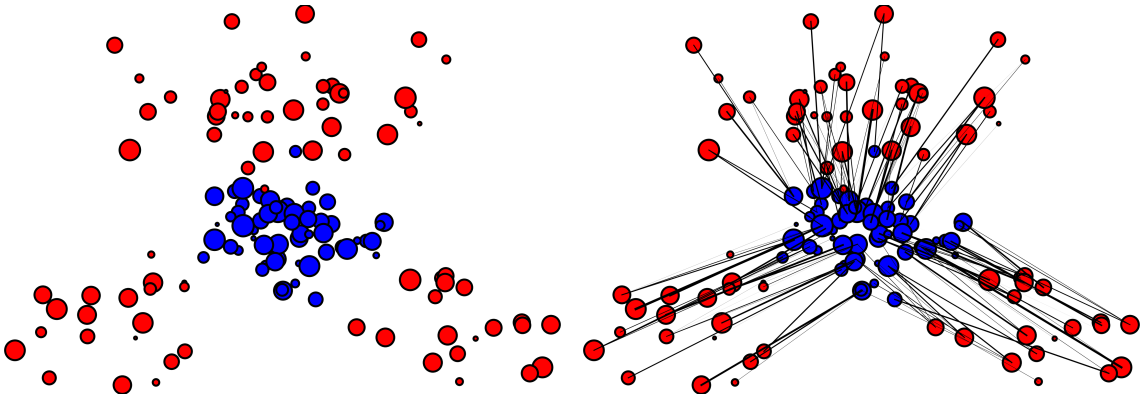
Numerical Tours

Gaspard Beugnot

Nov. 2019 - Jan. 2020

1 Optimal Transport with Linear Programming

In this notebook, we are introduced to a linear program solver. Given two weighted point clouds of mass 1 (ie. two discrete probability distributions), setting a cost matrix C, we can compute the optimal transport plan.



(a) Two distributions on $\mathbb{R}^2$ (b) The optimal transport for $\mathcal{L}_2$ norm

Figure 1: Two distributions and their optimal transport plan. Width of the lines are proportional to the quantity transported.

We have tools to check the validity of the solution:

- The coupling P should have no more than $n + m - 1$ non zero entries. Otherwise, the associated graph has at least one cycle, so P is not an extremal point of the simplex, and thus can't be optimal.
- The solution must be feasible, ie. P should meet: $P\mathbf{1}_m = a$ and $P^T\mathbf{1}_n = b$.
- If we had access to the dual variable, we would check that the complementary slackness is verified, but `perform_linprog.py` only returns the primal solution.

Most of the content of the notebook was already given, so there is not much to comment; still, we can see the influence of the **cost matrix** on the coupling P. We use a basic distribution, as in fig. 2a.

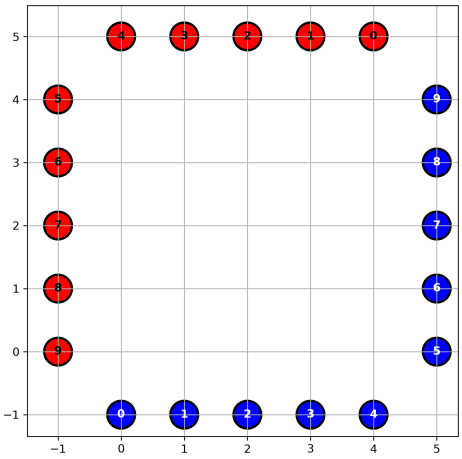
I tried various cost matrix C: the ones associated with the $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_\infty$ norm, and a constant cost. For each of these cost (except for the constant one, obviously), I also looked at the **inverse** of the distance, to mimic an **attractive cost**. Here, by "inverse", I mean:

$$C'_{i,j} = \frac{1}{C_{i,j}}$$

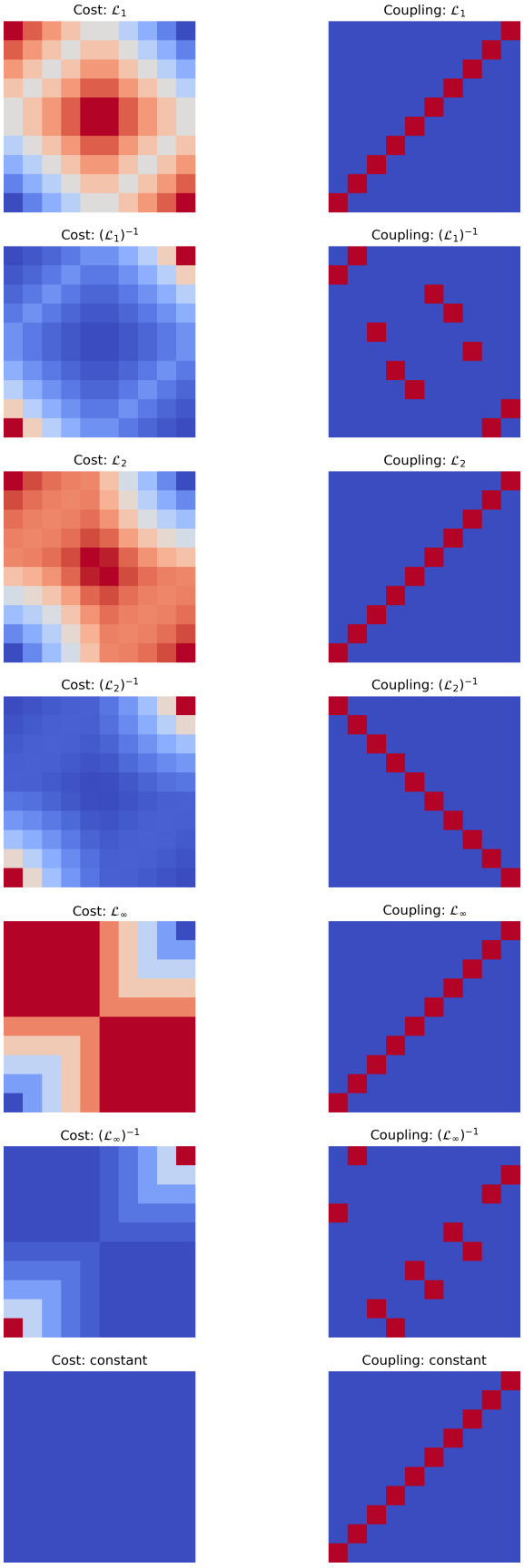
Of course, it is not a distance anymore, so the total cost has not the property of the Wassertein distance. Results are on fig. 2.

Several comments:

- All weights are uniform, so the optimal coupling will be a **permutation matrix.**
- I expected the behaviour to vary with the choice of p for the $\mathcal{L}_p$ norm. In fact, it does not, and what matters is only the relative order between the points.



(a) Two distributions on $\mathbb{R}^2$.



(b) Several cost and their associated optimal coupling.

Figure 2: Influence of the cost matrix C on the optimal coupling. Here, the cost matrix is either a $\mathcal{L}_p$ distance or an attractive metric, $1/||.||_{\mathcal{L}_p}$. For the inverse of the $\mathcal{L}_\infty$ norm, the linear program did not converge.

- However, they are multiple behaviour for the cost matrix I fashioned by taking the inverse of the $\mathcal{L}_p$ norms.
- The solver reported an error for the attractive cost derived from the $\mathcal{L}_\infty$ metric. Indeed, the output is not an optimum, as we can note that for instance:

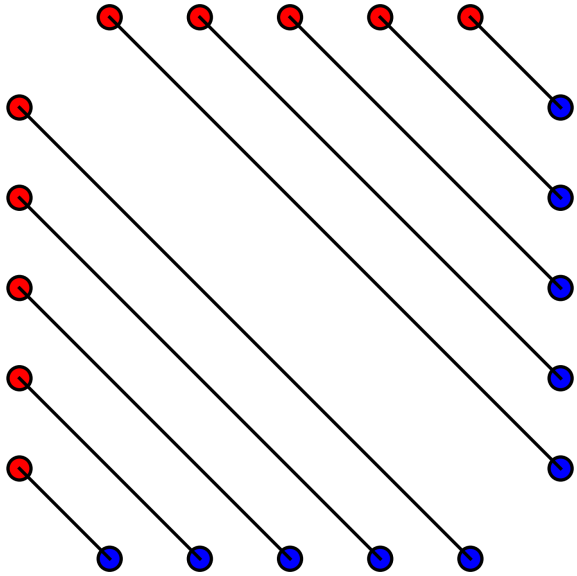
$$1.66 \approx \sum \mathbb{I}_{ij} C_{ij} < \sum P_{ij}^{\mathcal{L}_\infty^{-1}} C_{ij} \approx 2.48$$

  where $\mathbb{I}$ is the identity matrix, and corresponds for example at the optimal coupling for $\mathcal{L}_2^{-1}$ (fig. 2, 4th row).
- For the cost corresponding to the inverse of the $\mathcal{L}_1$ norm, I was surprised that the optimal coupling **was not symetric**. In fact, they are **multiple minima**. For instance:

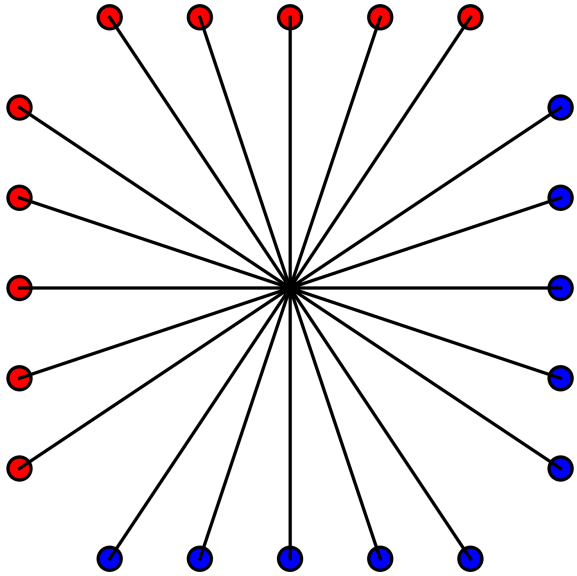
$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad B = A^T \quad C = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

  All these 3 matrices report a cost of ≈ 1.19.

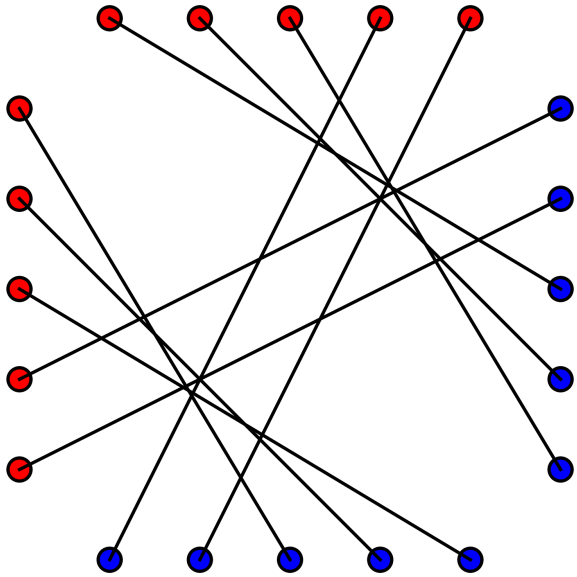
Finally, some of the couplings of 2 are shown on fig. 3.



(a) $\mathcal{L}_2$ cost



(b) $\mathcal{L}_2^{-1}$, attractive cost



(c) $\mathcal{L}_1^{-1}$, attractive cost

Figure 3: Various optimal coupling, depending on the associated cost matrix.

2 Entropic Regularization of Optimal Transport

This tour focused on the entropic regularization of the original optimal transport problem. Considering two discrete distributions $a, b \in \Sigma_n$, the Kantorovitch formulation reads:

$$W(a,b) \stackrel{\Delta}{=} \min_{P \in U(a,b)} \langle C, P \rangle$$

Whereas the entropic regularization reads:

$$W_\epsilon(a,b) \stackrel{\Delta}{=} \min_{P \in U(a,b)} \langle C, P \rangle - \epsilon H(P)$$

Where H is the entropy associated to the coupling P:

$$H(P) = \sum_{i,j} P_{ij} \left(\log(P_{i,j}) - 1 \right)$$

and ϵ is the strenght of this regularization parameter. The problem remains convex, we can show it amounts to taking the Kullback Leibler projection of the Gibbs kernel $K_{ij} = e^{-C_{ij}/\epsilon}$ on $U(a,b)$, and that the smaller the ϵ, the sparser the optimal coupling is and the closer it is from the optimal coupling related to the Kantorovitch formulation.

Small ϵ	**Big ϵ**
Sparse solution	Dense solution
Slow convergence	Fast convergence
Close to the unrelaxed problem	Far from the unrelaxed problem

That being said, let's see what this Numerical tour focused on.

2.1 Transport between point clouds

We first consider the case of transporting two point clouds. For that part, we keep the data suggested by the notebook, shown fig 4.

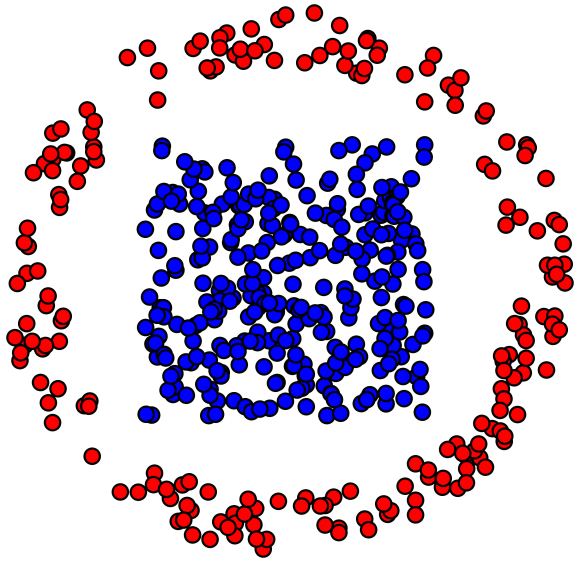


Figure 4: The point clouds we will transport.

We then implement the Sinkhorn's algorithm to solve the transport plan. The implementation is straightforward, as it only requires applying the update on the dual variable and iterating until a minimum tolerance for the equality constraints or a maximum number of iteration has been reached. One can refer to the notebook for the code.

2.1.1 Speed of convergence

We first check the **influence of ϵ on the convergence of the algorithm**. There is a trade off here between fast convergence and closeness to Kantorovitch solution. This is shown on fig. 5.

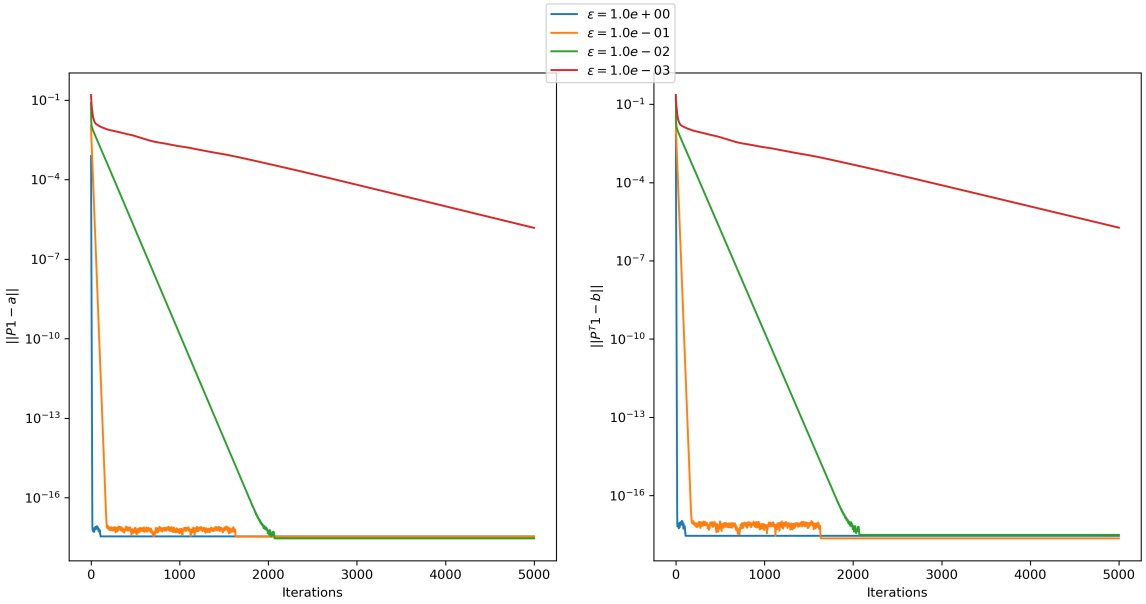


Figure 5: Speed of convergence function of ϵ.

We check our precedent statement. When ϵ is close to 1, we reach very quickly machine precision, whereas the slop is smaller as ϵ decreases. This behaviour is visible on the next plot, where we show the number of iterations needed to reach a certain precision function of ϵ value (fig. 6).

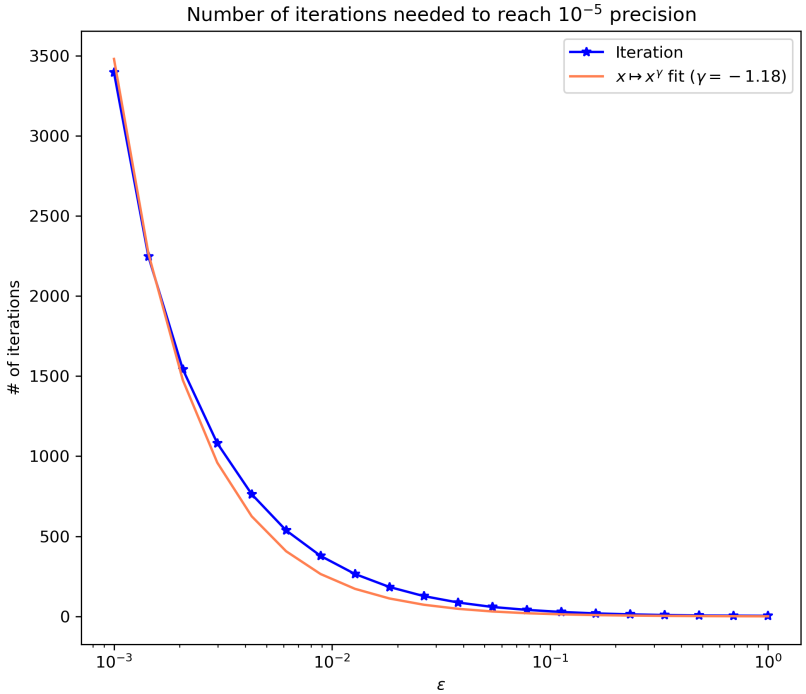


Figure 6: Number of iterations needed to reach a precision of 10^{-5}, function of ϵ. A polynomial fit is shown for information (obtained via least squares).

Interestingly, if ϵ is too low (I found 410^{-4} to be the limit), the algorithm **never converges**. This is simply because the Gibbs kernel gets huge and overflows, and this can be addressed by making the computations in the **log domain**.

2.1.2 Sparsity of the coupling

Let's now check the sparsity of the coupling. A visual is provided fig. 7.

The higher epsilon, the denser the coupling. We can choose to plot the number of non-zero entries, function of ϵ. This is shown fig. 8.

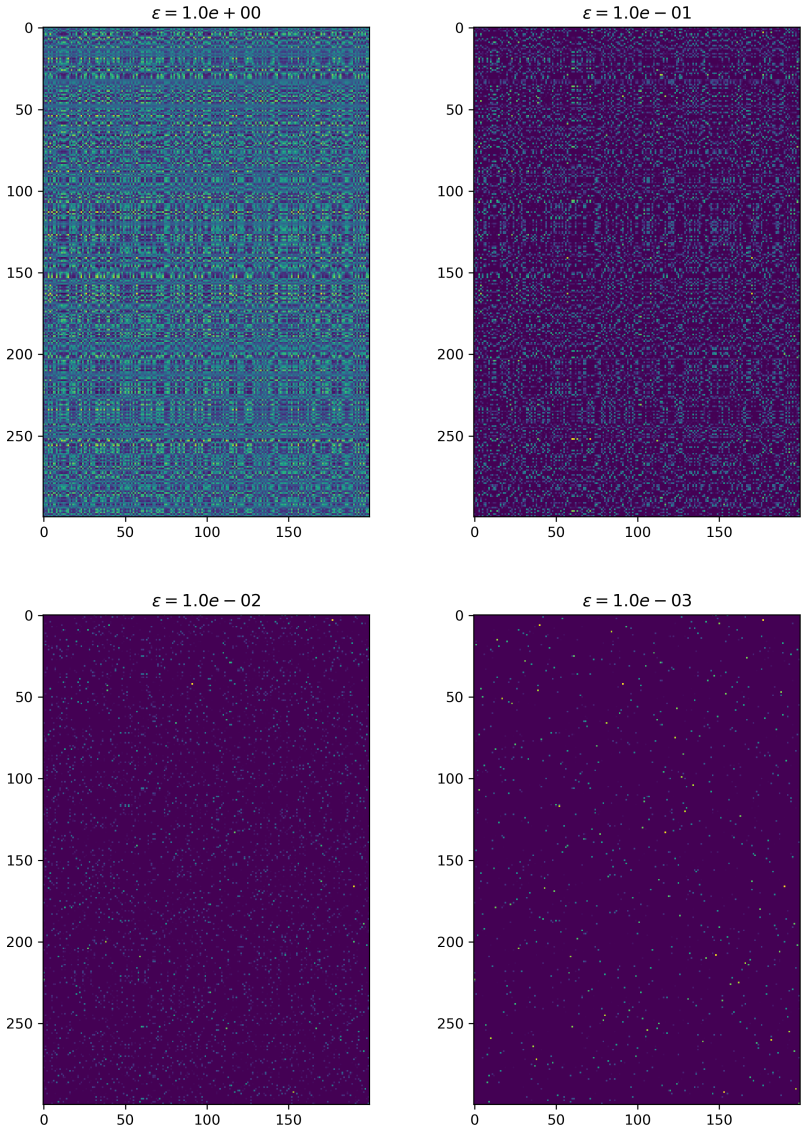


Figure 7: Coupling matrix P for various values of ϵ.

Finally, a more visual interpretation is provided fig. 9.

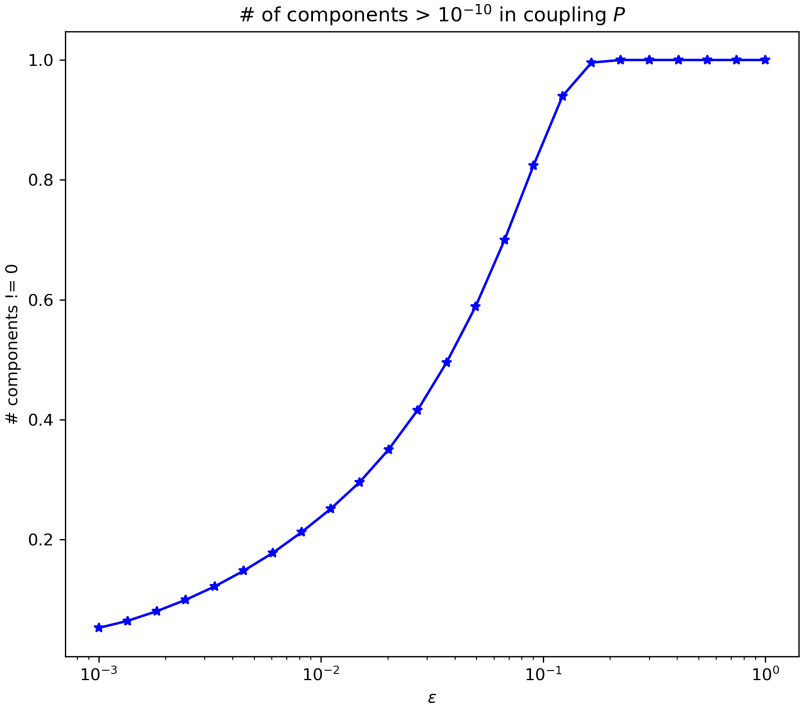
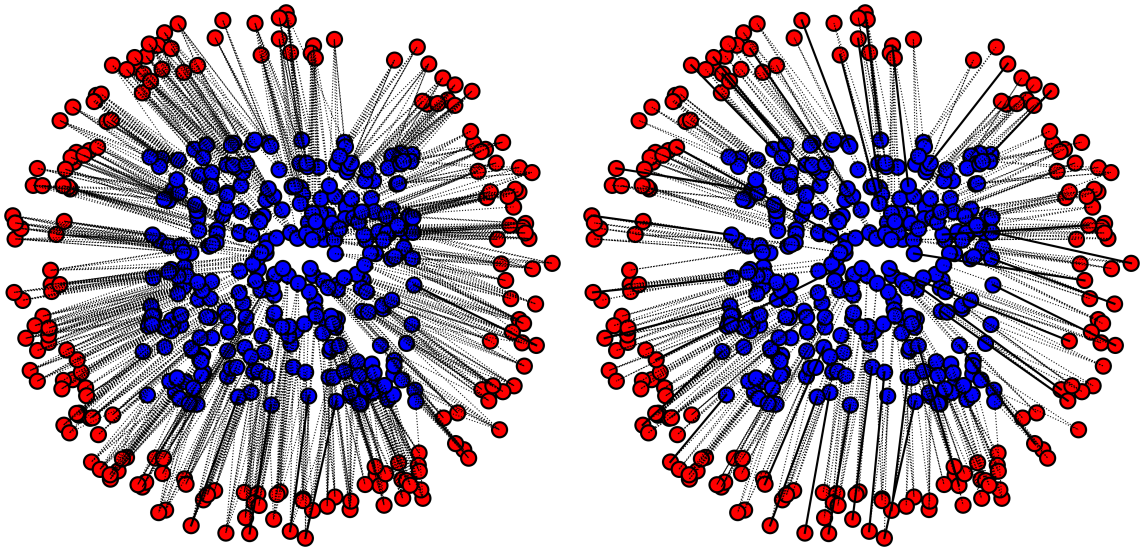


Figure 8: Fraction of non-zero components, function of ϵ.



(a) $\epsilon = 10^{-2}$ (b) $\epsilon = 10^{-3}$

Figure 9: Regularized coupling for the same distribution. For low values of ϵ, the coupling is sparse.

2.2 Transport Between Histograms

We now consider the problem of transporting two discrete measure whose support lies on a grid: $\left\{\frac{k}{n}, k \in \{0, \dots, n-1\}\right\}$. We first try when the two distributions are Gaussian (fig. 10).

Computing the Gibbs kernel yields a Gaussian kernel (fig. 11).

From there, we can compute the regularized transport plan. The barycentric projection map gives an idea of how the Monge map T looks like (fig. 12).

We can check the goodness of the approximation, function of the regularization parameter ϵ (fig. 13).

Finally, taking a look at the transportation of a unimodal distribution to a bimodal distribution gives an interesting hindsight on what happens to the *middle point*. We can suppose that the unregularized transport plan is never attained in this setting.

2.3 Using GPUs

It is little to say that I was disappointed by my implementation in PyTorch of Sinkhorn's algorithm. Comparing the execution time for 1000 iterations on histograms with N elements for a constant regularization $\epsilon = 0.06^2$, I obtained the graph shown fig. 15.

I am not a specialist of PyTorch, so my code may contain errors. Of course, I expected that for small values of N, the CPU would get ahead: building the computation graph implies some overhead. Yet, I thought that for large values, the GPU would completely destroy the CPU's performance. The code was ran on Google Colab. For this execution, the remote machine had an `Intel(R) Xeon(R) CPU @ 2.00GHz` for CPU and a `Tesla P100-PCIE` as a GPU. So that can't explain either why the performance are this low.

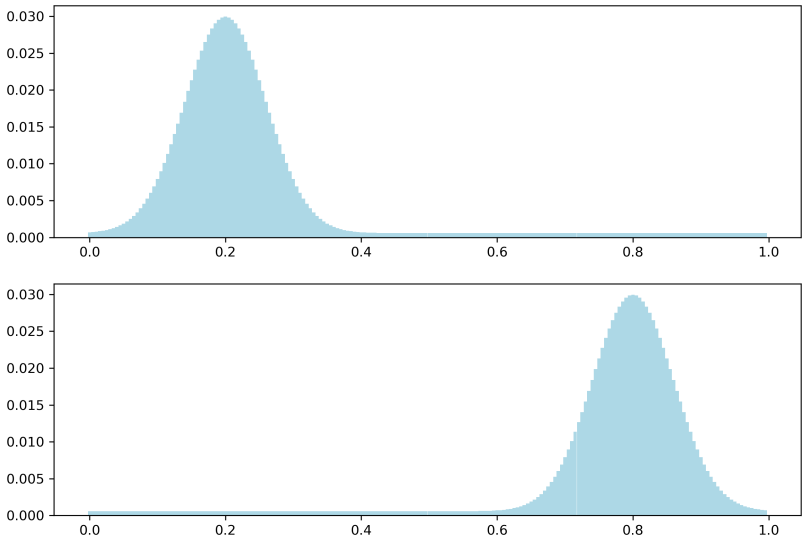


Figure 10: Two gaussian histograms we will transport.

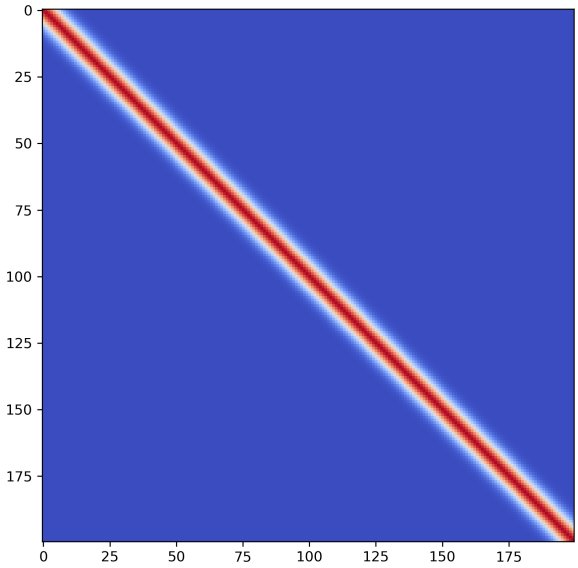


Figure 11: Computing the Gibbs Kernel of two histograms amounts to computing a Gaussian kernel.

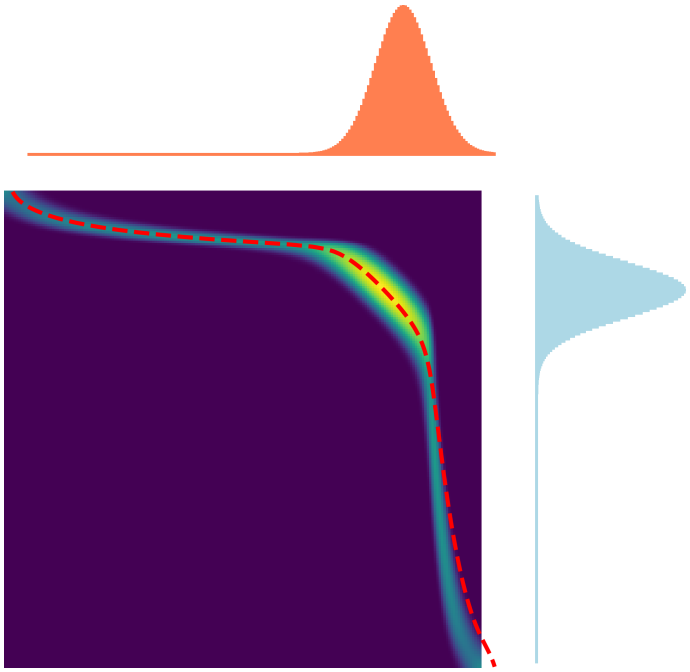


Figure 12: Log of the regularized transport plan. In dots, the barycentric projection gives an approximation of the Monge map.

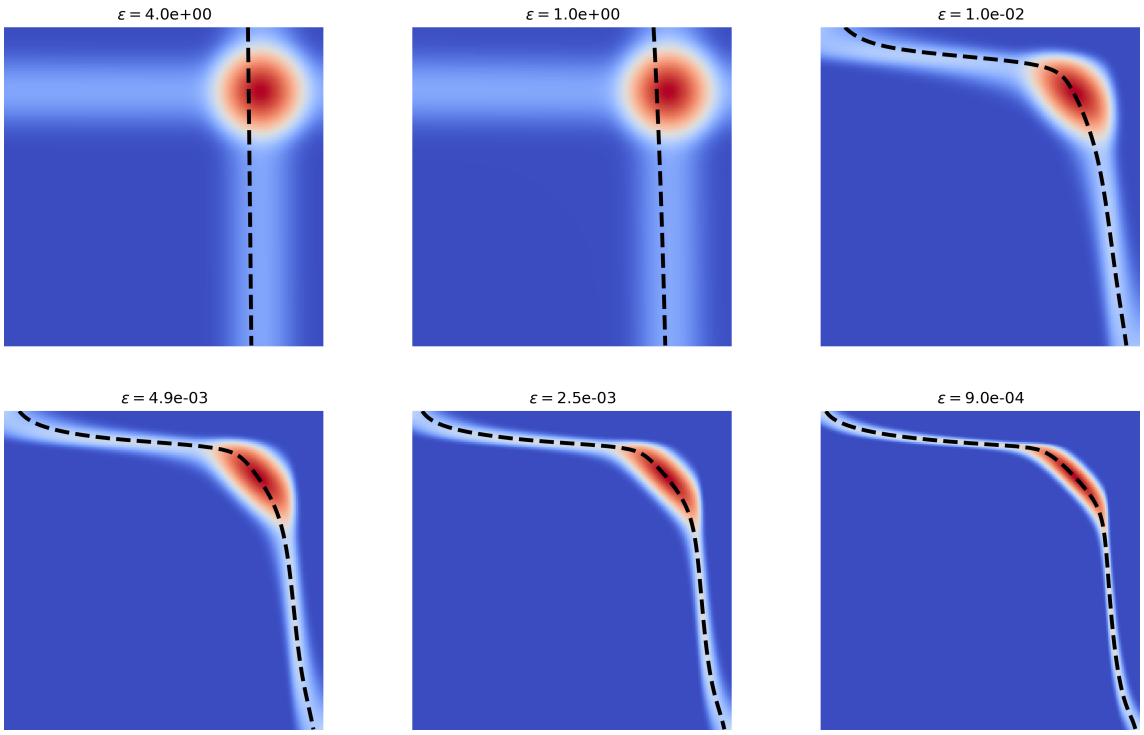
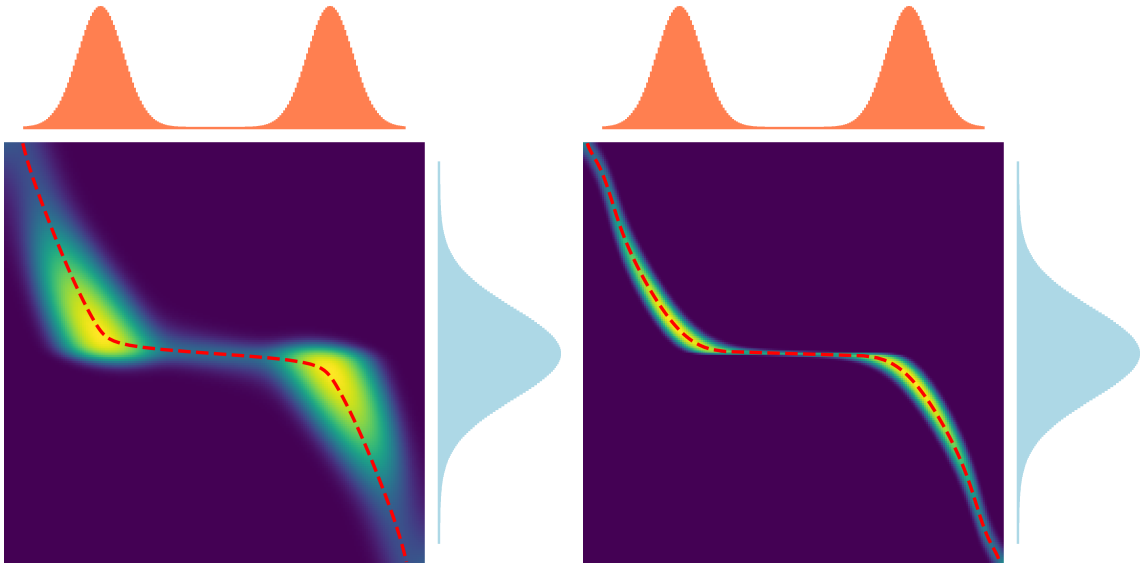


Figure 13: Barycentric maps for multiple values if ϵ. Notice how diffused the solution is as ϵ increases. The barycentric map is meaningless for these values.



(a) $\epsilon = 0.09^2$ (b) $\epsilon = 0.02^2$

Figure 14: Going from a unimodal distribution to a bimodal distribution. As ϵ goes to 0 and the coupling converges to the unregularized coupling, we may wonder what happens to the discontinuity in the middle.

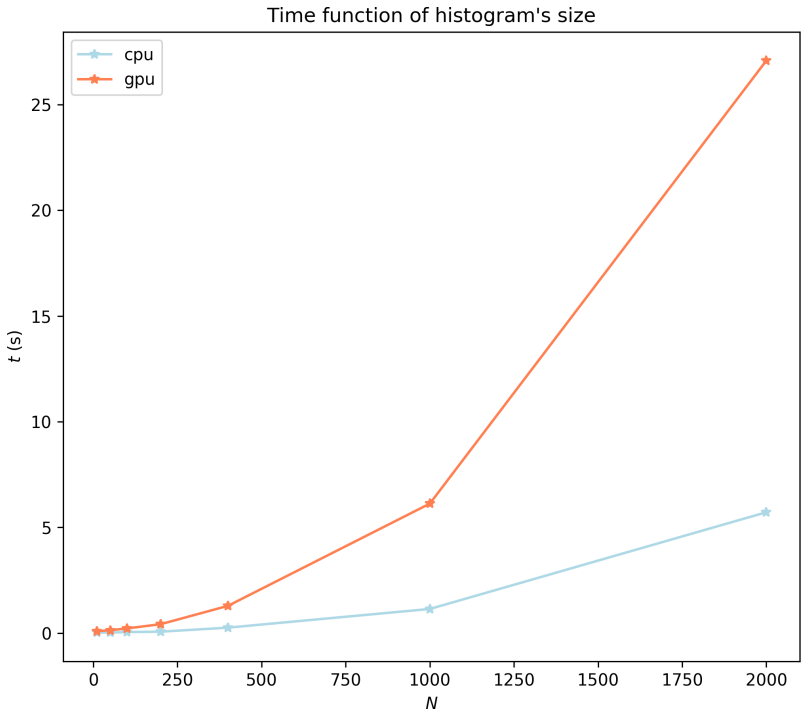


Figure 15: Comparing GPU vs. CPU execution times. To my utter disbelief, the red curve is in fact corresponding to the GPU performance.

2.4 Wasserstein barycenters

Here, I tried to see to what extent the Wasserstein barycenter could be used for image registration. Basically, let's say we have a set of brain scans. To compute statistics over this set, and give a meaning to mean, standard deviation, cluster or whatever, we need to specify a *distance* between our images. Optimal transport is a solution to that problem.

Concerning Wasserstein barycenters, I wondered if the barycenter of multiple brain scans could be used as a **mean** accross all our samples. Such mean is referred to as **atlas** in the corresponding literature.

So here was my question; can the Wasserstein barycenter of multiple brain scans be a relevant atlas?

2.4.1 Data

I did not have at my disposal fresh and real brain scans. So I generated *phantom brains*, which simply consist of ellipses at several specific points on the image. They correspond to a basic saggital cut of the brain. To denote the **variability** across individual, I dilated, translated, rotated each of these ellipses randomly.

The basis phantom is shown in fig. 16, along with samples from it.

Figure 16: The template (left) and three sampled phantoms.

In a second attempt, I chose simpler shapes as input. Those are disks whose center lies on a circle, center in the middle of the image. Samples are shown on fig. 17.

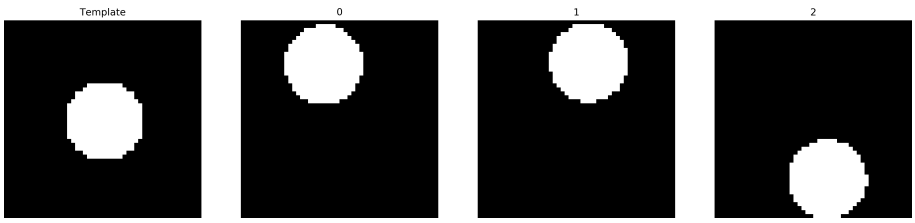


Figure 17: The template (left) and three sampled circles.

2.4.2 Result

I struggled quite a bit to get this part right. My first assumption was: "the more samples you have, the better is the barycenter". In fact, the number of samples is far from being the bottleneck here. Rather, it's the ϵ **parameter of the entropy** which is the real issue. Indeed, we quickly have enough samples to have a stable barycenter. However, we need to lower ϵ a lot to get a precise result.

More precisely:

- Refer to fig. 18 to get an overview of the evolution of the quality of the barycenter for various number of phantom brain samples and entropic parameter ϵ. Refer to fig. 19 for the same picture with the circles dataset.
- Refer to fig. 20 and 21 for the evolution of the loss, function of ϵ and the number of samples. You might notice that the L_2 loss was a poor choice[1].

[1]In fact, I truly lacked insights here: I searched for a while a good loss, trying L_1, L_2 or a "dice-score" loss. But it's precisely the purpose of OT to leverage a distance between points to yield a distance between distributions! My bad.

- Chronologically, I started with the phantom brains data set, with a fixed $\epsilon = 0.01$. I tried computing the barycenter with *a lot* of samples (up to a 1000), until I realized my error: we are not computing the optimal barycenter, but merely an entropic regularization of it.

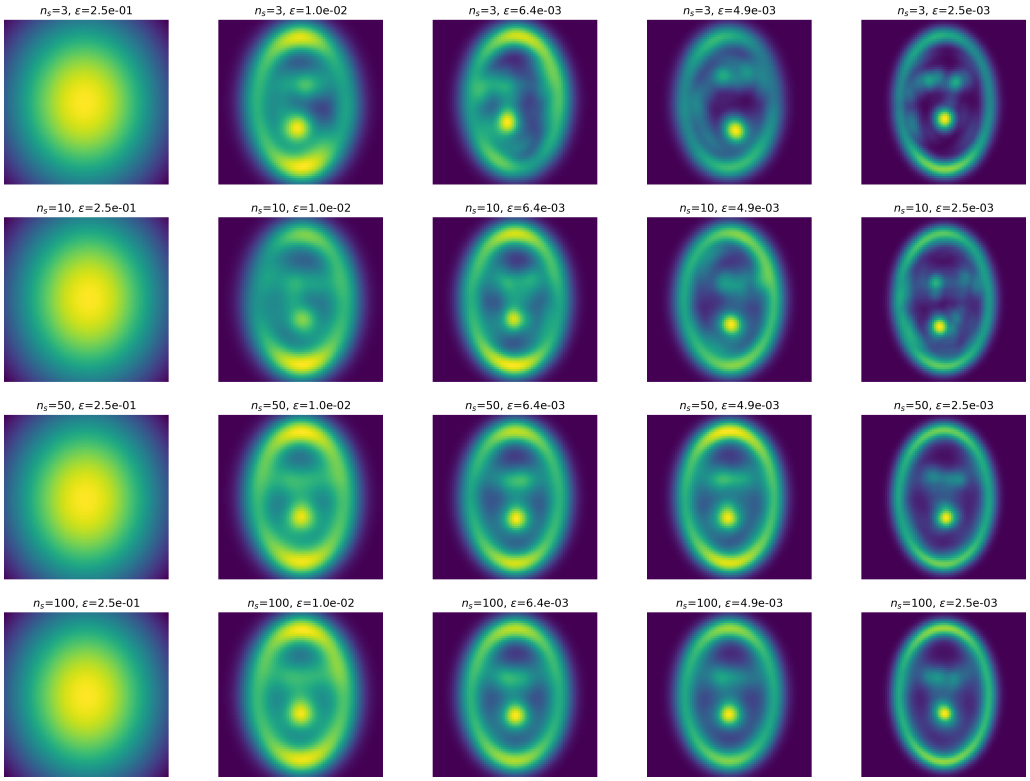


Figure 18: Barycenter of multiple sampled brains. On **rows** are varying number of samples n_s, ranging from 3 to 100. On columns are varying values of ϵ, ranging from 0.5^2 to 0.05^2. Notice how quickly having more samples becomes irrelevant.

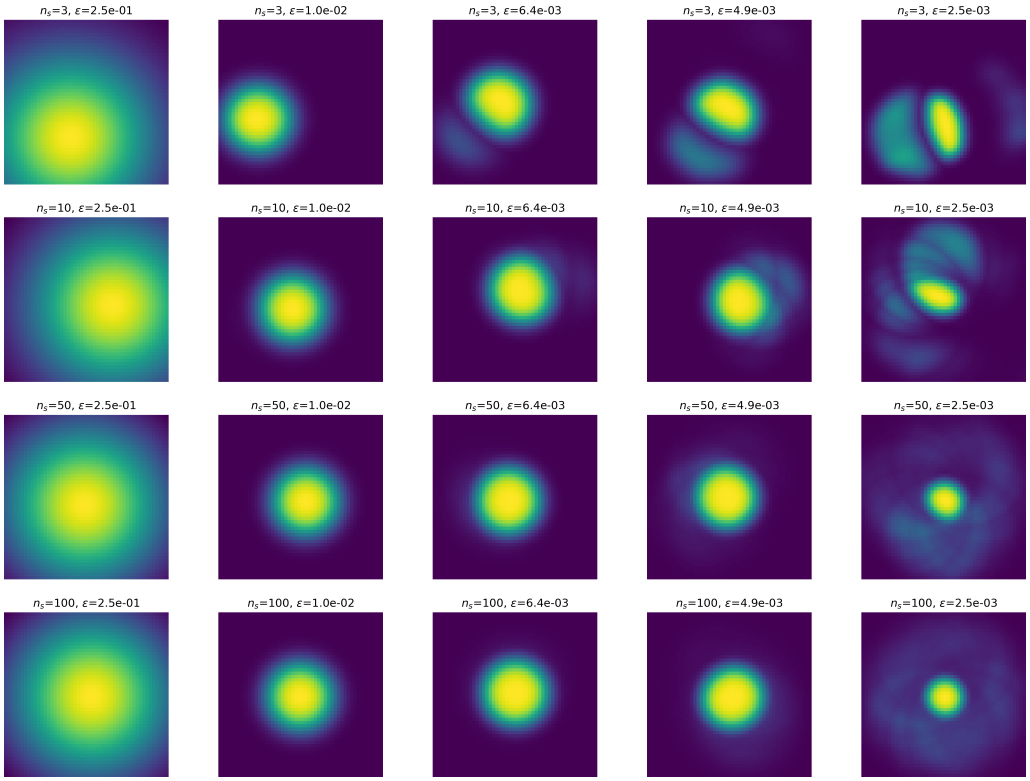


Figure 19: Barycenter of multiple sampled circles. On **rows** are varying number of samples n_s, ranging from 3 to 100. On columns are varying values of ϵ, ranging from 0.5^2 to 0.05^2. Here, we see that the L_2 loss is not appropriate, because the mass is concentrated in the middle. Trying a dice score was not helping. A more intricate loss which would consider the support of half the mass could be better (refer to foot note on page 13).

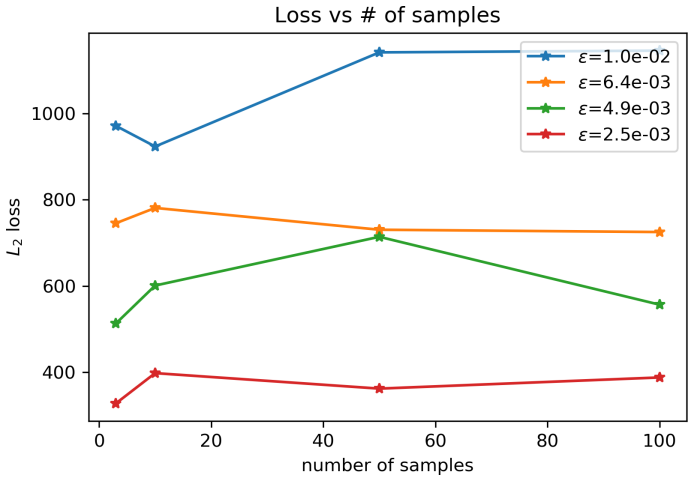


Figure 20: L_2 loss between the barycenter and the template, for the **phantom brains** dataset. As ϵ gets smaller, the barycenter becomes closer and closer to the expected template.

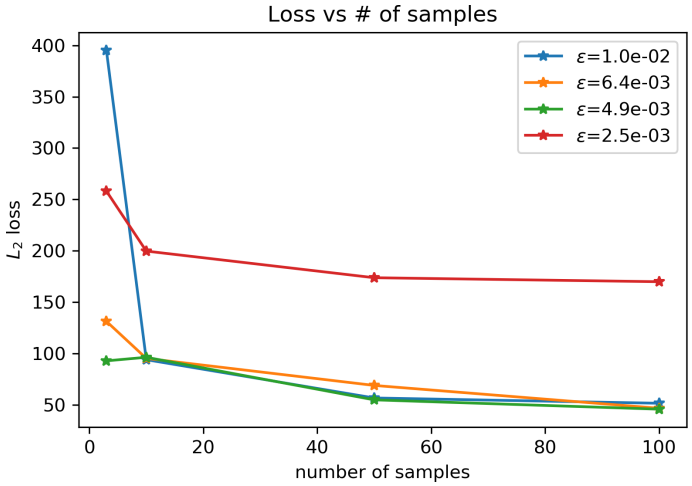


Figure 21: L_2 loss between the barycenter and the template, for the **circles** dataset. Here, we might be surprised that the loss does not decrease with the value of ϵ. In fact, it's due to the L_2 loss being ill suited to measure the discrepancy between the images.

3 Semi discrete Optimal Transport

This tour treated the case when one of the measure was discrete while the other had a continuous density. The primal and the dual reads:

$$W_c(\alpha,\beta) = \min_{\pi} \left\{ \int_{\mathcal{X}\times\mathcal{Y}} c(x,y)\mathrm{d}\pi(x,y) : \pi_1 = \alpha, \pi_2 = \beta \right\}$$
$$W_c(\alpha,\beta = \max_{f,g} \left\{ \int_{\mathcal{X}} f(x)\mathrm{d}\alpha(x) + \int_{\mathcal{Y}} g(y)\mathrm{d}\beta(y) : f(x)+g(y) \leq c(x,y) \right\}$$

If $\alpha \sum_i a_i \delta_{x_i}$ and we replace g by its c transform, we obtain the following problem which is much more tractable:

$$W_c(\alpha,\beta = \max_{f} \left\{ \sum_i f(x_i)a_i + \int_{\mathcal{Y}} f^c(y)\mathrm{d}\beta(y) \right\} \tag{1}$$

The c-transform of the discrete measure is a piece-wise smooth function on the **Laguerre** cells, defined by:

$$L_i(f) \triangleq \{y \mid \forall j, c(x_i,y) - f_i \leq c(x_j,y) - f_j\} \tag{2}$$

Differentating the score defined in (1), we obtain:

$$\partial_i E = a_i + \int_{L_i} f^c(y)\mathrm{d}\beta(y) \tag{3}$$

3.1 Idea

I lacked inspiration for this one, and got my idea from the news[2]. Madagascar is indeed claiming the ownership of the Scattered Island over France, which are a key strategic point, granting a large area of exclusive economic zone. I am not siding with any, but I was wondering how would look an **optimal assignment of sea resources** in that case. Indeed, we might want that all sides have equal access to the resources available; taking into consideration the distance between the land and the sea amounts to solving a transport problem.

I did not spent too much time on this problem: it is directly an application of the notebook and does not explore mathematical ideas. Thus, the presented results could be easily optimized in various ways.

3.2 Data

The `oceansdb` offers a database of the **tomography** of the oceans. We use a snapshot of the zone of interest, shown fig. 22. The discrete measure α represents points affiliated to the country in place. For big land part, I used multiple points; this could be improved by changing the distance between a point and the land. The continuous measure is simply the tomography of the ocean, ie. the ocean's depth. Indeed, we can model the **level of resources** with the **ocean's depth**.

On a first try, I used:

$$\beta(y) = \mathrm{depth}\,(y) \tag{4}$$

ie. the deeper the ocean, the more resources there are. On a second try, I used a Gaussian kernel centered on -1000m, to give less importance to parts which are too deep or too shallow. It seems that around 1000m under the level of water, one can start finding rare metals and still be able to exploit them[3] [4].

$$\beta(y) = \exp\left(-\frac{\mathrm{depth}\,(y) - \mu}{2\sigma^2}\right) \tag{5}$$

3.3 Gradient ascent

The optimization problem (1) can be solved by gradient ascent. Here, the number of variable is low so I did not notice any improvement in running a stochastic gradient ascent instead. The evolution of the Laguerre maps are shown fig. 23 and the dual value are fig. 24.

[2] Le Monde, *Madagascar organise une concertation nationale sur les îles Eparses*

[3] Again, this is just an example and does not reflect my views or motivations!

[4] National Geographics: the Ocean

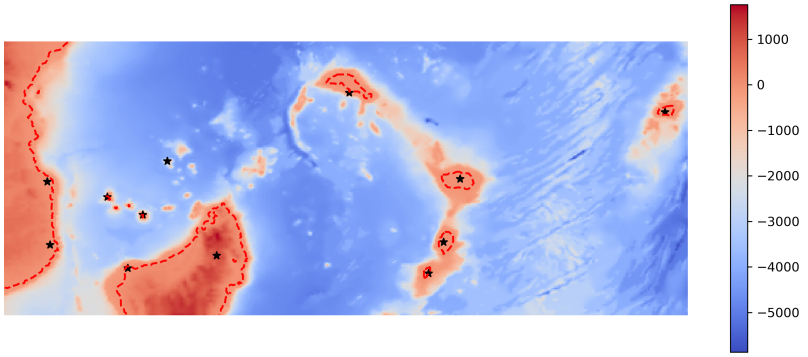
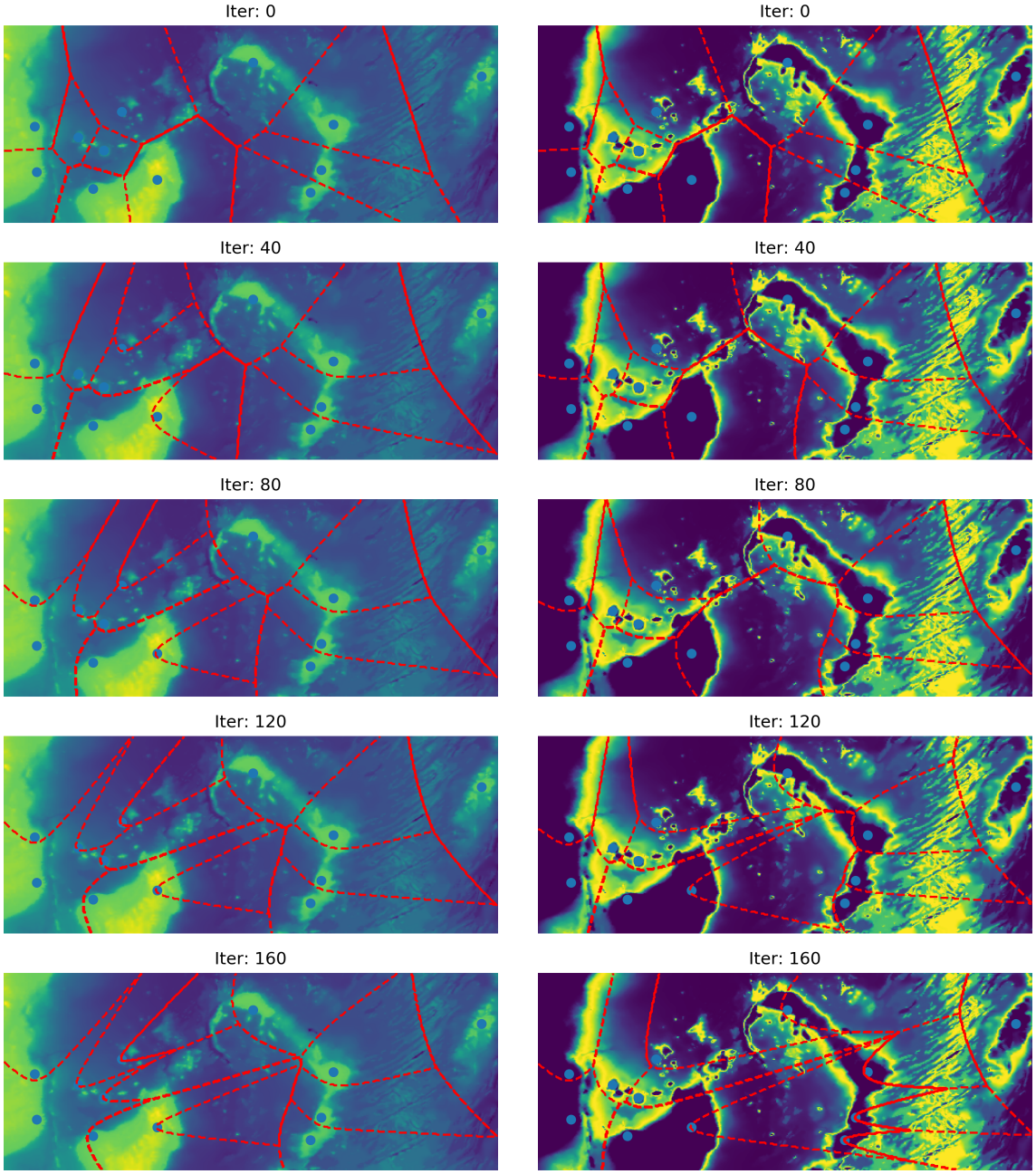


Figure 22: The area we are interested in, between latitude (-2, 19), and longitude (38, 73). On the west, the Mozambique. On the center, Madagascar. On the right, Seychelles and an island belonging to the UK. The Scattered Island are a few miles north of Madagascar. The support of the discrete measure are the 10 points scattered across the picture. The continuous measure (ie the ocean's tomography) is sampled on a fine grid of size (500, 300).

3.4 Results

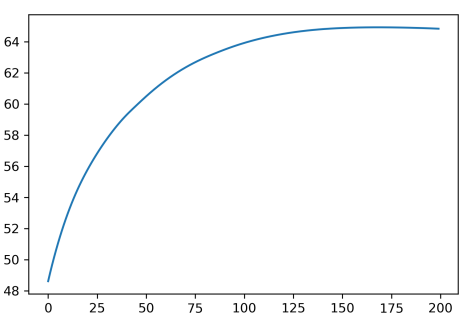
Finally, the results are shown fig. 25. Of course, they are terrible. But I find it still satisfying to know that each of these color players have an equal access to the ocean's resources. If we had taken a better distance, e.g. one which would compute the smallest distance to a point on the land, we would not have the Madagascar Island cut this way.



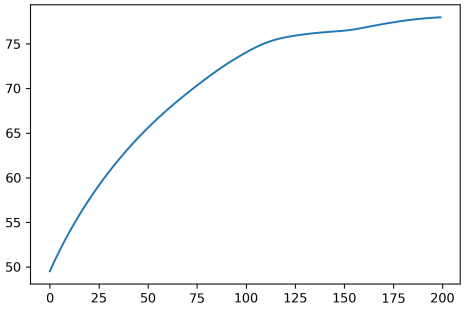
(a) With depth as the measure β (eq. 4).

(b) With a Gaussian kernel on depth as the measure β (eq. 5).

Figure 23: Evolution of the Laguerre maps during the gradient ascent, with two different measure for β.

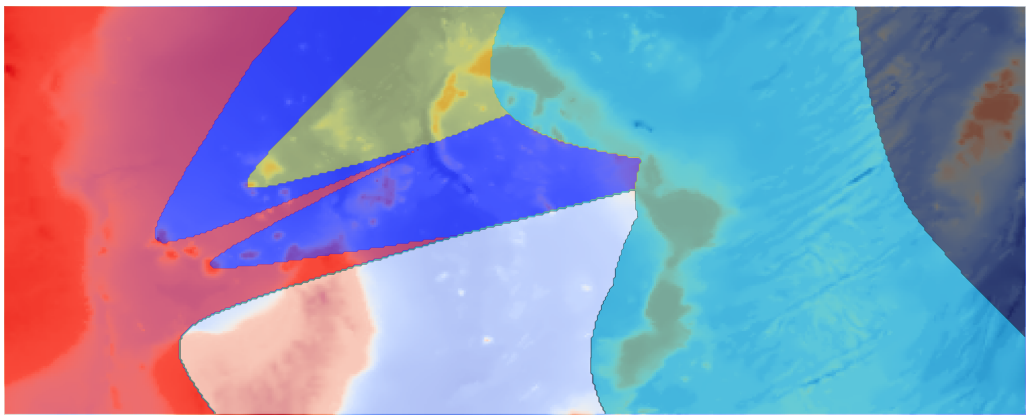


(a) With depth as the measure β (eq. 4).

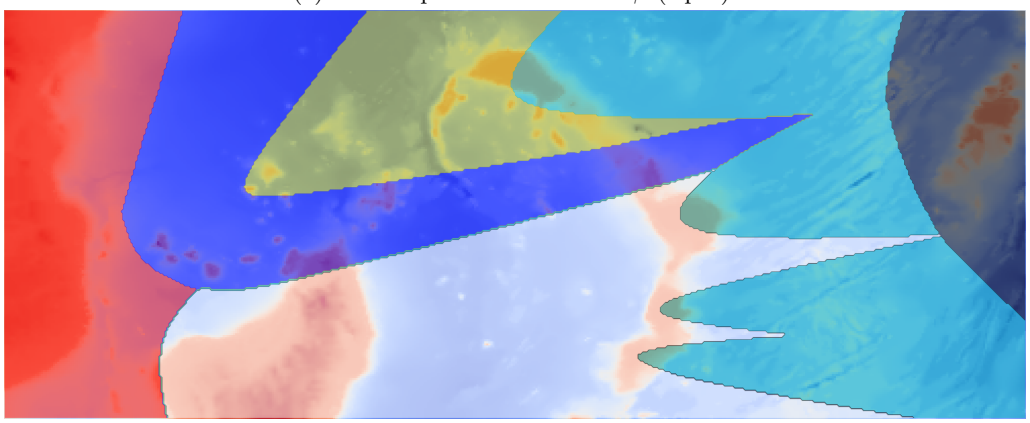


(b) With a Gaussian kernel on depth as the measure β (eq. 5).

Figure 24: Evolution of the dual value during the gradient ascent, with two different measure for β.



(a) With depth as the measure β (eq. 4).



(b) With a Gaussian kernel on depth as the measure β (eq. 5).

Figure 25: Final coupling found by dual ascent, for two different kind of distribution β.

4 Advanced Topics on Sinkhorn Algorithm

During this tour, we dived a bit more into Sinkhorn's algorithm and various adaptation.

4.1 Wasserstein Flow for Matching

There are a few things to say in this section. To begin with, I used the data provided at the beginning of the notebook, ie. $n = 200$ points sample on a square for α and $m = 100$ points sample on a circle for β, with uniform weights. The result of the flow, for various values of the regularization parameter ϵ and at different steps of the gradient descent are shown fig. 26.

Results A few things to say about the results:

- It seems that the bigger ϵ, the smaller the the step size for the gradient descent should be. That, or there is a **threshold** effect on ϵ, aboce which the entropy term always wins at the expense of the trace term.
- Below that threshold, it seems that all distribution seem to converge well. As expected, if one wants a **precise** final result, one needs to **lower** ϵ accordingly.

Finally, I plot the final score (at the end of the gradient descent) to see the contribution of the trace and of the entropy. Result is fig. 27.

Execution time The implementation did not posed any difficulties, except for a surprising result. Below, on table 1, I show the execution time for the multiple gradient descents.

At first, I thought: "totally fine: the smaller ϵ, the sparser the solution and the greater the computation time". However, each gradient step uses a **fixed** number of Sinkhorn steps. The stopping criteria is not about tolerance. Thus, this result was really surprising.

After looking up the code through and through, it is in fact due to a kind of exotic issue: it has to do with Numpy's `exp` function. Looking at the source code, it appears the computations work in two ways:

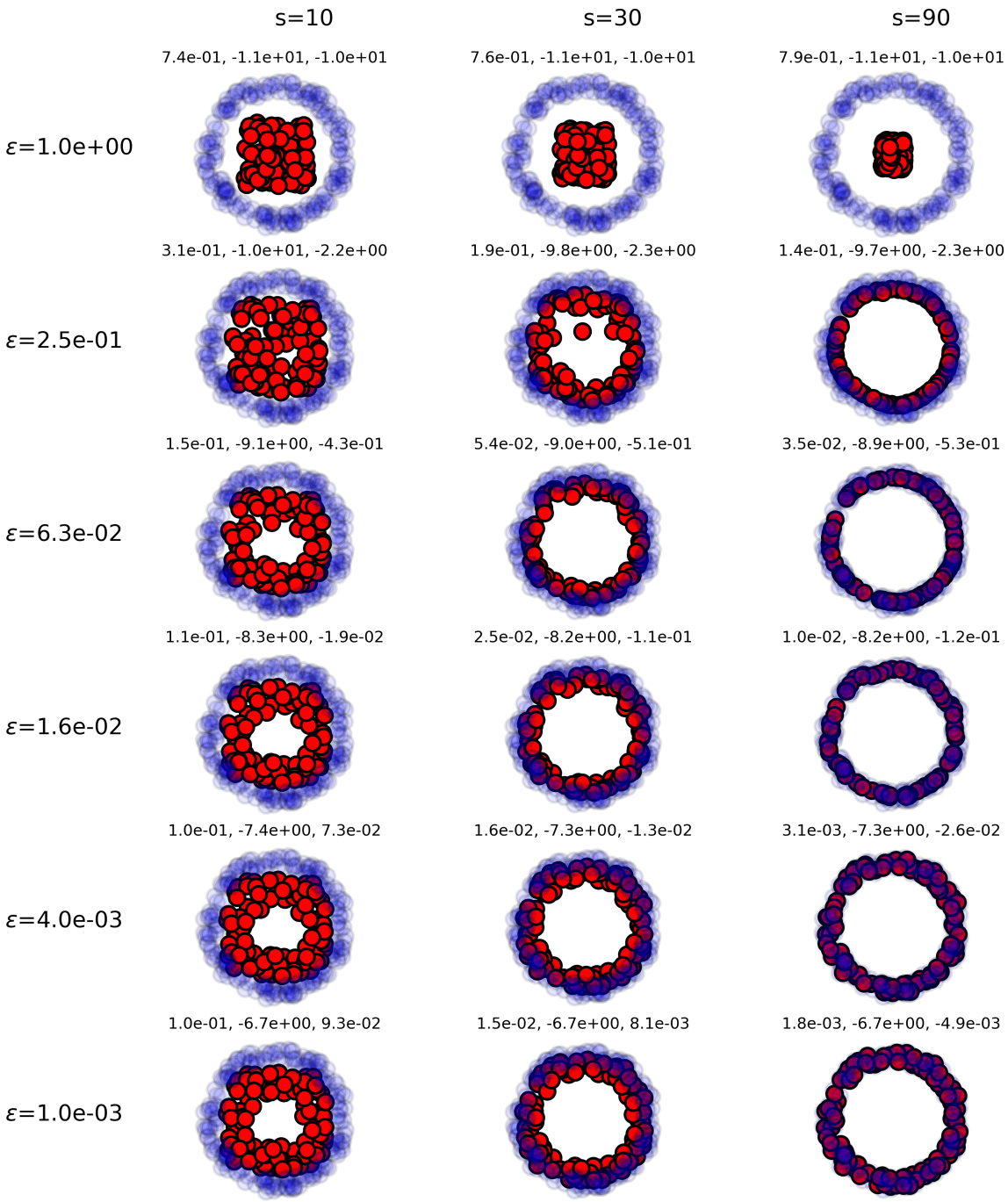


Figure 26: Wassertein flow. Rows are sorted with decreasing value of the regularization parameter ϵ. Columns are sorted with increasing values of timesteps during the descent. Above each plot is indicated the **energy** of the current z: first the trace, then the entropy, then the weighted sum of both with $(1, \epsilon)$. The best trace term is achieved with the lowest ϵ, at the end of the descent. The maximum entropy is reached with the biggest ϵ. During all simulation, the step size is set to $\tau = 0.5$. It seems too much for $\epsilon = 1$

ϵ	t (s)
1.0	21
$2.5\,10^{-1}$	25.3
$6.3\,10^{-2}$	27.8
$1.6\,10^{-2}$	23.7
$4.0\,10^{-2}$	85.1
$1\,10^{-3}$	259.1

Table 1: Execution time of the Wassertein flow for various values of ϵ. Each run used 100 gradient steps, and each used 200 Sinkhorn steps to compute the optimal transport plan P. The time should be roughly constant, as complexity does not depend of ϵ.

1. First, look up the values in a table, and compute the rest with the desired precision.
2. If that does not give a sufficient precision, it falls back on a classical Taylor expansion computation.

In fact, it appears it is exactly the problem we are facing here. To be sure, I computed $x \mapsto e^{x/\epsilon}$ for 10000 values of x between 0 and 1, and recorded the computation time. The result is shown fig. 28.

It is purely a numerical issue. Yet, it can be very annoying in our case, with a factor 10 in the computation time between for $\epsilon > 10^{-3}$ and smaller values.

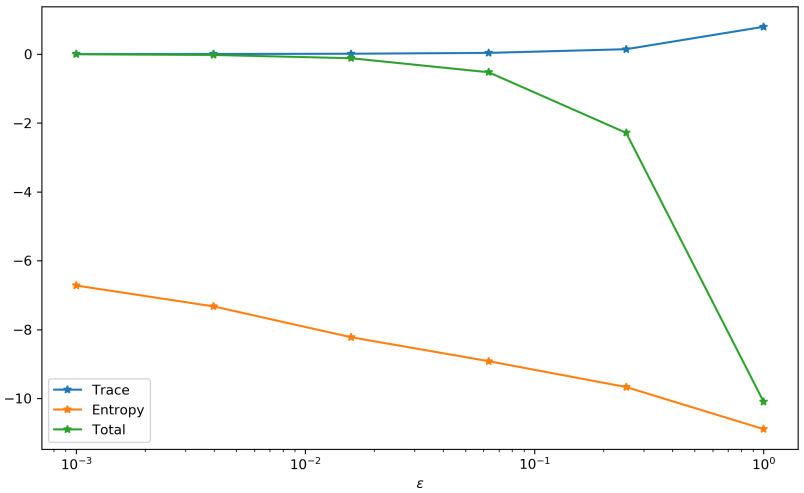


Figure 27: Value of the energy at the end of the descent. As ϵ grows, the entropy weight more and more in the energy term. When it is small, we are the closest to our objective.

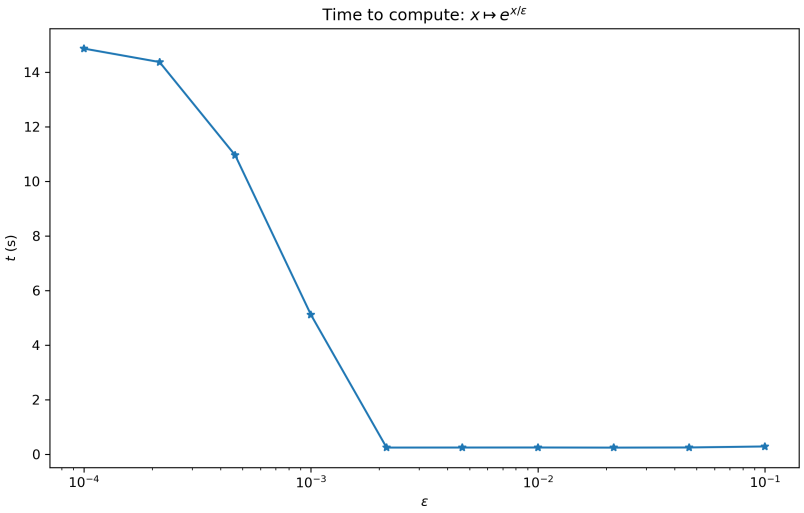


Figure 28: Execution time of $x \mapsto e^{x/\epsilon}$ for 10000 values of x between 0 and 1, function of ϵ. If its value is too small, Numpy computes a Taylor expansion which greatly increases the computation time.