Stochastic Optimization for Large Scale Optimal Transport

Aude Genevay

CEREMADE - Université Paris Dauphine
INRIA - Mokaplan project-team
DMA - Ecole Normale Supérieure

Journée Algorithmes Stochastiques - Dauphine - Decembre 2017

Joint work with F. Bach, M.Cuturi, G. Peyré

Recurrent issue in ML : Comparing probability distributions

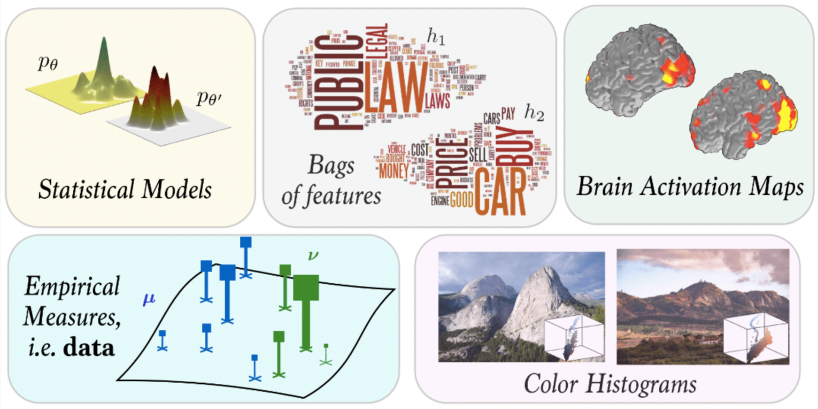
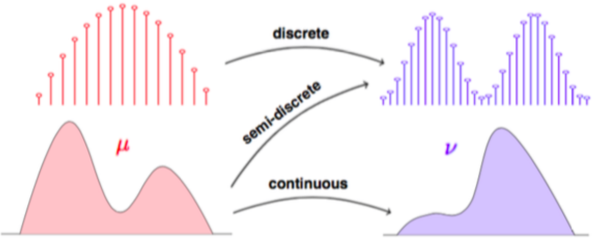


Figure 1: Many objects can be viewed as probability distributions (courtesy of M. Cuturi)

Optimal Transport and the Wasserstein Distance



- Optimal Transport : find coupling that minimizes total cost of moving μ to ν whith unit cost function **c**
- Constrained problem : coupling has fixed marginals
- Minimal cost of moving μ to ν(e.g. solution of the OT problem) is called the **Wasserstein distance** (it's an actual distance!)

OT for ML problems

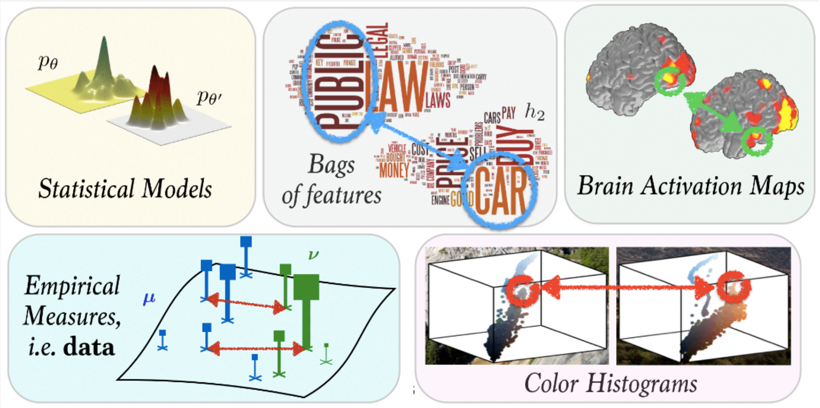


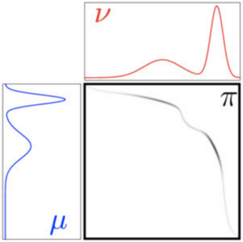
Figure 2: OT gives a natural framework for distances between probability distributions that takes geometry into account (courtesy of M. Cuturi)

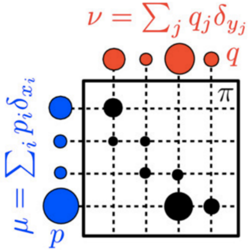
Optimal Transport

Two positive Radon measures μ on $\mathcal{X}$ and ν on $\mathcal{Y}$ of mass 1
Cost $c(x, y)$ to move a unit of mass from x to y
Set of couplings with marginals μ and ν

$$\Pi(\mu, \nu) \stackrel{\text{def.}}{=} \{\pi \in \mathcal{M}_+^1(\mathcal{X} \times \mathcal{Y}) \mid \pi(A \times \mathcal{Y}) = \mu(A), \pi(\mathcal{X} \times B) = \nu(B)\}$$

What's the coupling that minimizes the total cost?





Kantorovitch Formulation of OT

The optimal overall cost for transporting μ to ν is given by

$$W(\mu, \nu) = \min_{\pi \in \Pi(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{Y}} c(x, y) \mathrm{d}\pi(x, y) \qquad (\mathcal{P}_\varepsilon)$$

Kantorovitch Formulation of OT

The optimal overall cost for transporting μ to ν is given by

$$W_\varepsilon(\mu,\nu) = \min_{\pi\in\Pi(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{Y}} c(x,y)\mathrm{d}\pi(x,y) + \varepsilon\,\mathrm{KL}(\pi|\mu\otimes\nu) \qquad (\mathcal{P}_\varepsilon)$$

where

$$\mathrm{KL}(\pi|\mu\otimes\nu) \overset{\text{def.}}{=} \int_{\mathcal{X}\times\mathcal{Y}} \big(\log\big(\frac{\mathrm{d}\pi}{\mathrm{d}\mu\mathrm{d}\nu}(x,y)\big) - 1\big)\mathrm{d}\pi(x,y)$$

Entropy!

- Basically : Adding an entropic regularization smoothes the constraint
- Makes the problem easier :
  - yields an unconstrained dual problem
  - discrete case can be solved efficiently with alternate maximizations on the dual variables : Sinkhorn's algorithm (more on that later)
- For ML applications, regularized Wasserstein is better than standard one
- In high dimension, helps avoiding overfitting

Reminder on convex duality

Primal problem:

$$\begin{array}{ll} \min_x & f(x) \\ \text{subject to} & h_i(x) = 0 \quad \text{for i} = 1 \ldots \text{m} \end{array}$$

Lagrange dual function:

$$g(\lambda) = \min_x f(x) + \sum_{i=1}^{m} \lambda_i h_i(x)$$

Dual problem:

$$\max_\lambda g(\lambda)$$

Under good assumptions, both problems are equivalent.

Dual formulation of OT

$$W(\mu, \nu) = \max_{u \in \mathcal{C}(\mathcal{X}), v \in \mathcal{C}(\mathcal{Y})} \int_{\mathcal{X}} u(x) \mathrm{d}\mu(x) + \int_{\mathcal{Y}} v(y) \mathrm{d}\nu(y) - \iota_{U_c}(u, v) \quad (\mathcal{D}_\varepsilon)$$

where the constraint set U_c is defined by

$$U_c \overset{\text{def.}}{=} \{(u, v) \in \mathcal{C}(\mathcal{X}) \times \mathcal{C}(\mathcal{Y}) \; ; \; \forall (x, y) \in \mathcal{X} \times \mathcal{Y}, u(x) + v(y) \leq c(x, y)\}$$

Dual formulation of OT (with entropy)

$$W_\varepsilon(\mu,\nu) = \max_{u\in\mathcal{C}(\mathcal{X}),v\in\mathcal{C}(\mathcal{Y})} \int_\mathcal{X} u(x)\mathrm{d}\mu(x) + \int_\mathcal{Y} v(y)\mathrm{d}\nu(y) - \iota^\varepsilon_{U_c}(u,v)$$

and the smoothed indicator is

$$\iota^\varepsilon_{U_c}(u,v) \stackrel{\text{def.}}{=} \varepsilon \int_{\mathcal{X}\times\mathcal{Y}} \exp\left(\frac{u(x)+v(y)-c(x,y)}{\varepsilon}\right)\mathrm{d}\mu(x)\mathrm{d}\nu(y)$$

Semi-Dual formulation of OT

The dual problem is convex in u and v. We fix v and minimize over u.

Semi-Dual formulation of OT

The dual problem is convex in u and v. We fix v and minimize over u. This yields :

$$u(x) \stackrel{\text{def.}}{=} \min_{y \in \mathcal{Y}} c(x,y) - v(y)$$

Semi-Dual formulation of OT

The dual problem is convex in u and v. We fix v and minimize over u. This yields :

$$u(x) \overset{\text{def.}}{=} \min_{y\in\mathcal{Y}} c(x,y) - v(y)$$

Plugging back in the dual :

$$\begin{aligned} W_\varepsilon(\mu,\nu) &= \max_{v\in\mathcal{C}(\mathcal{Y})} \int_{\mathcal{X}} \min_{y\in\mathcal{Y}} \left(c(x,y) - v(y)\right) \mathrm{d}\mu(x) + \int_{\mathcal{Y}} v(y)\mathrm{d}\nu(y) - \varepsilon \\ &= \max_{v\in\mathcal{C}(\mathcal{Y})} \mathbb{E}_\mu \Big[\min_{y\in\mathcal{Y}} \left(c(x,y) - v(y)\right) + \int_{\mathcal{Y}} v(y)\mathrm{d}\nu(y) - \varepsilon\Big] \end{aligned}$$

Semi-Dual formulation of OT (with entropy)

The dual problem is convex in u and v. We fix v and minimize over u.

Semi-Dual formulation of OT (with entropy)

The dual problem is convex in u and v. We fix v and minimize over u. This yields :

$$u(x) \stackrel{\text{def.}}{=} -\varepsilon \log \left(\int_{\mathcal{Y}} \exp\left(\frac{v(y) - c(x,y)}{\varepsilon}\right) \mathrm{d}\nu(y) \right)$$

Semi-Dual formulation of OT (with entropy)

The dual problem is convex in u and v. We fix v and minimize over u. This yields :

$$u(x) \stackrel{\text{def.}}{=} -\varepsilon \log\left(\int_{\mathcal{Y}} \exp\left(\frac{v(y) - c(x,y)}{\varepsilon}\right)\mathrm{d}\nu(y)\right)$$

Plugging back in the dual :

$$\begin{aligned} W_\varepsilon(\mu,\nu) &= \max_{v \in \mathcal{C}(\mathcal{Y})} \int_{\mathcal{X}} -\varepsilon \log\left(\int_{\mathcal{Y}} \exp\left(\frac{v(y) - c(x,y)}{\varepsilon}\right)\mathrm{d}\nu(y)\right)\mathrm{d}\mu(y) \\ &\quad + \int_{\mathcal{Y}} v(y)\mathrm{d}\nu(y) - \varepsilon \\ &= \max_{v \in \mathcal{C}(\mathcal{Y})} \mathbb{E}_\mu\Big[-\varepsilon \log\left(\int_{\mathcal{Y}} \exp\left(\frac{v(y) - c(x,y)}{\varepsilon}\right)\right) \\ &\quad + \int_{\mathcal{Y}} v(y)\mathrm{d}\nu(y) - \varepsilon\Big] \end{aligned}$$

We consider 2 frameworks :

- Semi-Discrete : μ is continuous and $\nu = \sum_{j=1}^{M} \nu_i \delta y_j$ The optimization problem is

$$\max_{v \in \mathbb{R}^M} \mathbb{E}_\mu \left[-\varepsilon \log \left(\sum_{j=1}^{M} \exp\left(\frac{v(y_j) - c(x, y_j)}{\varepsilon}\right) \right) + \sum_{j=1}^{M} v(y_j) \nu_j - \varepsilon \right]$$

We consider 2 frameworks :

- Semi-Discrete : μ is continuous and $\nu = \sum_{j=1}^{M} \nu_i \delta y_j$ The optimization problem is

$$\max_{v \in \mathbb{R}^M} \mathbb{E}_\mu \left[-\varepsilon \log \left(\sum_{j=1}^{M} \exp\left(\frac{v(y_j) - c(x, y_j)}{\varepsilon} \right) \right) + \sum_{j=1}^{M} v(y_j)\nu_j - \varepsilon \right]$$

- Discrete : $\mu = \sum_{i=1}^{N} \mu_i \delta x_i$ and $\nu = \sum_{j=1}^{M} \nu_i \delta y_j$ The optimization problem is

$$\max_{v \in \mathbb{R}^M} \sum_{i=1}^{N} \left[-\varepsilon \log \left(\sum_{j=1}^{M} \exp\left(\frac{v(y_j) - c(x_i, y_j)}{\varepsilon} \right) \right) + \sum_{j=1}^{M} v(y_j)\nu_j - \varepsilon \right] \mu_i$$

Stochastic Optimization

Computing the full gradient is

- Hard in the semi-discrete setting (even impossible if we don't know μ explicitly)

Stochastic Optimization

Computing the full gradient is

- Hard in the semi-discrete setting (even impossible if we don't know μ explicitly)
- Very costly in the discrete case since we need to compute N gradients and sum them.

The idea of stochastic optimization is to use approximate gradients so that each iteration is inexpensive.

Stochastic Optimization I

- Goal : maximize $H_\varepsilon(v) = \mathbb{E}_\mu\left[h_\varepsilon(X, v)\right]$ over v in $\mathbb{R}^M$.
- Standard gradient ascent :

$$v^{(k)} = v^{(k-1)} + \nabla_v H_\varepsilon(v^{(k-1)})$$

- The whole gradient $\nabla_v H_\varepsilon(v)$ is too costly/complicated to compute
- Idea : Sample x from μ and use $\nabla_v h_\varepsilon(x, v)$ as a proxy for the full gradient in the gradient ascent.

Stochastic Optimization II

Algorithm 1 Averaged SGD

Input: C

Output: v

$v \leftarrow \mathbb{0}_M$, $\bar{v} \leftarrow v$

for $k = 1, 2, \dots$ **do**

Sample x_k from μ

$v \leftarrow v + \frac{C}{\sqrt{k}} \nabla_v h_\varepsilon(x_k, v)$ (gradient ascent step)

$\bar{v} \leftarrow \frac{1}{k} v + \frac{k-1}{k} \bar{v}$ (averaging)

end for

- cost of each iteration M
- convergence rate $O(1/\sqrt{(}k))$

Discrete OT : Sinkhorn's Algorithm I

State-of-the-art : Sinkhorn's Algorithm

- Two equivalent views
  - Alternate projections on the constraints of the primal
  - Alternate minimizations on the dual
- Iterates $a \stackrel{\text{def.}}{=} \exp(\frac{u}{\varepsilon})$ and $b \stackrel{\text{def.}}{=} \exp(\frac{v}{\varepsilon})$: $\begin{cases} a = \frac{1}{K(b \odot \nu)} \\ b = \frac{1}{K^T(a \odot \mu)} \end{cases}$

  where $K \stackrel{\text{def.}}{=} \exp \frac{-\mathbf{c}}{\varepsilon}$ and $\odot$ is coordinatewise vector multiplication.
- Linear convergence of the iterates to the optimizers

Discrete OT : Sinkhorn's Algorithm II

Algorithm 2 Sinkhorn

Output: $\mathbf{v}$

$\mathbf{b} \leftarrow \mathbb{1}_J$

for $k = 1, 2, \ldots$ **do**

$\quad \mathbf{a} \leftarrow \frac{\mathbb{1}_I}{K(\boldsymbol{\nu} \odot \mathbf{b})}$

$\quad \mathbf{b} \leftarrow \frac{\mathbb{1}_J}{K^\top(\boldsymbol{\mu} \odot \mathbf{a})}$

end for

$\mathbf{v} \leftarrow \varepsilon \log(\mathbf{b})$

$\Rightarrow$ Implies matrix vector multiplications at each iteration : cost $I \times J$ per iteration

Stochastic Optimization : Case of a Finite Sum I

When μ is also a discrete measure, we are minimizing a finite sum of N functionals :

$$\max_{v \in \mathbb{R}^M} \sum_{i=1}^{N} \left[-\varepsilon \log \left(\sum_{j=1}^{M} \exp\left(\frac{v(y_j) - c(x_i, y_j)}{\varepsilon}\right) \right) + \sum_{j=1}^{M} v(y_j)\nu_j - \varepsilon \right] \mu_i$$

Stochastic Optimization : Case of a Finite Sum II

A more efficient stochastic algorithm consists in using an average of the past gradients as a proxy for the full gradient :

- At iteration k, an index i is drawn. Its gradient $\nabla_v h_\varepsilon(x_i, v^{(k)})$ is updated in the vector of partial gradients (vector with N entries kept in memory).
- The average gradient is updated accordingly, and used in a step of the gradient ascent

Stochastic Optimization : Case of a Finite Sum III

Algorithm 3 SAG for Discrete OT

Input: C
Output: $\mathbf{v}$

$\mathbf{v} \leftarrow \mathbb{0}_M$, $\mathbf{d} \leftarrow \mathbb{0}_J$, $\forall i, \mathbf{g}_i \leftarrow \mathbb{0}_M$
for $k = 1, 2, \dots$ **do**
  Sample $i \in \{1, 2, \dots, I\}$ uniform.
  $\mathbf{d} \leftarrow \mathbf{d} - \mathbf{g}_i$
  $\mathbf{g}_i \leftarrow \boldsymbol{\mu}_i \nabla_v \bar{h}_\varepsilon(x_i, \mathbf{v})$
  $\mathbf{d} \leftarrow \mathbf{d} + \mathbf{g}_i$; $\mathbf{v} \leftarrow \mathbf{v} + C\mathbf{d}$
end for

- cost of each iteration M
- convergence rate $O(1/k)$

Stochastic Optimization : Case of a Finite Sum IV

$\Rightarrow$ Slower convergence rate than Sinkhorn but *online* algorithm, better for (very) large-scale problems

Numerical Results for Word Mover's Distance (Discrete OT)

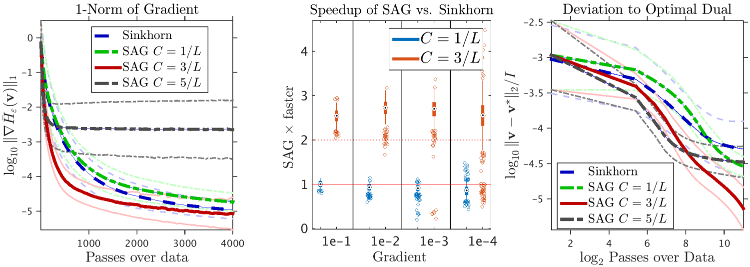
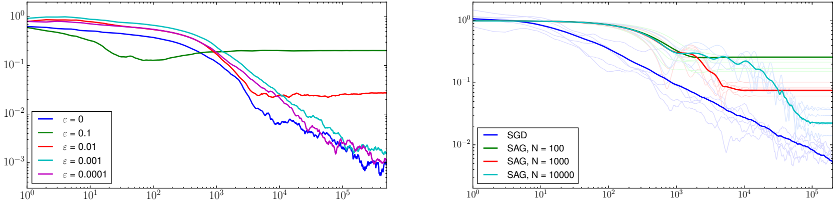


Figure 3: Results for the computation of 595 pairwise word mover's distances between 35 very large corpora of text, each represented as a cloud of $I = 20,000$ word embeddings.

Numerical Results for Density Fitting (Semi-discrete OT)



(a) SGD (b) SGD vs. SAG

Figure 4: (a) Plot of $\|\mathbf{v}_k - \mathbf{v}_0^\star\|_2 / \|\mathbf{v}_0^\star\|_2$ as a function of k, for SGD and different values of ε ($\varepsilon = 0$ being un-regularized). (b) Plot of $\|\mathbf{v}_k - \mathbf{v}_\varepsilon^\star\|_2 / \|\mathbf{v}_\varepsilon^\star\|_2$ averaged over 40 runs as a function of k, for SGD and SAG with different number N of samples, for regularized OT using $\varepsilon = 10^{-2}$.

Density Fitting

- Observed dataset $(y_1, \dots, y_n) \in \mathcal{X}$ (IID assumption)
- Empirical measure $\hat{\nu} = \frac{1}{n} \sum_{i=1}^n \delta_{y_i}$
- Parametric model $(\mu_\theta)_{\theta \in \Theta}$
- Goal : find $\hat{\theta} = \arg\min_{\theta \in \Theta} \mathcal{L}(\mu_\theta, \hat{\nu})$ where $\mathcal{L}$ is a loss on measures.
- If we assume (μ_θ) has density $(f_\theta)_{\theta \in \Theta}$ problem is solved with **Maximum Likelikood Estimator**

$$\hat{\theta} \stackrel{\text{def.}}{=} \underset{\theta \in \Theta}{\arg\min} - \sum_{i=1}^{n} \log f(y_i \mid \theta)$$

Generative Models

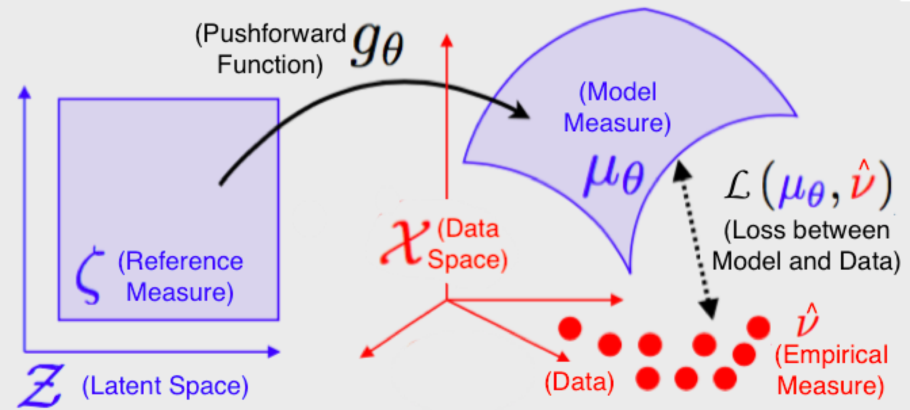


Figure 5: Illustration of Density Fitting on a Generative Model

Density Fitting for Generative Models I

- Parametric model : $\mu_\theta = g_{\theta\sharp}\zeta$
- ζ reference measure on (low dimensional) latent space $\mathcal{Z}$
- $g_\theta : \mathcal{Z} \to \mathcal{X}$ from latent space to data space
- Sampling procedure : $x \sim \mu_\theta$ obtained by $x = g_\theta(z)$ were $z \sim \zeta$
- Very popular topic in ML : image generation

Density Fitting for Generative Models II

- Generative Models usually supported on low dimensional manifolds (dim $\mathcal{Z}$ < dim $\mathcal{X}$)
- μ_θ doesn't have density wrt Lebesgue measure

$\Rightarrow$ **MLE can't be applied in this context!**

- 2 natural candidates emerge for $\mathcal{L}$
  - Maximum Mean Discrepency (based on Reproducing Kernel Hilbert Spaces) $\rightarrow$ Hilbertian norm
  - The Wasserstein Distance (based on Optimal Transport) $\rightarrow$ Non-Hilbertian distance

Density Fitting with Sinkhorn loss "Formally"

Define the Sinkhorn loss between two measures μ, ν as:

$$\bar{W}_{c,\varepsilon}(\mu,\nu) = 2W_{c,\varepsilon}(\mu,\nu) - W_{c,\varepsilon}(\mu,\mu) - W_{c,\varepsilon}(\nu,\nu)$$

Solve $\min_\theta E(\theta)$

where $E(\theta) \stackrel{\text{def.}}{=} \bar{W}_{c,\varepsilon}(\mu_\theta, \nu)$

$\Rightarrow$ Issue : untractable gradient

Approximating Sinkhorn loss

- Rather than approximating the gradient approximate the loss itself
- Minibatches : $\hat{E}(\theta)$
  - sample $x_1, \ldots, x_m$ from μ_θ
  - use empirical Wasserstein distance $W_{c,\varepsilon}(\hat{\mu}_\theta, \hat{\nu})$ where $\hat{\mu}_\theta = \frac{1}{N} \sum_{i=1}^{m} \delta_{x_i}$
- Use L iterations of Sinkhorn's algorithm : $\hat{E}^{(L)}(\theta)$
  - compute L steps of the algorithm
  - use this as a proxy for $W(\hat{\mu}_\theta, \nu)$

Computing the Gradient in Practice

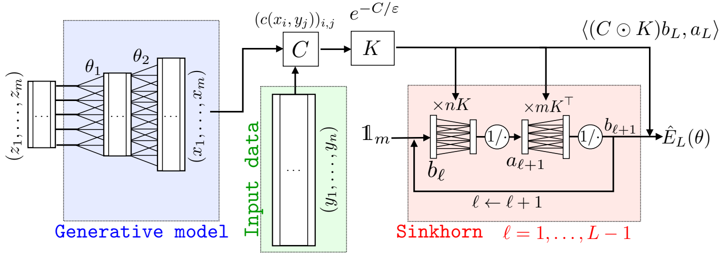
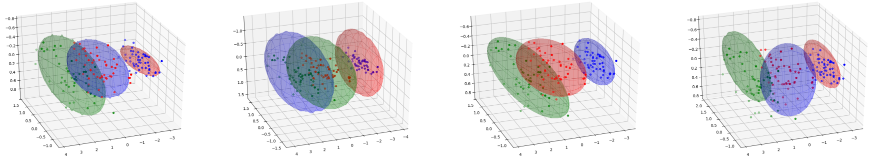


Figure 6: Scheme of the loss approximation

- Compute *exact* gradient of $\hat{E}^{(L)}(\theta)$ with autodiff
- Backpropagation through above graph
- Same computational cost as evaluation of $\hat{E}^{(L)}(\theta)$

Numerical Results : a toy example



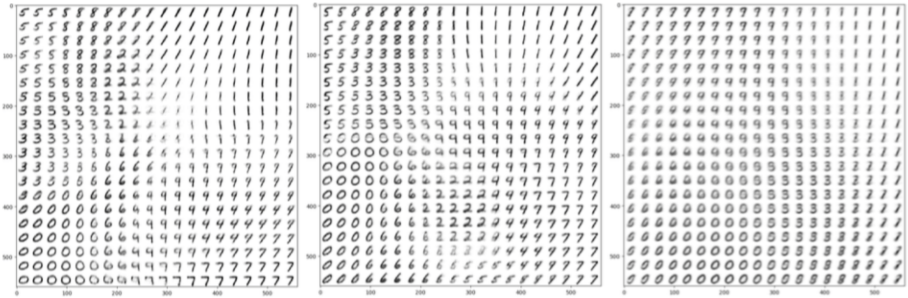
(a) MMD (b) $\varepsilon = 1$ (c) $\varepsilon = 0.1$ (d) $\varepsilon = 0.01$

Figure 7: Ellipses after convergence of the stochastic gradient descent with $L = 20$, $m = 200$

Numerical Results on MNIST (L2 cost)

Figure 8: Samples from MNIST dataset

Numerical Results on MNIST (L2 cost)



(a) $\varepsilon = 1, m = 200, L = 10$ (b) $\varepsilon = 10^{-1}, m = 200, L = 100$ (c) $\varepsilon = 10^{-1}, m = 10, L = 300$

Figure 9: Manifolds in the latent space for various parameters

Learning the cost [Li et al. '17, Bellemare et al. '17]

- On complex data sets, choice of a good ground metric c is not trivial
- Use parametric cost function $c_\phi(x, y) = \|f_\phi(x) - f_\phi(y)\|_2^2$ (where $f_\phi : \mathcal{X} \to \mathbb{R}^d$)
- Optimization problem becomes minmax (like GANs)

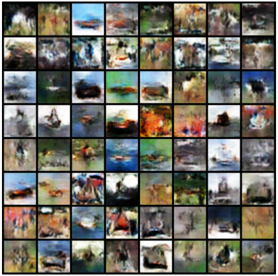
$$min_\theta max_\phi \bar{W}_{c_\phi,\varepsilon}(\mu_\theta, \nu)$$

- Same approximations but alternate between updating the cost parameters ϕ and the measure parameters θ

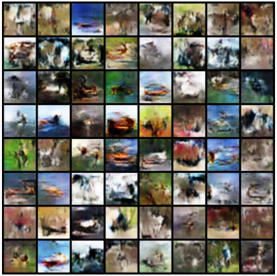
Numerical Results on CIFAR (learning the cost)

Figure 10: Samples from CIFAR dataset

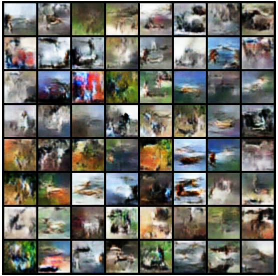
Numerical Results on CIFAR (learning the cost)



(a) MMD



(b) $\varepsilon = 1000$



(c) $\varepsilon = 10$

Figure 11: Samples from the generator trained on CIFAR 10 for MMD and Sinkhorn loss (coming from the same samples in the latent space)

Which is better? Not just about generating nice images, but more about capturing a high dimensional distribution... Hard to evaluate.