Computational Optimal Transport – Project report

Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration

Soufiane Fafe
École polytechnique, ENS Paris-Saclay

January 7, 2022

Abstract

Optimal Transport distances are very important quantities that are used in various fields. Since they can measure probability densities on a non-parametric setting. This is why we are interested in computing such distances which is a Linear Program with $O(n)$ linear constraints. Hence, it can be approximated in $O(n^3 \log n)$. This complexity is too high when considering a data that is coming from fields such as Machine learning where $n \geq 10^5$. In order to speed up the computations one can tolerate an ε approximation of OT distance. In this report, we study theoretically and numerically the work of Altschuler et Al. [1] in which they gave a quadratic time approximation of OT distance. Hence this affirmatively answers the problem of existence of a near linear time approximation for OT by using a new analysis to the Sinkhorn's Algorithm.

1 Introduction

Optimal Transport is a mathematical field that aim to study the best way to transport a set of points to another set with respect to a cost. It was first formoluted by G. Monge in 1781, but the major advances were made by L. Kantorovich in 1942 by relaxing the deterministic nature of the transportation as the following :

$$\min_{P \in \mathcal{U}_{a,b}} < P, C >, \qquad \mathcal{U}_{a,b} := \left\{ P \in \mathbb{R}_+^{n \times n} : P\mathbf{1}_n = a, P^\top \mathbf{1}_n = b \right\} \tag{1}$$

Which is a Linear Program, hence it can be exactly solved at $O(n^3)$. It is very expensive to deal with Data coming from field such Machine Learning where the number of the data n is so high ($n > 1e5$). This is why we are interested in an approximation of 1, where one can trade-off the optimality with speed.

Related Work

Notice that 1 is a linear program with $O(n)$ linear constraint, hence one can use the recent Lee-Sidford algorithm to find a solution in time $\tilde{O}(n^{2.5})$. While no practical Lee-Sidford algorithm is known, it provides a theoretical reference to [1] since they propose a quadratic time approximation to 1.

For practical algorithm to compute OT distances, one can use Orlin's algorithm for the *Incapacitated Minimum Cost Flow* problem via a standard reduction. Which it has a complexity of $O(n^3 \log n)$ like interior point methods.

Applications

Recently, OT distance 1 has been used in several fields such that : Deep-Learning, Statistics, etc. The Wasserstein Distance (for the definition see below) is the best distance to compare probability distributions in a non-parametric seeting. Since it costs time to compute it exactly. This is why we choose to trade-off the quality of the solution with running time by approximating it. This quantity is usually estimated by using Sinkhorn's iteration on the regularized problem by adding an entropic penalty.

New Contributions

In this work, we are interested in the approximation given by Altschuler et Al. in their work [1] for the Optimal Transport Distance by using a new analysis to the Sinkhorn's Algorithm. We gave in this report the main theorem as well as the algorithm and the key idea behind proofs used to improve the complexity of Sinkhorn algorithm. We also stress the case of the Wasserstein where Peyré et Al. [3] gave a Sinkhorn's iteration that evaluates directly the Dual of 1. Thus, they do not round the project matrix as it is done in [1].
Altschuler et Al. gave a new algorithm to approximate 1 called Greedy Sinkhorn = Greenkhorn. In this report we compare the performance of the two algorithm used in [1]. We also test the choice of ε given in [1] in Gaussian distributions in many dimensions.

2 Optimal Transport in Quadratic Time

In his section, we state the main theorem of [1], where the complexity of giving an ϵ approximation for 1 is $O(n^2\epsilon^{-3})$. We give also the key idea of the proof.

2.1 Regularized Kantorovich Problem

In order to solve efficiently the discrete Kantorovich problem 1, we need to introduce an enthropic penalty for 1. The Enthropy is defined as the following :

$$H(p) := \sum_{i=1}^{n} p_i \log(\frac{1}{p_i}) \tag{2}$$

Where $p \in \Delta_n := \left\{x \in \mathbb{R}_+^n : \sum_{i=1}^n x_i = 1\right\}$. For $p, q \in \Delta_n$, we define the Kullback-Leibler divergence $\mathcal{K}(p||q)$ between p and q by :

$$\mathcal{K}(p||q) := \sum_{i=1}^{n} p_i \log(\frac{p_i}{q_i}) \tag{3}$$

For a matrix P, we define the entropy $H(P)$ as $\sum_{i,j} P_{i,j} \log(\frac{1}{P_{i,j}})$.

We define the Entropic Regularization for Kantorovich problem 1 by :

$$\min_{P \in \mathcal{U}_{a,b}} < P, C > -\frac{1}{\lambda} H(P) \tag{4}$$

We note the solution of 4 by P_λ where :

$$P_\lambda := \operatorname*{argmin}_{P \in \mathcal{U}_{a,b}} < P, C > -\frac{1}{\lambda} H(P) \tag{5}$$

2.2 Sinkhorn's algorithm

Lemma 2.1. *[1] For any cost matrix C non-negative and $a, b \in \Delta_n$, the minimization program 4 has a unique minimum at $P_\lambda \in \mathcal{U}_{a,b}$ of the form $P_\lambda = diag(u)Kdiag(v)$, where $A = (e^{-\lambda C_{i,j}})_{1 \leq i,j \leq n}$ and $u, v \in \mathbb{R}_+^n$. The vectors (u, v) are unique up to a constant factor.*

Proof. The key idea of the proof is to use the KKT conditions for 4 by writing down the Lagrangien. For the unicity, we can clearly see that 4 is strictly convex. □

In order to solve 4, Sinkhorn proposed a simple iterative algorithm, which has the following simple updates :

$$u^{(l+1)} := \frac{a}{Kv^{(l)}} \tag{6}$$

$$v^{(l+1)} := \frac{b}{K^T u^{(l+1)}} \tag{7}$$

Where the initialization can be done with an arbitrary positive vector, for example we take $v^{(0)} = \mathbf{1}_n$. The division operator used above between two vectors is an entry-wise division.
For the convergence of Sinkhorn's algorithm, we refer the reader to [2].

2.3 Sinkhorn projection

By using the Kulbback-Leibler divergence, we can reformulate 4 as the following :

$$\min_{P \in \mathcal{U}_{a,b}} < P, C > + \frac{1}{\lambda} \mathcal{K}(P||a \otimes b) \tag{8}$$

8 is equivalent to 4 which is equivalent [2] to :

$$\min_{P \in \mathcal{U}_{a,b}} \mathcal{K}(P||K) \tag{9}$$

Hence, Sinkhorn's algorithm can be seen as the projection onto $\mathcal{U}_{a,b}$ of the Gibbs Kernel K.

2.4 Approximate Sinkhorn Approximation

In order to have a speed convergence of the projection of Gibbs Kernel onto $\mathcal{U}_{a,b}$, [**?**] replace the exact projection by an ε approximate projection which satisfy the condition $||B\mathbf{1}_n - a||_1 + ||B^T\mathbf{1}_n - b||_1 \leq \epsilon$ where $B = diag(u)Kdiag(v)$ the output of the approximate algorithm of the projection giving by [1].

2.5 Main theorem and analysis of the proof

[1] gave an approximation algorithm 2 that output an ε approximation for 1.
The given algorithm is as the following :
The rounding step on the algorithm 2 is as the following :

Theorem 2.2. *Algorithm 2 returns a point $\hat{P} \in \mathcal{U}_{a,b}$ satisfying :*

$$< \hat{P}, C > \leq \min_{P \in \mathcal{U}_{a,b}} < P, C > + \varepsilon$$

in time $O(n^2 L^3 (\log n) \varepsilon^{-3})$ if $||C||_\infty \leq L$.

Proof. The proof of the theorem is based on the theorem 2.3

Algorithm 1 Sinkhorn $(K, \mathcal{U}_{a,b}, \varepsilon)$

Initialize: $k \leftarrow 0$
$A^{(0)} \leftarrow K/\|K\|_1$, $x^0 \leftarrow \mathbf{0}$, $y^0 \leftarrow \mathbf{0}$
while dist $\left(K^{(k)}, \mathcal{U}_{a,b}\right) > \varepsilon$ **do**
if k odd **then**
$x_i \leftarrow \log \frac{a_i}{\left(K^{(k-1)}\mathbf{1}_n\right)_i}$ for $i \in [n]$
$x^k \leftarrow x^{k-1} + x$, $y^k \leftarrow y^{k-1}$
else
$y \leftarrow \log \frac{b_j}{\left(\mathbf{1}_n^T K^{(k-1)}\right)_j}$ for $j \in [n]$
$y^k \leftarrow y^{k-1} + y$, $x^k \leftarrow x^{k-1}$
end if
$K^{(k)} = \mathrm{Diag}\left(\exp\left(x^k\right)\right) K \mathrm{Diag}\left(\exp\left(y^k\right)\right)$
end while
Output: $B \leftarrow K^{(k)}$

Algorithm 2 APPROXOT(C, a, b, ε)

$\lambda \leftarrow \frac{4\log(n)}{\varepsilon}$ $\varepsilon^{'} \leftarrow \frac{\varepsilon}{8\|C\|_\infty}$
\\ Step 1 : Approximately project onto $\mathcal{U}_{a,b}$
1: $K \leftarrow \exp(-\eta C)$
2: $B \leftarrow PROJ(K, \mathcal{U}_{a,b}, \varepsilon^{'})$
\\ Step 2 : Round to feasible point in $\mathcal{U}_{a,b}$
Output: $\hat{P} \leftarrow ROUND(B, \mathcal{U}_{a,b})$

Algorithm 3 $ROUND(F, \mathcal{U}_{a,b})$

1: $X \leftarrow Diag(x)$ with $x_i = \frac{a_i}{(F\mathbf{1}_n)_i} \wedge 1$
2: $F^{'} \leftarrow XF$
3: $Y \leftarrow Diag(y)$ with $y_j = \frac{b_j}{(F^T\mathbf{1}_n)_j} \wedge 1$
4: $F^{"} \leftarrow F^{'}Y$
5: $err_a \leftarrow a - F^{"}\mathbf{1}_n$, $err_b \leftarrow b - F^{"T}\mathbf{1}_n$
6: Output $G \leftarrow F^{''} + err_a err_b^T / ||err_a||_1$

2.6 New analysis of the Sinkhorn algorithm

Before [1], the best analysis of the algorithm 1 showed that $\tilde{O}((\varepsilon^{'})^{-2})$ iterations suffice to obtain a matrix close to $\mathcal{U}_{a,b}$ in ℓ_2 distance, ie : $||K\mathbf{1}_n - a||_2 + ||K^T\mathbf{1}_n - b||_2 \leq \varepsilon^{'}$.

In order to have best analysis of the algorithm 2, [1] used the norm ℓ_1 instead of ℓ_2 since it is not an appropriate measure of closeness between probability.
The new bound found by [1] is independent from the dimension.

Theorem 2.3. *Algorithm 1 with* $dist(K,\mathcal{U}_{a,b}) = ||K\mathbf{1}_n - a||_1 + ||K^T\mathbf{1}_n - b||_1$ *outputs a matrix* B *satisfying* $dist(B,\mathcal{U}_{a,b}) \leq \varepsilon^{'}$ *in* $O((\varepsilon^{'})^{-2}\log(s/\ell))$ *iterations, where* $s = \sum_{i,j} K_{i,j}$ *and* $\ell = \min_{i,j} K_{i,j}$.

2.7 The squared Wasserstein distance

When the cost matrix C is derived from the squared-distance $c(x,y) = \frac{1}{2}||y-x||_2^2$, we can defined the Wasserstein distance between two probability densities μ, ν by $\mathcal{W}_2^2(\mu,\nu) = T_0(\mu,\nu)$ where :

$$T_\lambda(\mu,\nu) := \min_{\gamma\in\Pi(\mu,\nu)} \int_{\left(\mathbb{R}^d\right)^2} \frac{1}{2}\|y-x\|_2^2 \, \mathrm{d}\gamma(x,y) + \lambda \, \mathcal{K}(\gamma||\mu\otimes\nu) \tag{10}$$

Where $\Pi(\mu,\nu) = \left\{\gamma | \int_{y\in\mathbb{R}^d} \mathrm{d}\gamma(.,y) = \mu(.) \& \int_{x\in\mathbb{R}^d} \mathrm{d}\gamma(x,.) = \nu(.)\right\}$.

When μ and ν are discrete measures, the problem 10 is exactly the problem 4

Instead of using algorithm 2 to solve the problem 10. [3] evaluate directly the dual of the problem 10 and propose an algorithm that outputs an ε approximation within the same run time as [1]. The advantage of this procedure is that does not use the rounding step.

2.7.1 Dual of Program 10

By using the Lagrangian we found that the dual of the program 10 as the following :

$$F_\lambda(u,v) = \int_{\mathbb{R}^d} u \, \mathrm{d}\mu + \int_{\mathbb{R}^d} v \, \mathrm{d}\nu + \lambda\left(1 - \int_{\left(\mathbb{R}^d\right)^2} \exp((u(x)+v(y)-c(x,y))/\lambda)\mathrm{d}\mu(x)\mathrm{d}\nu(y)\right) \tag{11}$$

The strong duality holds, hence we have :

$$T_\lambda(\mu,\nu) = \max_{u,v} F_\lambda(u,v) \tag{12}$$

2.7.2 Sinkhorn update for Program 10

Sinkhorn iteration for program 10 becomes :
$u_0 = 0 \; v_0 = 0$

$$u_{k+1} = u_k - \lambda \log \int_{\mathbb{R}^d} \exp\left(\left(u_k(\cdot) + v_k(y) - c(\cdot,y)\right)/\lambda\right) \mathrm{d}\nu(y), \quad v_{k+1} = v_k \quad \text{if } k \text{ is odd} \tag{13}$$

$$v_{k+1} = v_k - \lambda \log \int_{\mathbb{R}^d} \exp\left(\left(u_k(x) + v_k(\cdot) - c(x,\cdot)\right)/\lambda\right) \mathrm{d}\mu(x), \quad u_{k+1} = u_k \quad \text{if } k \text{ is even} \tag{14}$$

When μ and ν are discrete, the update becomes:

$$u_{k+1}^i = u_k^i - \lambda \log \sum_{j=1}^{n} \exp\left(\left(u_k^i + v_k^j - C_{i,j}\right)/\lambda\right) \forall i, \quad v_{k+1} = v_k \quad \text{if } k \text{ is odd} \tag{15}$$

$$v_{k+1}^j = v_k^j - \lambda \log \sum_{i=1}^{n} \exp\left(\left(u_k^i + v_k^j - C_{i,j}\right)/\lambda\right) \forall j, \quad u_{k+1} = u_k \quad \text{if } k \text{ is even} \tag{16}$$

where u_k^i is the i-th entry of the vector u_k. The same for v_k^j.

Proposition 2.3.1. *Assume that μ and ν are discrete measures with n atoms s.t $p_i, q_j \geq \alpha/n$ for some α. If we fix $\lambda = \varepsilon/4(\log n + \log(1/\alpha))$, then the Sinkhorn's algorithm returns an ε-accurate estimation of T_λ in time $O(n^2 \log n ||c||_\infty/\varepsilon^2)$*

2.8 Greedy Sinkhorn

[1] propose in their article a new algorithm to do the projection of matrix K onto $\mathcal{U}_{a,b}$ named Greddy Sinkhorn = Grennkhorn. This new algorithm have the same convergence guarantee as Sinkhorn but practically it performs better as we show on the numerical tests (see below).

The performance of Greenkhorn came from replacing the phase of updating all rows and columns in Sinkhorn's algorithm to updating a single row or column at each step. Thus updates only $O(n)$ entries of K per iteration instead of $O(n^2)$.

Greenkhorn can be seen as stochastic version of Sinkhorn where choosing a row or column to update is not random. Greenkhorn chooses the best row or column to update greedily. It does an exact line search on the coordinates since there is a simple close form for the optimum.

Algorithm 4 *Greenkhorn*$(A, \mathcal{U}_{a,b}, \varepsilon^{'})$

$K^{(0)} \leftarrow K/||K||_1$, $x \leftarrow \mathbf{0}$, $y \leftarrow \mathbf{0}$
$K \leftarrow K^{(0)}$
while $dist(K, \mathcal{U}_{a,b}) > \epsilon^{'}$
  $I \leftarrow \text{argmax}_i \, \rho\left(a_i, (K\mathbf{1}_n)_i\right)$
  $J \leftarrow \text{argmax}_j \, \rho\left(b_j, (K^T\mathbf{1}_n)_j\right)$
  if $\rho\left(a_I, (K\mathbf{1}_n)_I\right) > \rho\left(b_J, (K^T\mathbf{1}_n)_J\right)$ **then**
    $x_I \leftarrow x_I + \log \frac{a_I}{(K\mathbf{1}_n)_I}$
  else
    $y_J \leftarrow y_J + \log \frac{b_J}{(K^T\mathbf{1}_n)_J}$
  $K \leftarrow \mathbf{Diag}(\exp(x)) K^{(0)} \mathbf{Diag}(\exp(y))$
Output $B \leftarrow K$

Where the distance function ρ is defined by:

$$\rho(a, b) = b - a + a \log \frac{a}{b} \quad \forall a, b > 0$$

3 Numerical Results

3.1 Greenkhorn vs Sinkhorn

3.1.1 Description of dataset

In order to compare Sinkhorn and Greenkhorn's algorithm. We used the same datasets of [1]. The first dataset is synthetic images as the following :

Figure 1: Synthetic images

The second one is from MNIST.
The cost matrix C is the matrix of ℓ_1 distances between pixel locations.

3.1.2 Performance of Greenkhorn and Sinkhorn

For the first dataset we have the following results :
The dimension of the synthetic image is $25 \times 25 = 625$.
As we see clearly on the performance of Greenkhorn and Sinkhorn. Despite the fact that they have the same theoritical guarantees, practically Greenkhorn is better that Sinkhorn.

3.2 Testing the choice of the ε and λ

We have seen that in order to have an ε approximation for OT problem. λ must be chosen as the following :

$$\lambda = 4 \log n / \varepsilon$$

Figure 2: MNIST image

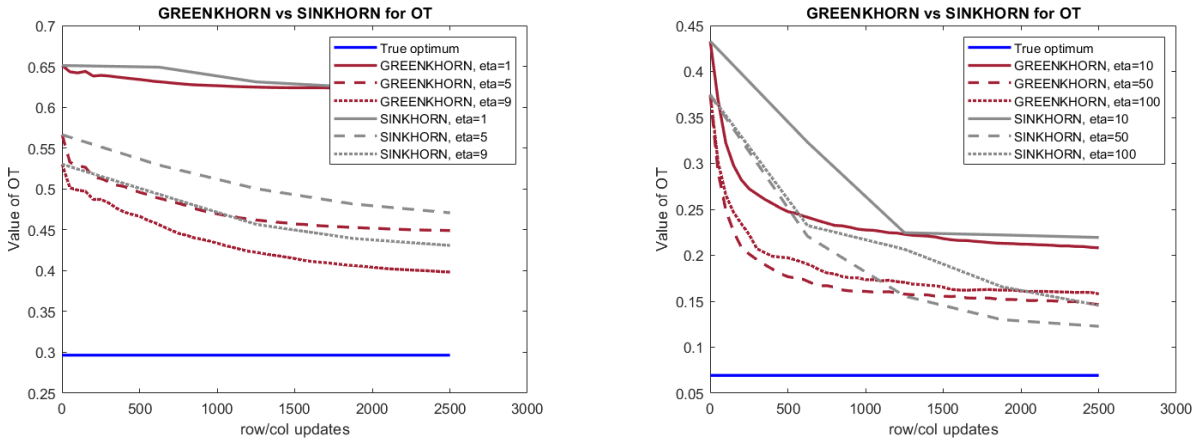

Figure 3: Performance of Greenkhorn and Sinkhorn for different values of λ = eta for the first dataset

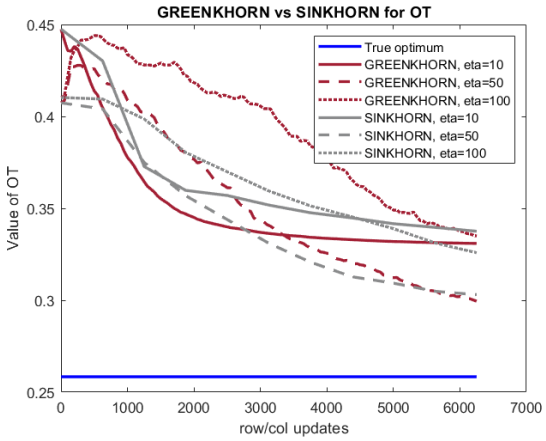

Figure 4: Performance of Greenkhorn and Sinkhorn for different values of λ = eta for the second dataset

3.2.1 Description of the dataset

The code for numerical results is available on my GitHub page [1].
In order to test the validation of the relation between λ and ε, we used Gaussian distributions in different dimension.

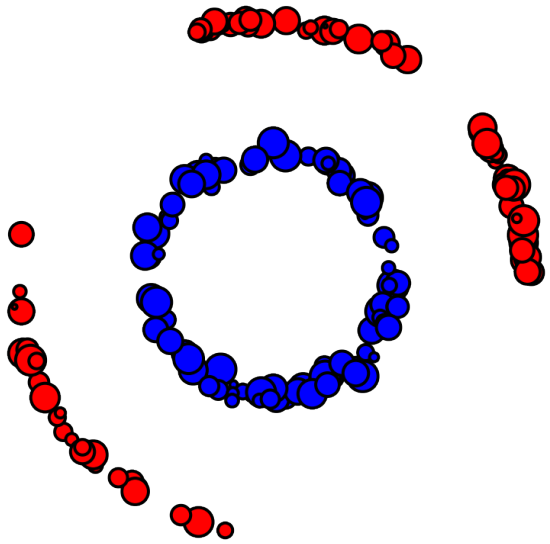

Figure 5: Gaussian distributions

3.2.2 Case of $d = 5$, $n = 1000$ and $\lambda = \log n/\varepsilon$

We plot in the figures below the theoretic ε which is the given precision and the practical error which is computed as the following :

$$\varepsilon_{pr} = < \hat{P}, C > - < P^*, C >$$

where $\hat{P}$ is the approximate solution and P^* is the exact solution.

[1] https://github.com/sfafe

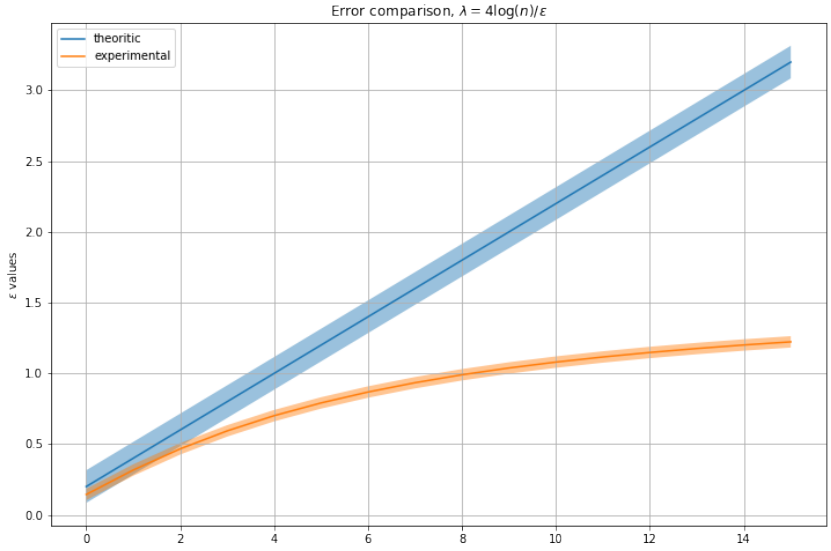

Figure 6: ε : blue, ε_{pr} : orange, $\lambda = 4 \log n/\varepsilon$

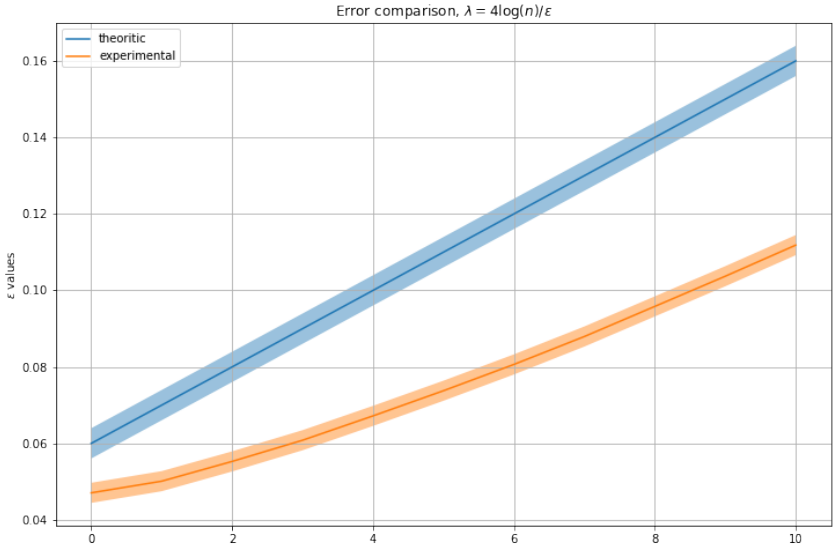

Figure 7: ε : blue, ε_{pr} : orange, $\lambda = 4 \log n/\varepsilon$

3.2.3 Case of $d = 10$, $n = 1000$ and $\lambda = \log n/\varepsilon^{1.5}$

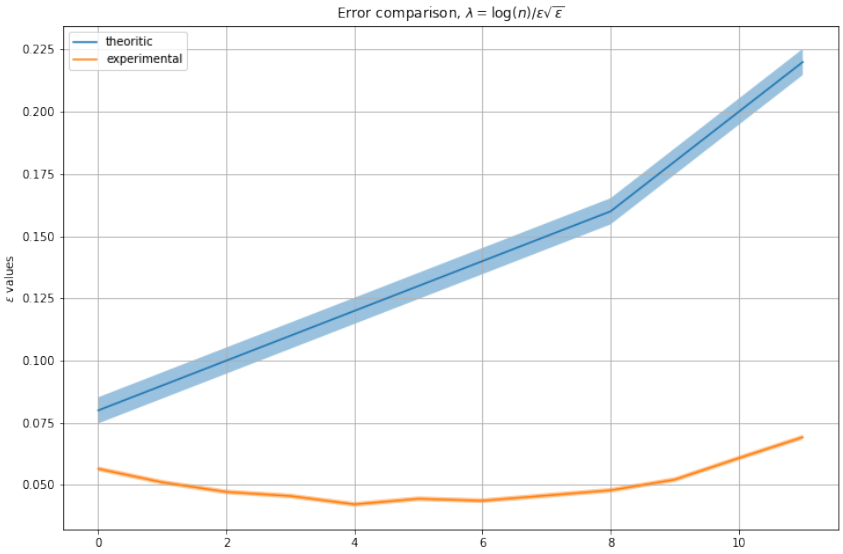

Figure 8: ε : blue, ε_{pr} : orange, $\lambda = \log n/\varepsilon^{1.5}$

3.2.4 Case of $d = 10$, $n = 1000$ and $\lambda = \log n/\varepsilon$

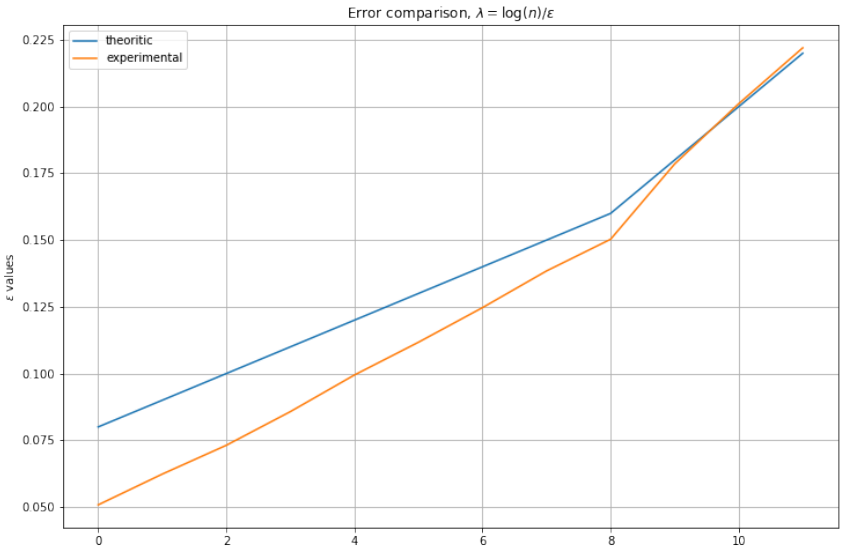

Figure 9: ε : blue, ε_{pr} : orange, $\lambda = \log n/\varepsilon$

3.2.5 Case of $d = 2$, $n = 1000$ and $\lambda = \log n/\varepsilon$

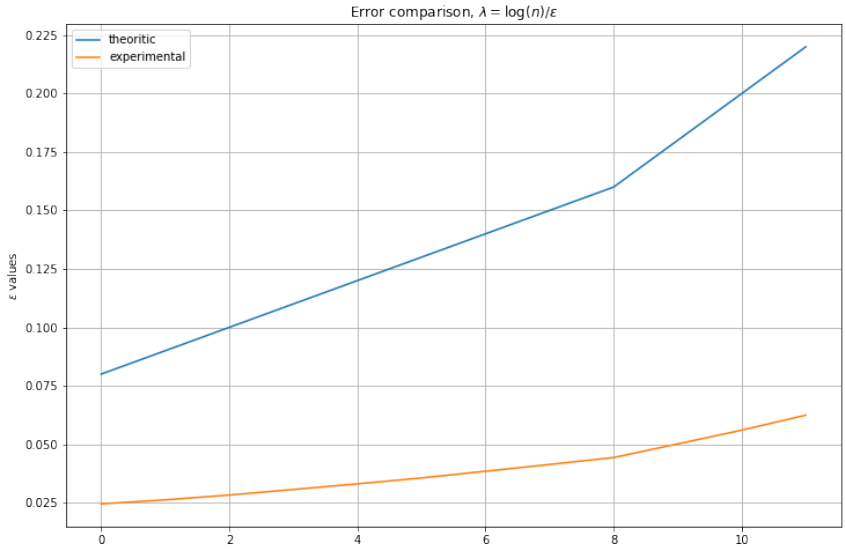

Figure 10: ε : blue, ε_{pr} : orange, $\lambda = \log n/\varepsilon$

3.3 Case of $d = 20$, $n = 500$ and $\lambda = \log n/\varepsilon$

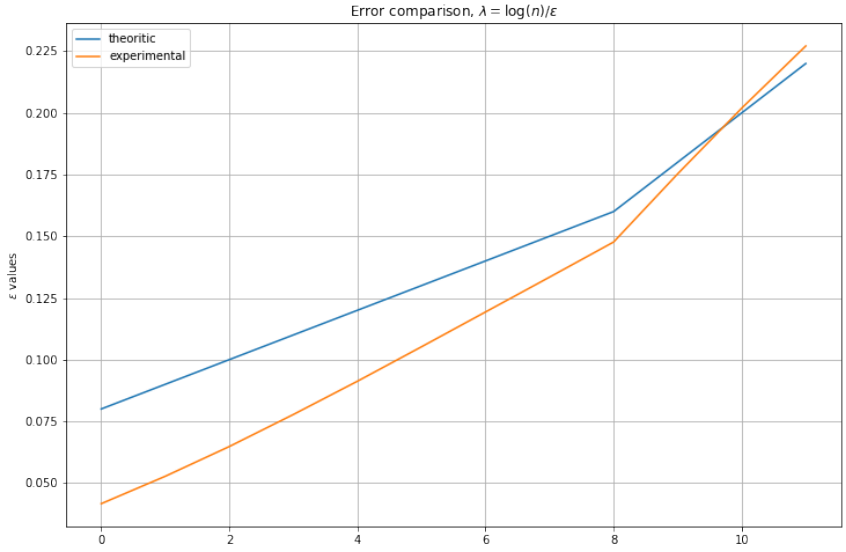

Figure 11: ε : blue, ε_{pr} : orange, $\lambda = \log n/\varepsilon$

3.3.1 Case of $d = 5$, $n = 500$ and $\lambda = \sqrt{\log n}/\varepsilon$

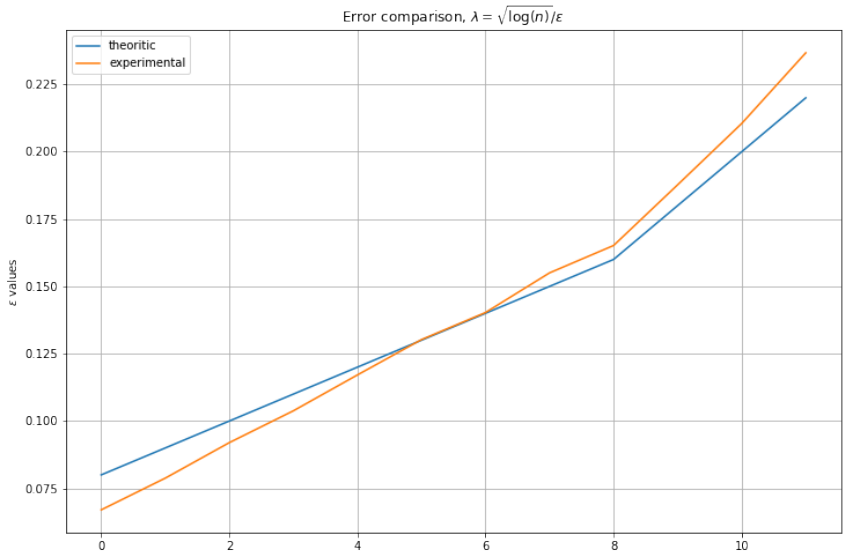

Figure 12: ε : blue, ε_{pr} : orange, $\lambda = \sqrt{\log n}/\varepsilon$

3.4 Interpretations

As we saw in the figures above, when λ has the same order of $\log n/\varepsilon$ the numerical results confirm that is the best choice as has been proven. We also observed that in practice λ can be more smaller according to Cuturi [4] but not much more small since when we tested with $\lambda = \sqrt{\log n}/\varepsilon$ we can not find an ε approximation when $\varepsilon \geq 0.15$.

We also observed the effect of the dimension when $\lambda = \log n/\varepsilon$. When d is small, theory and practice match. But when d is high we have to take much large λ in order to have an ε approximation.

4 Conclusion and perspectives

This work was an opportunity to deal with the fastest Sinkhorn's algorithm until now. We have seen that this algorithm reaches the complexity of $O(n^2/\varepsilon^3)$. We also studied the new algorithm proposed in [1] Greenkhorn, which have the same guarantees as Sinkhorn but it is more efficient in the practice . Despite this performance, Greenkhorn's algorithm has one major disadvantage compared to Sinkhorn which is not parallelizable.

We have seen also that by using the norm 1 instead of norm 2, Altschuler et AL. were able to give a new analysis to Sinkhorn's Algorithm. Hence, improving the running time of the algorithm. Peyré et Al. in their work [3], they were able to bypass the step of rounding proposing by Altschuler et AL. by evaluating directly the dual of the Wasserstein distance which is much more easier to implement in practice.

We were able to reproduce the the empirical results of the performance of the Greenkhorn vs Sinkhorn in the same datasets used in [1] with different entropic penalty.

We verified the dependence of the entropic penalty λ with ε. We have shown that the theoretical form of $\lambda = 4 \log n/\varepsilon$ is giving satisfying results in practice when the dimension of transported measures are not too high.

We have faced some problem when implementing Sinkhorns' algorithms. The nature of this problem is coming from the scaling. Since we are projecting Gibbs Kernel onto the set of transport plans between the two probabilities, and when the λ or C cost matrix have an entry to large the Gibbs Kernel read it as a 0. Hence, it divide by 0 which is a major problem.

According to Altschuler et AL., they said that they gave an near linear time approximation for OT. In fact, it is a quadratic time approximation. Thus, the question is still open whether there exist a near linear time approximation for OT. Can a new analysis using a new distance by probability measures give a new road to well approximate OT within a sub-quadratic time?

References

[1] Jason Altschuler, Jonathan Weed, Philippe Rigollet, "**Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration**", https://arxiv.org/abs/1705.09634

[2] Gabriel Peyré, Marco Cuturi."**Computational Optimal Transport**" https://arxiv.org/abs/1803.00567

[3] Lenaic Chizat, Pierre Roussillon, Flavien Léger, François-Xavier Vialard, Gabriel Peyré. "**Faster Wasserstein Distance Estimation with the Sinkhorn Divergence**", https://arxiv.org/abs/2006.08172

[4] Marco Cuturi. "**Sinkhorn Distances: Lightspeed Computation of Optimal Transport**", https://proceedings.neurips.cc/paper/2013/file/af21d0c97db2e27e13572cbf59eb343d-Paper.pdf

[5] Yin Tat Lee, Aaron Sidford. "**Path Finding I :Solving Linear Programs with $\tilde{O}$(sqrt(rank)) Linear System Solves**", https://arxiv.org/abs/1312.6677