Computational Optimal Transport Final Project: Optimal transport mapping via input convex neural networks

Elías Masquil
eliasmasquil@gmail.com

December 2021

1 Abstract

In [6] the authors study the problem of finding the optimal transport mapping between two probability distributions in $\mathbb{R}^d$ from samples. Thanks to the Optimal Transport (OT) framework, this allow us to handle both the discrete and continuous cases.

Kantorovich problem is studied, in the case where both distributions are in $\mathbb{R}^d$ and under Brenier's theorem assumptions. Starting from its classic dual formulation, the authors develop another valid dual which can be solved by stochastic minmax optimization. This optimization problem is in the space of convex functions, therefore the authors propose to leverage recent developments on Input Convex Neural Networks (ICNNs) [1] for solving it. The approximated functions are Kantorovich potentials from which can also be derived the W_2^2 distance and the optimal mapping between distributions.

The main contribution of the work is the development of a novel approach for finding the optimal transport mapping between two distributions, using convex neural networks. This approach can be used to train a generative model. The proposed algorithm offers mainly two advantages respect to other neural networks techniques such as GANs: the learned mappings are robust to different initialization and can also be discontinuous.

2 Introduction

Finding the optimal transport between two distributions P and Q is the core problem of Optimal Transport. In the discrete case, the problem is a linear program which can be solved by using the Simplex algorithm when the dimensions of the problem are not too large. If a faster solution is needed, or the dimensions are too large for using Simplex, the problem can also be solved by using entropic regularization via Sinkhorn algorithm [4].

It's in the case of continuous-continuous distributions or very large number of samples for the discrete case, where the problem is still open.

Let's frame the optimal transport problem with a probabilistic interpretation. We consider the specific case, where the cost of the transport is the quadratic cost. Consider P and Q two probability distributions in $\mathbb{R}^d$. Monge's problem is:

$$\underset{T:T_{\#}Q=P}{\text{minimize}} \frac{1}{2}\mathbb{E}_{X\sim Q}||T(X)-X||^2 \tag{1}$$

Kantorovich's relaxation of Monge's is:

$$W_2^2(P,Q) := \inf_{\pi\in\mathbb{M}_+(X,Y).\pi_1=P,\pi_2=Q} \frac{1}{2}\mathbb{E}_{(X,Y)\sim\pi}||X-Y||^2 \tag{2}$$

In the context of this work, both problems are equivalent, and will be solved (approximately) via stochastic optimization.

The idea of using stochastic optimization also appears in a previous work [2]. In this paper, the authors effectively rely on this technique to tackle large-scale discrete problems, semi-discrete problems, and continuous scenarios. A key difference respect to [6] is that in [2], stochastic optimization is used for solving the entropic regularization of Kantorovich, instead of the exact problem. Indeed, one of the contributions of [6] is that Kantorovich problem is solved in the case without regularization.

Another contribution of [6] is deriving an alternate dual of (2).

The dual of (2) is

$$W_2^2(P,Q) = \sup_{(f,g)\in\Phi_c} \mathbb{E}_P[f(X)] + \mathbb{E}_Q[g(Y)] \tag{3}$$

Where $\Phi_C = \{(f,g) \in L^1(P)\times L^1(Q) : f(x)+g(y) \leq \frac{1}{2}||x-y||^2 \ \forall(x,y) \ dP\otimes dQ$ a.e$\}$

Starting from (3), the authors derive an alternative minmax optimization over the space of convex functions. This novel minmax optimization problem is tackled by the usage of Input Convex Neural Networks, as defined in [1]. This architecture was shown to be rich enough for approximating any convex function [3].

The contribution of [6] is to present a novel algorithm for finding the optimal transport from samples. This approach is suitable for both discrete and continuous settings. Since the method learns the optimal transport between distributions, the results are the same despite the initialization, which makes the method robust to initialization in contrast to other generative models. The authors also provide a proof bounding the error in the learned mapping, with the minmax gap. Since the optimal mapping is found as the gradient of the optimized functions, this allow the method to find discontinuous transports which arise naturally in many scenarios. This

is another advantage of this approach when compared to other generative models.

In terms of its limitations, this paper presents a novel method but there are many things to explore and improve on how to initialize the models, which optimizers, architectures, and hyper-parameters to use, etc. The authors don't discuss any of these practical issues that are key for getting good results on challenging problems. Moreover, there are inconsistencies between what is reported in the paper and what is implemented on the project's repository [7].

There is still a challenge to effectively scale the method to high dimensions and large datasets. Although slower, with a large dataset in high dimension, this method is more accurate than the classic OT solver available in [5]. In smaller dimensions the classic solver is a clear winner because it's both accurate and blazingly fast. On the other hand, while in many continuous examples the method works correctly, it still can be improved in terms of speed. Currently it's not clear which architecture and parameters to select to get an optimal convergence. Moreover, there was one continuous example on which the approximation of W_2^2 seems to converge, but the mapping is not "as discontinuous" as it should be.

To sum up, in [6] the authors present a novel minmax problem for finding the optimal transport between (either continuous or discrete) distributions from samples. This problem is restricted to convex functions, and can be solved by using Input Convex Neural Networks. Empirically works well in many scenarios and has advantages respect to other neural networks generative models: robustness and discontinuity. In the discrete case it doesn't seem to be better than classic OT solvers in small datasets. For some large problems, although it's not fast, it produces accurate solutions. Scaling it to work with large problems is still an open challenge, and many improvements can be done in terms of batching, model selection, parameter tuning, optimization, and practical considerations.

3 Optimal transport mapping via input convex neural networks

The proposed method directly derives from the dual formulation of Kantorovich problem 3. Working a little bit over the set Φ_C, we can get a useful reparametrization of the problem.

First note that,

$$< x, y >\leq \frac{1}{2}[||x||^2 + ||y||^2 - ||x - y||^2]$$

Then, starting with the definition of the set Φ_C. For $(f, g) \in \Phi_C$

$$f(x) + g(y) \leq \frac{1}{2}||x - y||^2$$
$$< x, y > \leq \frac{1}{2}||x||^2 - f(x) + (\frac{1}{2}||y||^2 - g(y))$$

Redefining f and g as the terms on the RHS of the previous equation, we end up with the alternative formulation of the dual as:

$$W_2^2(P, Q) = \frac{1}{2}\mathbb{E}_P||X||^2 + \frac{1}{2}\mathbb{E}_Q||Y||^2 - \inf_{f,g}\ \mathbb{E}_P f(X) + \mathbb{E}_Q g(Y) \tag{4}$$

S.t $(f, g) \in L^1(P) \times L^1(Q) :\ \ f(X) + g(Y) \geq < X, Y > \ \ \forall x, y\ dP \otimes dQ$ **a.e**

Now we can use the idea of alternating optimization seen during the course to get rid of g and get an equivalent unconstrained optimization problem. Moreover, by the results cited by the authors, the optimization should be over the set of convex functions.

$$W_2^2(P, Q) = \frac{1}{2}\mathbb{E}_P||X||^2 + \frac{1}{2}\mathbb{E}_Q||Y||^2 - \inf_{f \in \mathbf{CVX}(P)} \mathbb{E}_P f(X) + \mathbb{E}_Q f^*(Y) \tag{5}$$

As we're under Brenier's theorem assumptions, we have that Kantorovich and Monge are equivalent problems. Solving the dual, we get a solution for the primal (Kantorovich) and the optimal mapping (Monge). The relationship between the different formulations is the following: whenever Q admits a density, the optimal coupling $\pi^* = (\mathrm{Id}, (\nabla f^*))_{\#Q}$. And the optimal mapping $T^* = \nabla f^*$.

From 5, the authors derive a novel minmax formulation for the problem. Whenever Q admits a density in $\mathbb{R}^d$, we have:

$$W_2^2(P, Q) = \sup_{f \in \mathbf{CVX}(P)} \inf_{g \in \mathbf{CVX}(Q)} V_{P,Q}(f, g) + C_{P,Q} \tag{6}$$

With $V_{P,Q}(f, g) = -\mathbb{E}_P f(X) - \mathbb{E}_Q[< Y, \nabla g(Y) > -f(\nabla g(Y))]$ and $C_{P,Q} = \frac{1}{2}\mathbb{E}_P||X||^2 + \frac{1}{2}E_Q||Y||^2$. Morover, the optimal is attained and ∇g^{opt} is the optimal transport map from Q to P, and ∇f^{opt} is the optimal transport map from P to Q.

Finally, the method of finding the optimal transport via ICNNs is obtained by solving 6 with f and g being ICNNs trained using loss functions derived from the above optimization problem.

An ICNN $f(., \theta)$ can be described by the following recurrence, where x is the input:

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

$$f(x;\theta) = z_L$$

We require all $W_l \geq 0$, σ_0 to be a convex function, and σ_l convex and non decreasing for $l = 1, ..., L-1$

The training loop implemented from scratch for this project is presented as Algorithm 1. Note that there are some differences between the proposed algorithm in the paper and the implementation reported by the authors. The function g that achieves the minimum in 6 is convex, in particular is f^*. That's why the authors relax the convexity constraint on g and add a regularization term to the loss of g for practical convenience:

$$R(\theta_g) = \lambda \sum ||max(-W_l, 0)||_F^2$$

Algorithm 1 Implementation of the method reported in [6]

Input: Source dist. Q, Target dist. P, Batch size M, Generator iterations K. Total iterations T. Evaluation iterations V.
Sample test batches $(X_i^{test})_i^M \sim P$ and $(Y_i^{test})_i^M \sim Q$ if continuous case, else $(X_i^{test})_i = P$ and $(Y_i^{test})_i^M = Q$
for t = 1, ..., T **do**
    Sample batches $(X_i)_i^M \sim P$ and $(Y_i)_i^M \sim Q$
    for k = 1, ..., K **do**
        Update the weights θ_g with Adam, to minimize $\frac{1}{M}\sum_i^M f(\nabla g(Y_i)) - < Y_i, \nabla g(Y_i) > + R(\theta_g)$
        Update the weights θ_f with Adam, to minimize $\frac{1}{M}\sum_i^M -f(\nabla g(Y_i)) + f(X_i)$
    Clip the weights W_l of θ_f to enforce the concavity constraint
    Every V iterations: evaluate the performance over X^{test} and Y^{test}

When solved exactly, problem 6 allow us to find the optimal transport between distributions. As a theoretical guarantee for the approximate solution obtained by stochastic optimization, the authors show that a bound on the error between the optimal and approximate transport can be derived. This error is bounded by the minimization and maximization gaps from 6.

4 Experiments

An implementation from scratch was used because of the poor usability of the project's original code. The method was evaluated in 7 different experiments in both continuous and discrete settings. In general, the results are good and comparable to the ones reported by the authors. However there is one particular dataset (triangle of Gaussians) on which the method doesn't converge to a discontinuous solution as expected. On the discrete case, for a small problem, the method is completely outperformed by a

classic OT solver, mainly because of the speed of convergence. On a large problem in which a closed form solution is known, again, the neural method is slower, but finds a more accurate solution.

All the experiments were run in Google Colab. Further experimentation on hyper-parameter tuning should be done to speed up the convergence. The weights of the networks were initialized using PyTorch's default initialization and clipped. Leaky ReLU with parameter 0.2 was utilized as activation function for all the layers (including the last one), and on the first layer was squared. Adam was used as optimizer, with parameters $\beta = (0.5, 0.9), \mathrm{lr} = 1e-4$. NumPy's and PyTorch's random seeds are set in 42 before running any experiment.

For all the continuous experiments samples from the distributions are shown. For discrete distributions, the full point cloud is presented. In the high dimension examples, only the first two coordinates are plotted. When displaying the training curves (evolution of W_2^2 and the norm of the difference between the mean of the distributions and its transport), the first point is omitted to have a better visualization. There are 500 iterations between each point of the training curves.

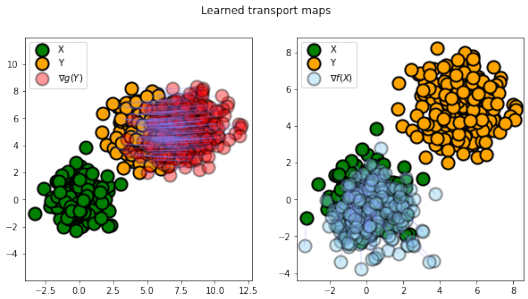
4.1 Continuous distributions

In the case where a closed form solution was known, it was used as ground-truth. We also include to the comparison the result obtained using `POT.emd` solver over the discrete samples from the test batch.

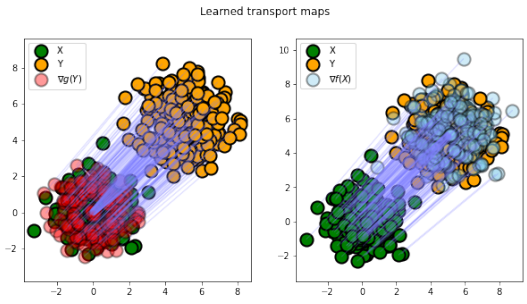
4.1.1 Translated Gaussians in 2D

- $P = \mathcal{N}([0, 0], Id)$, $Q = \mathcal{N}([5, 5], Id \times 1.5)$.
- 4 layers of size 64. Batch size = 256. 10000 total iterations. 10 inner iterations.

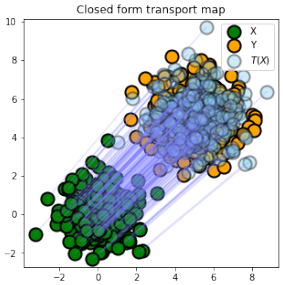
In this first experiment, the method is able to solve the problem around iteration 2500. The obtained solution is accurate. There are some minor fluctuations on the W_2^2 approximation, although the mappings are quite stable.



(a) Learned transport at iteration 1

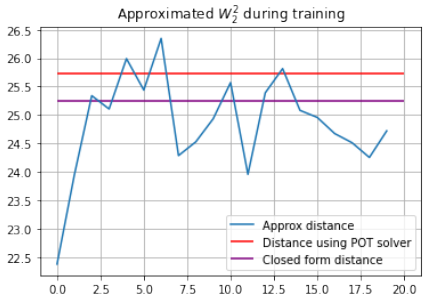


(b) Learned transport at iteration 10000

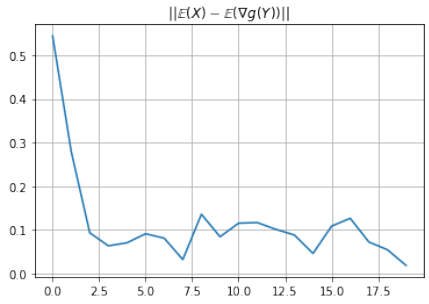


(c) Closed form optimal solution

Figure 1: 2D continuous Gaussian distributions



(a) W_2^2 distance



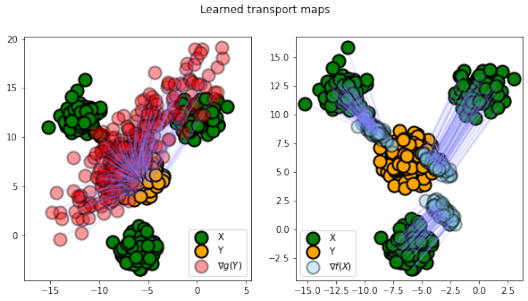
(b) Norm of the difference between the mean of X and $\nabla g(Y)$

Figure 2: 2D continuous Gaussian distributions training curves

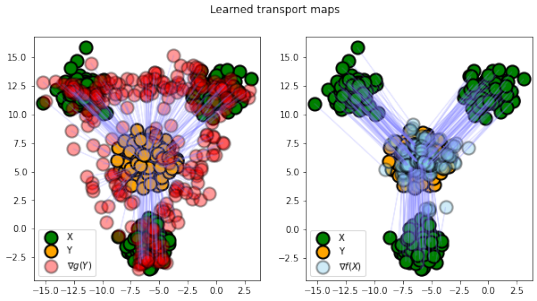
4.1.2 Triangle of Gaussians

- $P = \frac{1}{3}\mathcal{N}([-6, -1.5], Id) + \frac{1}{3}\mathcal{N}([-12, 12], Id) + \frac{1}{3}\mathcal{N}([0, 12], Id)$. $Q = \mathcal{N}([-6, 6], Id)$
- 4 layers of size 64. Batch size = 255. 10000 total iterations. 10 inner iterations.

Although at first sight it seemed a simple example, this turned out to be the case in which the proposed method didn't converge to the expected solution. At around iteration 1500 the approximation of the W_2^2 between the distributions seem to have converged, but the mapping still didn't look well. After iteration 10000, the map from X to Y looks correct, but the (discontinuous) map from Y to X doesn't. There are spurious probability masses in parts where they shouldn't be, and the map is "too continuous". In this example was far easier to learn the W_2^2 distance rather than the mappings.



(a) Learned transport at iteration 500



(b) Learned transport at iteration 10000

Figure 3: (Continuous) Triangle of Gaussians distributions

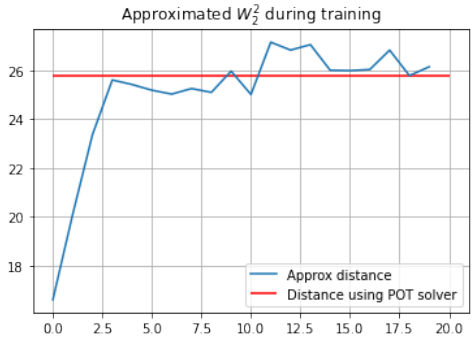


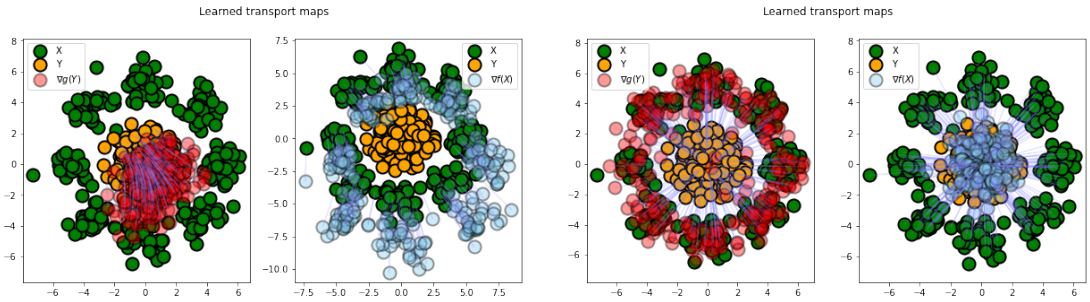
Figure 4: W_2^2 distance

4.1.3 Mixture of Gaussians

- $P = \frac{1}{8}\sum_{i=0}^{7}\mathcal{N}([5cos(\theta_i), 5sen(\theta_i], Id{\times}0.5), \theta_i = \frac{2\pi}{i+1}$. $Q = \mathcal{N}([0,0], Id)$
- 4 layers of size 64. Batch size = 256. 10000 total iterations. 10 inner iterations.

Again we observe some fluctuations on the W_2^2 distance between distributions. The learned maps seem accurate, there are some spurious probability

masses but this issue is smaller than it was on the previous example. This is expected since this example is easier, in the sense that the map is "less discontinuous". The reported results on [6] for this experiment actually look better and more discontinuous.



(a) Learned transport at iteration 1

(b) Learned transport at iteration 10000

Figure 5: Continuous mixture of Gaussians

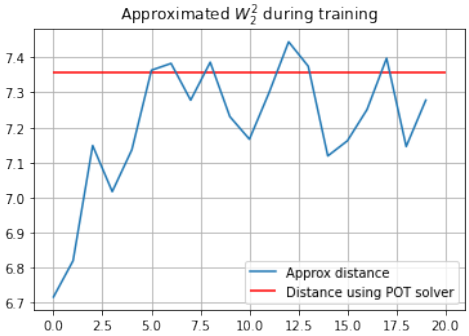
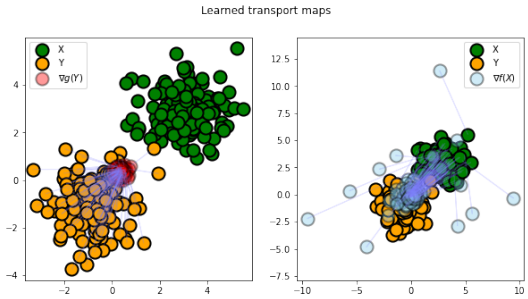


Figure 6: W_2^2 distance

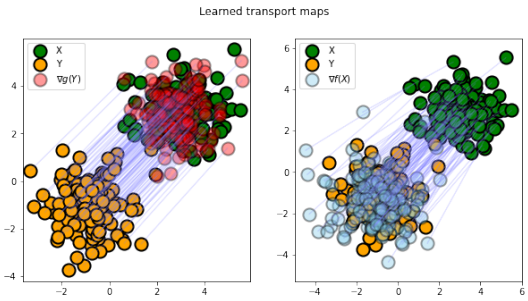
4.1.4 Translated Gaussian in dim 300

- $P = \mathcal{N}([3]_{300}, Id)$, $Q = \mathcal{N}([-1]_{300}, Id)$
- 4 layers of size 1024. Batch size = 64. 10000 total iterations. 10 inner iterations.

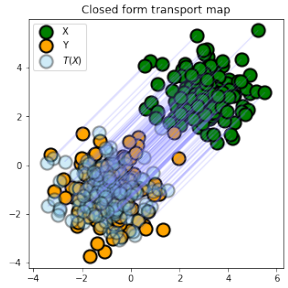
By observing the results over some dimensions, and also the training curves, it seems that this method can handle as well the continuous translated Gaussian setting in dimension 300. Not only the difference in the means between the distribution and its transport sharply decays, but the learned W_2^2 distance converges to the closed form solution. Moreover, the results are more accurate that the ones obtained by solving the discrete transport of the samples using POT.



(a) Learned transport at iteration 500 (only first 2 components)

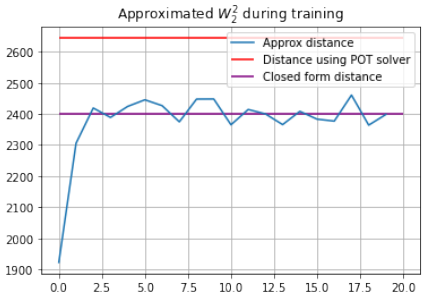


(b) Learned transport at iteration 10000 (only first 2 components)

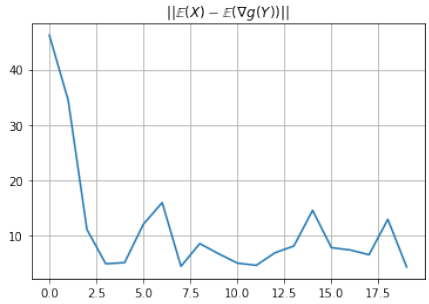


(c) Closed form optimal solution

Figure 7: 300-D continuous Gaussian distributions (only first 2 components)



(a) W_2^2 distance



(b) Norm of the difference between the mean of X and $\nabla g(Y)$

Figure 8: 300-D continuous Gaussian distributions training curves

4.2 Discrete distributions

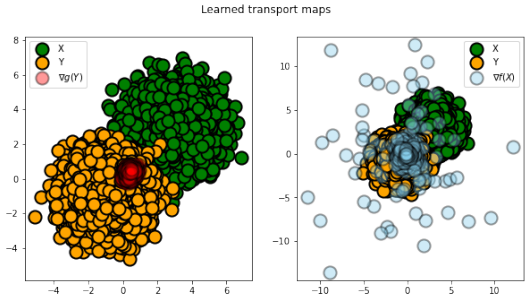
4.2.1 Translated Gaussian in dim 200

- Both P,Q are discrete point clouds with uniform weights. The position of the points of P were sampled from $\mathcal{N}([3]_{200}, Id)$. The position of

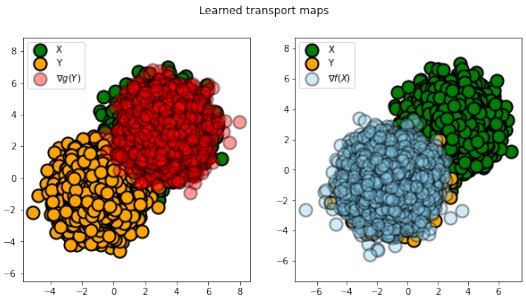
the points of Q were sampled from $\mathcal{N}([-1]_{200}, Id)$. We consider both point clouds of size $N = 10000$

- 4 layers of size 1024. Batch size = 64. 10000 total iterations. 10 inner iterations.

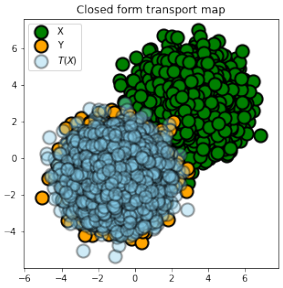
The method is able to handle this large scale discrete problem. Moreover, the results are much more accurate that the ones obtained by using POT solver.



(a) Learned transport at iteration 500 (only first 2 components)

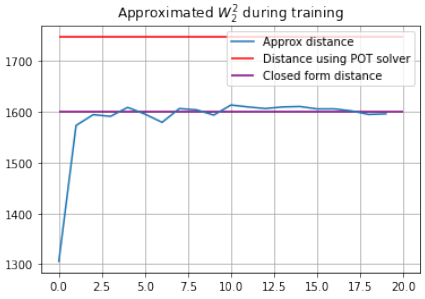


(b) Learned transport at iteration 10000 (only first 2 components)

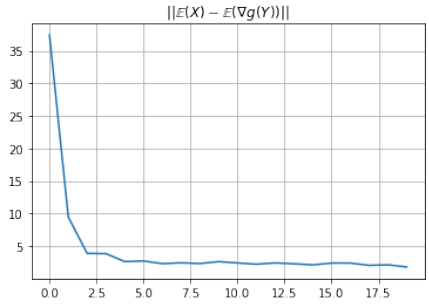


(c) Closed form optimal solution

Figure 9: 200-D discrete Gaussian distributions (only first 2 components)



(a) W_2^2 distance



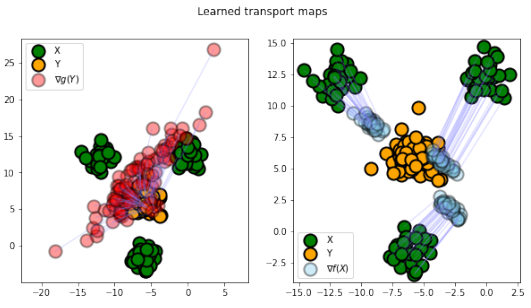
(b) Norm of the difference between the mean of X and $\nabla g(Y)$

Figure 10: 200-D discrete Gaussian distributions training curves

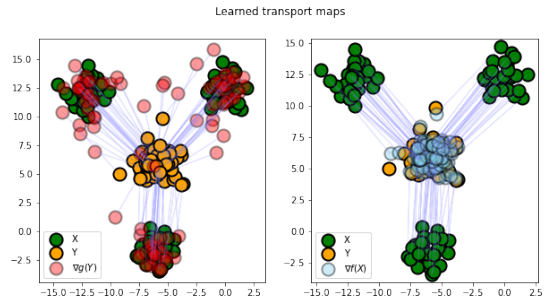
4.2.2 Triangle of Gaussians

- Both P,Q are discrete point clouds with uniform weights. The position of the points of P were sampled from $= \frac{1}{3}\mathcal{N}([-6, -1.5], Id) + \frac{1}{3}\mathcal{N}([-12, 12], Id) + \frac{1}{3}\mathcal{N}([0, 12], Id)$. The position of the points of Q were sampled from $\mathcal{N}([-6, 6], Id)$. We consider both point clouds of size $N = 90$
- 4 layers of size 128. Batch size = 90. 10000 total iterations. 50 inner iterations.

In this simpler version of the triangle of Gaussians, the method does perform better. The learned map is more discontinuous. This can be attributed to both: having a discrete problem, and having less points. However, although there is convergence to the W_2^2 distance, the solution still has some spurious masses. On the other hand, the solver from POT is able to easily solve this matching problem really fast.



(a) Learned transport at iteration 500



(b) Learned transport at iteration 10000

Figure 11: (Discrete) Triangle of Gaussians distributions

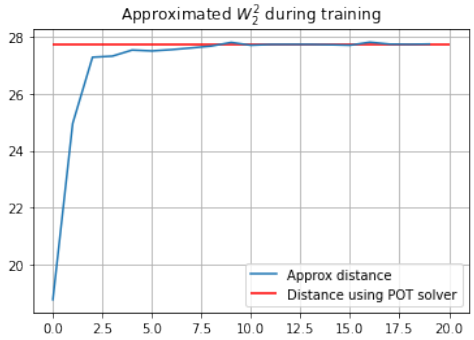


Figure 12: W_2^2 distance

5 Conclusion and perspectives

The authors present a practical novel method for finding the OT between two distributions from its samples. This method seems to converge to the correct solution in many examples. However, as problems get more difficult, the performance of the algorithm is sensible to practical details, such as: architecture of the networks, batch size, number of iterations, optimizer, etc. The impact of these practical details is not discussed in the paper.

One of the major flaws of this work are the inconsistencies between what is reported in the paper and what is implemented in the project's code. Some of the these are: in which part of the loop should batches be sampled, weight initialization, architecture of the networks, optimizer parameters. It's not straightforward to reproduce the reported results.

The proposed method introduces a convex minmax optimization problem, which is solved with ICNNs, but can be extended to other techniques of convex optimization. Its derivation is straightforward from the classic OT theory. The exact Kantorovich problem is treated, without the need to introduce entropic regularization.

It can be readily used to train a deep generative models, providing theoretical advantages respect to other methods: robustness to initialization, and ability to represent discontinuous maps. In practice, the capacity of representing discontinuous maps seem to depend on practical implementation details, and tends to be harder to get as the complexity of the problem increases (e.g continuous triangle of Gaussians).

The method can be used in both discrete and continuous scenarios. In small discrete cases, classic OT solvers are more efficient and accurate than the neural approach. However, as problems get larger, in some cases, the proposed method is more accurate than the classic OT solver. In the continuous case the method converges to accurate solutions as well, being closer to the closed form solution of the problem than the discrete OT solution.

Efficiently scaling this method is an open challenge. At this point is

not clear how to measure the efficiency, since it depends on many practical details which are not inherent to the proposed technique. I.e the same algorithm can be used with very different neural networks and the convergence will be different as well.

6 Connection with the course

This work can be completely understood with the contents from the course. Monge and Kantorovich formulations of the Optimal Transport problem are presented. From Kantorovich, the dual version of the problem, as seen on the course, is stated. Then, by Brenier's theorem an equivalence between Monge and Kantorovich is presented. Finally, the relationship between primal and dual is given, closing the circle between all formulations: solving the dual also solves the primal (Kantorovich) and allow us to compute the optimal transport map (Monge). In some of the derivations, notions like alternating optimization and c-transforms, are also used.

References

[1] Brandon Amos, Lei Xu, and J. Zico Kolter. Input convex neural networks. *CoRR*, abs/1609.07152, 2016.

[2] Genevay Aude, Marco Cuturi, Gabriel Peyré, and Francis Bach. Stochastic optimization for large-scale optimal transport, 2016.

[3] Yize Chen, Yuanyuan Shi, and Baosen Zhang. Optimal control via neural networks: A convex approach, 2019.

[4] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[5] Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z. Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, Léo Gautheron, Nathalie T.H. Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J. Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021.

[6] Ashok Vardhan Makkuva, Amirhossein Taghvaei, Sewoong Oh, and Jason D. Lee. Optimal transport mapping via input convex neural networks, 2020.

[7] Ashok Vardhan Makkuva, Amirhossein Taghvaei, Sewoong Oh, and Jason D. Lee. Optimal transport mapping via input convex neural networks. https://github.com/AmirTag/OT-ICNN, 2020.