Optimal Transport in Machine Learning

Yoann Coudert–Osmont Nemo Fournier

Report for the Machine Learning project
M1 IF 2018 – 2019

Table des matières

Chapter A

Overview of Optimal Transport

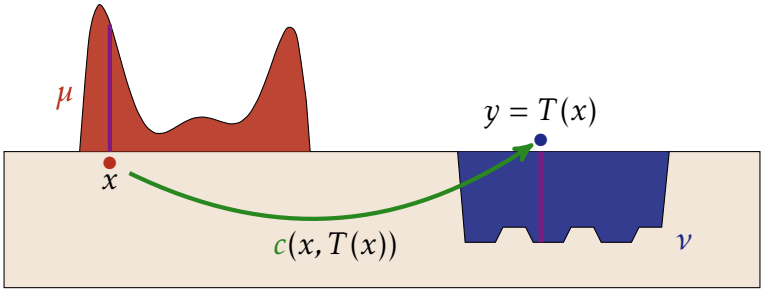
Table des matières

Optimal Transport theory aims to define a geometry along with a notion of distance on the set of probability measures over a space.

I - Introduction

1/ Monge's Problem

The history of optimal transport formulations can be traced back to Gaspard Monge's *Mémoire sur la théorie des déblais et des remblais (1781)*, studying how to move masses of sand or soil from one place to another. The main motivation of this *Mémoire* being that we can assign a cost to the act of moving a massic particle from one place to another, and it is of particular interest to seek for the most optimal way of moving a global quantity of mass from one place to another.



Formalism of Monge's problem We work in a probability space Ω. We consider two probability measures μ and ν on $\mathcal{P}(\Omega)$. We also consider a cost function $c : \Omega \times \Omega \to \mathbf{R}$. This function can for instance be the Euclidean distance. Monge's problem is therefore the following

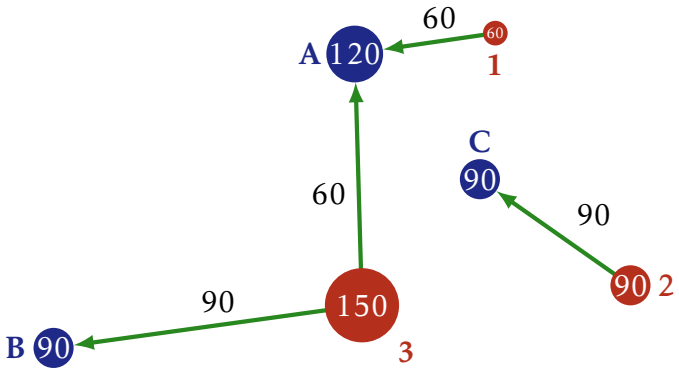
$$\min_{T:\Omega\to\Omega} \int_{\Omega} c(x, T(x))\, \mu(x)$$
$$\text{s.t.} \quad \forall A \in \mathcal{P}(\Omega),\ \nu(A) = \mu\left(T^{-1}(A)\right)$$

The function T is called a transport map. It carries the information regarding the displacement of each elementary unit of probability: the mass located in x is moved toward $T(x)$. The function c thus allows to quantify the cost of moving one elementary unit of mass. We want to minimize the average distance travelled by mass units during the optimal transport. The condition under which the optimization is performed ensures that all the mass of μ has been transported toward the distribution ν.

Limitations The issue with such a formulation of an optimal transport problem is that it does not handle discrete probability measures. Indeed, if μ has a Dirac impulse bigger than every Dirac impulse of ν, then the transport map cannot be build: we are only performing displacements of mass, and not actual repartition of mass.

2/ Kantorovich's problem

Kantorovich came up with an alternative formulation for the optimal transport problem, aiming to handle discrete distributions.



Change of transport plan The idea of Kantorovich is to change the function $T : \Omega \to \Omega$ into a probability P on the product space $\mathcal{P}(\Omega \times \Omega)$. On the above example, Monge's formulation forbids to split the masse of size 150 into two separate masses. With Kantorovich's formulation, this operation becomes accessible: the mass of 150 is splitted into a mass of 90 and a mass of 60, both sent to different places. In the discrete case, we can represent what happens through a matrix.

Transport Map

150	p_{1A}	p_{1B}	p_{1C}
90	p_{2A}	p_{2B}	p_{2C}
60	p_{3A}	p_{3B}	p_{3C}
	120	90	90

Distance matrix

d_{1A}	d_{1B}	d_{1C}
d_{2A}	d_{2B}	d_{2C}
d_{3A}	d_{3B}	d_{3C}

Discrete Case If we denote D the distance matrix of the problem, we obtain the following formulation in the discrete case:

$$\min_P \langle P, D \rangle = \sum_i \sum_j p_{ij} d_{ij}$$
$$\text{s.t.} \quad \begin{cases} \forall i & \sum_j p_{ij} = \mu_i \\ \forall j & \sum_i p_{ij} = \nu_j \end{cases}$$

The two conditions encode the fact that we aim to move all the mass of μ toward all the mass of ν.

Continuous Case In the continuous setting, we introduce the following set of joint probabilities, which encodes the analog of the two conditions in the discrete case, making sure that μ has been fully transported to ν.

$$\Pi(\mu, \nu) = \{P \in \mathcal{P}(\Omega \times \Omega) \mid \forall A, B \in \Omega,\, P(A \times \Omega) = \mu(A),\, P(\Omega \times B) = \nu(B)\}$$

We can rephrase the definition of $\Pi(\mu, \nu)$ as being the set of probability measures on $\Omega \times \Omega$) with marginals μ and ν.

We consider once again a cost function c defined as in the Monge's problem. Kantorovich's formulation now becomes the following optimization problem, where $X \sim \mu$ and $Y \sim \nu$:

$$\inf_{P \in \Pi(\mu,\nu)} \mathbb{E}_P [c(X, Y)] = \int \int c(x, y) P(dx, dy)$$

Existence of solutions One can wonder whether solutions of Kantorovich's problem as formulated above exist. A rich litterature exists on this topic. For instance, in the case of measures on the $\mathbf{R}^n$ space, with absolutely continuous measures μ and ν with respect to the Lebesgue measure, the existence (and uniqueness) of solutions will then depend on the properties of the cost function. If we consider the cost function $c_p(x,y) = |x-y|^p$, then we will have the following discussion:

- $p > 1$: the strict convexity of c_p ensures that there is a unique solution to the Kantorovich problem
- $p = 1$: here we can obtain the existence of a minimizer for the Kantorovich, but we won't have the unicity of the solution
- $p < 1$: in general, there will not exist solutions to the problem

More generally the existence of minimizers, as well as their uniqueness depends heavily on the structure of the space we consider, and the properties of both the measures and the cost function. A comprehensive study of those properties can be found in [Vil03].

3/ Wasserstein distance

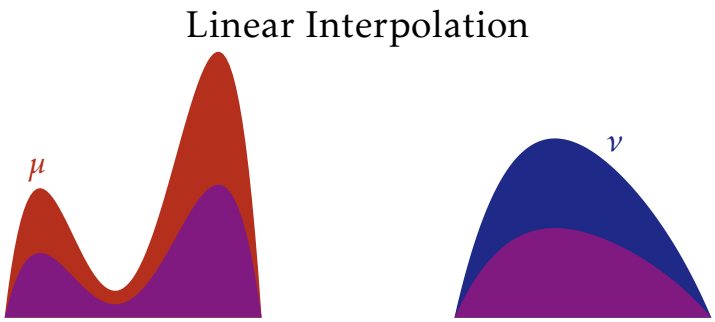
Having formulated this optimization problem allows us to define a distance between probability measures.

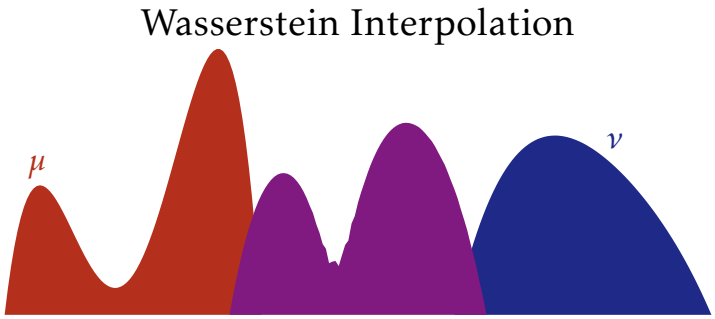
Definition

Given a probability space Ω, a cost function $c : \Omega \times \Omega \to \mathbf{R}$ and two probabilitiy mesures μ and ν in $\mathcal{P}(\Omega)$, the **p-Wasserstein-distance** between μ and ν is defined as:

$$W_p(\mu,\nu) = \left(\inf_{P \in \Pi(\mu,\nu)} \int\int c(x,y)^p P(dx,dy) \right)^{1/p}$$

This notion of distance allows to compute barycenters between several probability distributions, and obtain results that are way more natural than barycenters obtained with other norms, like the l^2 norm for instance. The following figure compares an interpolation (*i.e.* a progressive weighted barycenter of two distributions) obtained under Wasserstein-distance and a linear interpolation (the two target distributions being the blue and the red one, and the purple shape being an intermediate interpolation).





What emerges from those two interpolations is that the Wasserstein distance seems particularily well suited to manipulate distributions, as it appears to take into account the specificities (like modes, shape, etc) of both the source and the target distribution, and the resulting barycenter looks like a distribution combining those into one distribution, while the l^2 barycenter does not merge those specificities into one distribution but rather renormalize the source and target distributions.

We can extend this idea to the computation of the barycenter of an arbitrary number of distributions. Formally, the computing the barycenter μ of N distributions ν_i weighted by the λ_i boils down to the following minimization problem:

$$\min_{\mu \in \mathcal{P}(\Omega)} \sum_{i=1}^{N} \lambda_i W_p^p(\mu, \nu_i)$$

Now we can observe what the barycenter looks like with 4 distributions (seen as images) on $\mathbf{R}^2$ in the Figure A.1.

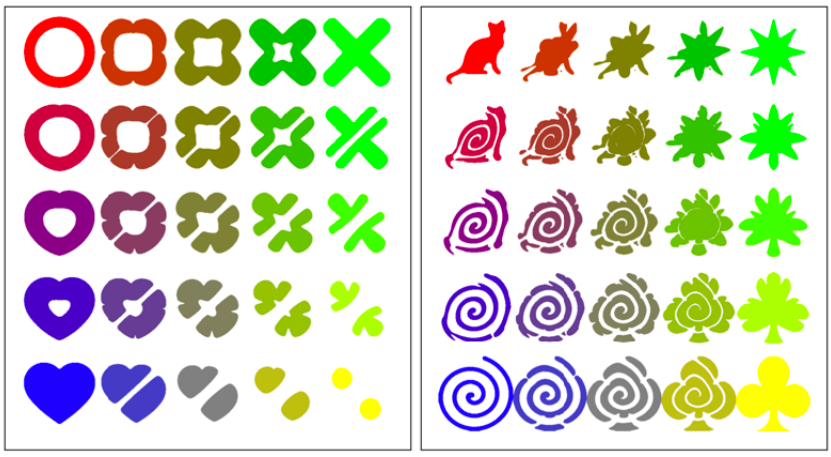


Figure A.1: Each image contains several barycenters of the four shapes that are in the corners

II - How to compute Optimal Transport?

Now that we have a convenient formulation of the optimal transport problem, we can study how to effectively compute minimizers of the optimal transport problem. For this we will introduce several new formulation, more adapted to effective solving. Sometimes, it is worth noting that we are only interested in computing the Wasserstein distance between two distributions, rather than the actual transport map.

1/ Duality

To effectively solve the optimal transport problem, we embrace the setting of discrete spaces. In this setting, probability measures can be seen as a sum of diracs impulses.

$$\mu = \sum_{i=1}^{n} a_i \delta_{x_i} \qquad \nu = \sum_{j=1}^{m} b_j \delta_{y_j}$$

The cost function is a matrix $D \in \mathbf{R}_+^{n\times m}$ such that $D_{ij} = d(x_i, y_j)^p$. Our joint distribution P is also a matrix of $\mathbf{R}_+^{n\times m}$ that has to satisfy several constraints that can be expressed matricially: if we denote by a and b the vector of the coefficients of the Dirac decompositions of, the set which P has to belong to is:

$$U(a,b) = \left\{ P \in \mathbf{R}_+^{n\times m} \mid P\mathbf{1}_m = a,\, P^\top \mathbf{1}_n = b \right\}$$

Definition

The Wasserstein distance in a discrete space is defined as:

$$W_p^p(\mu,\nu) = \min_{P\in U(a,b)} \langle P, D\rangle$$

Dual We now focus on the dual form of this optimization problem. For this we introduce the Lagrangian, using the Lagrange multipliers α and β

$$L(P,\alpha,\beta) = \langle P, D\rangle + \alpha^\top(a - P\mathbf{1}_m) + \beta^\top(b - P^\top \mathbf{1}_n)$$

As usual in a Lagrangian formulation, the objective function of the dual problem is:

$$g(\alpha,\beta) = \min_{P\geqslant 0} L(P,\alpha,\beta) = \min_{P\geqslant 0} \alpha^\top a + \beta^\top b + \langle P, D - \alpha\mathbf{1}_m^\top - \mathbf{1}_n\beta^\top\rangle$$

If the matrix $D - \alpha\mathbf{1}_m^\top - \mathbf{1}_n\beta^\top$ has a negative coefficient, then by having the corresponding coefficient of P grow toward $+\infty$ we obtain a minimum of $-\infty$. But if all the coefficients are non negative, then $\langle P, D - \alpha\mathbf{1}_m^\top - \mathbf{1}_n\beta^\top\rangle$ is non negative and we just have to take $P = 0$ to obtain a non zero value. We therefore obtain:

$$g(\alpha,\beta) = \begin{cases} \alpha^\mathsf{T} a + \beta^\mathsf{T} b & \text{si } D - \alpha \mathbf{1}_m^\mathsf{T} - \mathbf{1}_n \beta^\mathsf{T} \geqslant 0 \\ -\infty & \text{sinon} \end{cases}$$

Finally, the dual problem is the maximization of this function g.

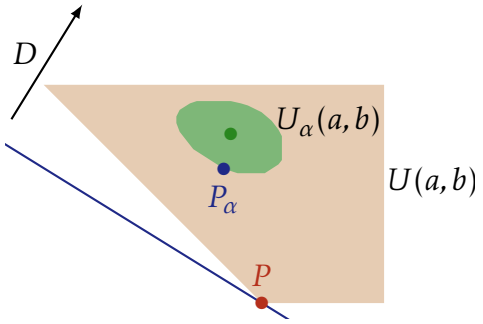
Definition

The **dual** of the optimal transport problem is:

$$W_p^p(\mu,\nu) = \max_{\substack{\alpha\in\mathbf{R}^n,\beta\in\mathbf{R}^m \\ \alpha_i+\beta_j\leqslant D(x_i,y_j)^p}} \alpha^\mathsf{T} a + \beta^\mathsf{T} b$$

We have reduced our initial problem into a linear programming formulation. Using a minimum flow solver, we can solve this linear optimization problem in $\mathcal{O}(n^3 \log(n))$. But the solution is not stable, as illustrated on the following figure. If we slightly modify our distance D or if we change the source and destination measures, the value of P can drastically change, since P is a vertex of our simplex. One solution is to regularize the problem to fall back on optimizing in a set $U_\alpha(a,b)$ strictly convex. This set is strictly contained in $U(a,b)$ and the optimal solution of this new problem will not necessarily be optimal for the initial problem, but will remain valid.

We will introduce in the next section a regularization of the original problem, that will us to exhibit an iterative algorithm that converges toward a meaningful approximation of the minimizer, as well as mainly relying on matrix-product kind of linear algebrea and thus potentially implemented on GPUs.



2/ Entropic Regularization

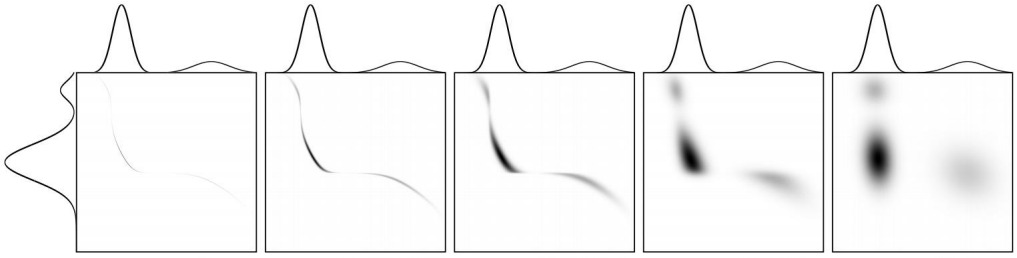
The regularization that we highlight consists in solving the optimal transport problem, while maximizing the entropy of the joint-probability P. We therefore exhibit a new parameter γ that parametrizes the regularization and we obtain the following distance.

Definition

The **regularized Wasserstein distance** is defined as follows:

$$W_\gamma(\mu,\nu) = \min_{P\in U(a,b)} \langle P, D\rangle - \gamma H(P)$$

Remark Recall that we can consider two random variables X and Y such that $X \sim \mu$, $Y \sim \nu$ and consider their joint distribution P such that $(X, Y) \sim P$. The entropy of P is maximal when X and Y are independant, *i.e.* when P factorizes as $P = ab^\top$ and this maximum entropy is given by $H(\mu)+H(\nu)$. The regularization therefore produces a matrix whose positive values are more distributed. In other words, we favor randomness of the transport map against determinism. The intuition behind this regularization comes from the observation that actual trafic patterns in networks are sparser than the theoretical solution obtained through a minimizer of the Kantorovich problem.



By a strict convexity argument, there exist a unique matrix P_γ minimizing the distance:

$$P_\gamma = \underset{P \in U(a,b)}{\arg\min} \langle P, D \rangle - \gamma H(P)$$

Proposition 1

There exist a unique couple of vectors u and v belonging to $\mathbf{R}_+^n$ and $\mathbf{R}_+^m$ such that:

$$P_\gamma = \mathrm{diag}(u) K \mathrm{diag}(v), \qquad K = e^{-D/\gamma}$$

Demonstration We write the Laplacian of our new optimization problem.

$$L(P, \alpha, \beta) = \sum_{ij} \Big(P_{ij} D_{ij} + \gamma P_{ij} \big(\log_2 P_{ij} - 1 \big) \Big) + \alpha^\top (a - P\mathbf{1}_m) + \beta^\top (b - P^\top \mathbf{1}_n)$$

We compute the partial derivative with respect to each P_{ij}

$$\frac{\partial L}{\partial P_{ij}} = D_{ij} + \gamma \log_2 P_{ij} - \alpha_i - \beta_j$$

The dual problem is the maximization of the minimum of the Laplacian with respect to P. We must look for P such that the partial derivative is equal to zero:

$$\frac{\partial L}{\partial P_{ij}} = 0 \Rightarrow P_{ij} = e^{\frac{\alpha_i}{\gamma}} e^{-\frac{D_{ij}}{\gamma}} e^{\frac{\beta_j}{\gamma}} = u_i K_{ij} v_j$$

■

Sinkhorn We can derive an iterative method to find this matrix P_γ. We use the condition $P_\gamma \in U(a,b)$:

$$P_\gamma \in U(a,b) \Leftrightarrow \left\{ \begin{array}{rcl} \mathrm{diag}(u)K\mathrm{diag}(v)\mathbb{1}_m & = & a \\ \mathrm{diag}(v)K^\top \mathrm{diag}(u)\mathbb{1}_n & = & b \end{array} \right.$$

$$P_\gamma \in U(a,b) \Leftrightarrow \left\{ \begin{array}{rcl} \mathrm{diag}(u)Kv & = & a \\ \mathrm{diag}(v)K^\top u & = & b \end{array} \right.$$

Then, by introducing $\odot$ the coefficient-wise product, we can rewrite this as

$$P_\gamma \in U(a,b) \Leftrightarrow \left\{ \begin{array}{rcl} u \odot Kv & = & a \\ v \odot K^\top u & = & b \end{array} \right.$$

We end up considering a problem which was already studied in the numerical analysis community, and known as the *matrix scaling problem*. From this formulation, rewritten as below using the coefficient-wise division $\oslash$, we can obtain an iterative algorithm performing Sinkhorn's updates.

$$P_\gamma \in U(a,b) \Leftrightarrow \left\{ \begin{array}{rcl} u & = & a \oslash Kv \\ v & = & b \oslash K^\top u \end{array} \right.$$

First we modify u such that it satisfies the first equation, and we then use this value to modify the value of v in order for it to satisfy the second. And we keep performing those updates until convergence. Sinkhorn's algorithm is thus the following:

Algorithm 1: Sinkhorn

repeat
  $u \leftarrow a \oslash Kv$;
  $v \leftarrow b \oslash K^\top u$;
until *convergence of* u, v;

Complexity It has been proved that the algorithm is a linear convergence scheme. Moreover, an iteration is naively done in $\mathcal{O}(mn)$ but the computation can easily be parallelized, and especially performed on GPUs, the bottleneck of one update being the two matrix products Kv and $K^\top u$. It is also possible to use grid convolution to obtain a complexity of $\mathcal{O}(n \log n)$ for one update. More practical precise and exhaustive details can be found in Gabriel Peyré and Marco Cuturi's *Computational Optimal Transport* [PC18].

Dual One can also notice the formulation of the dual thanks to the Proposition 1.

$$W_\gamma(\mu,\nu) = \max_{\alpha,\beta} \alpha^\top a + \beta^\top b - \gamma \left(e^{\alpha/\gamma}\right)^\top K \left(e^{\beta/\gamma}\right)$$

Link with the KL divergence Writing the kernel:

$$K_\gamma(i,j) = \exp\left(-\frac{d_{i,j}}{\gamma}\right)$$

We obtain a new formulation of the entropic regularized problem using the KL-divergence

$$W_\gamma(\mu,\nu) = \min_{P \in U(a,b)} \gamma KL(P|K_\gamma)$$

III - Optimal Transport and Machine Learning

Now that we have convinced ourselves of the good mathematical foundations of Optimal Transport, and of the fact that we can actually compute it to some extent, we can wonder how we can put this tool to use in the setting of machine learning.

1/ What do we want to optimally transport ?

In the previous section we have seen that we have at our disposal an algorithm to compute optimal transport maps (Sinkhorn's algorithm), and that this algorithm can be parallelized. We can take advantage of this fact to actually compute optimal transport among huge databases. Since computing optimal transport maps implies that we can compute a meaningful distance between two objects, as long as they can be interpreted as distributions, one of the application of this is similarity based retrieval.

Color histograms The color-histogram of an image is a tool that characterizes the distribution of colors of an image. Interestingly enough, it has been noticed that color-histograms are a good tool to describe the similarity of two images. Since we now have a tool to compute the distance between two distribution probability, we can apply it to histograms (that are essentially discretized distribution, so much so that the algorithms and methods that we described earlier are relevant even without any fondamental change of setting). For a given image we can therefore compute the distance between its histogram and the histogram of images comming from a bigger database: picking images that have the most similar histogram (in the sense of the Wasserstein distance) will therefore allow to perform some form of similarity based image retrieval.

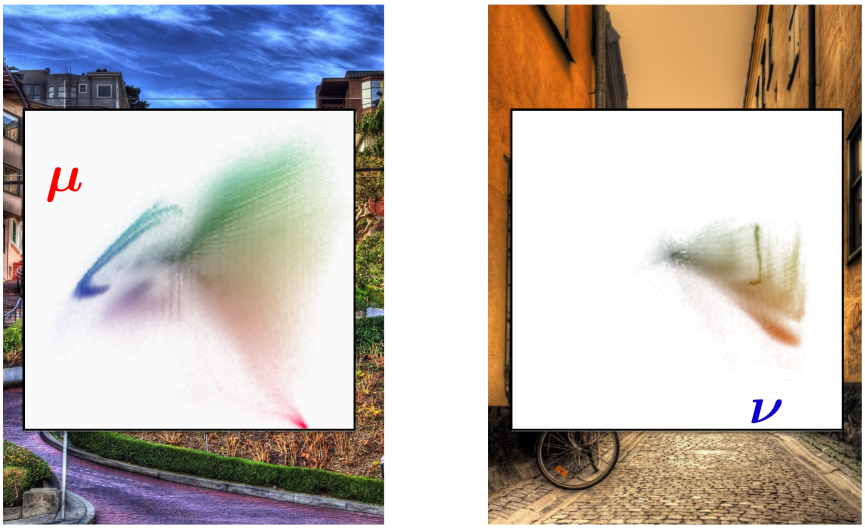


Figure A.2: Two images and their respective color-histograms, embedded in $\mathbf{R}^2$ (Image from Marco Cuturi's Primer on Optimal Transport at NIPS 2017)

Cloud of word We can apply the same technique as before, but instead of considering color-histograms, we can try to find some kind of distribution coming from other kind of documents, for example text documents. Some previous work, such as the `word2vec` framework allows for the embedding of natural language words into an Euclidean space. A text (seen as a collection of words) therefore somehow defines a probability distribution over this embedded space. As for images we can then compare this distribution to the distributions of other texts coming from a database (still in the sense of Wasserstein distance), and hopefully retrieve text dealing with similar topics, although there may be some significant differences in the actual vocabulary of those texts. We can take advantage of the geometry of the distribution of probability in the embedded text space, that tries to ensure that similar distributions are coming from texts dealing with similar topics.

2/ Wassersteinization

We can go further than just using the Wasserstein to compute distance between distributions and use this notion as distance as a main tool for already existing machine learning problems (we previously emphasized retrieval tasks for instance) by introducing the Wasserstein distance on well studied machine learning settings and problems. We therefore shed a new light on those problems, and the results are often fruitful. Some authors have coined the term **Wassersteinization** to describe this process.

Definition

Wassersteinization is the process of introducing optimal transport into an optimization or machine learning problem

One way to approach Wassersteinization is to introduce Wasserstein distance as a loss or fidelity term in optimization problems. This will motivate the interest toward not only

the very Wasserstein distance, but also its derivative. More precisely one can show that we can computationaly evaluate the derivatives of the Wasserstein distance. Therefore we can deal with it in a variety of optimization problems, including machine lerning related ones.

Averaging data We can also see optimal transport as a normalization tool for data. One of the setting where this point of view as been proposed is the one of data averaging. We have already discussed earlier the benefits of Wasserstein barycenters when compared to, say, linear interpolations or l^2 barycenters. One can put this in good use in the case of data acquired through real world experiment. For instance, we can imagine a neuro-imagery setting, where the goal is to study the reaction of the brain of a patient to a certain precise stimulus. To do this we subjet the patient to the stimuli several time and we aim to submit the measures of the brain activity to a machine algorithm to study some of its features. But because of the huge variety of data acquisition hazards (noise, imprecisions in the measures, patient variability etc), the result will slighty differ from one experiment to another although it concerns the same stimulus. Therefore computing the average of the brain activity (that can easily seen to be an analog of a probability distribution on the brain space) in the sense of Wasserstein barycenters can hopefully give access to more relevant data, without loosing the meaningful information.

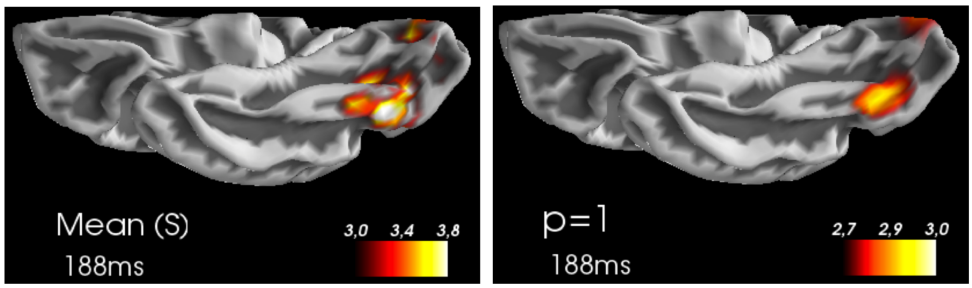


Figure A.3: On the left the standard average brain map among several experiments, on the right the Wasserstein average of the same brain activity data. (Image from [GPC15])

Aggregating distributions Instead of processing the input of our machine learning algorithms, one can use this idea of aggregating several distribution on the output of machine learning algorithms. Imagine that we want to use Bayesian learning (*ie.* learn a probability distribution, in a Bayesian framework), on a dataset so huge that it cannot even fit on a single machine, and would it fit that the computation time would be out of reach. We can imagine to split the dataset into J different parts and distribute it amongst J machines running the same learning algorithms. The result of this process is a set of J distributions learned from each part of the dataset. Although they represent the same truth, because of the variation of the data those distributions will be slightly different. Once again Wasserstein-averaging provides a useful tool to aggregate those distributions into an hopefully meaningful one, closer to the underlying real distribution. The good

news is that this Wasserstein posterior aggregation method comes with theoretic statistical guaranties!

Semi-supervised learning and Wasserstein propagation We are interested in smoothing a graph (V, E) where an histogram is associated μ_v to each vertex v, with a susbset S of the vertices that have a fixed histogram (known data). To write this more formally, we want to solve the following optimization problem:

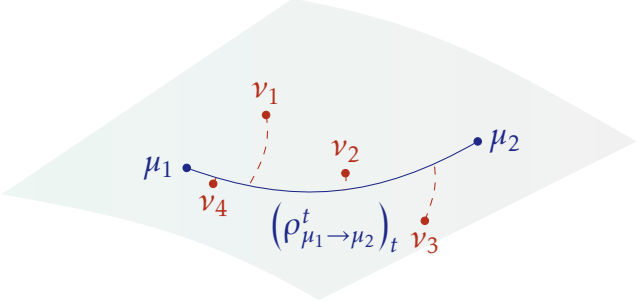
$$\min_{\substack{\mu_i \in \mathcal{P}(\Omega) \\ \text{for } i \in V \setminus S}} \sum_{(e_1,e_2)\in E} W_2^2\left(\mu_{e_1}, \mu_{e_2}\right)$$

That is, we want to propagate the distribution known at certain vertices along the graph, with a "Wasserstein manner". It is a formulation of several semi-suppervised learning problems, where we only have at our disposal the structure of the graph, and information a small portion of the vertices. The distribution that we are handling here can for instance be probability distribution of belonging to each class, in a classification problem (which leads us to solving the "stereotypical" semi-suppervised problem that is vertice labelling), but we could consider broader classes of problems with this formalism.

3/ Wasserstein PCA and Geodesics

The Wasserstein distance gives the space of probability measure a structure of geodesic metric space. Moreover, it has been shown that when Ω is an Hilbert space, then the space of probability measures endowed with the Wasserstein distance has a Riemannian manifold structure (see [AGS08]). Some authors such as [SC15] have proposed to take advantage of this structure to define the notion of **Principal Geodesic Analysis** over this space, which aims to be an analog of the good old Principal Component Analysis with Euclidean metrics.

The idea is that, as Ambrosio et al. shown in [AGS08], we can characterize the geodesic path between two distributions, that is the shortest path with respect to the Wasserstein distance. We denote $(\rho^t_{\mu_1\to\mu_2})_t$ the geodesic path between μ_1 and μ_2, parametrized by t. Now, given a family of measures $\nu_1, \nu_2, \ldots, \nu_N$, we would like to compute the pricipal geodesics of this family, that is find a set of geodesics between some measures (that also are to compute) that passes through the Wasserstein iso-barycenter of the ν_i and that are close to each of the ν_i.



We can formalize this idea more precisely as follows, using for example the second order Wasserstein distance W_2^2.

$$\min_{\mu_1,\mu_2\in\mathcal{P}(\Omega)} \sum_{i=1}^{N} \min_t W_2^2\left(\rho^t_{\mu_1\to\mu_2}, \nu_i\right)$$

Since a couple of measures (μ_1, μ_2) entirely characterizes a geodesic path, our optimization is performed over the space of probability measures over Ω. This problem can be solved using the methods that we presented earlier in our *How to compute Optimal Transport* section, and some projected gradient descent because of the overall non-convexity of this problem. We can iterate this process to gradually explain more and more features of our distributions.

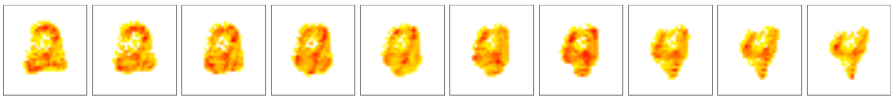


Figure A.4: Principal Geodesic Components obtained from a subset of 2000 images of the MNIST database, containing an equal proportion of 2 and 4.(Image from [SC15])

Chapter B

Review of papers

Table des matières

I - Sinkhorn Distances: Lightspeed Computation of Optimal Transport [Cut13]

In this paper published in 2013, Marco Cuturi proposed one of the currently most used algorithm to compute Wasserstein distances. He smoothed the classic optimal transport problem thanks to an entropic regularization and it leads to a new notion of distance which has a faster solver. This follows the regularization idea the we introduced in the first chapter. He also shows in this paper that using such a regularization gives better results on MNIST classification problems than using the standard Wasserstein distance (or obviously using the Euclidean distance). This paper is part of the search for adequate and tractable distances in machine learning. Wasserstein distance is more suited than Euclidean distance to many problems in machine learning, but because the complexity to compute it is $\mathcal{O}(n^3 \log n)$ when n is the size of the data, Wasserstein distance is rarely used. That is why such techniques to compute the Wasserstein distance had to be developped.

Idea The main ideas of this paper are the use of an entropic-regularization to simplify computation when using Sinkhorn iterative algorithm and the idea of computing the distances between one point and a family of other points in the same time, to use matrix-matrix multiplications instead of matrix-vector multiplications while computing distances between pair of points. Thus the algorithm can be implemented on GPU architectures.

Regularization We recall that the goal of the optimal transport problem in the Kantorovich formulation is to find a joint distribution $P \in \mathbf{R}^{n \times n}$ where the marginals equal the two distributions for which we want to compute the distance. That is to say $P\mathbf{1}_n = a$ and $P^\top \mathbf{1}_n = b$ when we compute the distance between a and b. Following the notations introduced in the first chapter, we call $U(a, b)$ the set of all possible values for P. By denoting the entropy by H we can show that:

$$H(P) \leqslant H(a) + H(b)$$

And $H(P) = H(a)+H(b)$ when $P = ab^\top$ (that is, when a and b are independant). We then restrict P to the new set:

$$U_\alpha(a, b) = \{P \in U(a, b) \,|\, \mathrm{KL}(P \| ab^\top) \leqslant \alpha\} = \{P \in U(a, b) \,|\, H(P) \geqslant H(a) + H(b) - \alpha\}$$

Two reasons are given to this restriction. The first one is that it will be easier to compute the distance. The second is that without regularization P has almost $2n - 1$ non-zero coefficients. Thus the transport is almost deterministic and it is not natural. That is why smoothing the transport plan with entropic regularization is presented as a good idea. In the paper, it is shown that this restriction induce a new distance:

$$d_{M,\alpha} = \min_{P \in U_\alpha(a,b)} \langle P, M \rangle$$

Where M is a distance matrix.

Computation By duality theory it is obtained that to α, corresponds a value $\lambda \geqslant 0$ such that the distance $d_{M,\alpha}$ is equal to the distance d_M^λ defined by:

$$d_M^\lambda(a, b) = \langle P^\lambda, M \rangle, \quad \text{where } P^\lambda = \operatorname*{arg\,min}_{P \in U(a,b)} \langle P, M \rangle - \frac{1}{\lambda} H(P)$$

Then Cuturi shows that this new distance can be computed with a much cheaper cost than the original Wassertein distance d_M. In fact the iterative Sinkhorn algorithm presented below converge in few steps to the transport plan P^λ. The Figure B.1 summarize the relationships between all these distances.

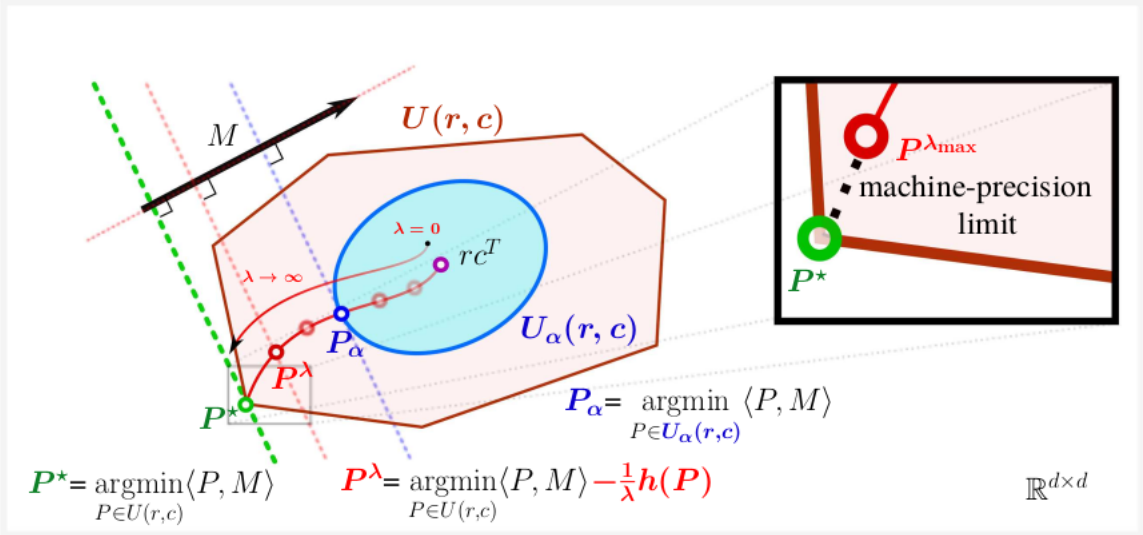


Figure B.1: Graphical representation of different distances and their relationships.

Algorithm 2: SINKHORN$\left(a, \{b_i\}_{i=1}^N, M, \lambda\right)$

$B \leftarrow [b_1, \ldots, b_N] \in \mathbf{R}^{n \times N}$;
$K \leftarrow \exp(\lambda M)$;
$u = [u_1, \ldots, u_N] \leftarrow \mathbb{1}_{n \times N} / n$;
repeat
  $u \leftarrow a / \left(K \left(B / \left(K^\top u\right)\right)\right)$;
until *convergence of* u;
$v = [v_1, \ldots, v_N] \leftarrow B / \left(K^\top u\right)$;
$P_i^\lambda \leftarrow \mathrm{diag}(u_i) K \mathrm{diag}(v_i)$;
return $d = \left[d_M^\lambda(a, b_1), \ldots, d_M^\lambda(a, b_N)\right] = \left[\left\langle M, P_1^\lambda \right\rangle, \ldots, \left\langle M, P_N^\lambda \right\rangle\right]$

Results The regularized Wasserstein distance is then used in a SVM to classifiy images of the MNIST database. The Figure B.2 compare the error on a test set for different distances used in the SVM. The distance EMD is the classical optimal transport. As we can see the regularized version beats all other distances. Furthermore he evaluated the execution time by stopping the iterations when $\|d_{i+1}/d_i - 1\| < 10^{-4}$ where d_i is the distance obtained at the end of the iteration i. He gave a comparison between EMD and regularized Wasserstein for different values of λ in the Figure B.3, highlighting the efficiency of computing the regularized optimal transport.

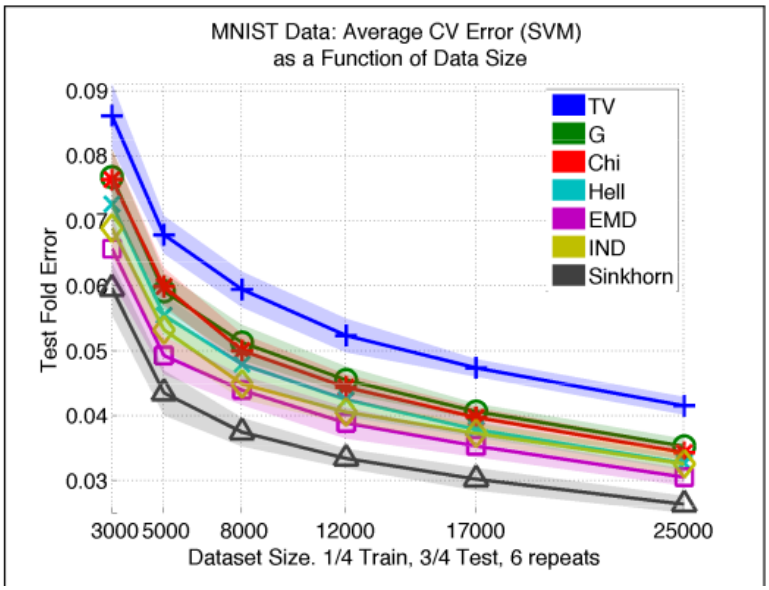


Figure B.2: Test error for different distances

II - Wasserstein GAN [ACB17]

This paper proposes a new GAN training algorithm based on Wasserstein distance. When we learn a probability distribution we need to have a notion of divergence between probability distributions. In a first time they show the advantages of the Wasserstein distance over all well-known other divergences on probabilities. Thanks to their WGAN (for Wasserstein GAN), they cure the problem of the need of maintaining a balance between training discriminator and training generator. They also observed that the mode dropping phenomenon is reduced. Furthermore their GAN allows to plot the Wasserstein distance continuously which is very usefull for debugging and having an estimation of the quality.

Advantage of Wasserstein distance They illustrate the interest of Wasserstein distance with a toy problem, that is the convergence of the probability $\mathbb{P}_\theta = g_\theta(Z) = (\theta, Z)$ toward $\mathbb{P}_0 = (0, Z)$ for $Z \sim U([0,1])$ when θ tends to 0. We have:

- $W(\mathbb{P}_0, \mathbb{P}_\theta) = |\theta|$ where W is the Wasserstein distance.
- $JS(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases}$ where JS is the Jensen-Shannon divergence.
- $KL(\mathbb{P}_0, \mathbb{P}_\theta) = JS(\mathbb{P}_\theta, \mathbb{P}_0) = \begin{cases} +\infty & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases}$ where KL is the Kullback-Leibler divergence.
- $\delta(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} 1 & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases}$ where δ is the total variation distance defined as $\delta(\mathbb{P}_0, \mathbb{P}_1) = \sup_{A \in \Sigma} |\mathbb{P}_0(A) - \mathbb{P}_1(A)|$ with Σ the set of all Borel subsets.

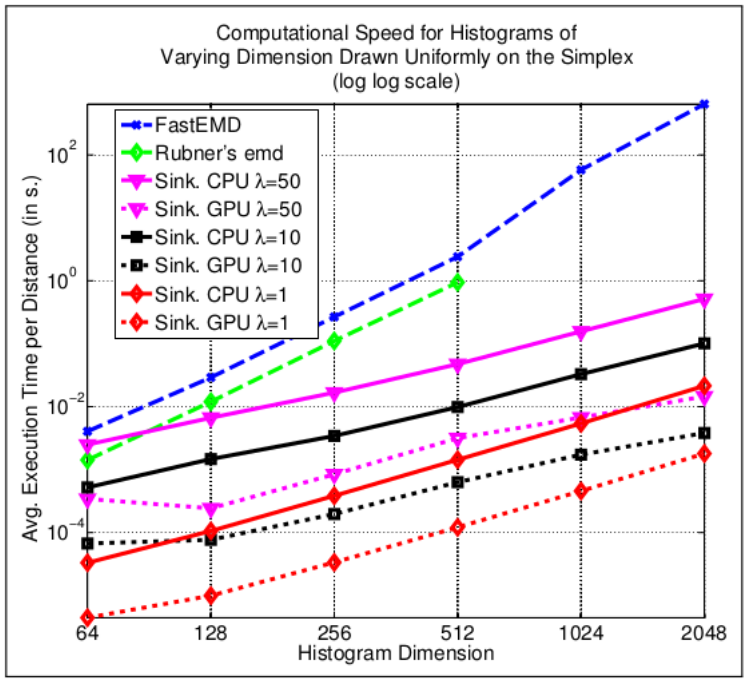


Figure B.3: Execution time for different distances

Furthermore the Wasserstein distance is the only one for which there is convergence. Then a gradient descent can be applied using the Wasserstein distance while it cannot be applied with other divergences. This is a just an example but they also proved the following theorem which shows the continuity under some assumptions.

Proposition 2

Let $\mathbb{P}_r$ be a distribution over $\mathcal{X}$ and let Z be a distribution over another space $\mathcal{Z}$ (e.g Gaussian). Let $g : \mathbf{R}^d \times \mathcal{Z} \to \mathcal{X}$ be a function that will be denoted $g_\theta(z)$ for $g(\theta, z)$. We denote by $\mathbb{P}_\theta$ the distribution of $g_\theta(Z)$. Then:

- If g is continuous in θ, so is $W(\mathbb{P}_r, \mathbb{P}_\theta)$.
- If g is locally Lipschitz and satisfies the following regularity assumption: By denoting the local Lipschitz constant $L(\theta, z)$ we have $\mathbb{E}_{z\sim Z}[L(\theta, z)] < +\infty$. Then $W(\mathbb{P}_r, \mathbb{P}_\theta)$ is continuous everywhere and almost differentiable everywhere.
- The two previous statements are false for Jensen-Shannon and KL divergences.

They also give another theorem that well show that the Wasserstein distance is a very natural divergence for probability distributions.

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

Figure B.4: Algorithm of the WGAN

Proposition 3

Consider a sequence of distributions $(\mathbb{P}_n)$ on a compact space $\mathcal{X}$. If $W(\mathbb{P}_n, \mathbb{P}) \to 0$ for a certain distribution $\mathbb{P}$, then $\mathbb{P}_n \xrightarrow{\mathcal{D}} \mathbb{P}$ where the limit is in the sense of the convergence in distribution.

WGAN To construct their GAN they need to compute the gradient of $W(\mathbb{P}_r, \mathbb{P}_\theta)$ with respect to θ. For that they consider the following dual formulation of the Wasserstein distance:

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \max_{\|f\|_L \leqslant 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$

Where $\|f\|_L$ denote the Lipshitz coefficient of f. If f is the solution to the maximization problem then the gradient is:

$$\nabla_\theta W(\mathbb{P}_r, \mathbb{P}_\theta) = -\mathbb{E}_{z\sim Z}[\nabla_\theta f(g_\theta(z))]$$

To find f they train a neural network parameterized with weights w in a compact space $\mathcal{W}$ that lead to a function f_w. Then they can backpropagate θ using $\mathbb{E}_{z\sim Z}[\nabla_\theta f_w(g_\theta(z))]$. To have w in a compact space, they use weight clamping. Their algorithm is in the Figure B.4.

Results First, the authors set up a small experiment to showcase the difference between GAN and WGAN (Figure B.5). There are two 1D Gaussian distributions, blue for real and green for fake. Train a GAN discriminator and WGAN critic to optimality, then plot their

values over the space. The red curve is the GAN discriminator output, and the cyan curve is the WGAN critic output.

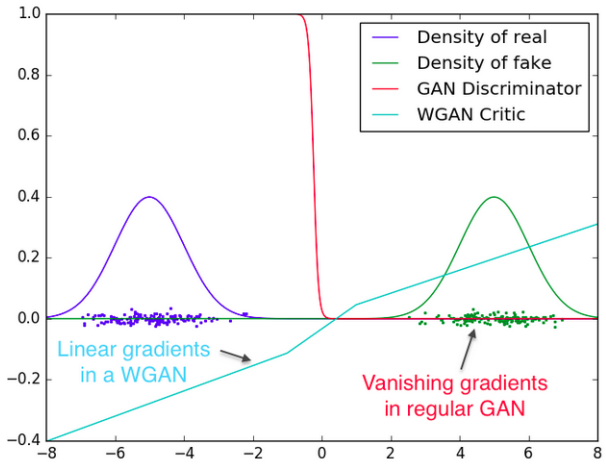


Figure B.5: Comparison between a GAN discriminator and the WGAN Critic

Both identify which distribution is real and which is fake, but the GAN discriminator does so in a way that makes gradients vanish over most of the space. In contrast, the weight clamping in WGAN gives a reasonably nice gradient over the whole space.

Another important point is that the quality of the generation is correlated to Wasserstein estimation as we can see in the Figure B.6. We can also see that the training of the critic function f works well as the Wasserstein estimator strictly decrease. The gradient of the loss function is well computed. But this is not a method to compare the quality of different GANs as the estimator is defined with a constant scaling factor depending on the critic's architecture. They finally observed an improvement of the stability.

III - Convolutional Wasserstein Distances: Efficient Optimal Transportation on Geometric Domains [SdGP+15]

Because a single step of Sinkhorn algorithm has a complexity of $\mathcal{O}(n^2)$, computing the Wassertein distance may take a while in many cases, even though the number of iterations needed to converge toward a good minimizer is small. That is why the researchers who wrote this paper thought about a more effective solution to perform a step of the algorithm. They found a way to use Gaussian convolution with a $\mathcal{O}(n \log n)$ complexity instead of a matrix-vector product. More generally they use a heat kernel. So instead of the convolution it can also be a sparse pre-factored linear system which can be computed in a complexity smaller than $\mathcal{O}(n^2)$. Furthermore this approach does not affect the overall convergence speed of the scheme, which remains linear. Of course this can not be applied to all the cases of optimal transport, but it can be used on very common geometric domains, like images or meshes. In addition of the computation of Wasserstein distance, the

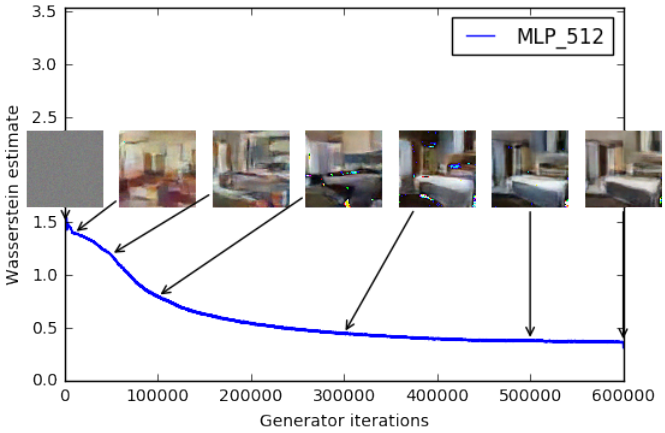


Figure B.6: Sample images at different iterations of the training

authors also propose a way to use convolution in the computation of Wasserstein barycenters and Wasserstein propagation.

Using a heat kernel in the entropy-regularization We recall what is the entropy-regularized Wasserstein distance between μ_0 and μ_1 on the domain M:

$$W_{2,\gamma}^2(\mu,\nu) = \inf_{\pi\in\Pi(\mu,\nu)} \left[\int\int_{M\times M} d(x,y)^2 \pi(x,y) dxdy - \gamma H(\pi) \right] = \gamma \left[1 + \min_{\pi\in\Pi(\mu,\nu)} \mathrm{KL}(\pi|\mathcal{K}_\gamma) \right]$$

Where have kept the same notations as the ones used in the first chapter, that is

$$\Pi = \{\pi \in \mathcal{P}(M\times M) \mid \pi(\cdot, M) = \mu_0, \pi(M,\cdot) = \mu_1\}$$

$$H(\pi) = -\int\int_{M\times M} \pi(x,y) \ln \pi(x,y) dxdy$$

$$\mathcal{K}_\gamma = e^{-d(x,y)^2/\gamma}$$

$$\mathrm{KL}(\pi|\mathcal{K}) = \int\int_{M\times M} \pi(x,y) \left[\ln \frac{\pi(x,y)}{\mathcal{K}(x,y)} - 1 \right] dxdy$$

This is a strictly convex problem thanks to the entropy. The idea of the paper is to use heat kernel because in some cases, the solution of the diffusion equation can be computed in a effective way. We denote by $\mathcal{H}_t(x,y)$ the diffusion between x and y after a time t in the heat kernel. Just remember that a heat kernel is a solution of the heat equation:

$$\frac{\partial f}{\partial t} = \Delta f$$

One can show the following that the following approximation holds

$$\mathcal{K}_\gamma \approx \mathcal{H}_{\gamma/2}(x, y)$$

We won't store $\mathcal{H}$ because it could have a space complexity in $\mathcal{O}(n^2)$ and we want a lower complexity. But we generally know a way to apply $\mathcal{H}$ to a vector. In the case of images we apply Gaussian convolution with $\sigma^2 = \gamma$. In the case of triangle meshes we associate a weight to faces proportional to their area. We denote by a the vector of those weights (for images of size $n \times m$ we set $a = 1/(nm)$). Then we denote by L the cotangent Laplacian and by D_a the diagonal matrix whose diagonal is a. By discretizing the heat equation, we obtain:

$$w = \mathcal{H}_t(v) \Leftrightarrow (D_a + tL)\, w = v$$

The linear system can be solved efficiently by pre-computing a sparse Cholesky factorization.

Algorithm With some computations we arrive at

Algorithm 3: CONVOLUTIONAL-SINKHORN(μ_0, μ_1, H_t, a)

$v, w \leftarrow 1$;
repeat
  $v \leftarrow \mu_0 \oslash H_t(a \otimes w)$;
  $w \leftarrow \mu_1 \oslash H_t(a \otimes v)$;
until *convergence of* v, w;
return $2ta^\mathsf{T}\left[(\mu_0 \otimes \ln v) + (\mu_1 \otimes \ln w)\right]$

Where $\oslash$ and $\otimes$ denote element-wise operations. As with classical Sinkhorn algorithm we obtain a simple iterative algorithm where, this time, each step can be computed in $\mathcal{O}(n \log n)$ in many cases, n being the size of the domain M.

Barycenter An algorithm for computing barycenters is also provided. We won't enter into details but we give their algorithm for computing the barycenter of distributions μ_i associated with weights α_i for $1 \leqslant i \leqslant k$:

Algorithm 4: CONVOLUTIONAL-BARYCENTER($\{\mu_i\}, \{\alpha_i\}, H_t, a$)

$v_1, \dots, v_k \leftarrow 1$;
$w_1, \dots, w_k \leftarrow 1$;
repeat
  $\mu \leftarrow 1$;
  for $i = 1, \dots, k$ **do**
    $w_i \leftarrow \mu_i \oslash H_t(a \otimes v_i)$;
    $d_i \leftarrow v_i \otimes H_t(a \otimes w_i)$;
    $\mu \leftarrow \mu \otimes d_i^{\alpha_i}$;
  // Optional ;
  $\mu \leftarrow$ ENTROPIC-SHARPENING$(\mu, \max_i H(\mu_i))$;
  for $i = 1, \dots, k$ **do**
    $v_i \leftarrow v_i \otimes \mu \oslash d_i$;
until *convergence of* v_i, w_i;
return $2ta^{\mathsf{T}}[(\mu_0 \otimes \ln v) + (\mu_1 \otimes \ln w)]$

Here entropic sharpening allows us to make the entropy of the result μ smaller than the maximum entropy of all μ_i. To do so if $H(\mu)$ is greater than $\max_i H(\mu_i)$, then we set μ to μ^β where β is such that $H(\mu^\beta) = \max_i H(\mu_i)$. We can find such a β using Newton-like methods.

Applications This has many applications in geometric domains. Applications that are given are:

- Shape interpolation (Figure B.7)

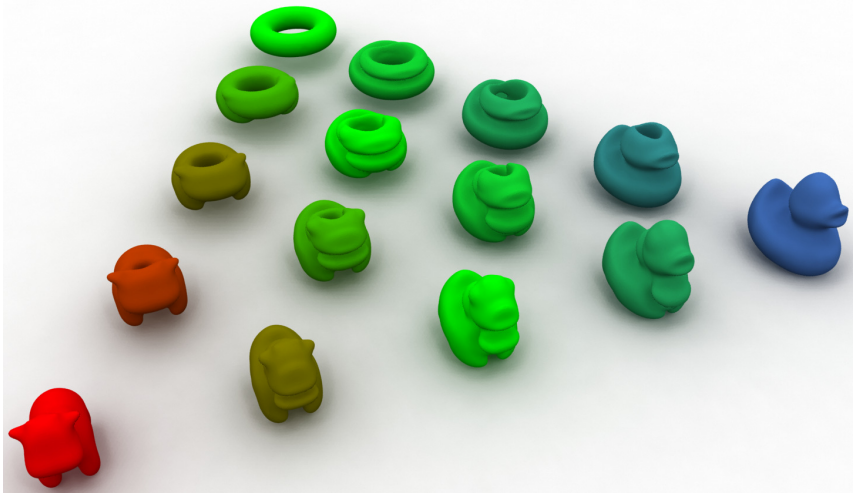


Figure B.7: Application to shape interpolation

- BRDF design
- Color histogram manipulation
- Skeleton layout
- Soft maps

Finally Figure B.8 and Figure B.9 are two images of barycenters that they have computed. The first one is an image where we find several barycenters of four images that are in the corners. The second is an interpolation between two cards. As we tried to reproduce it in our implementation part we put them here, for comparison sakes:

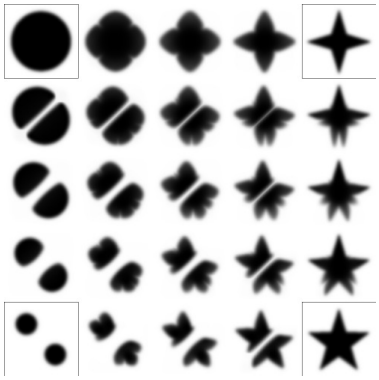


Figure B.8: Barycenters of four images

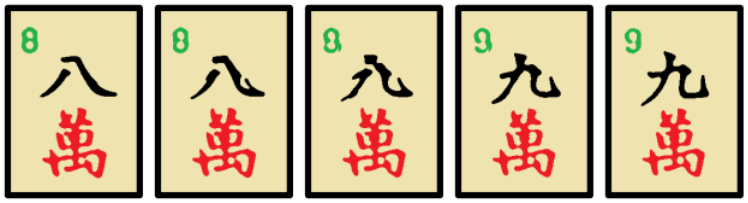


Figure B.9: Interpolation of two images of cards for times $t = 0, 0.25, 0.5, 0.75, 1$.

Chapter C

Implementation

We decided to implement the last paper to compute Wasserstein distances on images in an efficient way. We then use convolutional Sinkhorn in an SVM classifier as Marco Cuturi used Sinkhorn in his paper to classify images of the MNIST digit database [Cut13]. The implementation of the convolutional Wassertein distance has been written in C++. We then write the Gramm matrix of a subset of the MNIST database in a file and use it in a Python script to use SVM from Sklearn library. Our implementation is for example available at `http://perso.ens-lyon.fr/nemo.fournier/studies/M1/ML/implementation.tar.gz`.

How to use our code OpenMP has been used to parallelize the code. If you don't have OpenMP already installed, you can install it with `sudo apt install libomp-dev`. To compile the C++ code you just have to type `make`. Then you need to obtain the train image datatset and the train label dataset of the MNIST database `http://yann.lecun.com/exdb/mnist/`. To do so you have to run the script `get_mnist.sh` which will download and rename the files correctly. The executable `main` created by `make` command can be used in four ways.

- `./main bar` to reproduce an image similar to the Figure B.8; An image containing different barycenters of four shapes.
- `./main card` to reproduce images similar to the Figure B.9; An interpolation of two images of cards.
- `./main wass > kernel.txt` to compute a kernel with the convolutional Wasserstein distance of `N_TRAIN` images of the MNIST dataset. We project `N_TEST` other digit images of the MNIST dataset on the kernel space created. You can change the parameter `N_TRAIN` and `N_TEST` at the beginning of the file `src/main.cpp`. The kernel will then be written if the file `kernel.txt`.
- `./main eucl > kernel.txt` to do the same as the previous command but with Euclidean distance instead of Wasserstein distance.

Finally when a kernel file is computed you can test SVM on it thanks to the python script `learning.py`. Because computing Wasserstein kernel may take a while to compute we provide a pre-computed Wasserstein kernel `kernels/wass.txt` as well as an Euclidean kernel `kernels/eucl.txt`. You can then run:

```
python3 learning.py < [kernels/wass.txt | kernels/eucl.txt]
```

It will print the performances of these two kernels, show the projection of the data on the two principle components and finally show the kernel matrix.

Barycenters We first computed some barycenters to be sure that everything works. The code for computing Wasserstein distances and barycenters is in the file `src/wasserstein.cpp`. The code that produce the barycenters that we will present is in the file `examples.cpp`. The first example is the reproduction of Figure B.8. Here is what we obtained:

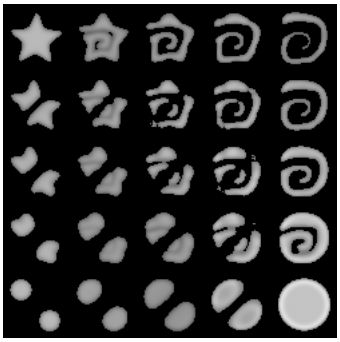


Figure C.1: Reproduction of Figure B.8

The result seems to be correct. We then try on larger images, the card images that we can find in the original paper presenting convolutional Wassertein [SdGP+15]. Here is what we obtained:

Figure C.2: Reproduction of Figure B.9

SVM usage As proposed in the paper of Cuturi [Cut13], we used the Wasserstein distance to compute the Gram matrix for 500 image and we used it as a kernel in a SVM. On a test set of 1500 images we obtained 92.4% of good answers. Whereas with a kernel using Euclidean distance we obtain a good answer on only 88.4%. To check these results

you can run the python script `learning.py` on the two kernels that are in the `kernels` folder.

Bibliography

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv e-prints*, page arXiv:1701.07875, Jan 2017.

[AGS08] L. Ambrosio, N. Gigli, and G. Savare. *Gradient Flows: In Metric Spaces and in the Space of Probability Measures*. Lectures in Mathematics. ETH Zürich. Birkhäuser Basel, 2008.

[Cut13] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. pages 2292–2300, 2013.

[GPC15] Alexandre Gramfort, Gabriel Peyré, and Marco Cuturi. Fast optimal transport averaging of neuroimaging data. In *International Conference on Information Processing in Medical Imaging*, pages 261–272. Springer, 2015.

[PC18] Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11:355–607, 2018.

[SC15] Vivien Seguy and Marco Cuturi. Principal geodesic analysis for probability measures under the optimal transport metric. In *Advances in Neural Information Processing Systems*, pages 3312–3320, 2015.

[SdGP+15] Justin Solomon, Fernando de Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas J. Guibas. Convolutional wasserstein distances: efficient optimal transportation on geometric domains. *ACM Trans. Graph.*, 34(4):66:1–66:11, 2015.

[Vil03] C. Villani. *Topics in Optimal Transportation*. Graduate studies in mathematics. American Mathematical Society, 2003.