Optimal transport mapping via input convex neural networks

Computational Optimal Transport

Lucas Versini

Abstract

Optimal transport (OT) is a mathematical framework that defines the optimal way to transfer mass between two probability distributions while minimizing a transport cost. Based on the formulations of Monge and Kantorovich, OT has found various applications in machine learning, including generative modeling, domain adaptation, and fairness. However, classical OT approaches often struggle with scalability, especially when dealing with high-dimensional data, and do not naturally produce computationally efficient mappings between distributions.

[10] builds on the dual formulation of the Kantorovich problem. By reformulating this dual problem into a min-max optimization task over pairs of convex functions, the authors introduce a framework where Input Convex Neural Networks (ICNNs) are used to parameterize these convex functions. ICNNs are well-suited for this task as they provide a flexible yet principled way to approximate convex functions while preserving computational efficiency.

In this report, we investigate the efficiency of this method, while showing its limitations.

1 Introduction

Optimal transport (OT) provides a framework for studying how to "transport" mass in an optimal way. Formally, given two probability distributions μ and ν, defined on spaces $\mathcal{X}$ and $\mathcal{Y}$ respectively, the OT problem seeks a mapping $T : \mathcal{X} \to \mathcal{Y}$ such that $T_{\#}\mu = \nu$ (i.e., the pushforward of μ under T is equal to ν), while minimizing a transport cost:

$$\min_{T|T_{\#}\mu=\nu} \int_{\mathcal{X}} c(x, T(x)) d\mu(x), \tag{1}$$

where $c : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is a cost function, often taken as the quadratic cost $c(x, y) = \frac{1}{2}\|x-y\|^2$ when $\mathcal{X} = \mathcal{Y} = \mathbb{R}^d$. Note that throughout [10], the authors use this cost c, with the constant $\frac{1}{2}$; we shall therefore do it as well for the theoretical part developed later on.

Monge's formulation (1) is the initial formulation of OT, and requires a mapping T that satisfies $T_{\#}\mu = \nu$. The issue with this formulation is that there may not exist such a mapping T; and even if such mappings exist, nothing guarantees that the minimum in (1) is actually reached.

This is essentially why Kantorovitch's relaxation is used. The idea is to consider couplings $\gamma \in \Pi(\mu, \nu)$, where $\Pi(\mu, \nu)$ denotes the set of joint distributions with marginals μ and ν. The objective then becomes:

$$\min_{\gamma \in \Pi(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{Y}} c(x, y) d\gamma(x, y). \tag{2}$$

Assuming that at least one of the two densities μ or ν has a density with respect to the Lebesgue measure, then the quadratic cost OT problem admits a solution γ^* via Brenier's

theorem, which states that the OT map T is the gradient of a convex potential ϕ, i.e., $T(x) = \nabla\phi(x)$. Moreover, if μ has a density with respect to the Lebesgue measure, then $\gamma^* = (\mathrm{Id}, T)_{\#}\,\mu$.

While OT has attractive theoretical properties, computing OT maps is challenging, especially in high dimensions, where traditional numerical methods scale poorly as the dimensionality or data size increases. Moreover, these methods often provide couplings γ instead of explicit mappings T, which limits their applicability in scenarios requiring functional mappings.

Several approaches have been proposed to address the computational bottlenecks of OT:

- [5] introduced entropic regularization (and the use of Sinkhorn algorithm) to make OT computations scalable by solving a regularized version of the Kantorovich problem. Although efficient, the method introduces bias and does not directly produce mappings T.
- In [11], the authors use the dual of Kantorovitch's problem $\sup_{f,g} \mathbb{E}_{(X,Y)\sim\mu\otimes\nu}\left[f(X) + g(Y)\right]$ subject to $\forall x, y, f(x) + g(y) \leq c(x, y)$, and replace the constraint by a strictly concave regularization term:

$$\sup_{f,g} \mathbb{E}_{(X,Y)\sim\mu\otimes\nu}\left[f(X) + g(Y) + F_\varepsilon\left(f(X), g(Y)\right)\right].$$

  Then, they solve this optimization problem using deep neural networks to represent f and g. Of course, whenever $\varepsilon > 0$ the solution to this problem will not be optimal for the dual of Kantorovitch's problem, so in practice the problem has to be solved for a decreasing sequence of ε, which can be costly.
- In [9], the authors use Generative Adversarial Networks (GANs) to approximate $W_2^2(\mu, \nu)$.

Neural networks have emerged as a promising tool for approximating OT maps due to their capacity to model complex functions, and the many efficient implementations that exist. However, enforcing the structural constraints (e.g., convexity) is sometimes nontrivial. Naively parameterized networks may violate these constraints, leading to invalid OT maps. The primary contribution of [10] is the use of *Input Convex Neural Networks* (ICNNs) [2].

Indeed, the dual of (2) can be reformulated as:

$$W_2^2(\mu, \nu) = \sup_{\substack{f \text{ convex} \\ f^* \in L^1(\nu)}} \inf_{g \text{ convex}} \mathcal{V}_{\mu,\nu}(f, g) + C_{\mu,\nu}, \tag{3}$$

where $C_{\mu,\nu}$ only depends on μ and ν, $\mathcal{V}_{\mu,\nu}(f, g) = -\mathbb{E}_\mu\left[f(X)\right] - \mathbb{E}_\nu\left[\langle Y, \nabla g(Y)\rangle - f\left(\nabla g(Y)\right)\right]$, and $f^*(y) = \sup_x\left(\langle x, y\rangle - f(x)\right)$.
Then, ICNNs are used to parameterize the convex functions involved in (3). Therefore this approach does not directly approximate the transport map $T(x)$ or the Brenier potential $\phi(x)$, and instead reformulates the OT problem into a min-max optimization problem over two convex potentials.
This framework is computationally efficient and well-suited for high-dimensional datasets, addressing a key limitation of classical OT methods that struggle in such settings.
By leveraging ICNNs to parameterize convex functions in the dual Kantorovich formulation, this method guarantees convexity, which is a theoretical requirement for OT theory and Brenier's theorem.

2 Convex neural networks and optimal transport

2.1 Problem formulation

We first briefly explain how (7) can be deduced from (2).

As said previously, we consider Kantorovitch's relaxation of Monge's problem in the case $c(x,y) = \frac{1}{2}\|x-y\|_2^2$, which admits the following dual:

$$W_2^2(\mu,\nu) = \sup_{(f,g)\in\Phi_c} \left(\mathbb{E}_\mu\left[f(X)\right] + \mathbb{E}_\nu\left[g(Y)\right]\right), \tag{4}$$

where the set of constraints is:

$$\Phi_c := \left\{(f,g) \in L^1(\mu)\times L^1(\nu) \mid f(x)+g(y) \leq \frac{1}{2}\|x-y\|_2^2 \ \forall (x,y)\, d\mu\otimes d\nu \text{ a.e.}\right\}.$$

Problem (4) is not easy, since it involves two functions f, g which have to satisfy a joint constraint. However, if we replace f by $\frac{1}{2}\|\cdot\|_2^2 - f(\cdot)$ and g by $\frac{1}{2}\|\cdot\|_2^2 - g(\cdot)$, then (4) becomes

$$W_2^2(\mu,\nu) = C_{\mu,\nu} - \inf_{(f,g)\in\tilde{\Phi}_c} \left(\mathbb{E}_\mu\left[f(X)\right] + \mathbb{E}_\nu\left[g(Y)\right]\right), \tag{5}$$

where $C_{\mu,\nu} = \frac{1}{2}\mathbb{E}\left[\|X\|_2^2 + \|Y\|_2^2\right]$, and where the set of constraints is
$\tilde{\Phi}_c := \left\{(f,g) \in L^1(\mu)\times L^1(\nu) \mid f(x)+g(y) \geq \langle x,y\rangle \ \forall (x,y)\, d\mu\otimes d\nu \text{ a.e.}\right\}$.

This change of variables shows that, in (5), an optimal couple (f,g) necessarily satisfies $g = f^*$, where f^* is the conjugate function of f defined by $f^*(y) = \sup_x \left(\langle x,y\rangle - f(x)\right)$, which leads to the following problem

$$W_2^2(\mu,\nu) = C_{\mu,\nu} - \inf_{f \text{ convex}} \left(\mathbb{E}_\mu\left[f(X)\right] + \mathbb{E}_\nu\left[f^*(Y)\right]\right). \tag{6}$$

The fact that we can restrict ourselves to convex functions f is not obvious, and is proven in [12].
However, the presence of f^* does not facilitate the minimization, so instead, we reformulate the previous problem (6) into (3), which we re-write here for convenience:

$$W_2^2(\mu,\nu) = \sup_{\substack{f \text{ convex} \\ f^*\in L^1(\nu)}} \inf_{g \text{ convex}} \mathcal{V}_{\mu,\nu}(f,g) + C_{\mu,\nu}. \tag{7}$$

with $\mathcal{V}_{\mu,\nu}(f,g) = -\mathbb{E}_\mu\left[f(X)\right] - \mathbb{E}_\nu\left[\langle Y, \nabla g(Y)\rangle - f\left(\nabla g(Y)\right)\right]$ and $C_{\mu,\nu} = \frac{1}{2}\mathbb{E}_{X\sim\mu, Y\sim\nu}\left[\|X\|_2^2 + \|Y\|_2^2\right]$.
Moreover, if μ has a density with respect to Lebesgue measure, then the optimal transport map from μ to ν is given by ∇f, where f is optimal for (7).

The question is to know how to optimize in practice the objective in (7). One could use neural networks for f and g, but then there would be no guarantee that both f and g are convex functions.
This is why the authors use Input Convex Neural Networks.

2.2 Input Convex Neural Networks (ICNNs)

An Input Convex Neural Network (ICNN) is a special type of neural networks designed to parameterize convex functions. To construct an ICNN, the following components are required:

- Weight matrices $\{W_l\}_{l=0}^{L-1}$ (with $W_0 = 0$), and $\{A_l\}_{l=0}^{L-1}$;
- Bias vectors $\{b_l\}_{l=0}^{L-1}$;
- Activation functions $\{\sigma_l\}_{l=0}^{L-1}$.

The network structure is then defined as:

$$z_{l+1} = \sigma_l(W_l z_l + A_l x + b_l),$$

where z_l represents the intermediate layer outputs, and $z_0 = x$ is the input. If we denote by $\theta = \{W, A, b\}$ the collection of parameters of the network, then the final network output is given by:

$$f_\theta(x) = z_L.$$

To ensure convexity of $f_\theta(x)$ with respect to the input x, the following constraints are imposed:

- All coefficients of the weight matrices W_l are non-negative.
- The activation functions σ_l are convex for $l = 0, \dots, L-1$.
- The activation functions σ_l are non-decreasing for $l = 1, \dots, L-1$.

It is straightforward to verify that the non-negative constraints on W_l and the use of convex and monotonic activation functions ensure that convexity is preserved throughout the network, which means that the function $x \mapsto f_\theta(x)$ is convex.

If we denote by $\mathrm{ICNN}(\mathbb{R}^d)$ the set of ICNNs over $\mathbb{R}^d$, then we can approximate the problem (7) by:

$$\sup_{f \in \mathrm{ICNN}(\mathbb{R}^d)} \inf_{g \in \mathrm{ICNN}(\mathbb{R}^d)} \mathcal{V}_{\mu,\nu}(f, g) + C_{\mu,\nu}. \tag{8}$$

Of course, the question is to know if the class of ICNNs is wide enough to approximate optimal functions f, g in (7).

A result established in [4] shows that, on a compact domain $\mathcal{D} \subset \mathbb{R}^n$, any convex Lipschitz function can be approximated uniformly by ICNNs to arbitrary precision.

However, it is important to note that for a given target function, there is no guarantee that the optimal solutions f and g of the original problem can be *exactly represented* by ICNNs. Moreover, like with usual neural networks, the width and depth required to approximate a given function can be very large in practice, which means that for a given architecture (that is, for a given depth and a given width), ICNNs may be unable to properly approximate the optimal functions f, g. Nevertheless, ICNNs provide an efficient and practical surrogate for approximating the solution.

Then, the advantage of ICNNs is that we can essentially optimize their parameters as is usually done for neural networks, using classical Deep Learning libraries such as PyTorch or TensorFlow.

2.3 Training and optimization

Therefore we now aim at solving (4.4) using samples from μ and ν. This is what Algorithm 1 does.

Algorithm 1 to solve (4.4)

Require: Samples from μ and ν, batch size M, generator iterations K, total iterations T

for $t = 1$ to T **do**
  for $k = 1$ to K **do**
    Sample a batch $\{X_i\}_{i=1}^M \sim \mu$, $\{Y_i\}_{i=1}^M \sim \nu$
    Update g to minimize (9) using Adam
  end for
  Sample a batch $\{X_i\}_{i=1}^M \sim \mu$, $\{Y_i\}_{i=1}^M \sim \nu$
  Update f to maximize (9) using Adam
  $w \leftarrow \max(w, 0)$ for all coefficients w of W_l in θ_f, $l = 1, \dots, L-1$
end for

More precisely, we define:

$$J(\theta_f, \theta_g) = \frac{1}{M}\sum_{i=1}^{M}\left(f\left(\nabla g(Y_i)\right) - \langle Y_i, \nabla g(Y_i)\rangle - f(X_i)\right) + \lambda \sum_{W_l \in \theta_g} \|\max(-W_l, 0)\|_F^2, \quad (9)$$

which is the quantity that is optimized in lines (5) and (7) of Algorithm 1, using Adam [7]. Note that we do not require g to be an ICNN: indeed, in (7), we can drop the requirement g convex, because g is automatically equal to f^*, and is thus convex. However, the authors of [10] found that penalizing negative coefficients of weight matrices of g leads to better results.

2.4 Theoretical guarantees

[10] also establishes some theoretical guarantees.
Let

$$\varepsilon_1(f, g) = \mathcal{V}_{\mu,\nu}(f, g) - \inf_{\tilde{g} \text{ convex}} \mathcal{V}_{\mu,\nu}(f, \tilde{g})$$

and

$$\varepsilon_2(f) = \sup_{\tilde{f} \text{ convex}} \inf_{\tilde{g} \text{ convex}} \mathcal{V}_{\mu,\nu}(\tilde{f}, \tilde{g}) - \inf_{\tilde{g} \text{ convex}} \mathcal{V}_{\mu,\nu}(f, \tilde{g}).$$

Then:

Theorem: Assume ν has a density with respect to Lebesgue measure, and let ∇g^* be the optimal transport from ν to μ.
Then for any (f, g) such that f is α-strongly convex, the following holds:

$$\|\nabla g - \nabla g^*\|_{L^2(\nu)}^2 \leq \frac{2}{\alpha}\left(\varepsilon_1(f, g) + \varepsilon_2(f)\right).$$

Basically, $\varepsilon_1(f, g)$ measures the optimality of (f, g), and $\varepsilon_2(f)$ measures the optimality of f. Under some assumptions, the error between ∇g and the true OT map can be upper-bounded by a quantity depending on $\varepsilon_1(f, g)$ and $\varepsilon_2(f)$.
So in a way, the error made by g as an OT map ($\|\nabla g - \nabla g^*\|_{L^2(\nu)}^2$) can be upper-bounded by the error of g as a solution to (7) ($\inf_f \frac{2}{\alpha_f}\left(\varepsilon_1(f, g) + \varepsilon_2(f)\right)$).

3 Other ideas

To scale down computations, and maybe have a better understanding, we can restrict ourselves to easier models.

3.1 Polynomials

Instead of approximating f and g with ICNNs, we could try and use convex polynomials.
Let us first focus on the one-dimensional case. $P \in \mathbb{R}[X]$ seen as a function from $\mathbb{R}$ to $\mathbb{R}$ is convex if and only if $\forall x \in \mathbb{R}, P''(x) \geq 0$. And it is known that:

Lemma: $Q \in \mathbb{R}[X]$ satisfies $\forall x \in \mathbb{R}, Q(x) \geq 0$ if and only if there exist $A, B \in \mathbb{R}[X]$ such that $Q = A^2 + B^2$.

So P is convex if and only if there exist $A, B \in \mathbb{R}[X]$ such that $P'' = A^2 + B^2$.
If we define $I_2 : \mathbb{R}[X] \to \mathbb{R}[X]$ such that for each Q, $I_2(Q)$ is the only polynomial satisying $I_2(Q)'' = Q, I_2(Q)(0) = I_2(Q)'(0) = 0$, then what we can do in practice is not directly optimize over P (which is a constrained problem), but over $A \in \mathbb{R}[X], B \in \mathbb{R}[X], a \in \mathbb{R}, b \in \mathbb{R}$ and set $P = I_2(A^2 + B^2) + aX + b$.

Note that in dimension $d \geq 2$, there is no easy characterization of convex polynomials.

So instead, we can simply use a sufficient condition, based on what was done in the one-dimensional case:

Lemma: If $f : \mathbb{R}^d \to \mathbb{R}$ is twice differentiable, and is such that there exists a polynomial function $S : \mathbb{R}^d \to \mathbb{R}^{d \times d}$ that satisfies $\forall x \in \mathbb{R}^d, \nabla^2 f(x) = S(x)^T S(x)$, then f is a convex polynomial function.

By "polynomial function", we mean that each coefficient of $S(x)$ is polynomial in the coefficients of the variable x.

In the above lemma, the fact that f is polynomial is simply obtained by integrating its hessian $\nabla^2 f(x)$, which is polynomial in x; the fact that f is convex comes from $\nabla^2 f(x) = S(x)^T S(x) \succeq 0$.

Now, in general (that is, for general degrees and general number of variables), a convex polynomial cannot be written as above. However, we may still try to use such polynomials to approximate f and g, hoping that the class of these polynomials is wide enough.
In fact, such polynomials have been studied. For instance, [1] focuses on giving a sufficient condition for a multivariate polynomial to be non-negative (which is closely related to our own problem), and [8] shows the approximation capability of such polynomials.
Note that if P is of degree $2m$, in d variables, then we can look for $S(x) \in \mathbb{R}^{d \times d}$, where each coefficient of S is a scalar polynomial of degree d in n variables.

More simply, one can also consider polynomials $x \mapsto \frac{1}{2} x^T A x$ with $A \succeq 0$. Since $\nabla(\frac{1}{2} x^T A x) = Ax$, we know that such polynomials will be sufficient for OT between Gaussian distributions.

3.2 One-hidden-layer networks

To speed up computations, and maybe have a better understanding, we can also restrict ourselves to neural networks with only one hidden layer.
We will do this in section 4, and compare both the run time and the results.
A good reference to understand more in details one-hidden-layer neural networks is [3], where the optimization problem is cast as a convex problem (which does not have much to do with the convexity we consider here).

4 Experiments

Here, we experiment on the previous method on various datasets to assess its efficiency.
We re-implemented the code, as the code from [10] was quite lengthy, and contained improvements that led to better results, but also to less interpretability.

The base code can be found on `https://github.com/lucas-versini/OT_ICNN`.

For the following experiments, unless specified otherwise, we use $T = 10,000$, $K = 10$ and $M = 1,000$ using the notations from Algorithm 1. The ICNNs are composed of 4 hidden layers (unless otherwise specified), each with 128 neurons, and we use the `Leaky ReLU` activation function, with parameter 0.2. We focus on Wasserstein-2 distance $W_2^2(\mu, \nu)$ (this time, without the factor $\frac{1}{2}$).

Also note that, unless specified otherwise, all the following experiments took about 10 minutes to run.

4.1 First experiments, comparison with POT

First, we run basic experiments on several datasets, and compare the results to the ones obtained with POT [6], that is by solving a linear program.

However, note that the comparison between the ICNN results and POT's results is biased: the ICNN model is trained on a large training dataset and then evaluated on the test set, whereas POT is only evaluated on the test dataset (and does not use the training dataset). It is not possible to use all the data with POT: for POT, all the data needs to be processed at the same time, whereas the ICNN method works with batches.

So, to compare the results, we split the entire data into batches, and for each batch, we use POT to compute the distance. Then, we look at the mean m of all these distances.
Below, we plot the result obtained by the ICNNs (`ICNN`), and the result obtained with POT (`POT`).

4.1.1 Gaussian distributions

Several results with Gaussian distributions and Gaussian mixtures are shown in Fig. 1, Fig. 2, Fig. 3.
Note that when plotting the estimation of $W_2^2(\mu, \nu)$, we do not show the results obtained for a small number of iterations, as the ICNNs' estimation can be very far away from the true value (typically, the estimated value can be negative with large absolute value).

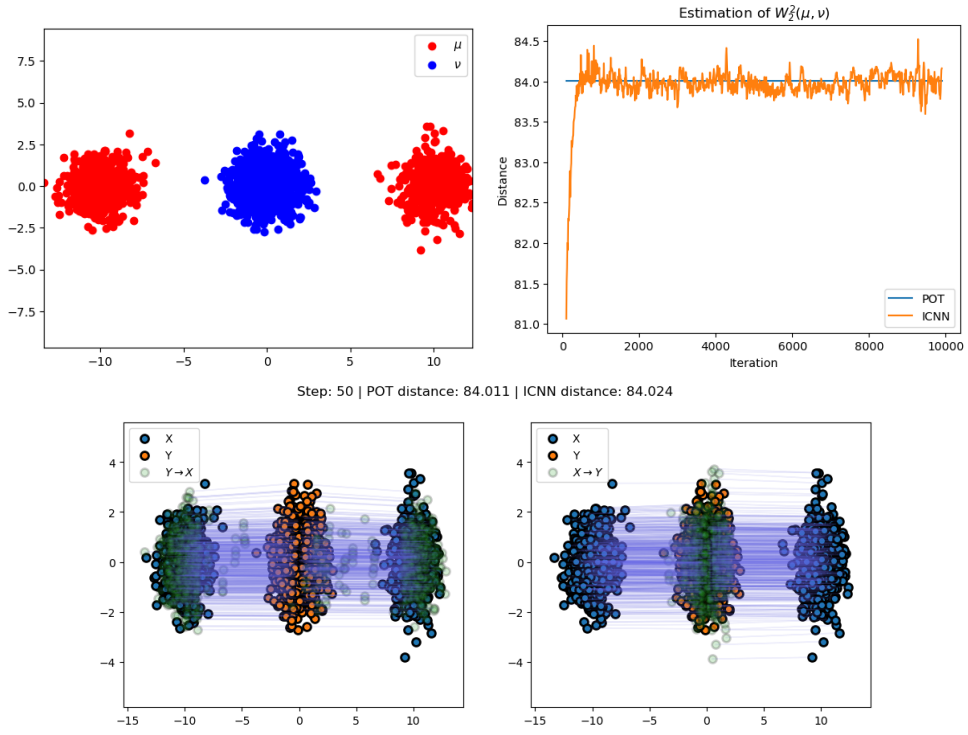


Figure 1: μ: Gaussian mixture with 2 components, ν: Gaussian. Estimated distance and optimal transport.

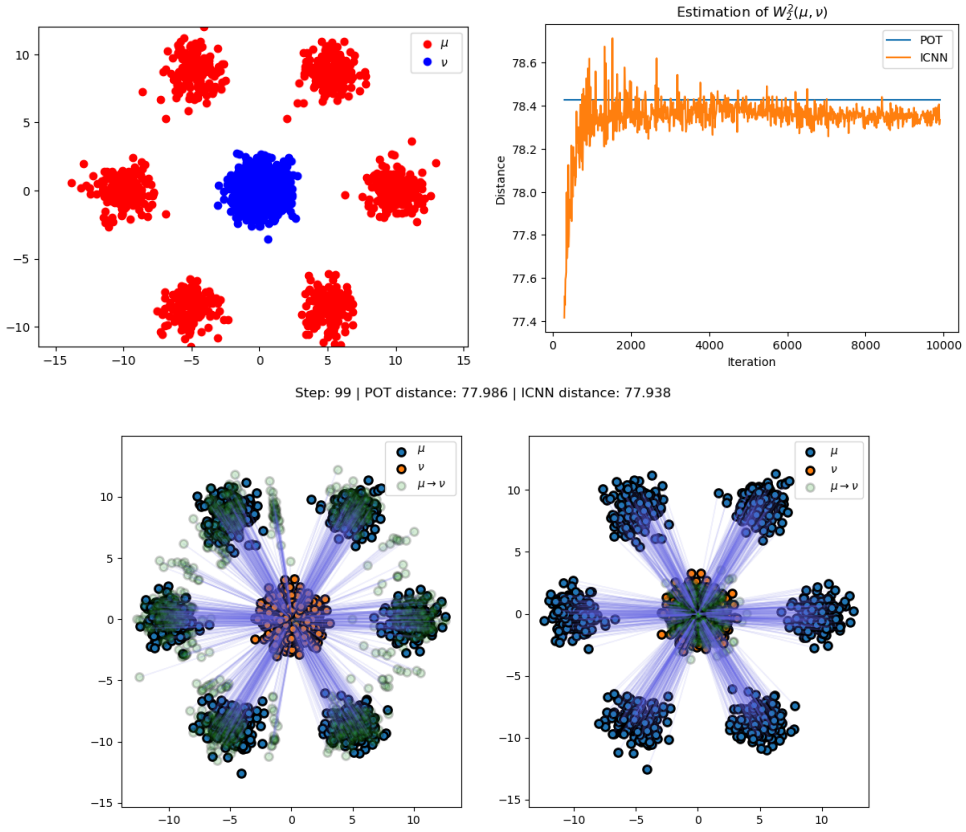
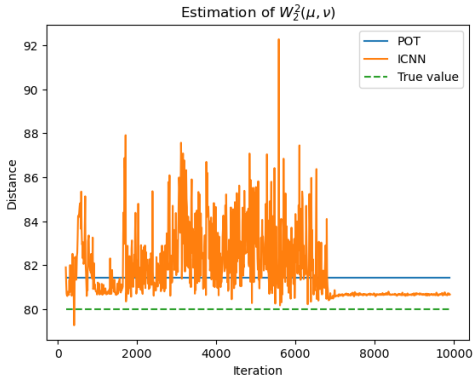


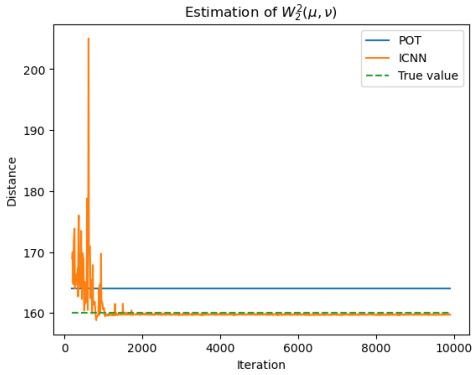
Figure 2: μ: Gaussian mixture with 6 components, ν: Gaussian. Estimated distance and optimal transport.

We observe that the values obtained with POT and with ICNNs can be quite different, and that very often, the ICNNs' result is closer to the true value.
However, as can be seen in Fig. 3 in $\mathbb{R}^5$ or $\mathbb{R}^{20}$ (where the covariance matrices are I_n, and the means are 0_n and $(4,\dots,4)$), the estimation of $W_2^2(\mu,\nu)$ by the ICNNs can sometimes stabilize at the wrong value after some iterations. This is not surprising: just like neural networks for other tasks, a local minimum (or here, a local minimum-maximum) can be encountered. So it is necessary to carefully tune some hyperparameters, including the learning rate.

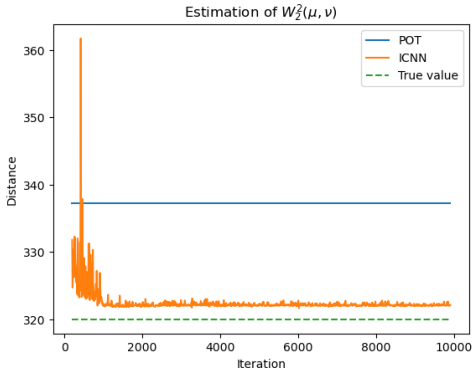
Fig 4 shows the results obtained on another 2D dataset. Though the estimated distance seems to oscillate, notice that this happens on a rather small scale.



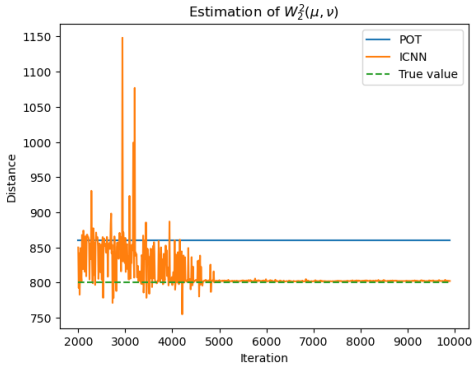
(a) Gaussian distributions in $\mathbb{R}^5$



(b) Gaussian distributions in $\mathbb{R}^{10}$

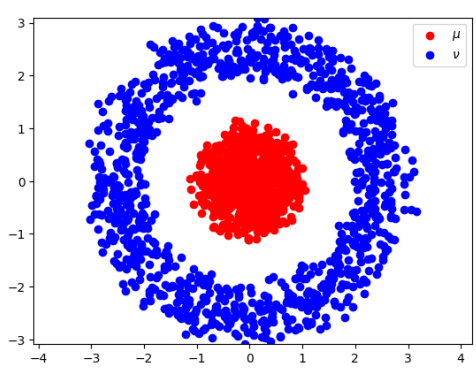


(c) Gaussian distributions in $\mathbb{R}^{20}$



(d) Gaussian distributions in $\mathbb{R}^{50}$

Figure 3: Translated Gaussian distributions in higher dimensions.



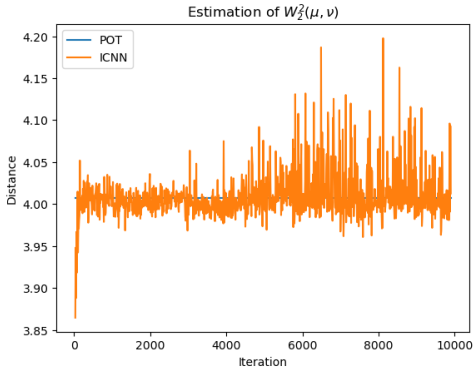


Figure 4: Illustration on another dataset

4.1.2 Voronoi partitions

Here, we can consider a toy example where μ has a density with respect to Lebesgue measure, but ν does not.
We work in $\mathbb{R}^2$, set μ the uniform measure on $[-10, 10]^2$, and $\nu = \frac{1}{4}(\delta_{-5,-5} + \delta_{-5,5} + \delta_{5,-5} + \delta_{5,5})$.
Then, we have explicitly (without using the $\frac{1}{2}$ factor in the definition of W_2^2):

$$W_2^2(\mu, \nu) = \frac{1}{10^2} \int\int_{[0,10]^2} \|(x, y) - (5, 5)\|_2^2 \, dxdy = \frac{50}{3} \approx 16.67.$$

The obtained result is shown in Fig. 5.

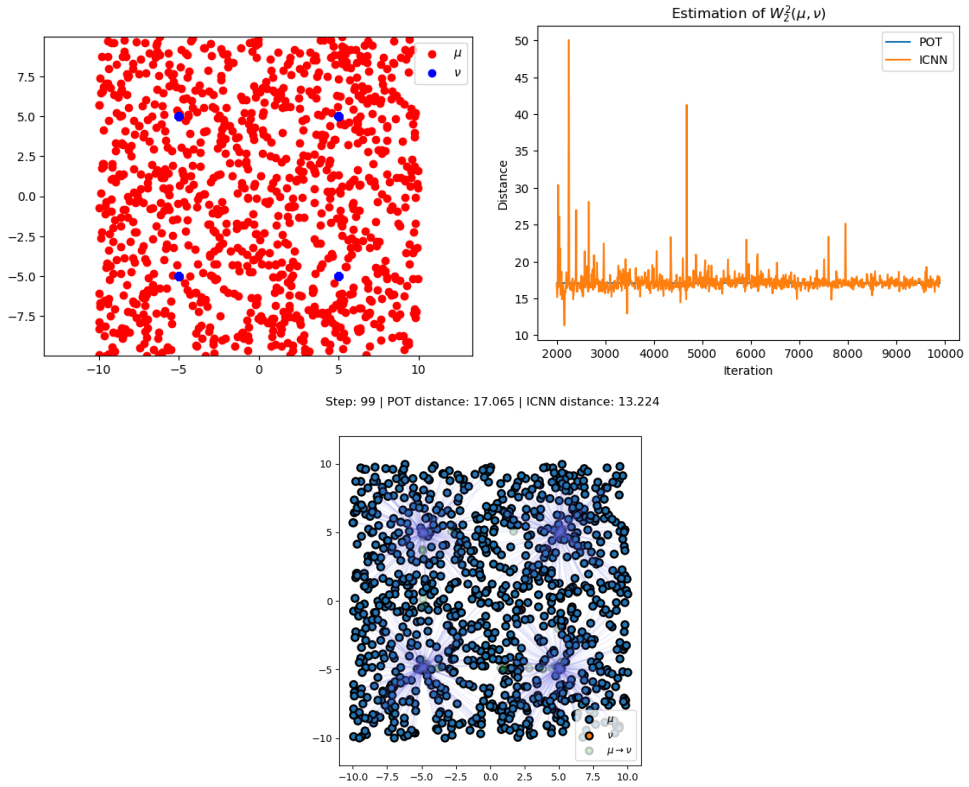


Figure 5: Voronoi configuration

We see that in this case as well the result is quite good, both for the estimated distance and for the estimated map.

4.1.3 μ and ν without density

We can also look at what happens when μ *and* ν do not have a density. For instance, we can take μ uniform on $[0, 1] \times \{0\}$, and ν uniform on $\{0\} \times [0, 1]$.
Then

$$W_2^2(\mu, \nu) = \int_0^1 2t^2 \, dt = \frac{2}{3}.$$

Fig. 6 shows that the ICNNs quickly correctly estimate $W_2^2(\mu, \nu)$. However, after many iterations, we observe oscillations and a divergence. This shows that the results obtained with ICNNs can be very unstable.
Moreover, the transport shown in Fig. 6 is not convincing at all, whereas it corresponds to $4,000$ iterations, when the estimation of $W_2^2(\mu, \nu)$ is accurate. So the distance is well estimated in this case, but the optimal transport is not.

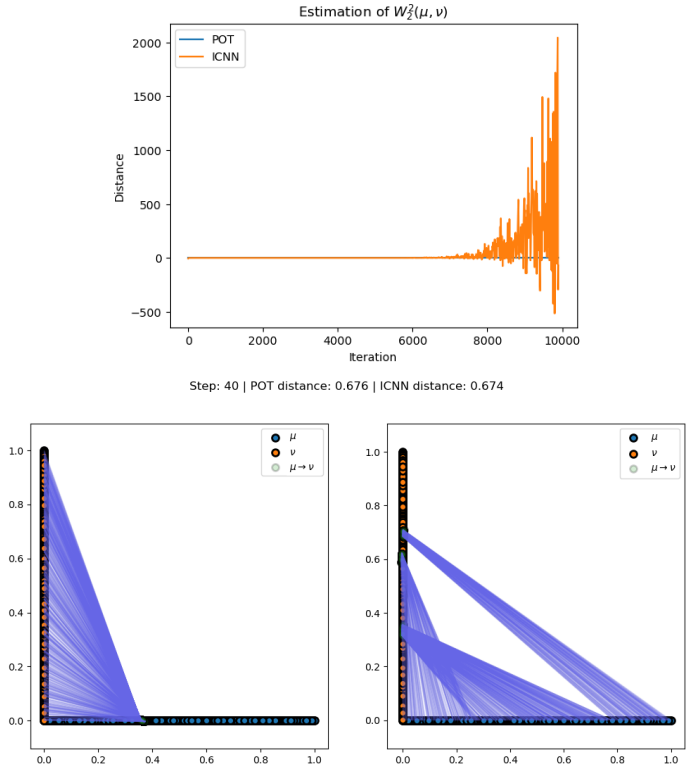


Figure 6: μ and ν without density

4.2 Number of layers

Here, we study the impact of the number of layers. We consider the case of two Gaussian distributions in $\mathbb{R}^5$, and plot the estimation of $W_2^2(\mu, \nu)$ as a function of the number of layers, for a Gaussian mixture in two dimensions. The results are shown in Fig. 7.
We see that when there are not enough layers (1), the estimated distance can be far from the true one.
However, if there are too many layers, then the result is quite unstable: for 20 hidden layers, we observe significant variations even after 6,000 iterations.
As for the time it took to train the models, see Table 1.

Number of layers	1	2	3	5	10	20	50
Time (seconds)	310	335	390	570	950	1740	4050

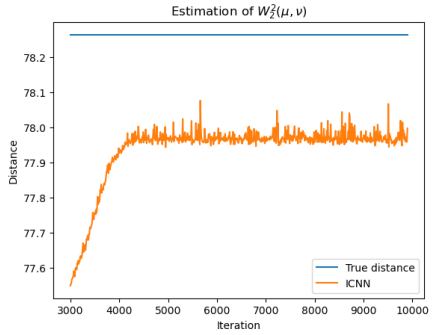
Table 1: Time to train the models for different numbers of layers

So just like with neural networks in usual contexts, the hyperparameters should be tuned very carefully.

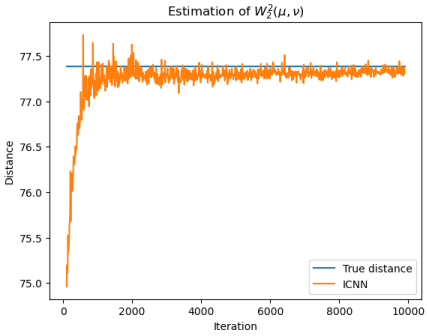
4.3 Should g be convex ?

As said in subsection 2.3, the article [10] does not enforce convexity of g, but instead penalizes non-convexity. Here, we compare three settings:

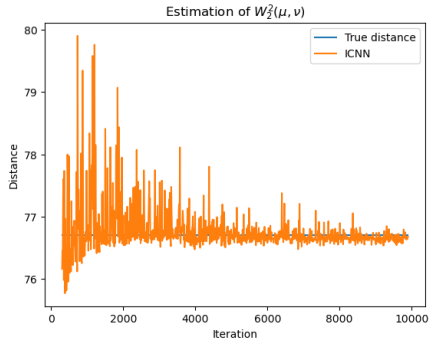
- No constraint nor penalization for g;
- The coefficients of g's weight matrices are constrained to be non-negative;
- We add a regularization term based on the coefficients of g (as in [10]).



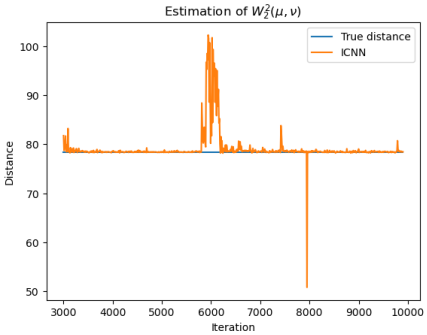
(a) 1 hidden layer



(b) 5 hidden layers



(c) 10 hidden layers



(d) 20 hidden layers

Figure 7: Impact of the number of layers

We use a dataset with μ, ν two Gaussian distributions in dimension 10. Results are shown in Fig. 8.
It appears that making g's coefficients positive leads to instability compared to using only a penalization. Moreover, not using any constraints nor penalization surprisingly leads to a stable result, but biased.
This explains the use of a penalty in [10].

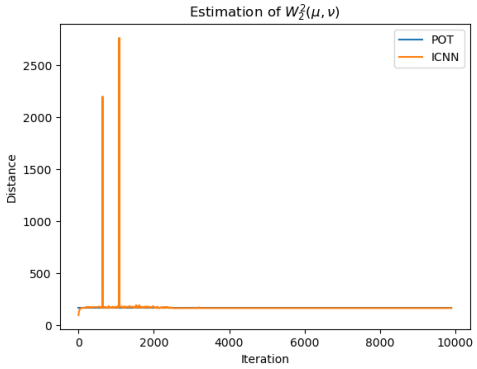
4.4 Using polynomial functions

We also ran experiments with f and g polynomial functions.
The results were not surprising: $W_2^2(\mu,\nu)$ and the optimal transport were very well estimated in the case of Gaussian distributions, which makes sense since the optimal transport is of the form $x \mapsto Ax$.
When using more complex distributions, such as Gaussian mixtures, the polynomials $x \mapsto \frac{1}{2}x^T Ax$ no longer worked, but more general polynomials still led to reasonable results.
However, when using more complex distributions (Voronoi, segments, etc.), the results were far from the true values.

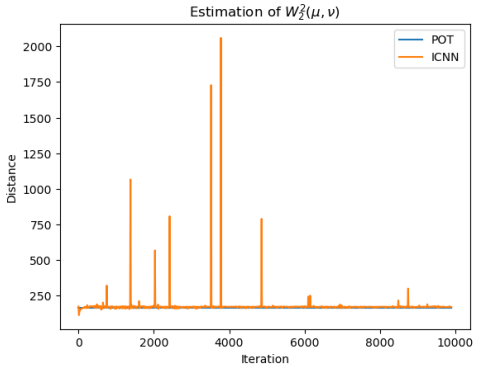
Therefore the approximation capability of ICNNs is crucial to ensure that problems (7) and are close enough.

4.5 Non-convex f and g

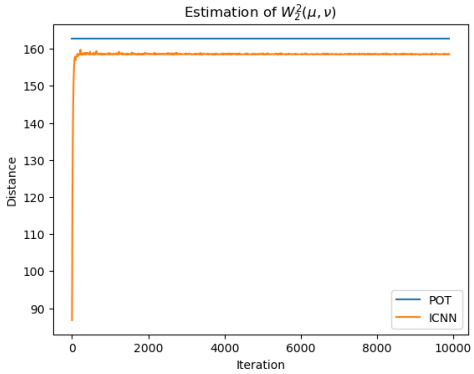
If we do not require the weight matrices of f and g to have non-negative coefficients, then we observed that the estimation of $W_2^2(\mu,\nu)$ keeps oscillating, due to the $\max - \min$ optimization.



(a) Penalty for g's coefficients



(b) Positive coefficients for g



(c) No constraints on g

Figure 8: Impact of enforcing positivity of g's coefficients

5 Conclusion and perspective

The authors of [10] introduce a new method to compute optimal transport between two distributions using sample data. The approach demonstrates convergence to correct solutions in various cases, but is turns out to be sensitive to practical implementation details, such as network architecture, iteration count, and optimization parameters. These aspects are not discussed comprehensively in the paper, but are usual in the field of Deep Learning.

The method frames Kantorovitch problem as a convex min-max optimization task, solvable with ICNNs (or other convex optimization techniques such as polynomials in some specific cases), directly addressing the Kantorovich OT problem without using entropic regularization. It is theoretically suitable for training deep generative models, offering robustness to initialization and the ability to model discontinuous maps.
For simple datasets, classic OT solvers remain more efficient and more accurate, but the proposed method outperforms them in some larger scenarios.

A Connexion with the course

It is quite straightforward to see the link between [10] and the course: Monge problem (1), Kantorovitch relaxation (2), its dual (4), optimal transports and Brenier theorem are key notions in what was presented previously, and were a main part of the MVA course.

References

[1] A. A. Ahmadi and P. A. Parrilo. Sum of squares and polynomial convexity. 2009.

[2] B. Amos, L. Xu, and J. Z. Kolter. Input convex neural networks. In *International conference on machine learning*, pages 146–155. PMLR, 2017.

[3] F. Bach. Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research*, 18(19):1–53, 2017.

[4] Y. Chen, Y. Shi, and B. Zhang. Optimal control via neural networks: A convex approach. 2018.

[5] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

[6] R. Flamary, N. Courty, A. Gramfort, M. Z. Alaya, A. Boisbunon, S. Chambon, L. Chapel, A. Corenflos, K. Fatras, N. Fournier, L. Gautheron, N. T. Gayraud, H. Janati, A. Rakotomamonjy, I. Redko, A. Rolet, A. Schutz, V. Seguy, D. J. Sutherland, R. Tavenard, A. Tong, and T. Vayer. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021.

[7] D. P. Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[8] J. B. Lasserre and T. Netzer. Sos approximations of nonnegative polynomials via simple high degree perturbations. *Mathematische Zeitschrift*, 256(1):99–112, 2007.

[9] J. Leygonie, J. She, A. Almahairi, S. Rajeswar, and A. Courville. Adversarial computation of optimal transport maps. *arXiv preprint arXiv:1906.09691*, 2019.

[10] A. Makkuva, A. Taghvaei, S. Oh, and J. Lee. Optimal transport mapping via input convex neural networks. In *International Conference on Machine Learning*, pages 6672–6681. PMLR, 2020.

[11] V. Seguy, B. B. Damodaran, R. Flamary, N. Courty, A. Rolet, and M. Blondel. Large-scale optimal transport and mapping estimation. *arXiv preprint arXiv:1711.02283*, 2017.

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