Application of Gaussian and Euclidean Cost Matrices to Matrix Rescaling Optimal Transport Algorithms

Jacob Fishman, Aneesh Maganti

December 16th, 2022

1 Abstract

For our project, we reviewed *Near-linear Time approximation algorithms for optimal transport via Sinkhorn iteration* from Jason Altschuler, Jonathan Weed, and Philippe Rigollet. The paper's main contributions are proving that the Sinkhorn algorithm proposed by Cuturi can converge in near-linear time. The theoretical approach also leads to a new greedy approach to Sinkhorn, referred to as Greenkhorn, which was faster in practice though with the potential cost of higher error under the same number of iterations. However, the testing provided in the results section of the paper was limited and notably only used l1 distances when it performed the image tests. There is a lack of empirical research discussing the effectiveness of Sinkhorn or Greenkhorn on alternative pre-set cost matrices such as gaussian or l2 distances.

The goals for this project were to re-analyze the Sinkhorn and Greenkhorn algorithms and then perform further analysis on Sinkhorn, Greenkhorn, Greenkhorn Random and Greenkhorn Reverse Gaussian, in terms of two defined cost matrices based on a gaussian distribut ion and l2 distance between the marginals. After the tests, there was no correlation between the gaussian cost matrices and the euclidean distance matrices on the optimal transport algorithm. Notably, there was no significant difference in time between any of the Greenkhorn versions, suggesting that the marginal row chosen during the update process has little relevance to runtime.

2 Introduction

We will start off our report by understanding the problem at hand and restating the analysis in a more digestible manner. The problem that Sinkhorn addresses is computing optimal transport distances [Vil09]. Optimal transport distances deal with the transport of one distribution in any dimension space to another distribution in the same dimension space. The best way to visualize this problem is to imagine a construction worker moving dirt from one pile to a hole. In this example, the worker is the optimal transport that is moving the dirt (the initial distribution) into the hole (the second distribution) in the most optimal way possible. The original optimal transport problem is:

$min_{P \in U_{r,c}} < P, C >$

In this case, the starting probability distribution is r and the final probability distribution is c. The $U_{r,c}$ is the subspace of all valid transport that would result in a matrix that compressing the rows would equal r, and compressing the columns would result in c. The paper talks about previous work on ways to solve this problem and introduces its new analysis of the Sinkhorn algorithm that would result in solving this problem in near-linear time.

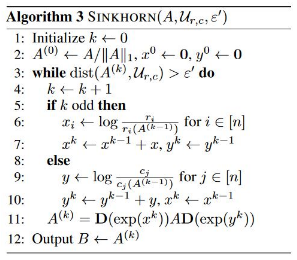
3 Sinkhorn Algorithm and Proof

The OT algorithm, using the Sinkhorn projection, will find the transport matrix that satisfies the following equation:

$\mathrm{P}_n = \mathrm{argmin}_{P \in U_{r,c}} < P, C > -\eta^{-1} H(P)$

This new algorithm adds an entropic penalty to the OT algorithm, which was proposed by Cuturi[Cut13]. The paper describes how the new transport matrix calculations will converge to the optimal transport in a quicker time than calculating the first equation natively.

We first will break down the algorithm at the high level, then we will look at the Sinkhorn and Greenkhorn algorithms. The pseudocode for the algorithm is:



```
Algorithm 3 SINKHORN($A, \mathcal{U}_{r,c}, \varepsilon'$)
1: Initialize $k \leftarrow 0$
2: $A^{(0)} \leftarrow A/\|A\|_1$, $x^0 \leftarrow \mathbf{0}$, $y^0 \leftarrow \mathbf{0}$
3: while $\mathrm{dist}(A^{(k)}, \mathcal{U}_{r,c}) > \varepsilon'$ do
4:     $k \leftarrow k + 1$
5:     if $k$ odd then
6:         $x_i \leftarrow \log \frac{r_i}{r_i(A^{(k-1)})}$ for $i \in [n]$
7:         $x^k \leftarrow x^{k-1} + x$, $y^k \leftarrow y^{k-1}$
8:     else
9:         $y \leftarrow \log \frac{c_j}{c_j(A^{(k-1)})}$ for $j \in [n]$
10:        $y^k \leftarrow y^{k-1} + y$, $x^k \leftarrow x^{k-1}$
11:    $A^{(k)} = \mathbf{D}(\exp(x^k)) A \mathbf{D}(\exp(y^k))$
12: Output $B \leftarrow A^{(k)}$
```

Figure 1: Sinkhorn Projection

The inputs to our algorithm is the Cost matrix (C), which this algorithm will work with any non negative cost matrix, the starting distribution (r), the final distribution (c), and the epsilon value, which represents the error by which we can be away from the optimal transport. We start out by determining our η and ϵ'. The second step is to determine the initial transport matrix, A. We use our cost matrix and our eta value, which is determined by our error we enter. The logic behind starting with these values is due to the cost dominating the transport cost. The starting matrix will be optimized over the cost, which will be entered into the Sinkhorn Projection.

We first normalize the starting transport matrix and initialize the update vectors based off of the rows and columns. The idea is for each iteration we determine the marginals for the transport matrix, and update each item in the marginal by taking the log of the ratio of the target marginal over the marginal we have now. We then multiply each item in the transport matrix by this resulting vector. This process iterates between rows and column updates until we are within ϵ' distance away from the subspace of transport matrix.

The analysis for run time of this algorithm is a bit hard to understand, so we will try our best to make it understandable. The main idea behind the algorithm is using the Kullback-Leibler divergence. This is an equation to find the difference between probability distributions [1]. The goal is to prove that we will find a transport matrix that will satisfy $||r(A) - r||_2 + ||c(A) - c||_2 \leq \epsilon'$ in O(np($\epsilon'^{-2} log(s/l)$)) where p is a number $\geq 1/$n. We want to understand the dimensionality independent analysis of the sinkhorn algorithm. First, we want to look at the improvement in each iteration of the algorithm, which the paper defines as $f(x, y)$ where the x and y here are associated with the scaling vector that we are alternating updating. The updates will add or subtract to the scaling of each item in the A matrix based off the r marginal or the c marginal. We use a form of the Lyapunov function, which is a function to help find an equilibrium point of a R dimension matrix [2]. The function here in the paper has been proven in many other papers, so we take it to be fact:

f(x,y) $= \sum_{ij} A_{ij} e^{x_i + y_i} - < r, x > - < c, y >$

The paper uses this fact of equilibrium to compare the update. The equation for this is:
$f(x^{k-1}, y^{k-1}) - f(x^k, y^k)$ (Which we can equate to the above equation) $= \sum_{ij} (A_{ij} e^{x_i+y_i})^(k-1) - (A_{ij} e^{x_i+y_i})^(k) - < r, x^{k-1} - x^k > - < c, y^{k-1} - y^k >$

$\sum_{ij} (A_{ij} e^{x_i+y_i})^{(k-1)} - (A_{ij} e^{x_i+y_i})^{(k)} + < r, x^k - x^{k-1} > + < c, y^k - y^{k-1} >$

In this analysis, we are able to assume k is odd without loss of generality [Paper]. This will let us look at every iteration after updating based off the r and the c marginals. Therefore c(A$^{(k-1)}) = c$ and $r(A^k) = r$. We also will use the facts that $||(A_{ij} e^{x_i+y_i})^(k-1)||_2$ and $(A_{ij} e^{x_i+y_i})^(k) = 1$. This is from the fact that we are moving probability distributions, where we know that each row and column marginal will equal 1. In this analysis, since k is odd we know that the y vector will not be changing throughout iterations, so therefore y$^k = y^{k-1}$. given this, we can start to simplify the equation:

$\sum_{ij} (A_{ij} e^{x_i+y_i})^(k-1) - (A_{ij} e^{x_i+y_i})^(k) + < r, x^k - x^{k-1} > + < c, y^k - y^{k-1} >$

$= \sum_{ij} (1-1) + < r, x^k - x^{k-1} > + < c, 0 > = < r, x^k - x^{k-1} >$

Now we want to look at this summation of this forbeneus norm:

$< r, x^k - x^{k-1} > = \sum_i r_i (x_i^k - x_i^{k-1})$

From here the we can rewrite the summation in different terms:

$\sum_i r_i log(\frac{r_i}{r_i(A^{(k-1)})}) = K(r||r(A^{(k-1)})$

Also, given the way that we are setting the problem up, we know that $K(c||c(A^{(k-1)}) = 0$ since c = c($A^{(k-1)}$); therefore, the update after each full iteration of the Sinkhorn projection algorithm is:

f($x^{k-1}, y^{k-1}) - f(x, y) = K(r||r(A^{(k-1)}) + K(c||c(A^{(k-1)})$

Now we need to look at the initial distance away from the minimum x and y vectors that will get us to the optimal vectors. The third lemma in the paper will help us with with this understanding:

$f(x^1, y^1) - min_{x,y \in R} f(x, y) \leq f(0, 0) - min_{x,y \in R} f(x, y) \leq log(s/l)$

In this equation, we look at the starting distance, since our x and y vectors start at 0, away from the minimum value for x and y that are in the subspace for the optimal transport. Now we know the starting distance away, and the update value after each iteration of the Sinkhorn algorithm. We can use Pinsker's inequality [3] to determine the distance away for probability measures to finalize the total iterations to update the starting transport matrix. Pinsker's Inequality is the difference from probability measures is bounded by 2 * the square root of the Kullback-Leibler divergence, which we saw that is equal to the improvement that we make during each iteration to getting the marginals closer to the target marginals. Given all of this, we will look at any step that is larger than the epsilon error that we set at the beginning of the algorithm. The any step will be designated as k:

$\epsilon' < (||r(A^{(k)}) - r||_1 + ||c(A^{(k)}) - c||_1$

Using the Pinsker's inequaltiy:

$\epsilon' < (||r(A^{(k)}) - r||_1 + ||c(A^{(k)}) - c||_1 \leq \sqrt{2(K(r||r(A^{(k)})) + K(c||c(A^{(k)}))}$
Now we can square both sides
$\epsilon'^2 < (||r(A^{(k)}) - r||_1 + ||c(A^{(k)}) - c||_1)^2 \leq 4(K(r||r(A^{(k)})) + K(c||c(A^{(k)})))$

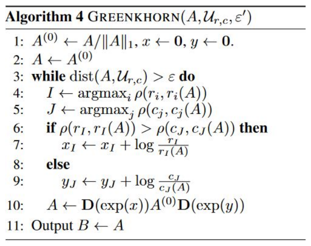
In order to solve for k*, we can use the lemmas proven in the paper. The analysis done on the scaling vectors shows us that each iteration will getting better by a factor of the Kullback-Leiber divergence. We also know that the starting distance from getting the minimized scaling vectors is $log(s/l)$. Therefore, we know that the starting k, when k = 0, is less than $log(s/l)$, meaning that the equation for the starting k is:

$\epsilon'^2 < (||r(A^{(0)}) - r||_1 + ||c(A^{(0)}) - c||_1)^2 \leq 4 * log(s/l)(K(r||r(A^{(k)})) + K(c||c(A^{(k)})))$

We then know that, since each iteration is improving by the Kullback-Leiber distance, that in order to get to the final k* step, we would need $4 * \epsilon'^{-2} log(s/l)$ steps, meaning that the overall run time for the Sinkhorn algorithm is proven to run in near linear time.

4 Greenkhorn Algorithm and Proof

The key difference in the Algorithm that the paper presents is the introduction of greedily updating either the x or y scaling vectors by finding the index of the marginal that is the farthest away from the Marginal space. The distance equation used in this case is $p(a, b) = b - a + alog(a/b)$. The difference in the algorithm is, since we are only updating one of the marginals, we only need to do O(n) actions for each of the iterations of the Greenkhorn Algorithm. Although there is improvement in the iteration time, the number of iterations needed to get to ϵ' error is increased by a factor of n, meaning that in theory, the algorithms should run in nearly identical time. The analysis for the Greenkhorn will use the same framework for the previous analysis to determine the amount of



```
Algorithm 4 GREENKHORN($A, \mathcal{U}_{r,c}, \varepsilon'$)
1: $A^{(0)} \leftarrow A/\|A\|_1$, $x \leftarrow \mathbf{0}$, $y \leftarrow \mathbf{0}$.
2: $A \leftarrow A^{(0)}$
3: while $\mathrm{dist}(A, \mathcal{U}_{r,c}) > \varepsilon$ do
4:     $I \leftarrow \mathrm{argmax}_i\, \rho(r_i, r_i(A))$
5:     $J \leftarrow \mathrm{argmax}_j\, \rho(c_j, c_j(A))$
6:     if $\rho(r_I, r_I(A)) > \rho(c_J, c_J(A))$ then
7:         $x_I \leftarrow x_I + \log \frac{r_I}{r_I(A)}$
8:     else
9:         $y_J \leftarrow y_J + \log \frac{c_J}{c_J(A)}$
10:    $A \leftarrow \mathbf{D}(\exp(x)) A^{(0)} \mathbf{D}(\exp(y))$
11: Output $B \leftarrow A$
```

Figure 2: Greenkhorn Projection

steps needed to get to ϵ' error. The first thing we notice is we can split up the updates based off if we will be updating the x or y scaling vector. Therefore:

$f(x', y') - f(x'', y'') = p(r_I, r_I(A'))$ for the update on the Ith row of the marginal
$f(x', y') - f(x'', y'') = p(c_J, c_J(A'))$ for the update on the Jth column of the marginal

The paper uses an extended form of the Pinsker Inequality for this new form for any value of $p(a, b) < 1$:

$||a - b||_1 \leq \sqrt{7p(a, b)}$

If we follow the same logic from the Sinkhorn projection proof, after each iteration of Greenkhorn, we will be updating the x or y scaling vector in a greedy fashion. If we use the logic from the Sinkhorn projection, we can see that for each update we will be improving by $1/2n(p(r, r(A)) + p(c, c(A)))$, and we also know that the both $p(r, r(A))$ and $p(c, c(A))$ are less than 1 (given how probability distributions work), so therefore we can extend Pinsker's Inequality yet again to determine the runtime for Greenkhorn:

$1/2n(p(r, r(A)) + p(c, c(A))) \geq 1/14n(||r(A) - r||_1 + ||c(A) - c||_1$

We also can use the previous analysis from the Sinkhorn proof to know that this hold for any iteration, k, that does not get us to the error:

$1/2n(p(r, r(A)) + p(c, c(A))) \geq 1/14n(||r(A) - r||_1 + ||c(A) - c||_1 \geq 1/28n\epsilon'^2$

The analysis for how far the Transport matrix starts from a transport matrix in the subspace is the same for Greenkhorn, meaning that we can use the same math for the Sinkhorn to determine the amount of iterations to get to the final number of iterations of k to be:

$k^* = 1/28n\epsilon'^- 2log(s/l)$

5 Practical Approach

This analysis of a near linear time for Sinkhorn and Greenkhorn was revolutionary when this paper came out, and is still used in practice today. The use of sinkhorn stretches farther than normal probability distributions, but into Adversarial Attacks [4], Change point Detection [3], and many more applications. Moreover, many papers examined Sinkhorn Projections for certain data sets [6] that have lead to many advancements in use of the Sinkhorn algorithm for certain problems.

Our approach to furthering the understanding of Sinkhorn and Greenkhorn Projections started with us understanding the literature, and trying to improve on the theoretical analysis. As we dug deeper into the Lemmas and inequalities, we realized that there were no improvements we could find for the analysis of the run time. After that, we tried to propose a different algorithm, one that used a batched approach to Greenkhorn. Although this showed promise, our research led us to a 2021 paper that already proposed the same algorithm [5]. Given our lack of success in proposing a new algorithm or adding theoretical work to the previous theory, we decided to look at an experimental approach.

During our breakdown of the paper and research into the subject, we noticed the lack of work done on the Greenkhorn algorithm, and the certain datasets where the algorithm would be optimal compared to Sinkhorn. Given the tight proof of run time for both projections, as well as the rounding algorithm to project the final transport onto the subspace [NIPS], we decided to look at these algorithms in practice. In the NIPS paper, the analysis is done for any problem with a non negative cost matrix, as well as for any transport problem [NIPS]. The problem we strive to help understand with our contribution is how can the understanding of the probability distributions and cost matrices help with choosing the algorithm to use. To do this, we first wanted to confirm the assertion made in the paper that the cost matrix chosen would have no impact on the runtime for the algorithms. Furthermore, we wanted to create modifications of Greenkhorn that would rely on randomness that based on our knowledge of the cost matrices. Given the need for computational time improvements, we hoped to see a correlation in our research that could lead into theoretical approaches.

5.1 POT

The goals was to do testing on the Sinkhorn and Greenkhorn algorithms that are highlighted in the NIPS paper as well as modifications to these algorithms on the cost matrices. In the original paper, there was limited testing on the comparisions between the Sinkhorn and Greenkhorn algorithms, sticking within the bounds of the real MNIST images and the randomly generated synthetic images (by randomly choosing a foreground square on top of a black background with intensities of the squares varying from [0, 1]); the cost matrix was kept the same, the same type of dataset was used, and it didn't compare to a network simplex solver, a method of solving optimal transport problems exactly. The eta terms were varied though.

We wanted to perform more thorough testing and decided against using the code that was provided from the paper, primarily due to the lack of optimizations from the age of the program and because the authors had stated that the provided program emphasized readability over speed. While there are many open source Sinkhorn and Greenkhorn implementations online, we decided to use the

Python Optimal Transport, or POT, library. POT is a general python library that has has many different solvers related to optimal transport and is regularly updated. Within this library are included functions for a network simplex solver, Sinkhorn-knopp and Sinkhorn-log (the algorithm that is used in the paper) and Greenkhorn implementation.

To facilitate the testing, the goal was to create a generalized testing suite that could benchmark different solvers and cost matrices and report the data back for analysis. Four different cost matrices were used with different purposes: a constant cost matrix (filled with all 1 values), a matrix filled with completely random values, a Gaussian matrix, and a l2 matrix. The first two matrices, the constant cost matrix and the random matrix, were primarily intended for baseline comparisions to the other two cost matrices. The gaussian matrix is square and was simulated with an exponentially modified Gaussian distribution, with larger values appearing closer towards the center of the matrix. The l2 cost matrix calculated the absolute l2 distances between each of the marginals and outputted a resulting a matrix.

From here, we decided to create some modifications of Greenkhorn to potentially better match the newly created cost matrices. Both of them targeted the same area: the selection of the indices for the rows that would be compared when deciding what the next row or column to update should be. In the first variant, Greenkhorn Random, the row or column is arbitrarily picked. In the second variant, Greenkhorn Reverse Gaussian, the row or column is picked based on a reverse gaussian distribution, or bounded gaussian from 0 to the length of the array that outputs values in the opposite frequency that a gaussian distribution would (this should not be confused with an inverse gaussian distribution). We had three datasets for our testing: the stock random datasets, a small selection of pairs of MNIST images to perform optimal transport on, and the pertubed marginal datasets. In the pertubed datasets, it contains a random dataset for the row marginal but for the column marginal, the row marginal is added or subtracted to a small percentage that ranges from 10 to 100% of the original value. To recap, there are 4 matrices (constant, random, gaussian, l2), five solvers (network simplex, Sinkhorn, Greenkhorn, Greenkhorn Random, Greenkhorn Reverse Gaussian), and two datasets (MNIST and pertubed dataset).

5.2 Results

The results, throughout all experiments, showed consistency with the error values the final transport matrix has outputted since each algorithm is running until the error between marginals of the projected transport matrix and the real marginal space goes below the set lambda value (0.1 in our code). For the pertubed marginal dataset, the percentage factors had little impact on the time for either Sinkhorn or Greenkhorn. The rest of the conclusions apply to both the MNIST and pertubed marginal datasets.

Turning our attention to the run time of each algorithm, we see that the base case, network simplex, is consistently slower than all of the other algorithms, confirming that projections are a faster way to get the optimal transport matrix if some amount of error is permitted. Comparing the times to reach below the error permitted between the Sinkhorn, Greenkhorn, Greenkhorn Random, and Greenkhorn Reverse Gaussian, Greenkhorn and its variants outperform Sinkhorn to output the optimal transport matrix regardless of the dataset which is consistent with the results seen in the paper. Furthermore, this applied to all of the cost matrices chosen, further confirming

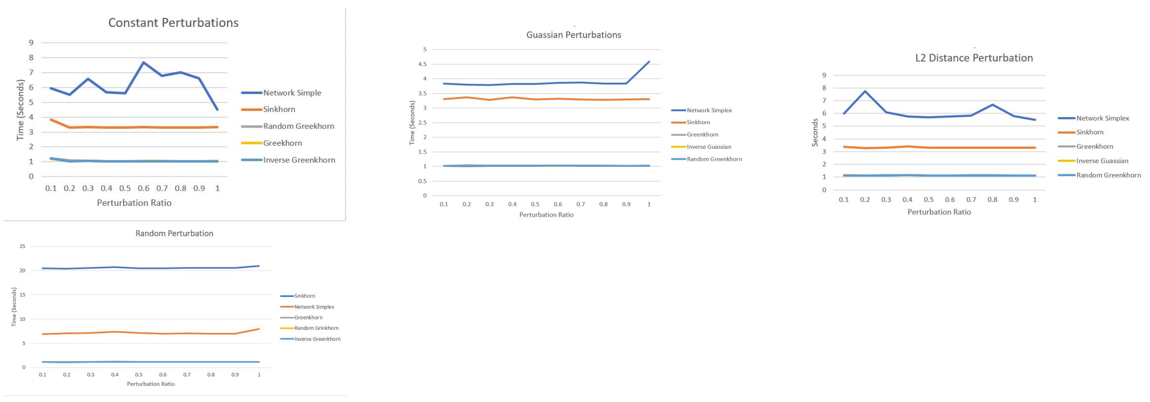


Figure 3: The Greenkhorn and Random Greenkhorn lines are hidden by the Greenkhorn Reverse Gaussian

the paper's assetion that the cost matrix did not have an impact on the runtime of either algorithm.

The runtime results for Greenkhorn, Greenkhorn Random, and Greenkhorn Reverse Gaussian were similar throughout each test, suggesting that either the choice of updating the marginal has no impact or the time saved in deciding the index of the marginal to update is the same amount of time for incurring extra iterations. Regardless, each iteration of Greenkhorn is faster than Sinkhorn, confirming the results the NIPS paper found.

6 Analysis

The analysis of the paper and the testing results have led to a couple of conclusions. Our thoughts that the Cost matrix's similarity to the maginals would have a direct impact on the run time of the Sinkhorn and Greenkhorn. However, we see that the cost matrix initialization and closeness to the marginals has no impact on the run time of the algorithms. We also thought that, as the perturbation of the data sets got larger, that Sinkhorn would outperform the Greenkhorn algorithms. The fact we do not see this means that, in choosing which algorithm to use, there is no benefit in analyzing the cost matrix and marginals before using one of the algorithms.

The analysis of Greenkhorn and its variants surprised us, where each of them perform in similar times. This data helps reaffirm the results and the argument we saw in the NIPS paper which is that it is unknown why Greenkhorn is outperforming Sinkhorn. We tried to come up with a theoritical approach that could explain the speed up in run time, but facing the results of the variant algorithms, we are not able to get a theoretical run time for the algorithm that would explain the out performance.

7 Further Goals

As we dive deeper into the Sinkhorn projection and Optimal Transport problem, we hope to add to the theoretical approach behind Greenkhorn, and to prove its superiority over Sinkhorn projections. Throughout our research, we have not found any theoretical argument that would match the results for Greenkhorn that we, and the NIPS paper, have seen. This research we have done to this point will help direct the theoretical argument we make in the future. On the practical side, having a greater number of datasets, more testing with epsilon values, and the inclusion of the l1 cost matrix for comparisons would make the comparisions more thorough.

8 References

[Vil09] - C. Villani. Optimal transport, volume 338 of Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]. Springer-Verlag, Berlin, 2009. Old and new.

[1] - Kullback-Leiber divergence: https://www.statlect.com/fundamentals-of-probability/Kullback-Leibler-divergence

[2] - Pinsker's Inequality: https://web.stanford.edu/class/ee363/lectures/lyap.pdf

[Cut13] - M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 26, pages 2292–2300. Curran Associates, Inc., 2013.

[4] - A. V. Subramanyam. Sinkhorn Adversarial Attack and Defense:
https://ieeexplore.ieee.org/abstract/document/9792616

[3] - Ahad, Nauman, et al. "Learning Sinkhorn divergences for supervised change point detection." arXiv preprint arXiv:2202.04000 (2022).

[5] - Kostic, Vladimir, Saverio Salzo, and Massimilano Pontil. "Convergence of batch Greenkhorn for Regularized Multimarginal Optimal Transport." arXiv preprint arXiv:2112.00838 (2021).

[6] - Dong, Yihe, et al. "A study of performance of optimal transport." arXiv preprint arXiv:2005.01182 (2020).

[$NIPS$] - Altschuler, Jason, Jonathan Niles-Weed, and Philippe Rigollet. "Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration." Advances in neural information processing systems 30 (2017).