Optimal Transport for Machine Learning

Pierre Alain Langlois - Thibault Groueix

Outline

1. Intro :
    a. General considerations
    b. Formulation
2. Discrete OT
    a. Use cases : point clouds, reordering… (paper on primitives)
    b. Relaxation : same solution
    c. Solver - duality - Hungarian algo
    d. Approximation : Auction algorithm
    e. Regularisation : Sinkhorn
3. Continuous case
    a. Wasserstein Distance formulation
    b. Kantorovitch duality -> what does it mean in the continuous case ?
    c. Gan discussion
    d. Other way to solve : sliced Wasserstein

Déblais et Remblais

« *Le prix du transport d'une molécule étant, proportionnel à la somme des produits des molécules multipliées par l'espace parcouru, il s'ensuit qu'il n'est pas indifférent que telle molécule du déblai soit transportée dans tel ou tel autre endroit du remblai, mais qu'il y a une certaine distribution à faire des molécules du premier dans le second, d'après laquelle la somme de ces produits sera la moindre possible, et le prix du transport total sera un minimum.*

[…]

C'est la solution de cette question que je me propose de donner ici ».

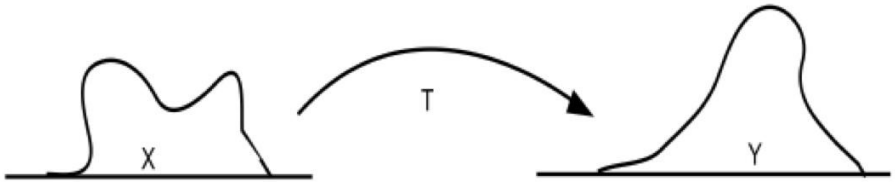
Gaspard Monge, 1776

Intro

Measure v on X : source distribution
Measure μ on Y : target distribution

T : transport map s.t T(v) = μ

Find T that minimizes a certain cost c



Intro

v : source distribution
μ : target distribution

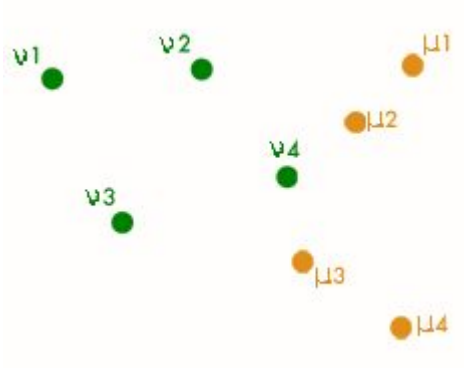
T : transport map s.t T(v) = μ

Find T that minimizes a certain cost c

Discrete case :

v and μ are sets of Dirac ⇔ point clouds

c is the euclidean distance between a point and its target.



Intro - Problem Formulation

Monge

$$\inf \left\{ \int_X c(x, T(x)) \, \mathrm{d}\mu(x) \middle| \, T_*(\mu) = \nu \right\}$$

Intro - Problem Formulation

Monge

$$\inf \left\{ \int_X c(x, T(x)) \, \mathrm{d}\mu(x) \middle| \, T_*(\mu) = \nu \right\}$$

Problem !

If v is a dirac and μ is not, there exist no T such that the condition is satisfied !

Intro - Problem Formulation

Monge

$$\inf \left\{ \int_X c(x, T(x)) \, \mathrm{d}\mu(x) \middle| \, T_*(\mu) = \nu \right\}$$

Kantorovitch

$$\inf \left\{ \int_{X \times Y} c(x, y) \, \mathrm{d}\gamma(x, y) \middle| \, \gamma \in \Gamma(\mu, \nu) \right\}$$

Γ(μ, ν) denotes the collection of all probability measures on $X \times Y$ with marginals μ on X and ν on Y

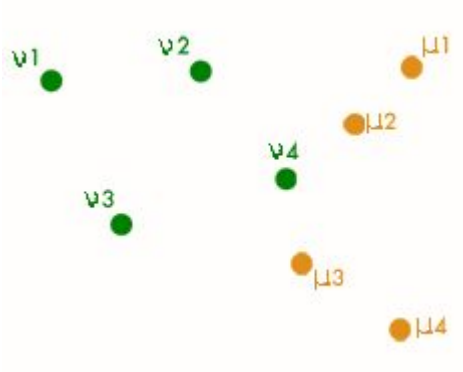
Discrete case : point clouds

$$\max_{X \in \mathbb{R}^{n \times n}} \quad \sum_{i=1}^{n} \sum_{j=1} A_{ij} X_{ij}$$

$$\text{subject to} \quad \sum_{i=1}^{n} X_{ij} = 1, \quad i = 1, 2, \ldots, n$$

$$\sum_{j=1}^{n} X_{ij} = 1, \quad j = 1, 2, \ldots, n.$$

$$X_{ij} \in \{0, 1\}.$$



Aij = -dist(vi, μj)

Relaxation : same solution

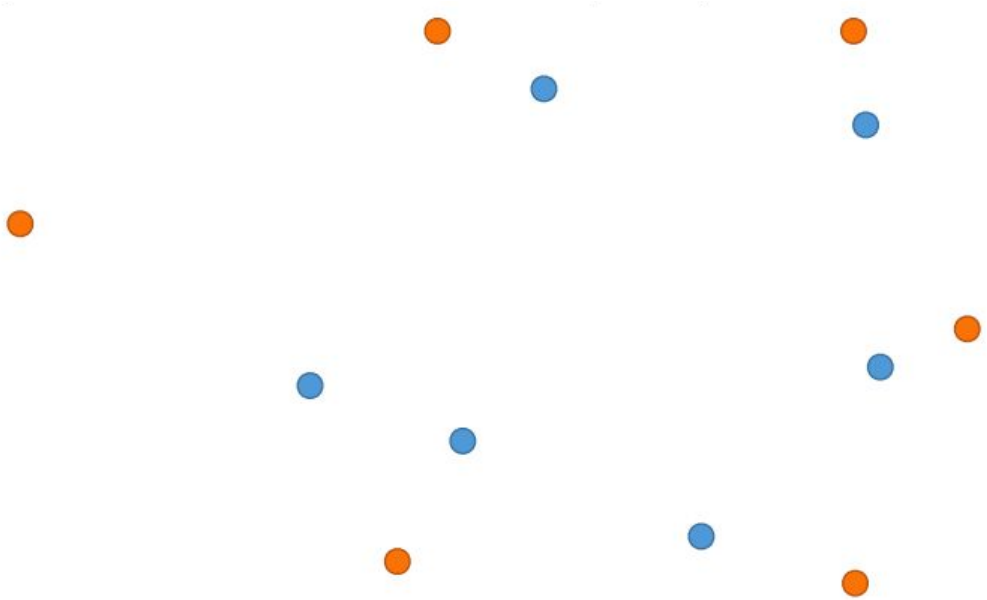
Formulation :

Proof :

1. 2n equality constraint are in fact 2n -1 equality constraint
2. Optimum -> n^2 saturated constraints -> 2n - 1 non zero values
3. Pigeonhole principle : there exists a line with at most 1 non zero value
4. By constraints : it’s 1. Remove the line and column and recurse

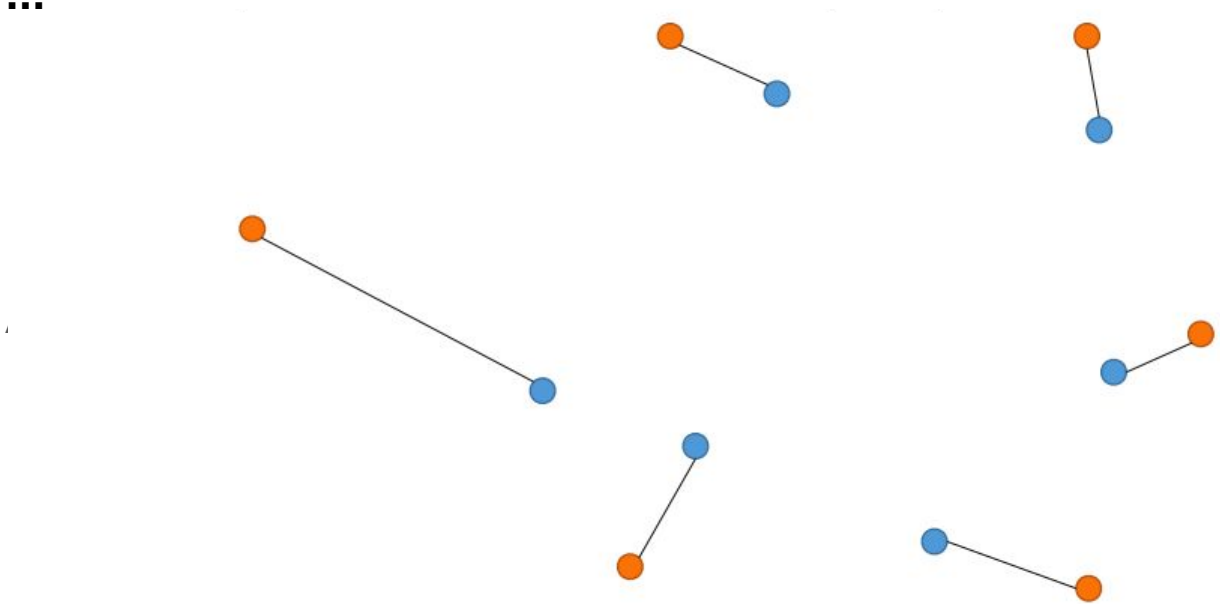
Use Cases 1

- Compare two pointclouds : PointSetGen [Fan2016], Atlasnet [Groueix2018], ...



Use Cases 1

- Compare two pointclouds : PointSetGen [Fan2016], Atlasnet [Groueix2018], ...



Use Cases 1

- Compare two pointclouds : PointSetGen [Fan2016], Atlasnet [Groueix2018], **...**

  Dimensions : 3-6 (normals…)

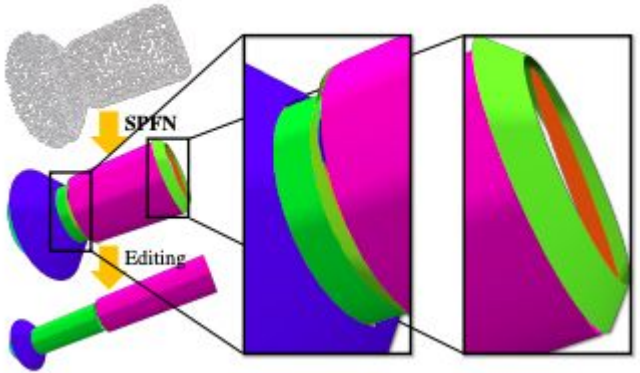
  Number of points : 1000 - 10000

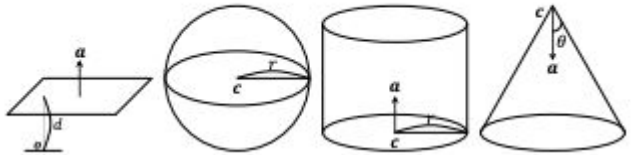
  Alternatives : Chamfer Distance

Use Cases 2

- Reorder a predicted vectors when you want to enforce invariance to permutations. (primitive discovery).

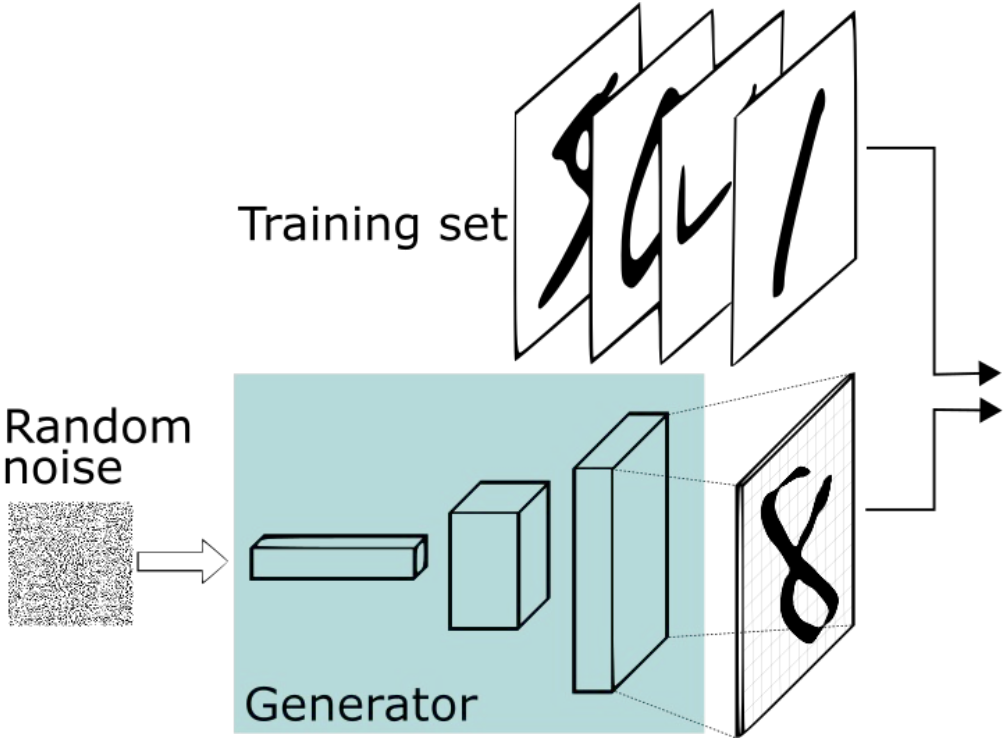
  Supervised Fitting of Geometric Primitives to 3D Point Clouds. [Li2018]





$$\mathrm{RIoU}(\mathbf{w}, \hat{\mathbf{w}}) = \frac{\mathbf{w}^{\mathbf{T}}\hat{\mathbf{w}}}{\|\mathbf{w}\|_1 + \|\hat{\mathbf{w}}\|_1 - \mathbf{w}^{\mathbf{T}}\hat{\mathbf{w}}}$$

Use Cases 3: Give a Monte Carlo estimate of the continuous case (MNIST)



$$\max_{X \in \mathbb{R}^{n \times n}} \quad \sum_{i=1}^{n} \sum_{j=1} A_{ij} X_{ij}$$

$$\text{subject to} \quad \sum_{i=1}^{n} X_{ij} = 1, \quad i = 1, 2, \ldots, n$$

$$\sum_{j=1}^{n} X_{ij} = 1, \quad j = 1, 2, \ldots, n.$$

$$X_{ij} \in \{0, 1\}.$$

credits : https://skymind.ai/

Solver - duality - Hungarian algo (Harold Kuhn 1955)

Complexity : O(n^3) [Edmonds and Karp]

Example : credit http://www.math.harvard.edu/archive/20_spring_05/handouts/assignment_overheads.pdf

Proof : https://theory.epfl.ch/courses/topicstcs/Lecture52015.pdf page 6

Code : https://gist.github.com/KartikTalwar/3158534

The Hungarian Method: The following algorithm applies the above theorem to a given $n \times n$ cost matrix to find an optimal assignment.

Step 1. Subtract the smallest entry in each row from all the entries of its row.

Step 2. Subtract the smallest entry in each column from all the entries of its column.

Step 3. Draw lines through appropriate rows and columns so that all the zero entries of the cost matrix are covered and the *minimum* number of such lines is used.

Step 4. *Test for Optimality:* (i) If the minimum number of covering lines is n, an optimal assignment of zeros is possible and we are finished. (ii) If the minimum number of covering lines is less than n, an optimal assignment of zeros is not yet possible. In that case, proceed to Step 5.

Step 5. Determine the smallest entry not covered by any line. Subtract this entry from each uncovered row, and then add it to each covered column. Return to Step 3.

Example 1: You work as a sales manager for a toy manufacturer, and you currently have three salespeople on the road meeting buyers. Your salespeople are in Austin, TX; Boston, MA; and Chicago, IL. You want them to fly to three other cities: Denver, CO; Edmonton, Alberta; and Fargo, ND. The table below shows the cost of airplane tickets in dollars between these cities.

From \ To	Denver	Edmonton	Fargo
Austin	250	400	350
Boston	400	600	350
Chicago	200	400	250

Where should you send each of your salespeople in order to minimize airfare?

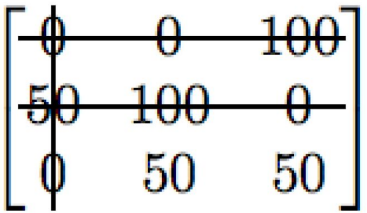
Step 1. Subtract 250 from Row 1, 350 from Row 2, and 200 from Row 3.

$$\begin{bmatrix} 250 & 400 & 350 \\ 400 & 600 & 350 \\ 200 & 400 & 250 \end{bmatrix} \sim \begin{bmatrix} 0 & 150 & 100 \\ 50 & 250 & 0 \\ 0 & 200 & 50 \end{bmatrix}$$

Step 2. Subtract 0 from Column 1, 150 from Column 2, and 0 from Column 3.

$$\begin{bmatrix} 0 & 150 & 100 \\ 50 & 250 & 0 \\ 0 & 200 & 50 \end{bmatrix} \sim \begin{bmatrix} 0 & 0 & 100 \\ 50 & 100 & 0 \\ 0 & 50 & 50 \end{bmatrix}$$

Step 3. Cover all the zeros of the matrix with the minimum number of horizontal or vertical lines.



Step 4. Since the minimal number of lines is 3, an optimal assignment of zeros is possible and we are finished.

Since the total cost for this assignment is 0, it must be an optimal assignment.

$$\begin{bmatrix} 0 & \boxed{0} & 100 \\ 50 & 100 & \boxed{0} \\ \boxed{0} & 50 & 50 \end{bmatrix}$$

Here is the same assignment made to the original cost matrix.

$$\begin{bmatrix} 250 & \boxed{400} & 350 \\ 400 & 600 & \boxed{350} \\ \boxed{200} & 400 & 250 \end{bmatrix}$$

Example 2: A construction company has four large bulldozers located at four different garages. The bulldozers are to be moved to four different construction sites. The distances in miles between the bulldozers and the construction sites are given below.

Bulldozer \ Site	A	B	C	D
1	90	75	75	80
2	35	85	55	65
3	125	95	90	105
4	45	110	95	115

How should the bulldozers be moved to the construction sites in order to minimize the total distance traveled?

Step 1. Subtract 75 from Row 1, 35 from Row 2, 90 from Row 3, and 45 from Row 4.

$$\begin{bmatrix} 90 & 75 & 75 & 80 \\ 35 & 85 & 55 & 65 \\ 125 & 95 & 90 & 105 \\ 45 & 110 & 95 & 115 \end{bmatrix} \sim \begin{bmatrix} 15 & 0 & 0 & 5 \\ 0 & 50 & 20 & 30 \\ 35 & 5 & 0 & 15 \\ 0 & 65 & 50 & 70 \end{bmatrix}$$

Step 2. Subtract 0 from Column 1, 0 from Colum 2, 0

from Column 3, and 5 from Column 4.

$$\begin{bmatrix} 15 & 0 & 0 & 5 \\ 0 & 50 & 20 & 30 \\ 35 & 5 & 0 & 15 \\ 0 & 65 & 50 & 70 \end{bmatrix} \sim \begin{bmatrix} 15 & 0 & 0 & 0 \\ 0 & 50 & 20 & 25 \\ 35 & 5 & 0 & 10 \\ 0 & 65 & 50 & 65 \end{bmatrix}$$

Step 3. Cover all the zeros of the matrix with the minimum number of horizontal or vertical lines.

$$\begin{bmatrix} 15 & 0 & 0 & 0 \\ 0 & 50 & 20 & 25 \\ 35 & 5 & 0 & 10 \\ 0 & 65 & 50 & 65 \end{bmatrix}$$

Step 4. Since the minimal number of lines is less than 4, we have to proceed to Step 5.

Step 5. Note that 5 is the smallest entry not covered by any line. Subtract 5 from each uncovered row.

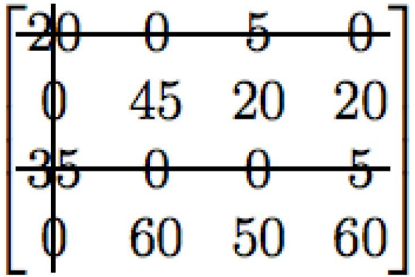
$$\begin{bmatrix} 15 & 0 & 0 & 0 \\ 0 & 50 & 20 & 25 \\ 35 & 5 & 0 & 10 \\ 0 & 65 & 50 & 65 \end{bmatrix} \sim \begin{bmatrix} 15 & 0 & 0 & 0 \\ -5 & 45 & 15 & 20 \\ 30 & 0 & -5 & 5 \\ -5 & 60 & 45 & 60 \end{bmatrix}$$

Now add 5 to each covered column.

$$\begin{bmatrix} 15 & 0 & 0 & 0 \\ -5 & 45 & 15 & 20 \\ 30 & 0 & -5 & 5 \\ -5 & 60 & 45 & 60 \end{bmatrix} \sim \begin{bmatrix} 20 & 0 & 5 & 0 \\ 0 & 45 & 20 & 20 \\ 35 & 0 & 0 & 5 \\ 0 & 60 & 50 & 60 \end{bmatrix}$$

Now return to Step 3.

Step 3. Cover all the zeros of the matrix with the minimum number of horizontal or vertical lines.



Step 4. Since the minimal number of lines is less than 4, we have to return to Step 5.

Step 5. Note that 20 is the smallest entry not covered by a line. Subtract 20 from each uncovered row.

$$\begin{bmatrix} 20 & 0 & 5 & 0 \\ 0 & 45 & 20 & 20 \\ 35 & 0 & 0 & 5 \\ 0 & 60 & 50 & 60 \end{bmatrix} \sim \begin{bmatrix} 20 & 0 & 5 & 0 \\ -20 & 25 & 0 & 0 \\ 35 & 0 & 0 & 5 \\ -20 & 40 & 30 & 40 \end{bmatrix}$$

Then add 20 to each covered column.

$$\begin{bmatrix} 20 & 0 & 5 & 0 \\ -20 & 25 & 0 & 0 \\ 35 & 0 & 0 & 5 \\ -20 & 40 & 30 & 40 \end{bmatrix} \sim \begin{bmatrix} 40 & 0 & 5 & 0 \\ 0 & 25 & 0 & 0 \\ 55 & 0 & 0 & 5 \\ 0 & 40 & 30 & 40 \end{bmatrix}$$

Now return to Step 3.

Step 3. Cover all the zeros of the matrix with the minimum number of horizontal or vertical lines.

$$\begin{bmatrix} \cancel{40} & \cancel{0} & \cancel{5} & \cancel{0} \\ \cancel{0} & \cancel{25} & \cancel{0} & \cancel{0} \\ \cancel{55} & \cancel{0} & \cancel{0} & \cancel{5} \\ \cancel{0} & \cancel{40} & \cancel{30} & \cancel{40} \end{bmatrix}$$

Step 4. Since the minimal number of lines is 4, an optimal assignment of zeros is possible and we are finished.

$$\begin{bmatrix} 40 & 0 & 5 & \boxed{0} \\ 0 & 25 & \boxed{0} & 0 \\ 55 & \boxed{0} & 0 & 5 \\ \boxed{0} & 40 & 30 & 40 \end{bmatrix}$$

Since the total cost for this assignment is 0, it must be an optimal assignment.

Here is the same assignment made to the original cost matrix.

$$\begin{bmatrix} 90 & 75 & 75 & \boxed{80} \\ 35 & 85 & \boxed{55} & 65 \\ 125 & \boxed{95} & 90 & 105 \\ \boxed{45} & 110 & 95 & 115 \end{bmatrix}$$

So we should send Bulldozer 1 to Site D, Bulldozer 2 to Site C, Bulldozer 3 to Site B, and Bulldozer 4 to Site A.

Approximation : Auction algorithm

https://web.eecs.umich.edu/~pettie/matching/Bertsekas-a-new-algorithm-assignment-problem-Mathematical-Programming-1981.pdf **[Bertsekas1981]**

Among the (numerous) variants of the Hungarian algorithm, this one is nice because it can be nicely parallelized

Parallel version
https://stanford.edu/~rezab/classes/cme323/S16/projects_reports/jin.pdf

Pytorch Code : (Thanks Stanford!)

https://github.com/fxia22/pointGAN/tree/master/emd

Approximation : Auction algorithm

1. Start with a set U of all bidders. U denotes the set of all *unassigned bidders*. We also maintain a set of prices which are initialized to all 0, and any structure that stores the current tentative (partial) assignment.

2. Pick any bidder i from U. Search for the item j that gives her the highest net payoff $A_{ij} - p_j$, and also an item k that gives her the second highest net payoff.

3. The price p_j of item j is updated to be $p_j \leftarrow p_j + \left(A_{ij} - p_j\right) - (A_{ik} - p_k)$. This update simply says that p_j is raised to the level at which bidder i is different (in terms of net payoff) bewteen item j and item k, i.e., the updated prices satisfy $A_{ij} - p_j = A_{ik} - p_k$.

4. Now assign item j to bidder i. If item j was previously assigned to another bidder s, then remove that assignment and add s to U.

Approximation : Auction algorithm

1. Start with a set U of all bidders. U denotes the set of all *unassigned bidders*. We also maintain a set of prices which are initialized to all 0, and any structure that stores the current tentative (partial) assignment.

2. Pick any bidder i from U. Search for the item j that gives her the highest net payoff $A_{ij} - p_j$, and also an item k that gives her the second highest net payoff.

3. The price p_j of item j is updated to be $p_j \leftarrow p_j + \left(A_{ij} - p_j\right) - (A_{ik} - p_k)$. This update simply says that p_j is raised to the level at which bidder i is different (in terms of net payoff) bewteen item j and item k, i.e., the updated prices satisfy $A_{ij} - p_j = A_{ik} - p_k$.

4. Now assign item j to bidder i. If item j was previously assigned to another bidder s, then remove that assignment and add s to U.

Approximation : Auction algorithm

1. Start with a set U of all bidders. U denotes the set of all *unassigned bidders*. We also maintain a set of prices which are initialized to all 0, and any structure that stores the current tentative (partial) assignment.

2. Pick any bidder i from U. Search for the item j that gives her the highest net payoff $A_{ij} - p_j$, and also an item k that gives her the second highest net payoff.

3. The price p_j of item j is updated to be $p_j \leftarrow p_j + \left(A_{ij} - p_j\right) - \left(A_{ik} - p_k\right)$. This update simply says that p_j is raised to the level at which bidder i is different (in terms of net payoff) bewteen item j and item k, i.e., the updated prices satisfy $A_{ij} - p_j = A_{ik} - p_k$.

4. Now assign item j to bidder i. If item j was previously assigned to another bidder s, then remove that assignment and add s to U.

Optimal Transport Dual

Problem statement

Integer Linear Problem : $\max_{X_{ij}} \sum_i \sum_j A_{ij} X_{ij}$

$$\text{sc} \begin{cases} \sum_i X_{ij} = 1 \\ \sum_j X_{ij} = 1 \\ \forall i,j \ , \ X_{ij} \in \{0,1\} \end{cases} \qquad (1)$$

Constraints $\{X_{ij}\}$ are totally unimodular so (1) is equivalent to the relaxed problem:

$$\max_{X_{ij}} \sum_i \sum_j A_{ij} X_{ij}$$

$$\text{sc} \begin{cases} \sum_i X_{ij} = 1 \\ \sum_j X_{ij} = 1 \\ X_{ij} \geq 0 \ . \end{cases} \qquad (2)$$

Let's write the lagrangian:

$$\mathcal{L}(X_{ij}, \lambda_i, p_j, \gamma_{ij}) = \sum_i \sum_j A_{ij} X_{ij} + \sum_j p_j \left(\sum_i X_{ij} - 1 \right) + \sum_i \lambda_i \left(\sum_j X_{ij} - 1 \right) + \sum_{ij} \gamma_{ij} X_{ij}$$

$$(2) = \max_{X_{ij}} \min_{\substack{\lambda_i, p_j \in \mathbb{R} \\ \gamma_{ij} \geq 0}} \mathcal{L}(X_{ij}, \lambda_i, p_j, \gamma_{ij}) \ .$$

The dual is: $\min_{\substack{\lambda_i, p_j \in \mathbb{R} \\ \gamma_{ij} \geq 0}} \max_{X_{ij}} \mathcal{L}(X_{ij}, \lambda_i, p_j, \gamma_{ij})$

$$= \min_{\substack{\lambda_i, p_j \in \mathbb{R} \\ \gamma_{ij} \geq 0}} \max_{X_{ij}} \sum_{ij} \left(A_{ij} + p_j + \lambda_i + \gamma_{ij} \right) X_{ij} - \sum_i \lambda_i - \sum_j p_j$$

$$= \min_{\lambda_i, p_j \in \mathbb{R}} - \sum_i \lambda_i - \sum_j p_j$$

$$\text{sc} \ A_{ij} + p_j + \lambda_i \leq 0 \ , \ \forall i,j$$

Dual formulation

$$\min_{\substack{\lambda_i, p_j \in \mathbb{R}}} \sum_i \lambda_i + \sum_j p_j \qquad (3)$$

$$\text{sc} \quad A_{ij} \leq \lambda_i + p_j \quad \forall i,j$$

Let's use complementary slackness to design a new optimization problem. If X_{ij}^* is optimal for (2) and λ^*, p^* optimal for (3)

then $\forall i,j, \quad X_{ij} \cdot (A_{ij} - \lambda_i - p_j) = 0$.

By constraints, $\forall i, \ \exists! \, j$ such that $X_{ij} = 1$

then $\lambda_i = \max_j \{ A_{ij} - p_j \}$.

Let's rewrite (3) in

$$\min_{p \in \mathbb{R}^n} \sum_i \max_j \{ A_{ij} - p_j \} + \sum_j p_j \qquad (4)$$

- p are called prices
- $\lambda_i = \max_j \{ A_{ij} - p_j \}$ is the marginal profit of bidder i

Relation between (3) and (4).

if (X_{ij}) feasible, and (X_{ij}), $\{p_j\}$ satisfy complementary slackness (CS)

ie: $X_{ij}\,(A_{ij} - d_i - p_j) = 0 \quad \forall i,j$

$\Leftrightarrow A_{i\sigma(i)} - p_{\sigma(i)} = \max_j (A_{ij} - p_j) \quad \forall i$

then $\{p_j\}$ is optimal for (4) and (X_{ij}) optimal for (2).

Proof We will show that (2) $\leq$ (4) always and are equal une CS assumption, hence are optimal.

- $\sum_i A_{i\sigma(i)} = \sum_i A_{i\sigma(i)} - p_{\sigma(i)} + p_{\sigma(i)}$
  $\leq \sum_i \max_j \{A_{ij} - p_j\} + \sum p_j$

  Hence if (X_{ij}) is feasible, then (2) $\leq$ (4) $\forall p$.
- with CS (2) = (4) then (X_{ij}^*) is optimum for (2) and p_j^* for (4).

Algorithm

1- Start with U: set of all unassigned bidder
   $p = 0$

2- Choose $i \in U$. find $j = \underset{k}{\operatorname{argmax}} \{A_{ik} - p_k\}$
   and $k = \underset{\ell \neq j}{\operatorname{argmax}} \{A_{i\ell} - p_\ell\}$.

3: Update the price $p_j \leftarrow p_j + (A_{ij} - p_j) - (A_{ik} - p_k)$,
   assign item j to bidder i,
   unassign potential other bidder s,

Comments & intuitions

* (4) is non increasing
* each item is likely to get assigned as they get more competitive as the price of the others goes up.
* may not converge in degenerate case
  → $p_j = p_j + (A_{ij} - p_j) - (A_{ik} - p_k) + \varepsilon$
  with $\varepsilon < 1/n$ and $A_{ij} \in \mathbb{N}$ $\Rightarrow$ approximation = optimal solution.

Parallel implementation.

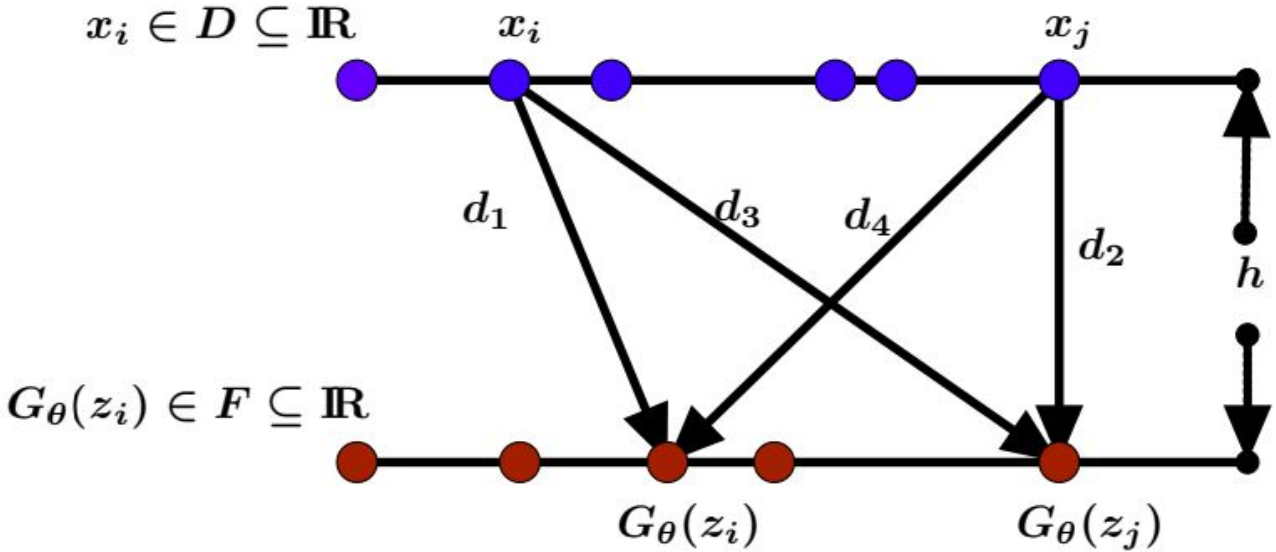
Gauss-Seidel : // on finding the best item for bidder i

Jacobi : // all unassigned bidders.

Approximation : Slice Wassertein distance

Generative Modeling using the Sliced Wasserstein Distance [Deshpande2018]

<u>Key idea</u> : project high dimensional data in 1D to get a fast approximation (nlog(n)) of EMD



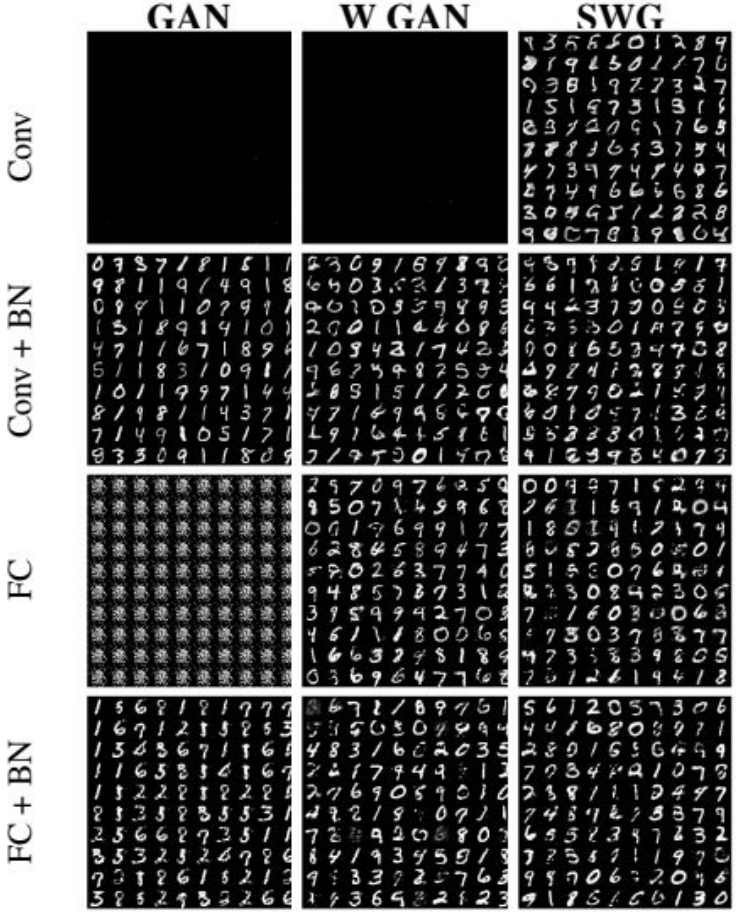


Figure 5. MNIST samples after 40k training iterations for different generator configurations. Batch size $= 250$, Learning rate $= 0.0005$, Adam optimizer

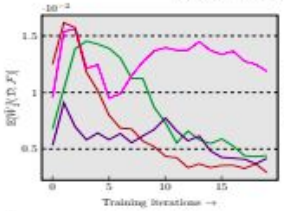
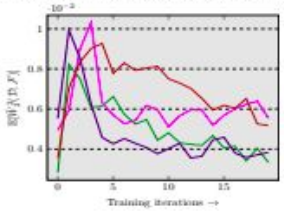
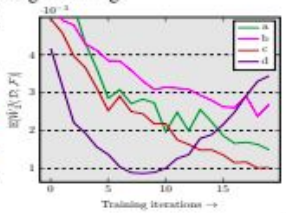
CIFAR-10 (32x32)	LSUN Bedrooms (64x64)	CelebA (64x64)
(a) G: DCGAN, D: DCGAN (with layernorm)		
		
(b) G: DCGAN, D: DCGAN with 64 filters in each layer (with layernorm)		
		
(c) G: DCGAN, D: DCGAN with 2x filters (with layernorm)		
		
(d) G: DCGAN with extra Conv2D layers, D: DCGAN (with layernorm)		
		
Evolution of the sliced Wasserstein distance through training		
		

Table 2. The SWG succeeds in training different architectures (a through d) on different datasets with same hyperparameters. Samples collected after 20 epochs of training with batch size $= 64$, learning rate $= 0.0001$, Adam optimizer

Almost Linear time Regularizations : Sinkhorn

M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport

J. Altschuler, J. Weed, and P. Rigollet. Near-linear time approximation algorithms for optimal transport via sinkhorn iteration.

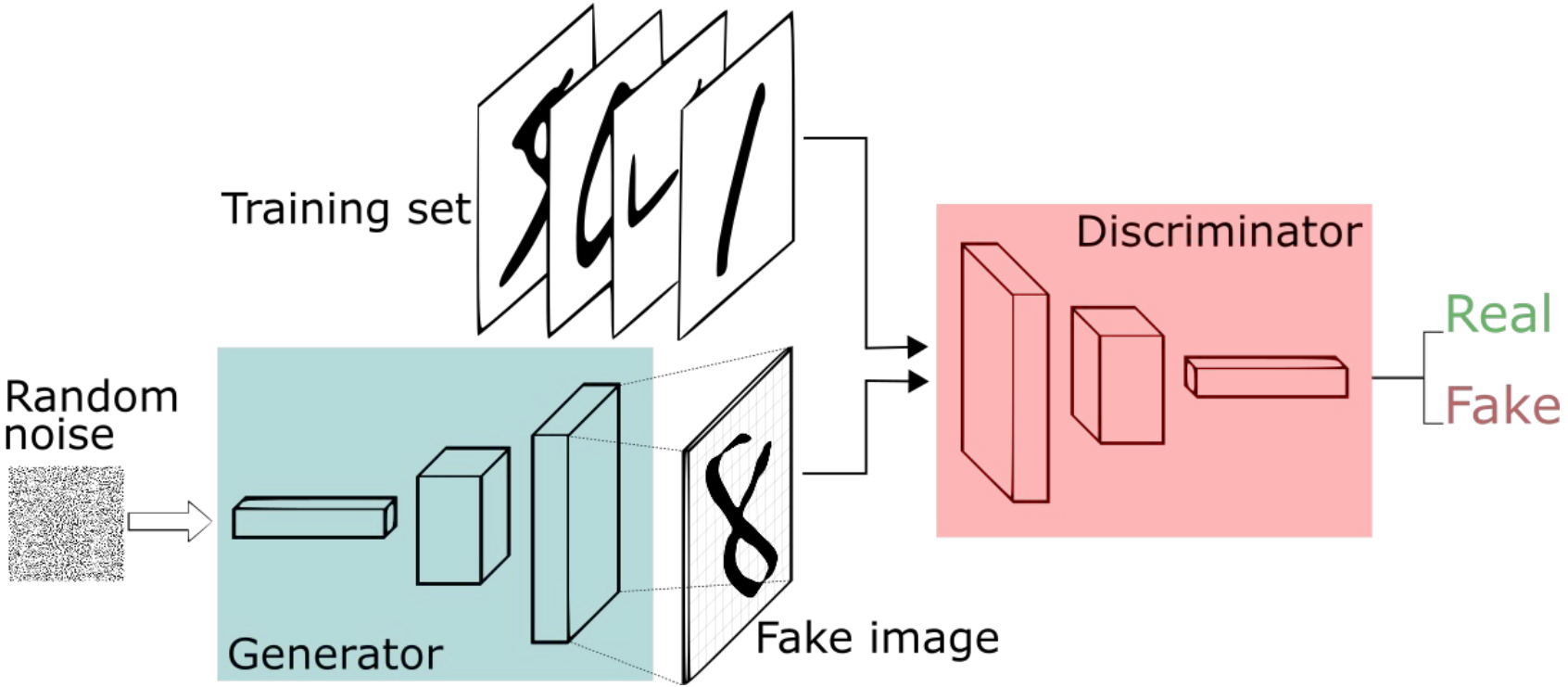
$$\max_{X \in \mathbb{R}^{n \times n}} \quad \sum_{i=1}^{n} \sum_{j=1} A_{ij} X_{ij} \boxed{+\lambda^{-1} \cdot H(X)}$$

$$\text{subject to} \quad \sum_{i=1}^{n} X_{ij} = 1, \quad i = 1, 2, \ldots, n$$

$$\sum_{j=1}^{n} X_{ij} = 1, \quad j = 1, 2, \ldots, n.$$

$$X_{ij} \in \{0, 1\}.$$

Wasserstein GAN



credits : https://skymind.ai/

Wasserstein GAN

- Goal of a GAN : learning a data distribution by learning a mapping between a random noise and the actual data space.
- Hard to train :
  - The loss is not a good indicator of the samples quality
  - Instable
  - Subject to mode collapse

Wasserstein GAN

- Goal of a GAN : learning a data distribution by learning a mapping between a random noise and the actual data space.
- Hard to train :
  - The loss is not a good indicator of the samples quality
  - Instable
  - Subject to mode collapse

Major interrogation : To what extent does the generated distribution approaches the real distribution ?

Wasserstein GAN

What notion of distance between 2 distributions ?

- The *Total Variation* (TV) distance

$$\delta(\mathbb{P}_r, \mathbb{P}_g) = \sup_{A \in \Sigma} |\mathbb{P}_r(A) - \mathbb{P}_g(A)| \ .$$

- The *Kullback-Leibler* (KL) divergence

$$KL(\mathbb{P}_r \| \mathbb{P}_g) = \int \log \left(\frac{P_r(x)}{P_g(x)} \right) P_r(x) d\mu(x)$$

- The *Jensen-Shannon* (JS) divergence

$$JS(\mathbb{P}_r, \mathbb{P}_g) = KL(\mathbb{P}_r \| \mathbb{P}_m) + KL(\mathbb{P}_g \| \mathbb{P}_m) \ ,$$

where $\mathbb{P}_m$ is the mixture $(\mathbb{P}_r + \mathbb{P}_g)/2$. This divergence is symmetrical and always defined because we can choose $\mu = \mathbb{P}_m$.

- The *Earth-Mover* (EM) distance or Wasserstein-1

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y) \sim \gamma} \big[\ \|x - y\| \ \big] \ , \qquad (1)$$

Wasserstein GAN - parallel lines example

Let $Z \sim U[0,1]$

Let $\mathbb{P}_0$ be the distribution of $(0, Z) \in \mathbb{R}^2$

Let $g_\theta(z) = (\theta, z)$ be a family of mappings between the distribution $U[0,1]$ and the distribution $\mathbb{P}_\theta$ with single parameter θ

In this case

- $W(\mathbb{P}_0, \mathbb{P}_\theta) = |\theta|,$
- $JS(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ , \end{cases}$
- $KL(\mathbb{P}_\theta \| \mathbb{P}_0) = KL(\mathbb{P}_0 \| \mathbb{P}_\theta) = \begin{cases} +\infty & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ , \end{cases}$
- and $\delta(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} 1 & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ . \end{cases}$

Wasserstein GAN - parallel lines example

Let $Z \sim U[0,1]$

Let $\mathbb{P}_0$ be the distribution of $(0, Z) \in \mathbb{R}^2$

Let $g_\theta(z) = (\theta, z)$ be a family of mappings between the distribution $U[0,1]$ and the distribution $\mathbb{P}_\theta$ with single parameter θ

In this case

- $W(\mathbb{P}_0, \mathbb{P}_\theta) = |\theta|,$
- $JS(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ , \end{cases}$
- $KL(\mathbb{P}_\theta \| \mathbb{P}_0) = KL(\mathbb{P}_0 \| \mathbb{P}_\theta) = \begin{cases} +\infty & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ , \end{cases}$
- and $\delta(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} 1 & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ . \end{cases}$

Wasserstein GAN

Theorem

Let $\mathbb{P}_r$ be a fixed distribution over $\mathcal{X}$.

Let Z be a random variable over another space $\mathcal{Z}$.

Let $g : \mathcal{Z} \times \mathbb{R}^d \to \mathcal{X}$ be a function.

Let $\mathbb{P}_\theta$ denote the distribution of $g_\theta(Z)$.

Then,

1. If g is continuous in θ, so is $W(\mathbb{P}_r, \mathbb{P}_\theta)$.
2. If g is locally Lipschitz and satisties some regularity assumption, then $W(\mathbb{P}_r, \mathbb{P}_\theta)$ is continous everywhere, and differentiable almost everywhere.
3. Statements 1-2 are false for the Jensen-Shannon divergence $JS(\mathbb{P}_r, \mathbb{P}_\theta)$ and all the KLs.

Wasserstein GAN

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[\, \|x - y\| \,\right] \longrightarrow$$ highly intractable

Wasserstein GAN

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[\, \|x - y\| \,\right] \longrightarrow$$ highly intractable

but we can consider Kantorovitch - Rubinstein duality !
https://vincentherrmann.github.io/blog/wasserstein/

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

Wasserstein GAN

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[\, \|x - y\| \,\right]$$ ⟶ highly intractable

but we can consider Kantorovitch - Rubinstein duality !
https://vincentherrmann.github.io/blog/wasserstein/

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

What about estimating f with a neural net ?

Wasserstein GAN

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[\, \|x - y\| \,\right]$$ ⟶ highly intractable

but we can consider Kantorovitch - Rubinstein duality !
https://vincentherrmann.github.io/blog/wasserstein/

$$\max_{w \in \mathcal{W}} \mathbb{E}_{x\sim\mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z\sim p(z)}[f_w(g_\theta(z)]$$

Wasserstein GAN

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

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

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

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

Wasserstein GAN -results

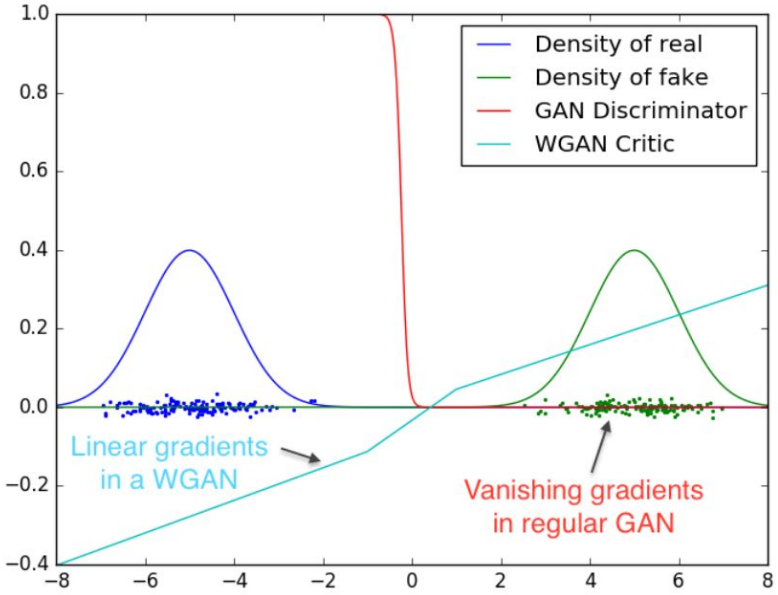


Figure 2: Optimal discriminator and critic when learning to differentiate two Gaussians. As we can see, the discriminator of a minimax GAN saturates and results in vanishing gradients. Our WGAN critic provides very clean gradients on all parts of the space.

Wasserstein GAN -results

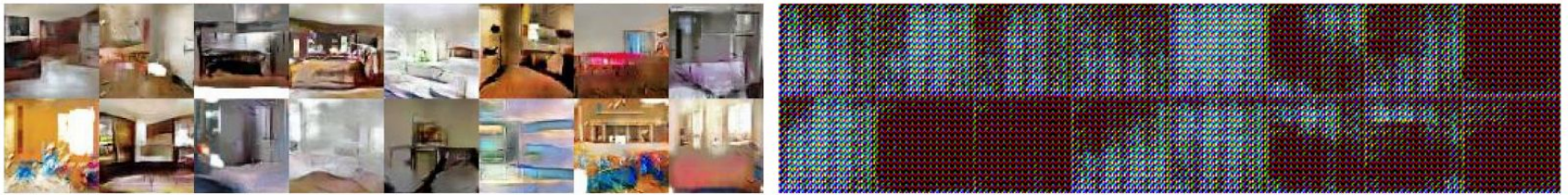


Figure 6: Algorithms trained with a generator without batch normalization and constant number of filters at every layer (as opposed to duplicating them every time as in [18]). Aside from taking out batch normalization, the number of parameters is therefore reduced by a bit more than an order of magnitude. Left: WGAN algorithm. Right: standard GAN formulation. As we can see the standard GAN failed to learn while the WGAN still was able to produce samples.

Gan Discussion

Why not estimate directly EMD in the primal ?

1. Use the discrete case on batch k -> Reach optimal solution of the primal of discrete EMD (noisy estimate) on the natural distributions on images
2. Use a discriminator : estimate of the dual of continuous EMD

(2) is better than (1) because of the curse of dimensionality.

(1) Relies on euclidean metric (not great on images) whereas (2) learns it’s metric (based on convolutions in the discriminator) as far as the exploration of the IPM space is not perfect.

Bibliography

http://www.math.harvard.edu/archive/20_spring_05/handouts/assignment_overheads.pdf
http://cedricvillani.org/wp-content/uploads/2012/08/preprint-1.pdf : Chapter 6
https://theory.epfl.ch/courses/topicstcs/Lecture52015.pdf
https://stanford.edu/~rezab/classes/cme323/S16/projects_reports/jin.pdf
https://vincentherrmann.github.io/blog/wasserstein/
[Fan2016] Fan, Haoqiang, Hao Su, and Leonidas J. Guibas. "A Point Set Generation Network for 3D Object Reconstruction from a Single Image." CVPR. Vol. 2. No. 4. 2017.
[Groueix2018] Groueix, T., Fisher, M., Kim, V., Russell, B., and Aubry, M. (2018, June). AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In CVPR 2018.
[Li2018] Supervised Fitting of Geometric Primitives to 3D Point Clouds.
[Arjovsky2017] Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein gan. arXiv preprint arXiv:1701.07875.
[Bertsekas1981] DP Bertsekas (1981) A new algorithm for the assignment problem
[Deshpande2018] Generative Modeling using the Sliced Wasserstein Distance
[Cuturi2013] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport
[Altschuler2017] J. Altschuler, J. Weed, and P. Rigollet. Near-linear time approximation algorithms for optimal transport via sinkhorn iteration.