9.41e-18	6.09e-16	4.95e-16	8.99e-4	3.93e-3
	err ν	2.7e-4	1.79e-15	2.27e-12	1.27e-15	1.17e-17	1.13e-17	8.40e-4	1.48e-16

Table 2: Numerical result of random dataset

$m = n$		M prim	M dual	M int	G prim	G dual	G int	ADMM prim	Sinkhorn
256	dist	2.37	2.27	1.88	2.28	2.12	2.26	2.22	2.41
	time	5.42e-1	9.80	4.10	6.57e-1	1.49	7.22e-1	37.5	20.8
	err μ	1.76e-5	9.54e-18	1.65e-12	0	0	0	1.42e-5	6.98e-17
	err ν	1.76e-5	9.54e-18	1.65e-12	0	0	0	1.40e-5	4.54e-16
512	dist	2.36	2.39	2.24	2.15	2.31	2.32	2.47	2.38
	time	1.65	5.01	1.49	4.87	4.62	4.01	2.48e+2	72.2
	err μ	4.05e-5	2.04e-17	2.01e-11	0	0	0	3.39e-5	1.05e-16
	err ν	4.05e-5	1.95e-17	2.01e-11	0	0	0	3.40e-5	6.26e-16
1024	dist	2.24	2.11	2.21	2.24	2.18	2.27	2.25	2.39
	time	7.89	70.4	9.73	64.5	21.9	16.6	8.41e+2	2.68e+2
	err μ	9.39e-5	1.59e-16	5.37e-10	0	0	0	1.02e-4	1.25e-16
	err ν	9.39e-5	1.48e-16	5.37e-10	0	0	0	1.02e-4	6.47e-16
2048	dist	2.38	2.23	2.28	2.27	2.22	2.37	2.31	*
	time	32.6	4.37e+2	47.9	1.16e+3	8.24e+2	81.9	7.88e+3	*
	err μ	2.42e-4	1.29e-16	1.52e-13	0	0	0	4.67e-3	*
	err ν	2.42e-4	1.31e-16	1.40e-13	0	0	0	4.65e-3	*

Table 3: Numerical result of ellipse example

$m = n$		M prim	M dual	M int	G prim	G dual	G int	ADMM prim	Sinkhorn
256	dist	4.06	3.91	4.03	4.08	3.86	4.15	4.02	4.24
	time	2.82e-1	8.80	6.54e-1	6.80e-1	1.96	7.63e-1	55.2	12.4
	err μ	1.67e-5	2.78e-17	1.98e-12	0	0	0	4.07e-6	6.46e-17
	err ν	1.67e-5	2.78e-17	1.98e-12	0	0	0	4.03e-6	4.59e-16
512	dist	3.95	3.97	4.05	4.10	3.97	3.96	3.99	4.21
	time	1.88	5.37	3.60	3.50	3.61	4.63	4.00e+2	50.3
	err μ	4.14e-5	7.55e-17	5.79e-12	0	0	0	4.03e-5	9.84e-17
	err ν	4.14e-5	7.59e-17	5.79e-12	0	0	0	4.08e-5	6.09e-16
1024	dist	3.99	4.04	4.09	4.00	4.00	3.95	3.97	4.35
	time	6.01	49.9	14.8	29.9	18.7	22.8	1.52e+3	1.72e+2
	err μ	9.80e-5	1.40e-16	8.04e-12	0	0	0	4.42e-4	1.33e-16
	err ν	9.80e-5	1.40e-16	8.12e-12	0	0	0	4.43e-4	6.29e-16
2048	dist	3.97	3.99	4.06	3.99	3.97	4.00	4.04	4.13
	time	27.9	4.93e+2	58.0	7.47e+2	1.02e+2	1.02e+1	1.06e+4	8.44e+3
	err μ	2.54e-4	3.95e-16	2.53e-13	0	0	0	8.60e-3	1.83e-16
	err ν	2.54e-4	3.96e-16	2.49e-13	0	0	0	8.60e-3	9.74e-16

Table 4: Numerical result of Caffarelli's example

Due to the limited time, we only tested a randomly chosen pair of images from each class with size 32×32, whose corresponding cost matirx is 1024×1024.

#		M prim	M dual	M int	G prim	G dual	G int	ADMM prim	Sinkhorn
1	dist	6.93e-4	6.93e-4	6.93e-4	6.93e-4	6.93e-4	6.93e-4	6.92e-4	6.62e-4
	time	9.50	10.0	11.9	15.6	11.3	1.73e+2	1.46e+3	1.49e+2
	err μ	1.02e-4	1.59e-16	3.39e-12	0	0	0	1.01e-4	7.90e-3
	err ν	1.02e-4	1.19e-9	1.20e-9	1.19e-9	1.19e-9	1.19e-9	1.01e-4	6.63e-17
2	dist	1.44e-3	1.44e-3	1.44e-3	1.44e-3	1.44e-3	1.44e-3	1.44e-3	1.27e-3
	time	7.34	15.7	10.3	14.4	14.4	26.5	1.59e+3	1.51e+2
	err μ	9.91e-5	1.10e-16	6.47e-13	0	0	2.58e-9	1.72e-4	1.15e-2
	err ν	9.91e-5	2.58e-9	2.58e-9	2.58e-9	2.58e-9	0	1.68e-4	1.29e-16
3	dist	3.98e-3	3.98e-3	3.98e-3	3.98e-3	3.98e-3	3.98e-3	3.98e-3	3.99e-3
	time	5.55	20.4	10.5	13.64	13.2	18.49	1.48e+3	1.52e+2
	err μ	9.84e-5	1.89e-16	7.38e-14	0	5.70e-10	0	1.16e-4	1.12e-3
	err ν	9.84e-5	5.70e-10	5.70e-10	5.70e-10	0	0	1.63e-4	1.82e-16
4	dist	2.09e-2	2.09e-2	2.09e-2	2.09e-2	2.09e-2	2.09e-2	2.09e-2	2.20e-2
	time	5.56	31.8	9.55	13.9	15.9	22.1	1.44e+3	1.25e+2
	err μ	9.78e-5	1.67e-16	9.80e-12	0	0	0	1.65e-4	1.23e-9
	err ν	9.78e-5	9.89e-8	1.24e-9	1.23e-9	1.23e-9	1.23e-9	1.63e-4	3.40e-16
5	dist	1.87e-2	1.87e-2	1.87e-2	1.87e-2	1.87e-2	1.87e-2	1.87e-2	1.97e-2
	time	5.99	20.4	10.3	14.2	16.8	29.4	1.54e+3	1.32e+2
	err μ	1.00e-4	1.55e-16	2.93e-12	0	0	0	1.16e-4	1.70e-9
	err ν	1.00e-4	1.59e-9	1.59e-9	1.59e-9	8.56e-10	1.59e-9	1.35e-4	3.07e-16
6	dist	1.65e-2	1.65e-2	1.65e-2	1.65e-2	1.65e-2	1.65e-2	1.65e-2	1.74e-2
	time	6.43	18.1	13.7	13.4	16.8	19.1	1.42e+3	1.31e+2
	err μ	1.00e-4	1.80e-16	4.20e-10	0	0	0	1.17e-4	4.11e-8
	err ν	1.00e-4	8.56e-10	1.28e-9	1.16e-9	8.56e-10	8.56e-10	1.20e-4	2.74e-16
7	dist	1.71e-2	1.71e-2	1.71e-2	1.71e-2	1.71e-2	1.71e	1.71e-2	1.80e-2
	time	8.66	29.7	12.5	13.4	18.0	26.4	1.54e+3	91.3
	err μ	1.09e-4	1.19e-16	4.09e-10	0	0	0	5.64e-4	1.16e-9
	err ν	1.09e-4	1.16e-9	1.57e-9	1.16e-9	1.16e-9	1.16e-9	4.35e-4	2.89e-16
8	dist	2.38e-2	2.38e-2	2.38e-2	2.38e-2	2.38e-2	2.38e-2	2.38e-2	2.50e-2
	time	5.17	6.84	6.33	13.3	12.2	12.2	1.56e+3	1.41e+2
	err μ	6.52e-5	1.17e-16	4.72e-11	0	0	0	8.20e-4	2.24e-8
	err ν	6.53e-5	2.24e-8	2.25e-8	2.24e-8	2.24e-8	2.24e-8	2.94e-4	4.48e-16
9	dist	6.12e-3	6.12e-3	6.12e-3	6.12e-3	6.12e-3	6.12e-3	6.13e-3	6.20e-3
	time	5.71	17.56	12.9	15.1	13.2	18.4	1.62e+3	1.51e+2
	err μ	9.90e-5	1.61e-16	9.08e-13	0	2.18e-11	0	1.16e-4	2.18e-11
	err ν	9.90e-5	2.18e-11	2.26e-11	2.18e-11	0	2.18e-11	1.21e-4	5.55e-16
10	dist	1.06e-2	1.06e-2	1.06e-2	1.06e-2	1.06e-2	1.06e-2	1.06e-2	1.09e-2
	time	4.20	7.32	6.00	12.8	13.9	20.9	1.71e+3	1.51e+2
	err μ	7.01e-5	1.16e-16	2.40e-11	0	0	5.94e-9	1.20e-4	1.23e-6
	err ν	7.01e-5	5.94e-9	5.95e-9	5.94e-9	5.94e-9	0	1.76e-4	2.24e-16

Table 5: Numerical result of 10 classes in the DOTmark

5 Conclusion and More

We have tried several first-order and second-order methods which have good performance in the context of discrete optimal transport, including ADMM-based methods and Entropy regularization-based methods, and both have their own merits and drawbacks. We run thorough experiments to test their performances and make comprehensive comparisons in many aspects in **??** .

As for our next move, we aim to refine some of our existing algorithms to have a more stable performance and higher convergence rate. For some of our algorithms, we find some empirical tuning skills and we will consider their inner relationships with the iteration process. We also have ambition to test multi-stage iteration step on dual of ADMM and other methods, holding the believe that on different scale we need to provide different parameters.

Due to the limited time and equipment, we don't test our algorithms at all scale of DOTmark dataset. At larger scale, there may be a need for different methods, which is to be explored by us.

Acknowledgments

We would like to thank Zhihan Li, Haochen Gan and Jason Jia for their useful instruction.

References

[1] Karlsson, J., & Ringh, A. (2017). Generalized Sinkhorn iterations for regularizing inverse problems using optimal mass transport. SIAM Journal on Imaging Sciences, 10(4), 1935-1962.

[2] Peyré, G. (2015). Entropic approximation of Wasserstein gradient flows. SIAM Journal on Imaging Sciences, 8(4), 2323-2351.

[3] Frogner, C., Zhang, C., Mobahi, H., Araya, M., & Poggio, T. A. (2015). Learning with a Wasserstein loss. In Advances in Neural Information Processing Systems (pp. 2053-2061).

[4] Chizat, L., Peyré, G., Schmitzer, B., & Vialard, F. X. (2016). Scaling algorithms for unbalanced transport problems. arXiv preprint arXiv:1607.05816.

[5] L. Ambrosio. Lecture notes on optimal transport problems. In Mathematical Aspects of Evolving Interfaces, volume 1812 of Lecture Notes in Mathematics, pages 1–52. Springer, 2003

[6] Bertsekas, D. P. (1992). Auction algorithms for network flow problems: A tutorial introduction. Computational optimization and applications, 1(1), 7-66.

[7] Schrieber, J., Schuhmacher, D., & Gottschlich, C. (2017). Dotmark–a benchmark for discrete optimal transport. IEEE Access, 5, 271-282.

[8] Villani, C. (2008). Optimal transport: old and new (Vol. 338). Springer Science & Business Media.

[9] Gerber, S., & Maggioni, M. (2017). Multiscale strategies for computing optimal transport. The Journal of Machine Learning Research, 18(1), 2440-2471.

[10] Cominetti, R., & San Martín, J. (1994). Asymptotic analysis of the exponential penalty trajectory in linear programming. Mathematical Programming, 67(1-3), 169-187.

[11] Brauer, C., Clason, C., Lorenz, D., & Wirth, B. (2017). A Sinkhorn-Newton method for entropic optimal transport. arXiv preprint arXiv:1710.06635.

[12] Sinkhorn, R., & Knopp, P. (1967). Concerning nonnegative matrices and doubly stochastic matrices. Pacific Journal of Mathematics, 21(2), 343-348.

[13] Peyré, G., & Cuturi, M. (2017). Computational optimal transport (No. 2017-86).