Computational Optimal Transport

Smooth and Sparse Optimal Transport

Professor : Gabriel Peyre

Author : Hugo Negrel

Abstract

In the new era of data science where very large data sets are required to be treated efficiently and quickly, Optimal Transport establish a very general formalism to give a hint on how to do it. As even the Kantorovich problem can be quite difficult to deal within very high dimension with the simplex algorithm, the usual approach consists in regularizing it with a Shannon-Boltzmann entropy term, making the problem solvable via the very famous Sinkhorn algorithm [1]. However, adding an entropy term tends to spread the data and very drastically reduces the sparsity of the transportation plan. This is bad news, because a non sparse matrix is numerically painful to deal with. Conversely, sparse matrices are the guarantee of good conditioning and therefore algorithmic efficiency, regarding stability or speed of computation [2]. This problematic is the main starting point of our approach. This work, based on [3], first sets a general formalism for regularisation, and not only for entropy one. We will be particularly interested in strongly convex regularisation, since they have very nice convergence properties which are conserved when switching to the dual. The dual can easily be derived and will be preferably solved, as it turns out to be unconstrained. Another approach is studied, inspired on discrete c-transform, with thus a third formulation. Eventually, a last approach where equality constraints are relaxed on the primal is derived. All these approachs have pros and cons, and it will be interesting to discuss the differences about them. The regular OT case will also be considered, in order to compare with the exact problem. The main technique employed is colour transfer [4], like in [3], but with my own images. The main optimisation algorithm used is L-BFGS, even though a try with FISTA was tempted.

1 Introduction

Optimal Transport (OT from now on) has become quite a trendy topic these last decades, whether it is regarding its theoretical aspects [5] or its applications, particularly in data science [6]. The problem, originally formulated by the french mathematician Monge :

$$\tilde{\mathcal{L}}_c = \inf_{T_\# \alpha = \beta} \int_{\mathcal{X}} c(x, T(x)) dx$$

with $T : \mathcal{X} \to \mathcal{Y}$ a bijection and $c(\cdot, \cdot) : \mathcal{X} \times \mathcal{Y} \to \mathbf{R}^+$ a cost. This formulation is too restrictive (e.g the feasible set can be empty), and it is too hard to solve in general. Kantorovich then got the idea to relax the constraint $\{T \quad \text{bijection with} \quad T_\# \alpha = \beta\}$ by making them convex , and consider the Cartesian product of the space $\mathcal{X} \times \mathcal{Y}$ with the following formulation :

$$\mathcal{L}_c = \inf_{\substack{P_{1\#}\pi = \alpha \\ P_{2\#}\pi = \beta}} \int_{\mathcal{X}\times\mathcal{Y}} c(x, y) d\pi(x, y) \tag{1}$$

with P_1 and P_2 respectively the projector on first and second component. This formulation has the good property to always have non empty feasible set, and infimums are always reached on compact set. In most practical case, data is only discrete, because computers can only handle discrete quantities. Theoretical results of existence or stability on the continuous case are still of interest though. Let's consider our data are $(x_i, y_j)_{i \in [\![1,m]\!] \times [\![1,n]\!]} \in \mathcal{X} \times \mathcal{Y}$. In this work, $\mathcal{X} \simeq \mathbf{R}^m$ and $\mathcal{Y} \simeq \mathbf{R}^n$. Therefore, the usual way to deals with (1) is with the LP :

$$\min_{T \in \mathcal{U}(\mathbf{a}, \mathbf{b})} (T, C) \tag{P}$$

where $\mathcal{U}(\mathbf{a}, \mathbf{b}) = \{T \in \mathbf{R}_+^{m \times n} | T\mathbf{1}_n = \mathbf{a}, T^\top \mathbf{1} = \mathbf{b}\}$ can obviously be put under the form of a polytope $\{x | Ax = b, x \geq 0\}$. As a LP, a regular simplex algorithm can be applied. $C \in \mathbf{R}_+^{m \times n}$ is the cost matrix whose elements are $C_{i,j} = c(x_i, y_j)$. However, whenever m and n are very large, this kind of approach is computationally expensive (polynomial temporal complexity on average). Another issue is the possibility to have an infinity of solutions, and so we can not *a priori* choose one in particular. A clever trick is to introduce an entropy regularization, which can sounds similar to an interior point method for the positivity constraint at first. Therefore, we consider the Schrodinger problem :

$$\min_{T \in \mathcal{U}(\mathbf{a}, \mathbf{b})} (T, C) - \gamma H(T)$$

where H is the entropy of T i.e $H(T) = -\sum_{i,j} T_{i,j} \log T_{i,j}$. This approach has many advantages. First, it makes the objective γ-strongly convex, so we are assured to have a unique solution, second it makes it possible to solve it with the so-called Sinkhorn algorithm, which is consecutive KL-projection on the closed convex sets : $\mathcal{C}_a = \{T \in \mathbf{R}_+^{n \times m} | T\mathbf{1}_n = \mathbf{a}\}$ and $\mathcal{C}_b = \{T \in \mathbf{R}_+^{n \times m} | T^\top \mathbf{1}_m = \mathbf{b}\}$. However, the main drawback of the schrodinger problem is that it tends to make the data non-sparse. The definition of sparsity is not really formal, but a matrix is said to be sparse if most of its elements are equal to 0. In a computational context (not a theoretical one), this is a big issue in HPC, since it is well known that sparse matrix are very algorithmic-friendly, because of their (forward and backward) stability and the speed of computations associated [2]. The l_1-norm regularisation in LASSO was coined for this purpose. Conversely, if the matrix is non-sparse, it makes the algorithms much unstable on very large data sets, and very ill-conditioned. In a more OT context, sparsity is demanded for parsimony and interpretability. Indeed, non-zero components shall attract our interest since it points us where the transport is optimal and gathered. If the matrix is non sparse, it can be very challenging to determine where the real quantities of interests are. Eventually, in compressed sensing, one often wants to solve the problem :

$$\begin{array}{ll} \min\limits_{x \in \mathbf{R}^n} & ||x||_0 \\ \text{s.t.} & Ax = b \end{array} \tag{2}$$

where $||x||_0$ is the counting norm i.e $||x||_0 = \sum_i \mathbf{1}_{x_i \neq 0}$. By the way, this problem is usually approached by the so-called LASSO problem. On a side note, sparsity is a big pro of the Finite Element Method. Another entropy regularisation exists, based on the Kullback-Leibler divergence

$$\min_{T \in \mathcal{U}(\mathbf{a},\mathbf{b})} (T, C) - \gamma KL(T||\mathbf{a} \otimes \mathbf{b}).$$

but I won't consider this form, since it does not have any interest in this context.

As a consequence, we could be interested in others approach to penalise (P) which encourage sparsity while keeping the desirable strongly convex property for optimisation purposes. [3] proposes to consider the generic form

$$\min_{T \in \mathcal{U}(\mathbf{a},\mathbf{b})} (T, C) + \sum_{j=1}^{n} \Omega(T_{\cdot,j}) \tag{3}$$

with Ω γ-strongly convex on either $\text{dom}(\Omega) \cap \mathbf{R}_+^m$ or $\text{dom}(\Omega) \cap \Delta^m$. Δ^m here is the m-simplex, and $\mathbf{t}_j$ is the j-th column of T. Recall that the effective domain is defined as : $\text{dom}(f) = \{x | f(x) < +\infty\}$. Let's first notice we naturally recover the Schrodinger problem for $\Omega_{\text{ent}} = -\gamma H$, which will be helpful for the comparison of our methods. [3] proposes two functions to compare with the entropy one : squared-2 norm and LASSO group. Squared-2 norm is the usual euclidean norm squared $\Omega_{norm} = \frac{\gamma}{2}|| \cdot ||_2^2$ while group LASSO is a bit more specific : $\Omega_{LASSO} = \gamma \sum_{i=1}^m y_i \log(y_i) + \mu \sum_{G \in \mathcal{G}} ||y_G||$, where y_G is the restriction of y to this subset and not the LASSO one, but both have the same objective : increase sparsity. The point is now to exploit as much as possible the convenient formulation (4) to solve it efficiently. In particular, we will explore two equivalent formulations based on the dual and semi-dual of (4), as well as a relaxed and semi relaxed version. The dual, semi-dual and relaxed formulations are the key points of this report, as they allow to improve previous works. In particular, one do not really need Ω to be strongly convex and differentiable to derive their dual (even if it is algorithmically desirable). It is done differently in [7], as the regularization is added in the constraint of the primal and therefore appear in the objective value in the dual. In this case, the differentiable and strongly convex property was necessary to interpret (4) as a Bregman divergence. As we will see, the very main advantages being that the dual and semi-dual are unconstrained, and can therefore be minimised via an arbitrary solver. The LASSO group will not be treated here, by lack of time, so I'll focus on comparing squared-2 norm and entropy regularization.

This reformulation of the unregularized OT (P) found some application on the domain adaptation problem [8]. Domain adaptation problem starts from the observation than not all positions of sensing devices during acquisition are equivalent in terms of efficiency and conditions of observations. Suppose one has data acquisition on day N at location X, and one would like to compare it or use it with an other data set taken on day M located at Y. This issue happens a lot for example in Computer Vision or NLP. Having learned an empirical decision function from the first data set, one would like to practice inference on the second one. Problem : there is no reason, and it is actually not the case, that probability distributions on day N at X is the same as day M at Y[8]. However, this property is crucial for any inference activity. To solve this problem, one would like to *transport* the probability distribution of the learning data set to the one of the testing data set. Optimal Transport can be formulated to be relevant in this context. This kind of issue helps to fathom the issue of what it is to be identically distributed and independent. Figure 1 illustrates the usual setting in optimal transport.

This report focused on colour transfer. An impressive example on video is here [4]. Colour transfer is an image processing technique on transferring color characteristics from a 'reference' image onto a 'source' image. Thus, the source image keeps it mains visual features (a tree remains a tree or a lake remains a lake) while changing of colors. It can be certainly useful for old videos only available on white and black, see Figure 2.

Concerning my personal work, all the code used was written by myself. Colour transfer was done on images found on https://www.flickr.com/explore/. Since the color transfer technique is not actually the subject on which my work is focused I did not really consider to perform something else, as I did not have really an interest in doing so for comparing methods. Concerning algorithms, I wrote the very

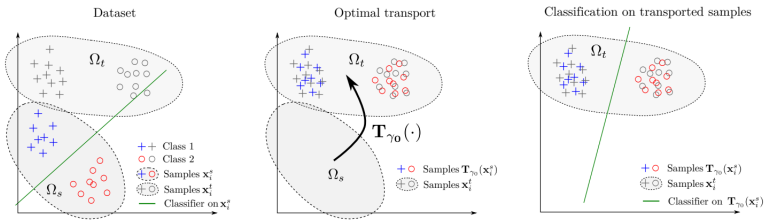


Figure 1: Domain adaptation setting in Optimal Transport. The classification line is transported by T_{γ_0}.Credits : [8]

Figure 2: Example of colour transfer. The reference image is located on the bottom left corner. Credits : [4]

majority of my code. The only algorithms I couldn't code are naturally the optimizer, since it would have taken a great amount of time, and it is once again the main objective of this work. The main optimizers I used are from scipy and CVXPY. The objective here was to verify the sparsity property of the 2-squared norm compared to the entropy regularization and study its speed of convergence and how close we are from the true transport plan $T^\star$. Therefore, the point is to study how relevant an regularization is with respect with each other. Many interesting properties emerge, as we can explore its connections with a certain duality for each regularization. Some theoretical bounds are given in [3], it will be interesting to check them at the end.

The plan will be the following :

1. I'll explain the principle of colour transfer and how we introduce the optimal transfer formalism for this image processing technique.

2. I will present the method of regularization, as well as its theoretical properties and its links with duality. I will give certain technical details on the algorithmic aspects of my code.

3. Eventually, I'll compare the regularized problems with visual graphics, and try to be critic regarding their computational performance.

I will not discuss how the algorithms solely related to colour transfer technique behave for large n and m, like k-means or barycentric projection (see Section 2). I will only be interested on how efficiently we can solve (P).

2 Colour transfer technique

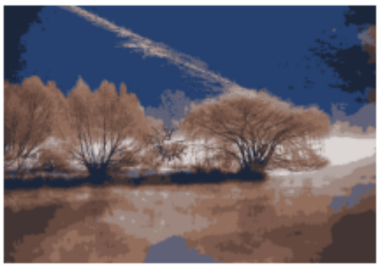
Colour transfer is an image processing technique whose objective is to give a new color to an image, while keeping a certain coherency overall. To perform it, one needs a 'reference' image and a 'source' image (Figure 2). The 'source' image is the one we wish to modify the colour, while the reference image is the one we use to to modify it. For all the report, the reference and source image are displayed on Figure 3

(a) Reference image

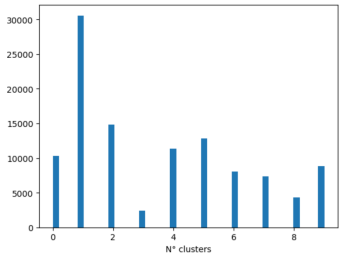
(b) source image

Figure 3

We process as follows. First, we convert reference and source image in list, where we store for each pixel its 'RGB' decomposition. Then, we use **k-means** on these $3D$ points to gather them in a given number of cluster. The number of cluster does not have to be the same in principle for reference and source image, there will be noted respectively m and n. I tried to implement myself the k-means algorithm (also called LLoyd's algorithm) but it would only stay blocked on local minima with 3 or 4 non-null locations. Indeed, k-means algorithm can be quite sensitive to initialization, and an other algorithm have to be employed for that, names 'k-means++'. I did not try to implement this one, and decided to use the command provided by sklearn. Along this work, I imposed that $n = m$. Once the pixels of the picture are clustered, we can compute a histogram based on it, storing the number of points on each cluster, as well as their location. Let's precise that we know to what cluster each pixel is assigned to. With this histogram, we introduce two vectors $\mathbf{a} \in \mathbf{R}_+^m$ and $\mathbf{b} \in \mathbf{R}_+^n$ for respectively the reference and source image, with the definition $a_i = \#number\ of\ pixel\ in\ i\text{-}th\ cluster$. Finally, we normalize them, in such a way that $\mathbf{a} \in \Delta^m$ and $\mathbf{b} \in \Delta^n$. An example of histogram is shown on Figure 4b



(a) Original image after clustering of colours n = 10



(b) Histogram of the clustering. Ordering is arbitrary. The data of this histogram is noted as **b** from now on.

Figure 4

Once we have our histograms **a** and **b**, we want to compute the optimal transportation plan associated to them, that is solve the problem. The cost matrix $C \in \mathbf{R}_+^{m \times n}$ is naturally defined as $C_{i,j} = ||\hat{\mathbf{x}}_i^{\text{ref}} - \hat{\mathbf{x}}_j^{\text{src}}||^2$, with the obvious notations that $\hat{\mathbf{x}}_i^{\text{ref}}, \hat{\mathbf{x}}_j^{\text{src}}$ are respectively the i-th, j-th cluster of the reference and source image. In practice, we implement the stabilized version of it i.e $C_{i,j} = ||\hat{\mathbf{x}}_i^{\text{ref}}||^2 + ||\hat{\mathbf{x}}_j^{\text{src}}||^2 - 2(\hat{\mathbf{x}}_i^{\text{ref}}|\hat{\mathbf{x}}_j^{\text{src}})$. The problem (P) is solved here with CVXPY, the preliminary being that we have to put this problem as a standard LP, which was a little bit tricky for the constraint,

as we have to move from the space of matrix $\mathcal{M}_{m\times n}(\mathbf{R})$ to $\mathbf{R}^{nm}$. We then solve

$$\begin{array}{ll} \min\limits_{t} & (\mathbf{c}, \mathbf{t}) \\ \text{s.t.} & At = d \end{array} \tag{4}$$

where $d = [a, b]^\top$, $T_{i,j} = \mathbf{t}_{i+(j-1)m}$, $C_{i,j} = \mathbf{c}_{i+(j-1)m}$. A have the adequate structure to ensure that $\sum_i \mathbf{t}_{i+(j-1)m} = b_j$, $\sum_j \mathbf{t}_{i+(j-1)m} = a_i$. Once (4) is solved, we compute the barycentric projection, i.e $\hat{\mathbf{x}}_i = \arg\min_{\mathbf{x}\in\mathbf{R}^3} \sum_j T_{i,j}||\mathbf{x} - \hat{\mathbf{x}}_j^{ref}||$. Eventually, each pixels which were assigned to $\hat{\mathbf{x}}_i^{src}$ will be now assigned to $\hat{\mathbf{x}}_i$. Reconstructing the image from these new clusters, the result is displayed on Figure 5.

(a) Colour transfer on source image for $n = m = 10$. (b) Colour transfer on source image for $m = 10, n = 50$.

Figure 5

An important technical detail is the **scaling issue** of objective function when using numerical solver. Indeed, since the scalar values in c are in order of magnitude $\approx 1e3$ while d is ≈ 1, the numerical solver is not well designed to handle value with several order of magnitude of difference. Therefore, the optimizer couldn't work at first. It is a common issue in computational optimization. As a result, it was necessary to re-scale C by dividing every cluster center $\hat{\mathbf{x}}^{ref}$ and $\hat{\mathbf{x}}^{src}$ by $2^8 = 256$, to bring the values between 0 and 1. Only this way it would works, and it is obvious it does not change at all the optimal set by linearity.

To sum up, this part presented the colour transfer technique and the standard mean to solve it. A direct application was displayed. By solving the original Optimal Transport problem, we consider knowing $T^\star$, and therefore will be useful for comparisons in section 3. I'll give some mathematical and numerical details, while remaining concise.

3 Approximation methods for Kantorovich's Optimal Transport problem

In this section, I will explain the mathematics necessary to understand the numerics employed in the code. I am conscious it might be redundant with [3], but it is still necessary in my opinion.

3.1 Convex Regularization

As discussed in introduction, solving (4) can be quite computationally expensive for large n and m. A common way to circumvent the difficulty is to introduce a regularizer, most often a negative entropy for example and using computationally-cheap method like Sinkhorn's algorithm [1]. An other common way is to find a dual problem, i.e a new problem extracted the primal while verifying the inequality : $(D) \leq (P)$. In this section, we will be combining both approach. Let's at first notice that adding a strongly convex function to the objective function make it automatically strongly convex. This is a precious property one needs to keep if we want to hope to remain simple. A general way to do it is displayed in (3). γ is called regularization parameter, and we of course have the property that $\lim_{\gamma\to 0} T_\gamma^\star = T^\star$. The fact that Ω is strongly convex assures us that the problem, not

only remain convex, but the whole objective function becomes actually strongly convex. A well-known result states that for a convex problem whose constraints are qualified (which is the case as the constraints are affine), strong duality holds. As a consequence, we have $(D) = (P)$. This property is key, since solving dual does not (only) provide a lower bound, but is actually equivalent to (P). Out of the three sets of constraints ($T \geq 0$, $T\mathbf{1}_n = \mathbf{a}, T^\top \mathbf{1}_m = \mathbf{b}$)), we are going to 'dualize' two of them, i.e we associate a Lagrange multiplier to each of them. We introduce the Lagrangian : $\mathcal{L}(T, \alpha, \beta) = (C, T) + \Omega(T) + (\alpha, T\mathbf{1}_n - a) + (\beta, T^\top \mathbf{1}_m - b)$. I noted here $\Omega(T) = \sum_{j=1}^m \Omega(T_{.,j})$. We state that $(P) \Leftrightarrow \inf_{T \geq 0} \sup_{\substack{\alpha \in \mathbf{R}^m; \\ \beta \in \mathbf{R}^n}} \mathcal{L}(T, \alpha, \beta) \overset{\text{strong duality}}{\Longleftrightarrow} \sup_{\substack{\alpha \in \mathbf{R}^m; \\ \beta \in \mathbf{R}^n}} \inf_{T \geq 0} \mathcal{L}(T, \alpha, \beta)$. By introducing the support function of the positive orthant $\delta(\mathbf{x}) = \sup_{y \geq 0} \mathbf{y}^\top \mathbf{x}$ and $\delta_\Omega(x) = \sup_{y \geq 0} \mathbf{y}^\top \mathbf{x} - \Omega(\mathbf{y})$, one can easily derive the following dual problem :

$$\sup_{\substack{\alpha \in \mathbf{R}^m; \\ \beta \in \mathbf{R}^n}} \alpha^\top a + \beta^\top b - \sum_{j=1}^n \delta_\Omega(\alpha + \beta_j - C_{.,j}) \tag{D}$$

An important property is that if Ω is $\gamma-$strongly convex, then δ_Ω is $\frac{1}{\gamma}-$smooth i.e its gradient is $\frac{1}{\gamma}-$Lipschitz. To prove it, it suffices to state that $\nabla \delta_\Omega(\mathbf{x}) = \mathbf{y}^\star$, where $\mathbf{y}^\star = \arg\max_{y \geq 0} \mathbf{y}^\top \mathbf{x} - \Omega(\mathbf{y})$, coming from the (non-trivial) result that in the differentiable case (this is the case here), one have $\nabla \delta_\Omega(\mathbf{x}) = \nabla_{\mathbf{x}}(\mathbf{y}^{\star\top} \mathbf{x} - \Omega(\mathbf{y}^\star))$ with the same notation. On a side note, we are assured that $\arg\max_{y \geq 0} \mathbf{y}^\top \mathbf{x} - \Omega(\mathbf{y})$ is a singleton because of strong convexity. By definition of $\gamma-$strong convexity, one has $||\nabla \Omega(\mathbf{y}^\star) - \nabla \Omega(\mathbf{z}^\star)|| \geq \gamma ||\mathbf{y}^\star - \mathbf{z}^\star||$. Since $\nabla_{\mathbf{y}}(\mathbf{y}^{\star\top} \mathbf{x} - \Omega(\mathbf{y}^\star)) = 0 \Leftrightarrow \nabla \Omega(\mathbf{y}^\star) = \mathbf{x}$ and dividing by γ, it follows that $||\nabla \delta_\Omega(\mathbf{x}) - \nabla \delta_\Omega(\mathbf{w})|| \leq \frac{1}{\gamma} ||\mathbf{x} - \mathbf{w}||$. So δ_Ω is smooth. Another very important statement comes from strong duality i.e $\alpha^{\star\top} a + \beta^{\star\top} b - \sum_{j=1}^n \delta_\Omega(\alpha^\star + \beta_j^\star - C_{.,j}) = (T^\star, C)$ Deriving by $C_{.,j}$, one has

$$T^\star_{.,j} = \nabla \delta_\Omega(\alpha^\star + \beta_j^\star - C_{.,j})$$

As a consequence, we can retrieve the optimal $T^\star$ by solving the dual, which turned out to be much faster in this case, as the problem is constraint-free. Of course, assuming we know how to compute δ_Ω efficiently, analytically in the best scenario. Let's compare with the usual dual of (P), which is

$$\begin{array}{lll} \sup\limits_{\substack{\alpha \in \mathbf{R}^m; \\ \beta \in \mathbf{R}^n}} & \alpha^\top a + \beta^\top b & \\ \text{s.t.} & \alpha_i + \beta_j \quad \leq \quad C_{i,j} & \forall (i,j) \in [\![1, m]\!] \times [\![1, n]\!] \end{array} \tag{5}$$

which can be re-written

$$\sup_{\substack{\alpha \in \mathbf{R}^m; \\ \beta \in \mathbf{R}^n}} \alpha^\top a + \beta^\top b - \sum_{j=1}^n \delta(\alpha + \beta_j - C_{.,j}) \tag{D'}$$

Therefore, it becomes evident here that regularizing by Ω is equivalent to replace δ by δ_Ω. Moreover, (D) is unconstrained compared to (D'), at the price of having a supplementary smooth term in the objective function. Of course, dealing with (D) remains much more interesting than (D'), as there is many solvers for an optimization problem with such suitable properties. To solve this problem, in accordance with [3], I used a BFGS-L solver (see section 3). An other less common way to proceed is by taking the semi-dual i.e dualizing only one constraint e.g $T\mathbf{1}_n = \mathbf{a}$. To do that, we introduce another transform of Ω, denoted as $\max(\mathbf{x}) \coloneqq \max_{j \in \{1 \cdots m\}} (x_j) = \sup_{\mathbf{y} \in \Delta^m} \mathbf{y}^\top \mathbf{x}$ and $\max_\Omega(\mathbf{x}) = \sup_{\mathbf{y} \in \Delta^m} \mathbf{y}^\top \mathbf{x} - \Omega(\mathbf{y})$. Actually, δ and max can be used as the equivalent of respectively the Legendre-Fenchel and c-transform for the derivations of the dual and semi-dual. On an other hand, δ_Ω and $\max_\Omega$ are considered as their regularized version. In this semi-dualization formalism, the lagrangian is $\mathcal{L}^\star(T, \alpha) = (C, T) + (\alpha, a - T\mathbf{1}_n)$. The semi-dual problem therefore derived as $\sup_{\alpha \in \mathbf{R}^m} \inf_{\substack{T \geq 0; \\ T^\top \mathbf{1}_m = b}} \mathcal{L}^\star(T, \alpha) = (C, T) + (\alpha, T\mathbf{1}_n -$

$a) \iff \sup_{\alpha \in \mathbf{R}^m} \alpha^\top a - \sup_{\substack{T \geq 0; \\ T^\top \mathbf{1}_m = b}} (C,T) - (\alpha, T\mathbf{1}_n)$. It can be shown (see Appendix B from [3]) that $\sup_{\substack{T \geq 0; \\ T^\top \mathbf{1}_m = b}} (C,T) - (\alpha, T\mathbf{1}_n) = \sum_{j=1}^n b_j \max_{\Omega_j}(\alpha - C_{\cdot,j})$, where $\Omega_j(\mathbf{x}) = \frac{1}{b_j}\Omega(b_j\mathbf{x})$. Eventually, we arrive at the smoothed semi-dual problem

$$\sup_{\alpha \in \mathbf{R}^m} \alpha^\top a - \sum_{j=1}^n \max_{\Omega_j}(\alpha - C_{\cdot,j}) \tag{SD}$$

Once again, strong duality holds because of convexity and the constraints are affine. (SD) is unconstrained, and solving it is much easier, given $\max_\Omega$ is cheap to compute. Moreover, the optimal transport can be retrieved from $\alpha^\star$ with the formula :

$$T^\star_{\cdot,j} = b_j \nabla \max_{\Omega_j}(\alpha^\star - C_{\cdot,j})$$

For the exact same reasons as before, $\nabla \max_\Omega(\mathbf{x}) = \mathbf{y}^\star$, where $\mathbf{y}^\star = \arg\max_{y \in \Delta^m}(\mathbf{y}^\top \mathbf{x} - \Omega(\mathbf{y}))$ and if Ω is $\gamma-$strongly convex, then $\max_\Omega$ is $\frac{1}{\gamma}-$smooth. This assures us suitable properties for numerical solvers. (SD) will also be solved via the BFGS-L algorithm. Now that everything is set up, it is necessary to treat the specific cases : $\Omega = \Omega_{\text{ent}}$ and $\Omega = \Omega_{norm}$.

3.1.1 The case $\Omega = \Omega_{\text{ent}}$

Let's recall that $\Omega_{\text{ent}}(\mathbf{y}) = \gamma \sum_i y_i \log(y_i)$ In this case, we retrieve the entropy regularization, and we of course won't expect any sparsity of the solution. Let's compute at first $\delta_{\Omega_{\text{ent}}}(\mathbf{x})$. Firstly, computing that the gradient w.r.t $\mathbf{y}$ of $\mathbf{y}^\top \mathbf{x} - \Omega(\mathbf{y})$ must be zero, one finds : $\nabla\Omega(\mathbf{y}^\star_\delta) = x \iff x_i = \gamma(\log(y^\star_{i,\delta}) + 1)$. The result is :

$$\mathbf{y}^\star_\delta = \exp\big(\frac{\mathbf{x}}{\gamma} - \mathbf{1}_m\big) = \nabla \delta_{\Omega_{\text{ent}}}(\mathbf{x})$$

We rapidly check that $\mathbf{y}^\star_\delta \geq 0$. Then, putting it back into $\delta_{\Omega_{\text{ent}}}$, we find $\delta_{\Omega_{\text{ent}}}(\mathbf{x}) = \mathbf{y}^{\star\top}_\delta \mathbf{x} - \gamma \sum_i y^\star_{i,\delta} \log(y^\star_{i,\delta}) = \gamma \sum_i y^\star_{i,\delta} = \gamma \sum_i \exp\big(\frac{x_i}{\gamma} - 1\big)$. To conclude,

$$\delta_{\Omega_{\text{ent}}}(\mathbf{x}) = \gamma \sum_i \exp\big(\frac{x_i}{\gamma} - 1\big)$$

Regarding $\max_{\Omega_j}$, we also set the derivative w.r.t to $\mathbf{y}$ equal to zero and we have the same result as before i.e $\mathbf{y}^\star_{\max} = \exp\big(\frac{\mathbf{x}}{\gamma} - \mathbf{1}_m\big)$. Except that there is no reason *a priori* for $\mathbf{y}^\star_{\max}$ to be in Δ^m. Since $\mathbf{y} \mapsto \mathbf{x}^\top \mathbf{y} - \Omega(\mathbf{y})$ is concave and Δ^m is convex, the supremum over Δ^m is simply the euclidean projection of $\mathbf{y}^\star_{\max}$ onto it. The optimal point is thus $\mathbf{y}^\star_{\max} = \text{Proj}_{\Delta^m}(\exp\big(\frac{\mathbf{x}}{\gamma} - \mathbf{1}_m\big))$. This projection can actually be done analytically. Indeed, this is simply a scaling issue because the direction does not need to be changed. As $\exp\big(\frac{\mathbf{x}}{\gamma} - \mathbf{1}_m\big) \geq 0$, we expect only a scalar variable $\alpha \geq 0$ would be sufficient to ensure it belongs to the m-simplex, that is $\mathbf{y}^\star_{\max} = \omega \exp\big(\frac{\mathbf{x}}{\gamma} - \mathbf{1}_m\big)$. Finding ω is easy, $\mathbf{y}^{\star\top}_{\max}\mathbf{1}_n = 1 \Rightarrow \omega = \frac{1}{\sum_{i=1}^n \exp\big(\frac{\mathbf{x}_i}{\gamma} - 1\big)}$. Eventually,

$$\mathbf{y}^\star_{\max} = \frac{\exp\big(\frac{\mathbf{x}}{\gamma} - \mathbf{1}_m\big)}{\sum_i \exp\big(\frac{\mathbf{x}_i}{\gamma} - 1\big)} = \frac{\exp\big(\frac{\mathbf{x}}{\gamma}\big)}{\sum_i \exp\big(\frac{\mathbf{x}_i}{\gamma}\big)} = \nabla \max_{\Omega_j}(\mathbf{x})$$

Plugging $\mathbf{y}^\star_{\max}$ back into the expression of $\max_{\Omega_j}$ and simplifying everything, one gets :

$$\max_{\Omega_j}(\mathbf{x}) = \gamma\big(\log\big(\sum_{i=1}^n \exp\big(\frac{x_i}{\gamma}\big)\big) - \log(b_j)\big)$$

We then have everything in hand to compute dual and semi dual analytically. A last technical detail related for the implementation is the beloved log-sum-exp trick, since it stabilises very well the numerical calculation of the exponential. It consists in writting : $\gamma \log \sum_i \exp \frac{x_i}{\gamma} = \gamma \log \sum_i \exp \frac{x_i - c}{\gamma} + \log(c)$ for $c > 0$. Usually, we take $c = \bar{x} = \sum_i \frac{x_i}{n}$. This trick is necessary in most practical case, especially when γ and x_i are not of the same order of magnitude. However, all the x_i's have to be of the same order of magnitude for it to work, this is the case in the majority of practical situation though.

3.1.2 The case $\Omega = \Omega_{norm}$

Ω_{norm} is defined as $\Omega_{norm} = \frac{\gamma}{2}||\cdot||_2^2$ and leads to different calculations. Let's first calculate $\nabla\delta_\Omega(\mathbf{x}) = \mathbf{y}^\star$. By setting the derivative to 0, on arrives at $\Omega(\mathbf{y}_\delta^\star) = \mathbf{x} \Leftrightarrow \mathbf{y}_\delta^\star = \frac{1}{\gamma}\mathbf{x}$. However, we are not ensured that $\mathbf{y}_\delta^\star > 0$ since $\mathbf{x} \in \mathbf{R}^n$. So, we need to actually take :

$$\mathbf{y}_\delta^\star = \frac{1}{\gamma}\text{Proj}_{\mathbf{R}_+^m}(\mathbf{x}) = \frac{1}{\gamma}[\mathbf{x}]_+$$

where $[\mathbf{x}]_+ = (\max(x_i, 0))_{i \in [\![1,n]\!]}$. This reasoning is valid because $\mathbf{y} \mapsto \mathbf{x}^\top\mathbf{y} - \Omega(\mathbf{y})$ is concave and $\mathbf{R}_+^m$ convex.

Therefore, since $\delta_\Omega(\mathbf{x}) = \mathbf{y}_\delta^{\star\top}\mathbf{x} - \Omega(\mathbf{y}_\delta^\star)$, we get immediately :

$$\delta_\Omega(\mathbf{x}) = \frac{1}{2\gamma}\sum_{i=1}^m [x_i]_+^2$$

Concerning $\max_{\Omega_j}(\mathbf{x})$, with the same method as before, one finds :

$$\mathbf{y}_{\max}^\star = \text{Proj}_{\Delta^m}\big(\frac{\mathbf{x}}{\gamma b_j}\big)$$

There is no analytical expression of $\mathbf{y}_{\max}^\star$, and so we can solely affirm that :

$$\max_{\Omega_j}(\mathbf{x}) = \mathbf{x}^\top\mathbf{y}_{\max}^\star - \Omega(\mathbf{y}_{\max}^\star)$$

Let's emphasize, the derivations of the expressions of $\mathbf{y}_{\max}^\star$ and $\mathbf{y}_\delta^\star$ were ambiguous. Setting the derivative to 0 to find an optimum is not correct for an optimisation problem subject to constraints. The right way to derive their expressions would have been to solve the KKT conditions or its dual problem, but I did not proceed as such for concision. However, the technique which consists to set the derivative to 0 and only after taking its projection works for convex problem. An interesting question is how to compute the projection on the m-simplex Proj_Δ^m. There is not one way to do it. Its existence $\forall x \in \mathbf{R}^n$ is assured because Δ^m is a closed convex set. I personally started by writing the definition of the projection :

$$\text{Proj}_{\Delta^m}(\mathbf{x}) = \arg\min_{y \in \Delta^m} \frac{1}{2}||\mathbf{y} - \mathbf{x}||_2^2$$

The problem being convex and differentiable, solving the dual is equivalent to solving the primal. If we dualize only the constraint $\mathbf{x}\top\mathbf{1}_n - 1 = 0$, one has the lagrangian : $\mathcal{L}(\mathbf{x}, \mu) = \frac{1}{2}||\mathbf{y} - \mathbf{x}||^2 - \mu^\top\mathbf{y}$, and the dual is computed as $g(\mu) = \inf_{\mathbf{y} \geq 0} \mathcal{L}(\mathbf{y}, \mu) = \inf_{\mathbf{y} \geq 0} \frac{1}{2}||\mathbf{y} - \mathbf{x}||^2 - \mu^\top\mathbf{y}$. If we derive by $\mathbf{y}$, set it to 0 and take its projection on the positive orthant, exactly like what was done for $\mathbf{y}_\delta^\star$, one finds that $\mathbf{y}^\star = (\mathbf{x} - \mu^\star)_+$. Let's emphasize that such a way to do it is valid because both the positive orthant and $\mathcal{L}$ are convex. We just need to know $\mu^\star$ to find the projection on Δ^m. The easiest way to derive $\mu^\star$ is to use the equality constraint condition : $\sum_i (x_i - \mu^\star)_+ - 1 = 0$. Let's note $h(\mu) = \sum_{i=1}^n (x_i - \mu)_+ - 1$, with $x_n < x_{n-1} < x_{n-2} < \cdots < x_1$. Since one have $h(x_1) = -1 < 0$, one wants to find an index i such that $h(x_i) < 0$ and $h(x_{i+1}) > 0$. This way, one knows by continuity that $\exists \mu^\star \in [x_{i+1}, x_i]$ such that

$h(\mu^\star) = 0$. Then, one have in this interval $h(\mu) = \sum_{j=1}^{i}(x_j - \mu) - 1 = \sum_{j=1}^{i} x_j - i\mu - 1$. setting it to at $\mu^\star$, 0, one have directly :

$$\mu^\star = \frac{\sum_{j=1}^{i} x_j - 1}{i}$$

This method worked pretty well and seemed rather efficient, the main difficulty being to sort the element of $\mathbf{x} \in \mathbf{R}^m$. Once it is done, finding the right index i can be done with a $\mathcal{O}(m)$ temporal complexity. Concerning the sorting, the best algorithms all have $\mathcal{O}(m\log(m))$ in worst-case scenario. On an other hand, some degenerate cases have to be taken into consideration (like for example if we already have $\mathbf{x}\Delta^m$), but for the sake of concision, I won't talk about it.

3.2 Convex Relaxation and Semi-relaxation

Another very employed technique is the penalisation i.e one removes the constraints, and add a function in the objective value to represent how violated is the constraint. In theory, an exact penalisation is for example :

$$\min_{T \in \mathbf{R}^{n\times m}} (T, C) + \mathbf{1}_{\mathcal{U}(\mathbf{a},\mathbf{b})}(T)$$

where $\mathbf{1}_{\mathcal{U}(\mathbf{a},\mathbf{b})}(T) = \begin{cases} +\infty & \text{if} \quad T \notin \mathcal{U}(\mathbf{a},\mathbf{b}) \\ 0 & \text{otherwise} \end{cases}$. In practice, such a problem is not easier to solve because it is not differentiable. Moreover, 'There is no free lunch', so if we abandon exact penalisation for an inexact one, gaining by the occasion differentiability, the solution of the penalised problem will not be a solution of the original problem. Yet, This is a common way to deal with constraints, despite this downside, if we know the right penalisation to use. The inexact penalisation will take the form in our case :

$$\min_{T \in \mathbf{R}_+^{m\times n}} (T, C) + \Phi(T\mathbf{1}_n, \mathbf{a}) + \Phi(T^\top \mathbf{1}_m, \mathbf{b}) \tag{ROT}$$

[3] proposes two classical function for affine constraints : $\Phi(\mathbf{x},\mathbf{y}) = \frac{1}{2\gamma}||\mathbf{x}-\mathbf{y}||_2^2$ and $\Phi(\mathbf{x},\mathbf{y}) = \frac{1}{\gamma}\mathrm{KL}(\mathbf{x}||\mathbf{y})$. A guarantee of the relevance of this method is that $\lim_{\gamma\to 0} T_\gamma^\star = T^\star$, or for a sub-sequence at least. This is a general property for penalised problems under very reasonable hypothesis, and is absolutely not specific to optimal transport[9]. Some theoretical bounds in [3] states how fast the convergence is.

The semi-relaxed was also considered, where we penalise only one constraint i.e

$$\min_{\substack{T \in \mathbf{R}_+^{m\times n}; \\ T^\top \mathbf{1}_m = \mathbf{b}}} (T, C) + \Phi(T\mathbf{1}_n, \mathbf{a}) \tag{SROT}$$

The relaxed problem (ROT) was solved with a L-BFGS-B method, that is a BFGS method adapted for box-like constraints. For (SROT), I decided to implement FISTA, a proximal method with a projection step on the closed convex set : $\mathcal{M}(\mathbf{b}) = \{T \in \mathbf{R}_+^{m\times n} | T^\top \mathbf{1}_m = \mathbf{b}\} = \{T \in \mathbf{R}_+^{m\times n} | \bigcap_{j=1}^{n} \{\sum_{i=1}^{m} T_{i,j} = b_j\}\}$. The point here being that one can re-write it cartesian product of scaled simplex $\mathcal{M}(\mathbf{b}) = b_1\Delta^m \times \cdots \times b_n\Delta^m$, which will turned out usefull for FISTA. $\Phi(\mathbf{x},\mathbf{y}) = \frac{1}{2\gamma}||\mathbf{x}-\mathbf{y}||_2^2$ being $\frac{1}{\gamma}-$smooth will be a key property to apply FISTA. This method was not compulsory, as I could also have used the L-BFGS-B method (which I did).

To sum up this section, I presented many alternatives formulations to the regular optimal transport. The basic approaches of regularizing, taking the dual or penalisation are all classical in optimization theory. In most cases, the implementation is rather direct from the mathematics formula, except a few technical difficulties which were previously briefly mentioned or explained. I also explained which solver I intended to use for each optimization problem, I intend to describe them a little bit more on

the next section. Whether it concerns regularized or relaxed problem, γ have to be tuned properly, to avoid numerical instabilities, or the problems can become very ill-conditioned for γ small. For example, Thikonov regularization has a criteria called Morozov's principle to choose it optimally.

4 Numerical applications

This section presents the numerical applications of the mathematics presented previously, where **a** and **b** represents the histograms associated with the reference and source image.

4.1 Visual results

Visually speaking, the best way to tell the difference between the methods is by using two images with radically different colours. $n = m = 50$ is more than enough. The change between the images is subtle, one can detect it on sharp changes of colour and nuance e.g around sky-cloud edges. Unfortunately, solving semi-relaxed with FISTA did not work very well, in particular for the sky. However, on the other images, we can without any problem say they all are very similar, and that the slight changes of details are hard to distinguish with our eyes.

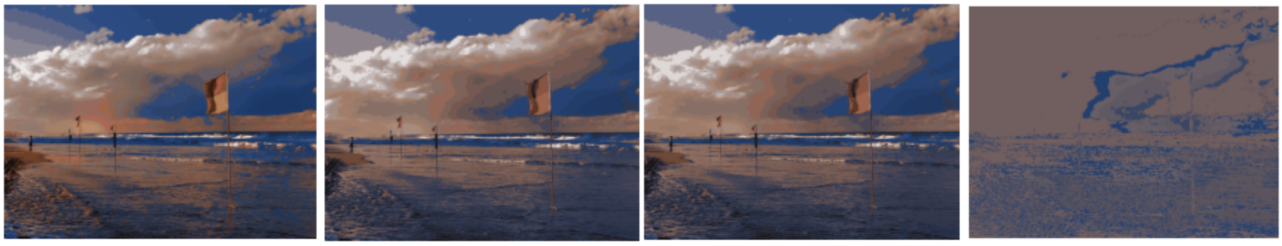
(a) Squared-norm regularization for dual problem (D)

(b) entropy regularization for dual problem (D)

(c) Squared-norm regularization for (SD)

(d) entropy regularization for (SD)

Figure 6: Visual comparisons of the squared-norm and entropy regulazition for the differents methods. They were all solved via L-BFGS algorithm. Parameters : $n = m = 50$ and $\gamma = 10^{-2}$.



(a) Entropy regularization for primal problem, numerically solved with Sinkhorn algorithm

(b) Solution to relaxed problem (ROT)

(c) Solution to semi-relaxed problem (SROT) with L-BFGS-B method

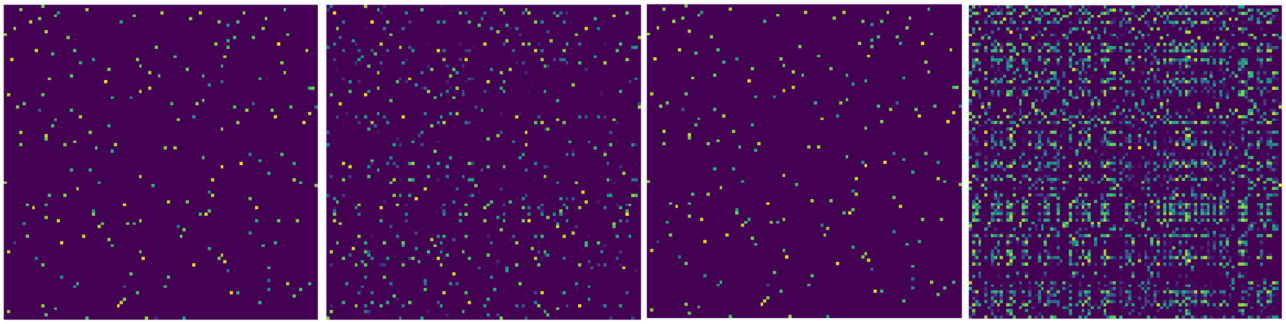
(d) Solution to semi-relaxed problem (SROT) with FISTA

Figure 7: Visual comparisons for relaxed and semi-relaxed problems. Parameters : $n = m = 30$ and $\gamma = 10^{-2}$.

4.2 Sparsity of Transportation plan

Despite what previous section could let us think, there is a fundamental difference between these Figure 6,7 and Figure 5, their transportation plan is sparse.

Also, I implemented Sinkhorn's algoithm to solve (P) (in the case $\Omega = \Omega_{\text{ent}}$ of course) to compare it with the resolution of (D) on Figure 6)a). Since it is its dual, so they are expected to be exactly equal because strong duality holds.



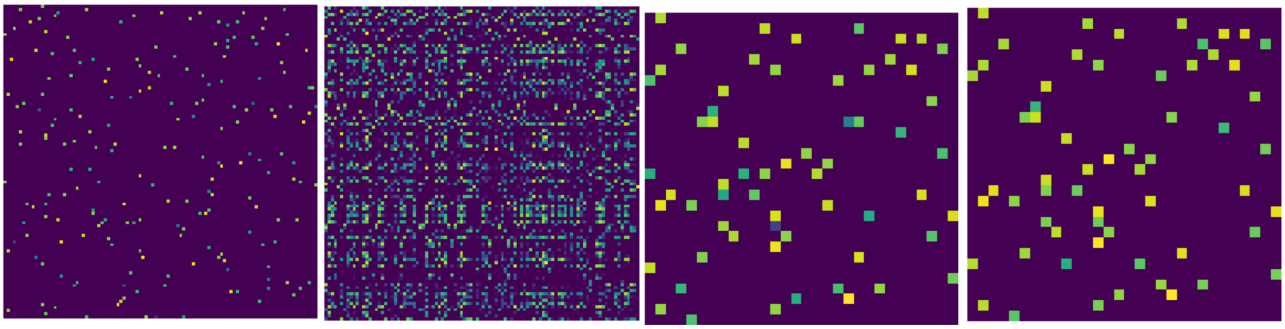
(a) Regular optimal transport. Sparsity = 0.98.

(b) Primal (P) with entropy regularization solved by Sinkhorn's algorithm. Sparsity = 0.

(c) Dual (D) with squared-norm regularization. Sparsity = 0.98

(d) Dual (D) with entropy regularization. Sparsity = 0.

Figure 8: Comparisons of the different transportation plans between regular and entropy/squared-norm regularization. Parameters : $n = m = 100$ and $\gamma = 10^{-2}$.



(a) Semi-dual (SD) with norm-squared regularization for $n = m = 100$. Sparsity = 0.

(b) Semi-dual (SD) with entropy regularization for $n = m = 100$. Sparsity = 0.98.

(c) (ROT) for $n = m = 30$. Sparsity = 0.18

(d) (SROT) for $n = m = 30$. Sparsity = 0.7

Figure 9: Comparisons of the different transportation plans between regular and entropy/squared-norm regularization. Parameters : $\gamma = 10^{-2}$.

Even relaxations conserves natural sparsity, I couldn't try on larger n and m, because of the time the calculations would take. Indeed, the L-BFGS-B method takes much more time, and even though FISTA would work much faster, the result was rather bad.

4.3 Description of the solvers employed

For this report, I wrote most of the codes, except k-means, and the solver of CVXPY for unregularized OT (which should apply Simplex algorithm or Interior Points method), and L-BFGS from scipy. The starting point of the algorithm might be important, but knowing the true transportation plan was of the order of unity, I started all my algorithms close to unity or at 0 or . I will not detail Sinkhorn's algorithm, but I will give a quick presentation of FISTA.

4.3.1 FISTA Algorithm

FISTA algorithm [10] is a proximal algorithm to solve an optimisation problem of the form :

$$\min_x F(x) + G(x)$$

where F is a proper convex smooth function and G a proper l.s.c convex (possibly non differentiable) function whose proximal operator is well-known. At first, one can understand the proximal algorithms

as being the generalisation of implicit gradient descent. The proximal operator is defined as such :

$$\mathcal{P}_G(x) = \arg\min_y G(y) + \frac{1}{2}||x-y||^2$$

The FISTA algorithm consists in the following. *1. Choose an initial condition* $u^{(0)} = v^{(0)} \in dom(G)$ *2. at step k, choose a step* $\epsilon^{(k)} > 0$ *Update the couple (u,v) by:*

$$u^{(k+1)} = \mathcal{P}_{\epsilon^{(k)}G}(v^{(k)} - \epsilon^{(k)}\nabla F(v^{(k)})); v^{(k+1)} = u^{(k+1)} + \frac{k-1}{k+2}(u^{(k+1)} - u^{(k)})$$

3. If not convergence, return to step 2 with k ← k + 1

In this case, one set $F(T) = \frac{1}{2\gamma}||T\mathbf{1}_n - a||^2$ which is $\frac{1}{\gamma}$-smooth. Therefore, we'll take $G(T) = (T, C)$, it satisfies the smoothness hypothesis. Its proximal operator is very simple to compute, it is

$$\mathcal{P}_G(X) = \arg\min_Y (Y, C) + \frac{1}{2}||X-Y||^2 = X - C$$

Concerning the step size, it is possible to show that for $\epsilon^{(k)} = \epsilon < \gamma$, convergence is assured. The main trick is to take into account the constraint $T^\top \mathbf{1}_n = b, T \geq 0$. For that I have decided to do something similar to the projected gradient method and so to project it onto $\mathcal{M}(\mathbf{b})$. This is relevant only because the problem is convex. So a generic step will finally look like :

$$u^{(k+1)} = \mathcal{P}_{\epsilon G}(v^{(k)} - \epsilon\nabla F(v^{(k)})) = v^{(k)} - \epsilon\nabla F(v^{(k)}) - \epsilon C$$

$$v^{(k+1)} = \text{proj}_{b_1\Delta^m \times \cdots \times b_n\Delta^m}(u^{(k+1)} + \frac{k-1}{k+2}(u^{(k+1)} - u^{(k)}))$$

For the projection, I use the exact same algorithm as before, but the m-simplex are now scaled by the elements of b. Let's precise the expression of $\nabla F \in \mathbf{R}^{m\times n}$. Component-wise, we have : $(\nabla F)_{i,j} = \frac{1}{\gamma}(\sum_k T_{i,k} - a_i)$, so the value is the same for all column. On a side note, we can state there is actually a link between proximal and Sinkhorn's algorithms, see [6] p.67. Unfortunately, this method did not work very well, as we can see on Figure 7)d). I did not manage to determine why, the algorithm converges though. We can of course notice that in this linear case, FISTA algorithm is completely equivalent to a projected gradient descent. The main pro of this algorithm is supposed to be its speed of convergence.

4.3.2 BFGS method

The Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [9] is part of the family of Quasi-Newton methods, i.e iterative methods where one uses approximations of the successive derivatives of interest to compute the descent direction. This is particularly interesting in the following cases : the hessian is not known or too computationally expensive, the function is not twice differentiable or for spatial complexity reasons (there is $\mathcal{O}(n^2)$ elements to store)... Therefore, the hessian is approximated by a symmetric matrix defined by the fundamental Quasi-Newton condition :

$$\nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k) = M_{k+1}(\mathbf{x}_{k+1} - \mathbf{x}_k)$$

This equation does not uniquely determines M_k, there is still some degrees of freedom. The main idea developed by BFGS was to find a formula between M_{k+1} from M_k by solve the following optimization problem :

$$\begin{cases} \min & \psi(M_k^{-1/2} M M_k^{-1/2}) \\ s.t & y_k = M s_k \\ & M \in \mathcal{S}_{++}^n \end{cases} \tag{6}$$

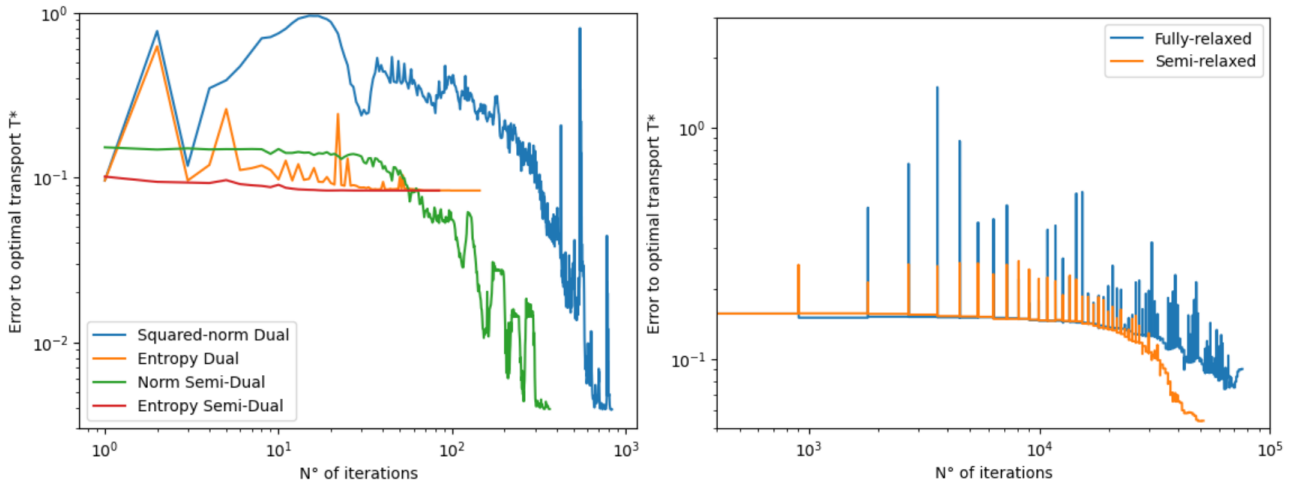
where $y_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$ and $s_k = \mathbf{x}_{k+1} - \mathbf{x}_k$, and $\psi(M) = \text{Tr(M)} + \text{ld}(M)$, ld being the log-determinant. Solving this problem lead to the famous BFGS recurrence formula :

$$M_{k+1} = M_k + \frac{y_k y_k^\top}{y_k^\top s_k} - \frac{M_k s_k s_k^\top M_k}{s_k^\top M_k s_k}$$

Once we have M_{k+1}, we use the usual second-order Newton step : $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha M_{k+1} \nabla f(\mathbf{x}_k)$, where α is possibly determined by backtracking. The implemented BFGS is certainly much more complicated but I gave here the main idea. The L-BFGS method is an extension more adapted to problem with a lot of variables.

4.4 Speed of convergence

Speed of convergence is also key when choosing an algorithm or an approached formulation of our problem. It is often related to the well conditioned the original problem is, so is a good indicator whether our problem is well-posed or not. The (semi-)relaxed problems were undoubtedly the longer to solve, it is easy to understand as they are constrained. Each of them would take around 15 minutes. The influence of γ is important, whether it is regarding its speed of convergence or precision. Moreover, it was difficult to go above $n = m = 30$, as the solver would take much more time to converge. Accordingly to what was said earlier, it has to be tuned with care. This is the main difference I noticed with [3], I could not choose γ as high as 10^4 with numerical instabilities. This oscillatory effect observed on both figure can be quite hard understand, it seems quite heretical, a little bit like iterations in stochastic optimization. Also, entropy-based regularization on Figure 10)a) converges faster, but are much less precise, because entropy solution can not imitate the natural sparsity of the optimal transport $T^\star$. Eventually, regarding Figure 10)b), something rather unexpected is noted. The semi-relaxed problem, which has m more constraints converges faster and with more precision.

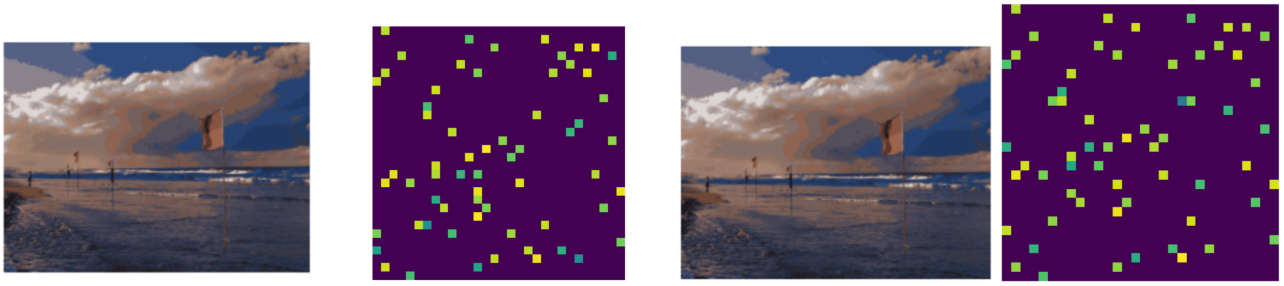


(a) $n = m = 50$ (b) $n = m = 30$

Figure 10: Speed of convergence towards $T^\star$. In log-log scale and with $\gamma = 10^{-2}$

4.5 A few more results

Until now, the relaxed formulation was presented with Kullback-Leibler penalisation $\Phi = \frac{1}{2\gamma}||\mathbf{x} - \mathbf{y}||^2$. It can be interesting what differences make $\Phi(\mathbf{x}, \mathbf{y}) = \frac{1}{\gamma}\mathrm{KL}(\mathbf{x}||\mathbf{y})$. The relaxed formulation on Figure 11)a) gives once more less good result that on Figure 11)b), despite being less constrained.



(a) Colour transfer for relaxed formulation with KL penalisation

(b) Transportation plan for Figure 11)a). Sparsity = 0.72

(c) Colour transfer for semi-relaxed formulation with KL penalisation

(d) Transportation plan of Figure 11)c). Sparsity : 0.61

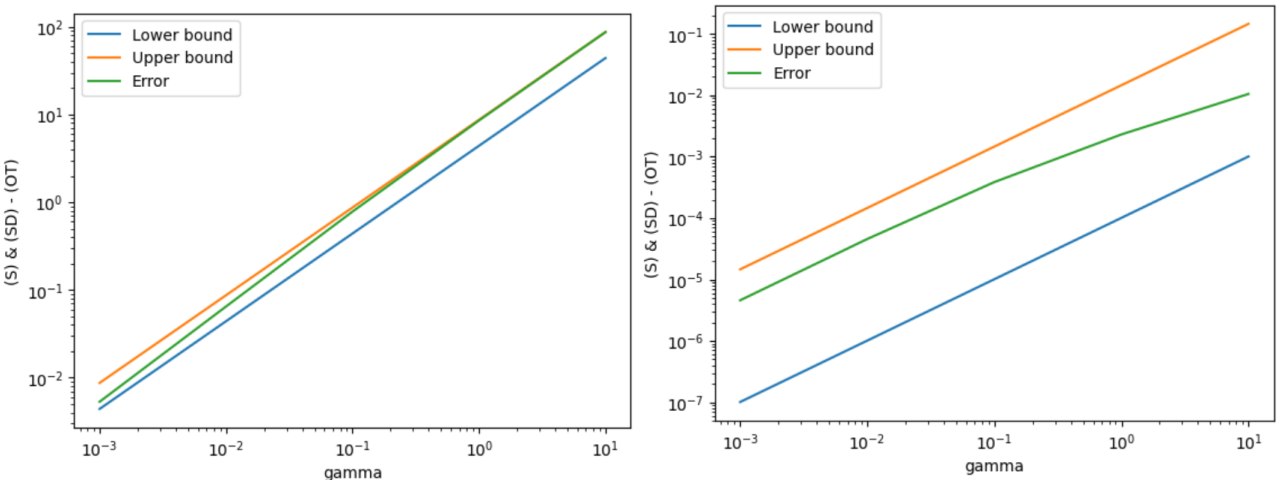
Figure 11: One observes KL penalisation makes it less sparse others formulation except for Entropy regularizer.

The last parameter important for comparison is γ. Indeed, it is the parameter which sets up the strength of the penalisation or regularization. Some theoretical bounds are given by [3] :

$$L\gamma \leq \mathrm{OT}_\Omega(\mathbf{a}, \mathbf{b}) - \mathrm{OT}(\mathbf{a}, \mathbf{b}) \leq U\gamma$$

where L and U are constants depending on the regularization, Figure 12. Moreover, one also have for respectively relaxed and semi-relaxed formulations

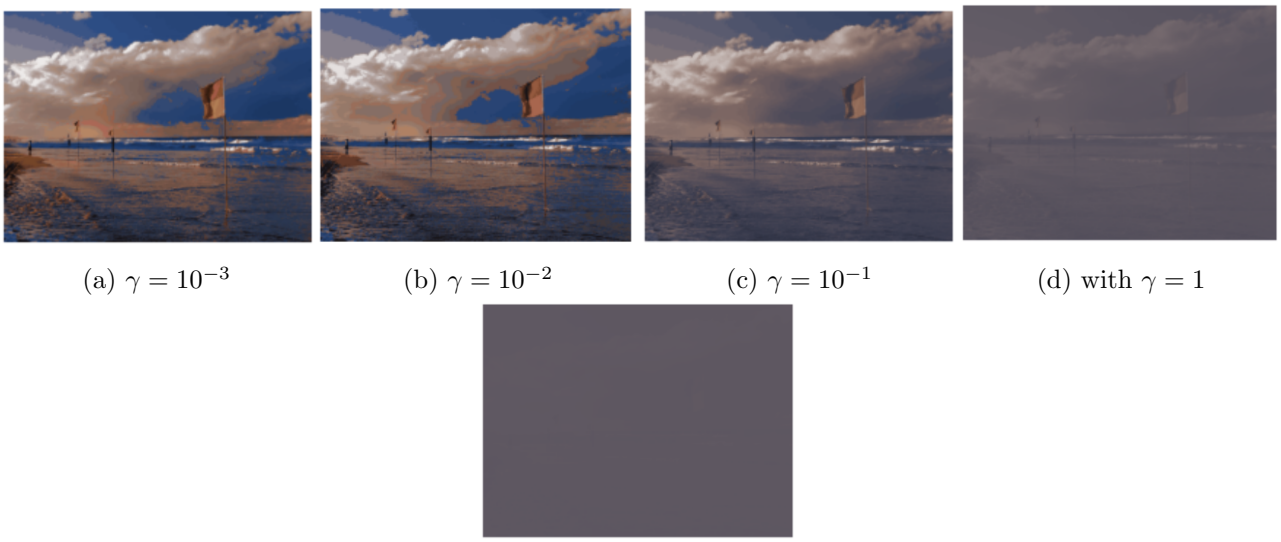
$$0 \leq \mathrm{ROT}_\Omega(\mathbf{a}, \mathbf{b}) - \mathrm{OT}(\mathbf{a}, \mathbf{b}) \leq U_{\text{rel}}\gamma$$
$$0 \leq \mathrm{SROT}_\Omega(\mathbf{a}, \mathbf{b}) - \mathrm{OT}(\mathbf{a}, \mathbf{b}) \leq U_{\text{s-rel}}\gamma$$



(a) $||OT_{\Omega_{\text{ent}}}(\mathbf{a}, \mathbf{b}) - OT(\mathbf{a}, \mathbf{b})||$ for entropy regularization (b) $||OT_{\Omega_{\text{norm}}}(\mathbf{a}, \mathbf{b}) - OT(\mathbf{a}, \mathbf{b})||$ for squared-norm regularization

Figure 12: Error of the objective function for entropy and squared-norm regularizer.

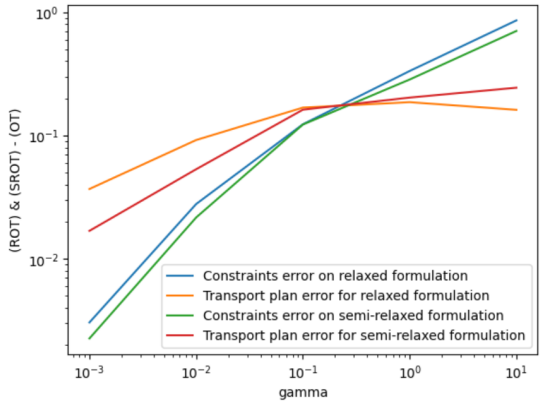
The bounds are respected, but entropy regularization seems to put more constraints and is more sensitive to the variations of γ.



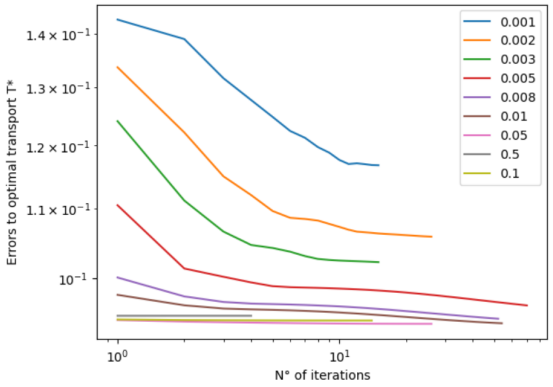
(a) $\gamma = 10^{-3}$ (b) $\gamma = 10^{-2}$ (c) $\gamma = 10^{-1}$ (d) with $\gamma = 1$

(e) $\gamma = 10^{1}$

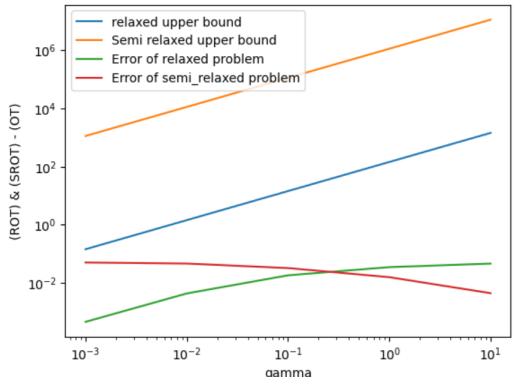
Figure 13: Entropy (D) formulation. Effect of the variation of γ on the visual aspect.



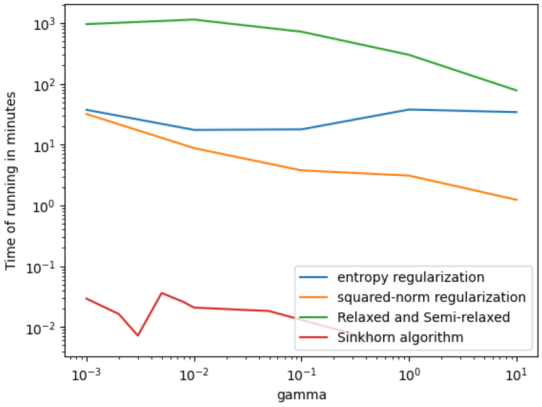
(a) Constraints errors : $(||T\mathbf{1}_n - \mathbf{a}||^2 + ||T^\top \mathbf{1}_m - \mathbf{b}||^2 / (||\mathbf{a}||^2 + ||\mathbf{b}||^2))^{1/2}$. Transport plan error : $||T^\star - T||$



(b) Error w.r.t true transport plan for Sinkhorn algorithm : $||T - T^\star||/||T^\star||$. The legend indicates the value of γ. One observes it quickly reaches a plateau but have in general higher errors.



(c) Objective function error for (SROT) and (ROT) : $||ROT_\Phi - OT||$, $||SROT_\Phi - OT||$ with $\Phi = \frac{1}{2}||\mathbf{x} - \mathbf{y}||^2$. The bounds are respected.



(d) Running times w.r.t γ in log-log scale

Figure 14

5 Conclusion

In conclusion, I had the opportunity to use many alternatives formulations of Optimal Transport in two contexts, with a penalizer and a regularizer. The regularization approach turned out to be a better option in my opinion, because takes relatively little time to run, a few minutes maximum, and showcase good results. The most interesting case is undoubtedly the squared-norm regularizer, as it is the most precise while keeping sparsity. Penalisation can works well for small γ, but is very long to run with L-BFGS, at least 10 minutes for a similar result. Moreover, the theoretical bounds are better for regularization than penalisation. We can first conclude that penalisation is not a very good idea to relaxed the problem. In general, the natural sparsity of $T^\star$ is well conserved, except of course in the entropy case. Solving the dual (D) and semi-dual (SD) is much easier than in the classical approach, since we do not have the usual constraints over f and g : $\{f \in \mathbf{R}^n, g \in \mathbf{R}^m | f_i + g_j \leq C_{i,j}\}$. The difficulty is passed on the computation of δ_Ω and $\max_\Omega$; which was possible for entropy and squared-norm, but not for LASSO Group [3]. In this case, the computation of these generalized dual functions might be more expansive than solving the classical duals with constraints. Using squared-norm instead of the classical entropy approach works very well in the sense that the transport matrix remains very sparse, so this objective is attained. Even for penalisation with a KL divergence, sparsity remains above 60%. The main improvement I could suggest would be to actually test the LASSO-Group or other kind of regularizer (maybe another, e.g 'stronger', norm). One point suggested by [3] is to use FISTA to solve (SROT) on the scaled simplex, however I couldn't have it work. It could be possible to extend this work by explicitely seeing the time of running for earch formulation, and not only its capacity to convgerge. Moreover, my errors seem rather sensitive to γ, it is clearly not relevant to take any value over 0.1. A key difference between [3] and my work is that I couldn't tune γ up to 10^3 because otherwise, it wouldn't converge anymore. Another important feature which would has been interesting to look at is the role of γ in the speed of convergence. Indeed, we see that in general, increasing γ leads to a faster convergence, or has no excessive effect. Another parameter which would have been interesting to play on is the number of cluster, i.e m and n. However, in the context of colour transfer, this is not really relevant to do so, because in practice, 10 clusters is enough.

5.1 Connection with the course

There is several direct connections with the course. The most obvious one is the treatment of optimal transport in discrete case and the use of Sinkhorn algorithm. The other one is surely about approachind the originial Kantorovich optimal transport by different technique. The more common si by taking its dual and introducing the c-transform to remove the explicit constraint from the problem. Here, the approach developed was a little bit different, as one introduced a regularizer into the objective function and take its dual and semi-dual. The regularizer employed is more general, and so, the dual and semi-dual (semi-dual being equivalent to the formulation with f and f^c as set of variables while dual has f and g) is similar but smoother. However, the relaxed formulation was new in my opinion and was not seen in the course.

References

[1] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transportation distances. *Stat-ML*, 2013.

[2] Marc Bonnet. *Methodes numeriques matricielles avancees: analyse et experimentation*. ENSTA Paris.

[3] Matthieu Blondel Vivien Seguy Antoine Rolet. Smooth and sparse optimal transport. *Stat-ML*, 2013.

[4] Optimal transportation for example-guided color transfer. URL `https://oriel.github.io/color_transfer.html`.

[5] Cedric Villani. *Optimal Transport: Old and New*. 2020.

[6] Gabriel Peyre Marco Cuturi. *Computational Optimal Transport*. 2020.

[7] Arnaud Dessein Nicolas Papadakis Jean-Luc Rouas. Regularized optimal transport and the rot movers distance. *Stat.ML*, 2018.

[8] Devis Tuia Nicolas Courty, Rmi Flamary. Optimal transport for domain adaptation. *IEEE*, 2016.

[9] Jean-Charles Gilbert. *Optimisation differentiable et algorithmes*. ENSTA Paris, 2020.

[10] Amir Beck Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM*, 2009.