Learning Wasserstein Embeddings : a report

Maxime Duval
Master MVA
Ecole des Ponts ParisTech
Marne-la-Vallée
maxime.duval@eleves.enpc.fr

Abstract

Using the Wasserstein distance for machine learning holds great promises, notably for its principled way of comparing distributions. Its adoption is unfortunately hard because of its heavy computational cost. This report, based on the paper by Courty and al. (1) shows that it is possible to accurately learn Wasserstein embeddings by using a siamese architecture encoding scheme coupled with a decoder. It also introduces a Kullback-Leibler divergence based regularization. This papers proves experimentally that the computation of Wasserstein embeddings can be extremely fast on some datasets. This is specially interesting for the W_2 (or more generally $W_p, p > 1$) case. While it provides a good numerical tool, it lacks theoretical support.

1 Introduction

The Wasserstein distance is an interesting mathematical object, as it represents one good way to compare distributions. Two major reasons explain its power : (i) it works on data distributions in a non-parametric way (ii) The geometry of the underlying space can be leveraged to compare the distributions in a geometrically sound way (for example we can use Wasserstein Barycenters or Principal Geodesic Analysis).

1.1 Problem

The Wasserstein distance is defined between probability measures. Let X be a metric space endowed with a metric d. Let $p \in (0, \infty)$ and $\mathcal{P}_p(X)$ be the set of Borel probability measures μ with finite moments of order p, *i.e.* $\forall x_0, \int_X d(x, x_0)^p d\mu(x) < \infty$. Then, the p-Wasserstein distance between distributions μ and ν is defined as :

$$W_p(\mu, \nu) = \Big(\inf_{\pi \in \Pi(\mu,\nu)} \int_{X \times X} d(x, y)^p d\pi(x, y) \Big)^{\frac{1}{p}}$$

where $\Pi(\mu, \nu)$ is the set of all probability couplings between μ and ν, that is, the set of all joint probabilities on $X \times X$ whose projections onto the first and second variable are respectively equal to μ and ν. W_p defines a metric over $\mathcal{P}_p(X)$ as soon as $p \geq 1$ (e.g. (2), definition (6.1)).

When $p = 1$, the W_1 distance is known as Earth Mover's Distance (EMD). This distance has been thoroughly studied, and there exists good methods for computing this distance for point sets in $\mathbb{R}^k$. However, the W_2 ($p = 2$) distance is more interesting for a number of applications, notably because we can use $d(x, y) = \|x - y\|^2$, which is more natural for computer vision (images) or any other field where the data space has dimension > 1.

1.2 Related work

Because of the computational complexity of computing the Wasserstein distance for $p > 1$, the deployment and adoption of this distance to a wide range of tasks has been limited. For discrete distributions (most real-world applications), the W_2 Wasserstein distance computation has complexity $O(n^{\frac{3}{2}})$ (see (3)). Some recent strategies use slicing techniques, entropic regularization or involve stochastic optimization. According to the paper under study, the cost of computing pairwise Wasserstein distances for a large number of distributions remains prohibitive. This may have become less true with a recent paper (4), where they propose an algorithm which achieves an ϵ error and runs in $O(n)$ (the full complexity is $O(\frac{n}{\epsilon^3}(\frac{C \log n}{\epsilon})^d)$, where data points live in $\mathbb{R}^d$). This would make the paper less interesting, as many Wasserstein distance applications only need W_2 computation. Still, the method derived in the paper applies to any W_p.

Metric embedding This paper is part of the general question of metric embedding, that is, finding a function ϕ that embeds points from some space $\mathcal{X}$ to another space $\mathcal{Y}$ while preserving distances. Mathematically, we say that ϕ is a mapping with distortion at most $D \geq 1$ if there exists $\alpha > 0$ such that $\forall x, y \in \mathcal{X}, \alpha d_{\mathcal{X}}(x, y) \leq d_{\mathcal{Y}}(\phi(x), \phi(y)) \leq D\alpha d_{\mathcal{X}}(x, y)$. The distorsion D of ϕ is the infimum over D where D respects the previous inequality. The paper mentions a good introduction on metric embedding published by Matoušek, 2013 (5).

Wasserstein embeddings Theoretically, embedding the Wasserstein space into a normed metric space is still open. We know that for absolutely continuous measures on $\mathbb{R}$, there is an embedding of W_1 into L^1. For W_2, in (6) they show that there does not exist an embedding into L^1 with less than $O(\sqrt{\log n})$ distortion. In the paper, they show experimentally that it is possible to embed distributions whose distance is W_2 into a Euclidean space where the distance is simply the euclidean norm, with constant distortion. This is probably because the manifold on which the Wasserstein distance operates in a small subset of the whole subset of images.

Wasserstein distance in Machine Learning Wasserstein distances are increasingly used in the Machine Learning community for their principled way of comparing distributions. They are used successfully in data generation (7), (8), auto-encoding data (9), graphs embedding (10), blue-noise generation (11), barycenter computation (12), shape interpolation (13).

1.3 Contribution

The contribution of the paper is to propose a new neural network architecture 1, called Deep Wasserstein Embedding (DWE) that is able, combined with a specific loss, to learn Wasserstein distances.

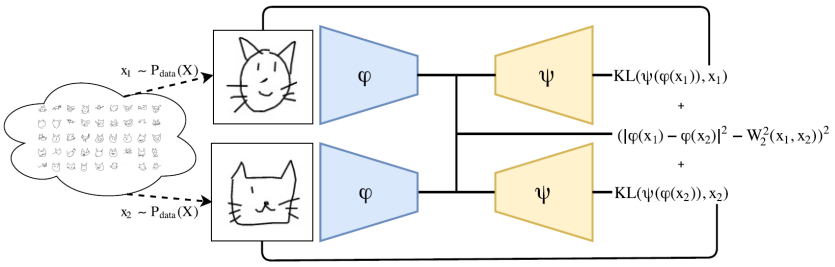


Figure 1: Architecture developed in the paper. The training is done using a Siamese type model and pairs of images. An encoder and a decoder are used in tandem.

More precisely, the neural network has an encoder ϕ which projects the input distribution x onto a Euclidean latent space $\phi(x)$. This latent space is such that the Wasserstein distance between two distributions x_1, x_2 is approximated by the Euclidean norm between the projections of those distributions into the latent space $\|\phi(x_1) - \phi(x_2)\|$:

$$W_p(x_1, x_2) \approx \|\phi(x_1) - \phi(x_2)\|$$

To make it work, the neural network tries to reconstruct the input from the latent space with a decoder : this allows us to reconstruct the probability distribution from the latent space.

Once learnt, the DWE provides a significant speed-up for computing Wasserstein distances. Its encoders ϕ and ψ also allow what they call Wasserstein data mining in the embedding space. It is the use of the latent space for techniques or tasks that require the computation of Wasserstein distances. Here are some examples : Wasserstein barycenters, Principal Geodesic Analysis, Wasserstein K-means.

Personal contributions : in this report, I investigate multiple things. (i) I replicate results on the MNIST dataset (14) for the tasks of learning the Wasserstein distance, computing Wasserstein Barycenters and using Principal Geodesic Analysis. (ii) I also implement Wasserstein k-means (15). (iii) Finally, I look at the quality of the latent space learnt and its generative potential for approximating the input distribution.

2 Methods

2.1 Network architecture and Loss

The key idea of the paper is to use a kind of auto-encoder neural network in a Siamese architecture to learn the Wasserstein distance. Denoting by ϕ the encoder and ψ the decoder, the loss between two inputs x_1, x_2 and their precomputed y Wasserstein distance is :

$$\begin{aligned} L(x_1, x_2) = (& \underbrace{\underbrace{\|\phi(x_1) - \phi(x_2)\|^2}_{\text{estimated Wasserstein distance}} - y}_{\text{Wasserstein dist. error}})^2 \\ & + \underbrace{\text{KL}(\psi(\phi(x_1)), x_1)}_{\text{Reconstruction error for input 1}} \\ & + \underbrace{\text{KL}(\psi(\phi(x_2)), x_2)}_{\text{Reconstruction error for input 2}} \end{aligned}$$

The fact that the reconstruction error term uses a Kullback-Leibler divergence is arbitrary but is supported by the fact that we compare probability distributions. In practice, a regularization term λ is multiplied to the reconstruction term to control how much we favor reconstruction quality over the precision of the Wasserstein distance.

2.2 Wasserstein barycenters

For a set of inputs $x_1, ..., x_n$ and a set of weights $\alpha_1, ..., \alpha_n$, where $\alpha_i > 0$ and $\sum_{i=1}^n \alpha_i = 1$, their Wasserstein barycenter ((16)) $\bar{x}$ is defined as :

$$\bar{x} = \arg\min_{\alpha} \sum_{i=1}^{n} \alpha_i W(x, x_i)^2$$

We can approximate this barycenter with our neural network :

$$\bar{x} = \arg\min_{\alpha} \sum_{i=1}^{n} \alpha_i \|\phi(x) - \phi(x_i)\|$$

which can be further approximated in a closed form solution by :

$$\bar{x} = \psi(\sum_{i=1}^{n} \alpha_i \phi(x_i)) \tag{1}$$

When $\alpha_i = \frac{1}{n}$, the Wasserstein barycenter is also known as the Wasserstein population mean or Fréchet mean ((17), proposition 4.1). Results are shown in the Numerical Experiments section.

2.3 Principal Geodesic Analysis

Principal Geodesic Analysis (PGA) was introduced by Fletcher et al. (18). It is a generalization of Principal Component Analysis (PCA) for general Riemannian manifolds. Like PCA, its goal is to find a set of directions, called geodesic directions, that maximally explain the variability of the data. In the article, they explain how Fletcher's formulation is intractable and propose a new PGA approximation : (i) Find and subtract the approximate Fréchet mean (computed with Wasserstein barycenters) to all samples. (ii) Recursively construct a subspace $V_k = \text{span}(v_1, ..., v_k)$ in the embedding space :

$$v_1 = \arg\max_{\|v\|=1} \sum_{i=1}^{n} \langle v, \phi(x_i) \rangle$$

and

$$v_k = \arg\max_{\|v\|=1} \sum_{i=1}^{n} \Big(\langle v, \phi(x_i) \rangle + \sum_{j=1}^{k-1} \langle v_j, \phi(x_i) \rangle \Big)$$

This subspace corresponds exactly to the PCA in the embedding space. Thus, it is much faster to run thanks to our DWE framework than with classical tools.

2.4 Wasserstein k-means

Wasserstein k-means was first introduced in Cuturi et al. (15). My implementation only uses the case where $\Theta = \Sigma_n$ (free assignments). It is simply a classical k-means where distances are Wasserstein distances and objects are distributions.

My implementation (see Algorithm 1) profits from the DWE framework : I compute projections into the Wasserstein latent space only once. This makes following computations extremely fast. An other advantage is the possibility to compute the corresponding image of centroids, thanks to the decoder ψ.

Here are my two equations that support Wasserstein k-means :

$$c_j = \frac{1}{|C(j)|} \sum_{i \in C(j)} \phi(x_i) \tag{2}$$

with $C(j) = \{i \mid l_i = j\}$, and :

$$l_i = \arg\min_{1 \leq j \leq K} \underbrace{\|\phi(x_i) - c_j\|^2}_{\sim W_2^2(x_i, \psi(c_j))} \tag{3}$$

A measure of the quality of the clustering can be obtained with

$$L(c) = \sum_{i=1}^{n} \|\phi(x_i) - c_{l_i}\|^2$$

Data: Dataset $\{x_1, ..., x_n\}$, DWE (ϕ, ψ), integer $K > 1$, threshold ϵ, maximum iteration M
Result: Centroids $\{c_1, ..., c_k\}$, Assignments $(l_i)_{1 \leq i \leq n}$, $l_i \in \{1, ..., K\}$
Compute latent vectors $\{\phi(x_1), ..., \phi(x_n)\}$;
Initialize centroids with k-means++;
iter = 0;
while $L(c)$ *improves by more than* ϵ *and iter* < M **do**
    Compute assignments with (3);
    Compute centroids with (2);
    iter++;
end

Algorithm 1: Wasserstein K-means

3 Theoretical Guarantees

As mentioned in the discussion concerning the acceptation of the paper at ICLR 2018, its weakness is its lack of theoretical analysis. This is probably explained because of the lack of results on neural network generalization error : the paper may be summarized as a neural network (with some tweaks) that can learn a function : the Wasserstein distance.

In (6), some results give intuition on the state of the research. Here I recap some of the results. Let $\mathcal{P}_p(X)$ be the set of all Borel probability measures μ on X satisfying

$$\int_X d_X(x, x_0)^p \mathrm{d}\mu(x) < \infty$$

Let (X, d_X) and (Y, d_Y) be metric spaces, $D \in [1, \infty]$. A mapping $f : X \to Y$ is said to have distortion at most D if there exists $s > 0$ such that $\forall x, y \in X$ we have :

$$sd_X(x, y) \leq d_Y(f(x), f(y)) \leq Dsd_X$$

The infimum over those D is the distortion of f noted dist(f). We say that (X, d_X) embeds with distortion D into (Y, d_Y). The infimum of dist(f) over all mappings between X and Y is denoted $c_{(Y,d_Y)}(X, d_X)$. We also introduce the term "θ-snowflake" : the metric space (X, d_X^θ) is called the *θ-snowflake* of (X, d_X).

Theorem 1 asserts that for any *finite* metric space (X, d_X),

$$c_{(\mathcal{P}_p(\mathbb{R}^3),W_p)}(X, d_X^{\frac{1}{p}}) = 1$$

This means that $(\mathcal{P}_p(\mathbb{R}^3), W_p)$ is snowflake-universal. : it can be used as the metric space used for modeling almost everything we encounter in the real world.

Theorem 2 states that for any fixed $p \in (1, \infty)$, for any $n \in \mathbb{N}$, the metric space X of measures on $\mathbb{R}^3$ supported by at most n points equipped with the W_p metric is such that any embedding of X into L_1 must incur distortion $\Omega(((p-1)\log n)^{1/p})$: the distortion is not constant.

Concerning our problem, we would like to have asymptotic results for subsets X_n of n elements $x_1, ..., x_n \in X$ on $c_{(\mathcal{P}_p(\mathbb{R}^k),W_2)}(X_n, d_X)$. To make the problem easier :

- one may consider that the true data distribution of X_n lives on a manifold $\mathcal{M}$ with smaller dimension.
- We could also say that we don't want general results for all mappings but only for class of functions like neural networks (which are a succession of linear transforms and non-linear activation functions)
- Finally, maybe only considering euclidean distances $d_Y(x, y) = \|x - y\|$ and embeddings into Euclidean or Hilbert spaces may help.

In Numerical Optimal Transport (19), Proposition (8.2) states that the p-Wasserstein distance is not Hilbertian (which means it cannot be embed with constant distortion) for $X = \mathbb{R}^d$ with $d > 1$ and $p = 1, 2$, i.e. $c_{(X,d_X)}(\mathcal{P}_p(\mathbb{R}^{d,d\geq 2}), W_{p,p\in\{1,2\}})$ is not fixed. This is not the case for $d = 1$ as the inverse cumulative function gives an embedding. In the next section (4.1), we show that with 1d gaussians, we have good approximation of the true Wasserstein distance.

Finally, in (20), they define an explicit embedding of the 2-Wasserstein distance into a Hilbert space, which is Hölder-continuous, using the Monge map (from Monge's formulation of the optimal transport). This Monge map can be seen as a generalization of the 1d quantile function. For a target distribution μ and a fixed distribution ρ it is defined as :

$$T_\mu = \arg\min_T \{ \int_X c(x, T(x))\rho(x)dx | T : X \to Y, T_\# \rho = \mu \}$$

They prove that this embedding $\mu \to T_\mu$ is a bi-Hölder embedding of probability distributions into $L^2(\rho)$ (the space of functions f such that $\int_X f(x)^2\rho(x) < \infty$). More precisely, for all $\mu, \nu \in \mathcal{P}(Y)$, for all $p \geq 1$:

$$W_2(\mu, \nu) \leq \|T_\mu - T_\nu\|_{L^(\rho)} \leq CW_p(\mu, \nu)^{\frac{2}{15}}$$

4 Numerical Experiments

For all experiments, I computed Wasserstein distances ground truths with the python package POT (Python Optimal Transport) (21).

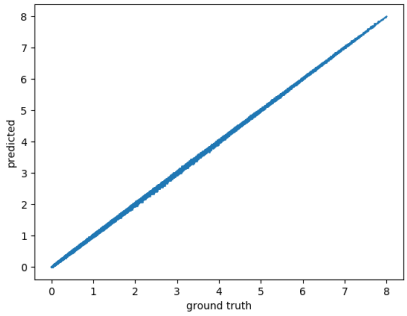
4.1 Gaussians

I first show and test the precision accuracy of the DWE architecture. For that purpose, I work with gaussians. This also answers one of the demands of the reviewers of the paper at ICLR 2018. Working with Gaussians has an intrinsic quality : we know the W_2 distance ground truth. Indeed, we have the following equality :

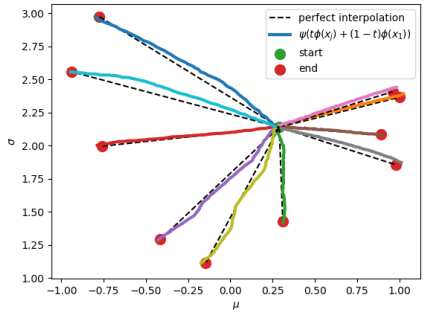
$$W_2(\mathcal{N}(m_1, \Sigma_1), \mathcal{N}(m_2, \Sigma_2))^2 = \|m_1 - m_2\|_2^2 + \text{Tr}(\Sigma_1 + \Sigma_2 - 2(\Sigma_1^{\frac{1}{2}}\Sigma_2\Sigma_1^{\frac{1}{2}})^{\frac{1}{2}})$$

This allows me to compute exact data distributions and Wasserstein distances, to assess the capacity of the DWE network at accurately predicting Wasserstein distances. The main result is that a DWE architecture is able to embed one dimensional Gaussians with very good prediction (figure 2 left) : on a test sample the predicted Wasserstein distance has relative mean absolute error 1.3%, relative MSE 0.023%. 99% of errors are below 2.7% relative error, 50% are below 0.56% relative error.

Interpolations are also very good. Another advantage of using 1d Gaussians is that I can represent their distributions as points in the 2d space of means and variances. I compute several interpolations trajectories $(\psi(t\phi(x_1) + (1-t)\phi(x_2)))_{0 \leq t \leq 1}$ and show them Figure 2 (right). The path is quite straight, and never diverges that much from the perfect interpolation. The remaining noise may be due to the errors coming from the estimation of the mean and variance of the decoded input : I sample from it, by considering it as a probability distribution function.



(a) Scatter plot of the (ground truth, predicted) Wasserstein distances.



(b) Interpolation trajectories for Gaussians. I sample a "start" distribution and "end" distributions, compute the interpolated distributions and extract the mean and variance.

Figure 2: DWE results for 1d gaussians

4.2 MNIST

I replicated results of the paper on the MNIST dataset (14). I train the DWE for 10 epochs on 100000 image tuples. The architecture of the encoder is simple : 3 convolutions followed by 2 fully connected layers. The architecture of the decoder is the inverse of that of the encoder. My learning rate is 0.001 and I use the Adam optimizer (22).

Numerical precision In terms of the approximation performance of the Wasserstein distance, here are some results that I got : I have a mean absolute error (MAE) of 0.64, a relative MAE of 6.7%, a mean squared error (MSE) of 0.71, a relative MSE of 6.8%, a median error of 0.5 for a mean Wasserstein distance of 11.2. The correlation between my predictions and the ground truths is 0.990 Figure 3.

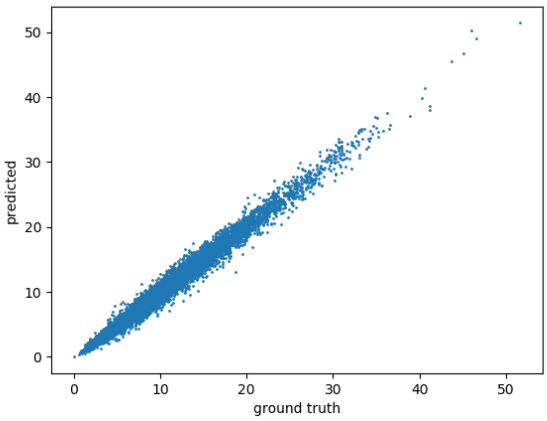


Figure 3: Scatter plot of the (ground truth, predicted) Wasserstein distances.

Method	W_2^2/s
OT solver	136
DWE, 1 CPU	745
DWE, 1 GPU	20840

Table 1: Speed performance for computing the W_2 Wasserstein distance for different methods, for the MNIST dataset

I also tested the speed increase for computing Wasserstein distances 1. I only report "Independent" (see original paper) results. My Machine has i7-6700HQ with 2.60 GHz frequency CPUs, and a Nvidia GTX 960M GPU.

In the paper, they also mention "Pairwise" computation, an obvious speed increase that the method allows : in the case where we want to compute all Wasserstein distances between 2 sets of distributions, we can simply compute the Wasserstein embeddings of each of the samples and then compute their euclidean distances. More generally, for a set of n distributions, instead of naively computing each of the $n(n-1)/2$ distances, we can compute Wasserstein embeddings for each of the n distributions (which is the costly operation) and then compute pairwise euclidean distances (which is almost free in terms of computational cost) of those embeddings to get the Wasserstein distances. This makes the computation of distances $O(n)$ faster.

Concerning the computation of single distances, because of my prehistoric GPU, the performance speed-up is not as impressive as in the paper, but is still around x150 faster. It is obviously highly dependent on the complexity of the neural network encoder used.

Wasserstein Barycenters I then make use of the equation (1) to produce the Wasserstein population mean of each digit. I compare them to the simple Barycenter taken in euclidean space given by $x = \frac{1}{n}\sum_{i=1}^{n} x_i$. The results are shows Figure 4.

I also show digit interpolation results Figure 5. This proves how once the DWE is learnt, it is really easy to compute shape interpolations.

Principal Geodesic Analysis Similarly to the Wasserstein Barycenter analysis, I compare results between the classical euclidean framework of PCA and the Wasserstein distance based PGA. I compute variations from the mean digit along the first three vectors. For PCA, images shown are simply $x = \bar{x} + \alpha\sqrt{\lambda_i}v_i$ where $i \in 1, 2, 3$ (first three eigenvectors with corresponding eigenvalue λ_i), $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ (mean digit) and $\alpha \in [-1.5, 1.5]$ (discrete values are $\{-1.5, -1, -0.5, 0, 0.5, 1, 1.5\}$). For PGA, images show are $x = \psi(\overline{\phi(x)} + \alpha\sqrt{\tilde{\lambda}_i})\tilde{v}_i$ where $\overline{\phi(x)} = \frac{1}{n}\sum_{i=1}^{n}\phi(x_i)$ is the mean digit latent vector and $(\tilde{\lambda}, \tilde{v})$ are the eigenvalues-eigenvectors of

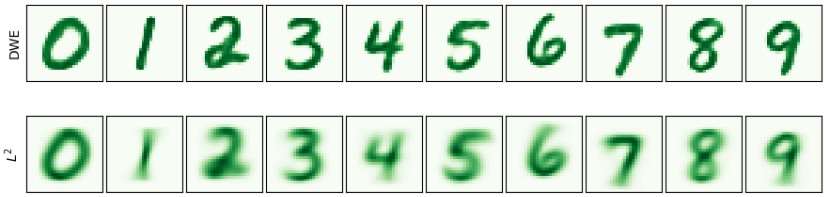


Figure 4: Upper row : Wasserstein barycenters. Lower row : Euclidean barycenters. We can see that Wasserstein barycenters are sharper than the Euclidean ones, which proves that the Wasserstein embedding learnt has a good approximation of the latent space of handwritten digits.

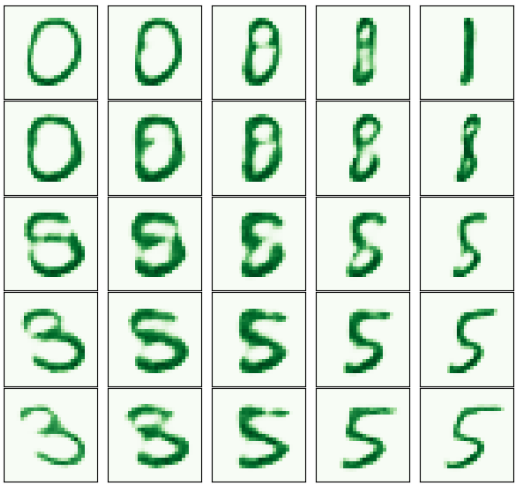
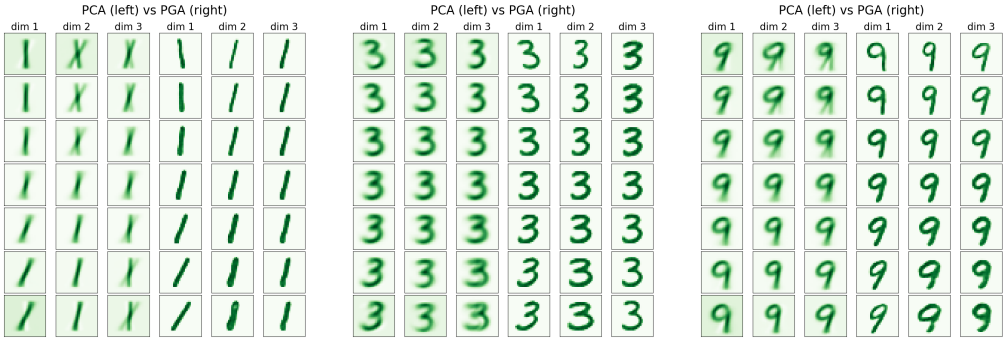


Figure 5: Wasserstein shape interpolation. Corners are reconstructed digits $\psi(\phi(x))$.

the covariance matrix of the samples projected into the latent space (because computing the PGA is equivalent to computing the PCA in the latent space as seen in section (2.3)).



(a) Digit 1 (b) Digit 3 (c) Digit 9

Figure 6: Principal Geodesic Analysis vs PGA for 3 different digits

Results are shown Figure 6 for digits 1, 3 and 9. We can clearly see that similarly to the comparison between Euclidean barycenters and Wasserstein barycenters, images are sharper with the Wasserstein setting. We also see that each geodesic component encodes well semantics. This shows how well the Wasserstein space is for representing data : it encodes more semantic and non linear transformation

of the data. For example, for digit 1, the first dimension of PGA encodes clockwise rotation and the second encodes width of the shape. For digit 3, the first dimension encodes clockwise rotation, the second encodes the width of the whole digit and the third encodes the width of the lines.

Wasserstein k-means Here I present results that are purely mine, in the sense that I did not try to replicate previous results.

In order to increase the probability of having good results, I use a k-means++ (23) initialization scheme, run 10 iterations of the same k-means, and keep the one which got the best error $L = \sum_{j=1}^{K} \sum_{i \in C(j)} \|c_j - x_i\|^2$, where K is the number of clusters, x_i are the data points, c_j the clusters and $C(j)$ the data points assigned to c_j.

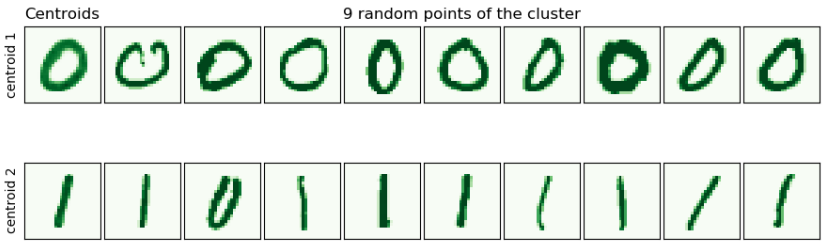


Figure 7: Results of Wasserstein K-means for 2 clusters on digits $0, 1$. Left column : centroids $\psi(c_j)$. Right columns : random samples of each cluster.

I first test clustering with only two digits, 0 and 1. The centroids and random samples taken in their clusters are shown Figure 7. The unsupervised classification accuracy is 99.1%, which is not so surprising since those digits are very different.

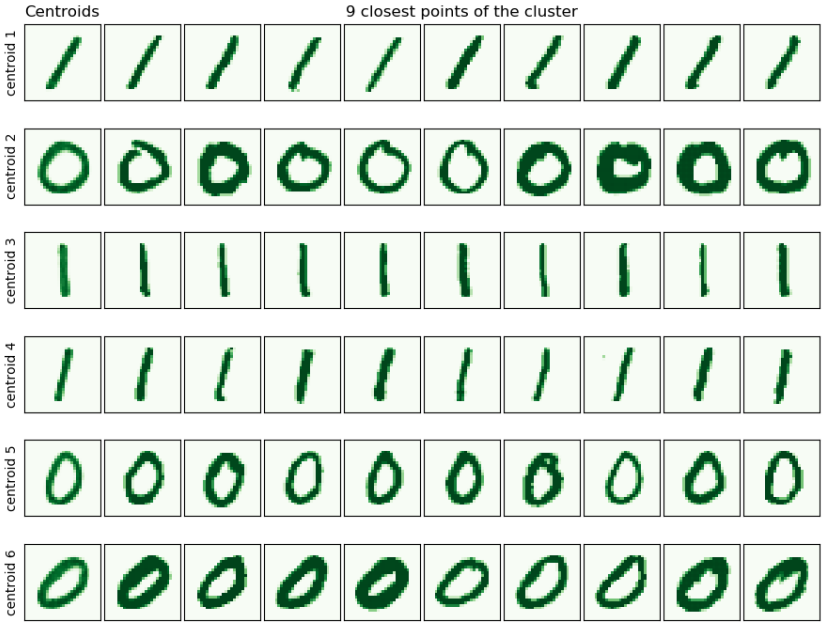


Figure 8: Results of Wasserstein K-means for 6 clusters on digits $0, 1$. Left column : centroids $\psi(c_j)$. Right columns : closest samples of each cluster.

I also test how this Wasserstein k-means (WKM) is able to retrieve different style for different digits. I compute 6 clusters on the $0, 1$ digits dataset. Results are shown Figure 8. We observe that 3 typical styles are identified for each digit. For digit 0, the clustering finds heavy nearly spherical zeros, tighter zeros and tilted to the right zeros. For digit 1, the clustering finds tilted, slightly tilted and

straight ones. A remarkable feat of the algorithm is the quality of the reconstructed centroids $\psi(c_j)$: they are sharp, and express well the semantic of the cluster.

(a) $K = 3$

(b) $K = 5$

(c) $K = 8$

Figure 9: Centroids results for WKM for different K on digits $0, 1, 2, 3, 4$

In a third experiment, I show a limit of the algorithm : for the 5 digits $0, 1, 2, 3, 4$, if I run K-means with $K = 5$, WKM is unable to retrieve the digit 2, which is kind of merged with the digit 3. The algorithm favors clustering two different versions of 1, which to us are semantically similar but to him are two different distributions. When $K = 8$, however, the digit 2 is retrieved. I plot Figure 9 the centroids obtained for different values of K. For $K = 3$, the first two centroids are a mix of $0, 2, 3$ and 4 while the last contains only 1.

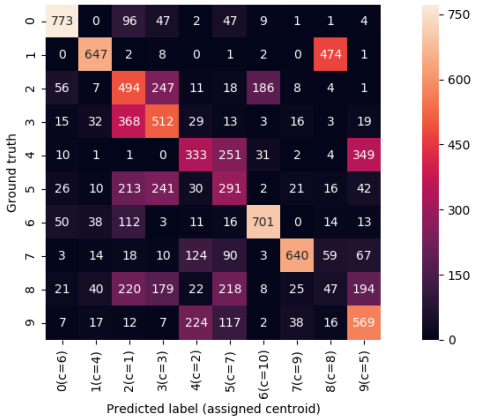


Figure 10: Confusion matrix of the unsupervised classification for the 10 digits.

In a final experiment, I run the algorithm on all 10 digits for $K = 10$. My unsupervised classification accuracy (since clusters are unlabeled, I find the labeling permutation that maximizes accuracy) is 50.1%. I plot the confusion matrix Figure 10. Centroids and random samples of each cluster are shown Figure 12. We can see that 1 and 0 are approximately recovered, but other clusters are mixes of digits, but with a tendency to favor one digit. For example, centroid 5 is a majority of 9 with a big minority of 4 and some 8. I finally compare WKM with K-means (in the image space). In terms of running time, on a single CPU, once projections are computed, WKM runs at 1.9 iterations/s

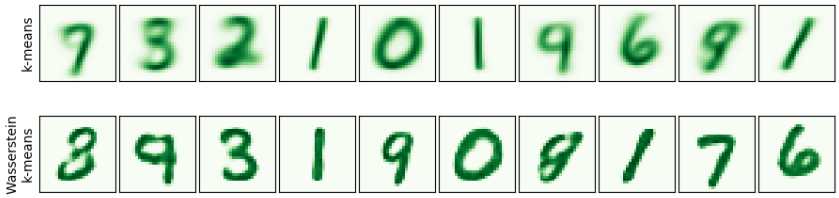


Figure 11: Centroids for WKM and K-means

Figure 12: Results of Wasserstein K-means for 10 clusters on digits $0, ..., 9$. Left column : centroids $\psi(c_j)$. Right columns : random samples of each cluster.

(K-means is usually run with multiple different initializations) which is faster than K-means (1.4 iterations/s). I compare centroids for both methods Figure 11. Centroids for K-means are noisy, the semantic is unclear, contrary to WKM, for which they are sharp and express semantic. However, the semantic is sometimes hard to interpret, as centroids may be a Wasserstein barycenter of different digits.

5 Conclusion

This report finds the conclusion of the original paper to be coherent. The computation of the Wasserstein distance, with the Deep Wasserstein Embedding architecture, can be made very fast. This could be particularly interesting in online applications that can manage a long learning phase once and then offer very fast services. This method also allows the use of different data mining tasks at a low computational price. Two major drawbacks present themselves : the lack of theoretical results for such Wasserstein embeddings, and the lack of general principles for finding the complexity a neural network encoder has to have in order to capture the Wasserstein distance.

References

[1] N. Courty, R. Flamary, and M. Ducoffe, "Learning Wasserstein Embeddings," in *ICLR 2018 - 6th International Conference on Learning Representations*, (Vancouver, Canada), pp. 1–13, Apr. 2018.

[2] C. Villani, *Optimal transport: old and new*, vol. 338. Springer Science & Business Media, 2008.

[3] P. K. Agarwal and R. Sharathkumar, "Approximation algorithms for bipartite matching with metric and geometric costs," in *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pp. 555–564, ACM, 2014.

[4] J. Altschuler, F. Bach, A. Rudi, and J. Weed, "Approximating the quadratic transportation metric in near-linear time," *arXiv preprint arXiv:1810.10046*, 2018.

[5] J. Matoušek, "Lecture notes on metric embeddings," tech. rep., Technical report, ETH Zürich, 2013.

[6] A. Andoni, A. Naor, and O. Neiman, "Impossibility of sketching of the 3d transportation metric with quadratic cost," in *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[7] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[8] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems 30* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), pp. 5767–5777, Curran Associates, Inc., 2017.

[9] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf, "Wasserstein auto-encoders," *arXiv preprint arXiv:1711.01558*, 2017.

[10] D. Zhu, P. Cui, D. Wang, and W. Zhu, "Deep variational network embedding in wasserstein space," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2827–2836, ACM, 2018.

[11] F. De Goes, K. Breeden, V. Ostromoukhov, and M. Desbrun, "Blue noise through optimal transport," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, p. 171, 2012.

[12] N. Bonneel, J. Rabin, G. Peyré, and H. Pfister, "Sliced and radon wasserstein barycenters of measures," *Journal of Mathematical Imaging and Vision*, vol. 51, no. 1, pp. 22–45, 2015.

[13] N. Bonneel, M. Van De Panne, S. Paris, and W. Heidrich, "Displacement interpolation using lagrangian mass transport," in *ACM Transactions on Graphics (TOG)*, vol. 30, p. 158, ACM, 2011.

[14] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[15] M. Cuturi and A. Doucet, "Fast computation of wasserstein barycenters," in *International Conference on Machine Learning*, pp. 685–693, 2014.

[16] M. Agueh and G. Carlier, "Barycenters in the wasserstein space," *SIAM Journal on Mathematical Analysis*, vol. 43, no. 2, pp. 904–924, 2011.

[17] J. Bigot, R. Gouet, T. Klein, A. López, *et al.*, "Geodesic pca in the wasserstein space by convex pca," in *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, vol. 53, pp. 1–26, Institut Henri Poincaré, 2017.

[18] P. T. Fletcher, C. Lu, S. M. Pizer, and S. Joshi, "Principal geodesic analysis for the study of nonlinear statistics of shape," *IEEE transactions on medical imaging*, vol. 23, no. 8, pp. 995–1005, 2004.

[19] G. Peyré, M. Cuturi, *et al.*, "Computational optimal transport," *Foundations and Trends® in Machine Learning*, vol. 11, no. 5-6, pp. 355–607, 2019.

[20] Q. Mérigot, A. Delalande, and F. Chazal, "Quantitative stability of optimal transport maps and linearization of the 2-wasserstein space," *arXiv preprint arXiv:1910.05954*, 2019.

[21] R. Flamary and N. Courty, "Pot python optimal transport library," 2017.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[23] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035, Society for Industrial and Applied Mathematics, 2007.