

Optimal Transport for Machine Learning

Wenbo Gong and Mark Rowland

November 16, 2017

Metrics on probability distributions

Measuring distances between probability distributions (or measures) is important for many domains, including:

Machine Learning

- Classification (computing losses/gradients on predictions)

- Domain adaptation

- Distributional reinforcement learning

Computational Statistics

- Studying convergence properties of Monte Carlo samplers (“How far is my sampler from convergence?”)

- Hypothesis testing

Operations Research

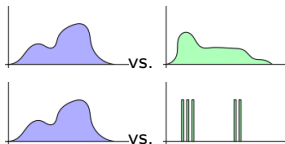
- Solving supply chain/transportation problems

And more...

There is a lot of flexibility in choosing a probability metric; it can be easy to not give as much thought as we ought to as to *how* we should measure these distances...

Motivating problems

How to compare probability distributions?



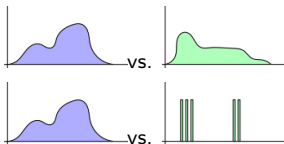
Kullback-Leibler divergence

$$KL(q||p) = \mathbb{E}_{X \sim q} [\log(q(X)/p(X))]$$

+ Connections to information
theory/max-likelihood

Motivating problems

How to compare probability distributions?



Kullback-Leibler divergence

$$KL(q||p) = \mathbb{E}_{X \sim q} [\log(q(X)/p(X))]$$

+ Connections to information
theory/max-likelihood

q target distribution, p approximating distribution.

With samples $x_1, \dots, x_n \stackrel{iid}{\sim} q$, minimising the KL estimator

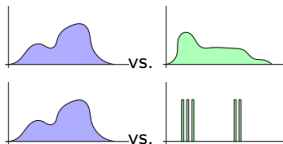
$$\min_p \widehat{KL}(q||p) = \sum_{i=1}^n \log(q(x_i)/p(x_i))$$

is *exactly* maximum likelihood estimation.

Very well-understood, supported by a lot of statistical theory e.g. Cramér-Rao information-theoretic lower bound.

Motivating problems

How to compare probability distributions?



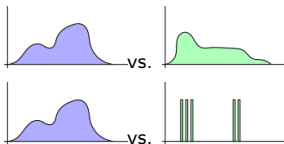
Kullback-Leibler divergence

$$KL(q||p) = \mathbb{E}_{X \sim q} [\log(q(X)/p(X))]$$

- + Connections to information theory/max-likelihood
- + Unbiased (gradient) estimators via samples from p

Motivating problems

How to compare probability distributions?



Kullback-Leibler divergence

$$KL(q||p) = \mathbb{E}_{X \sim q} [\log(q(X)/p(X))]$$

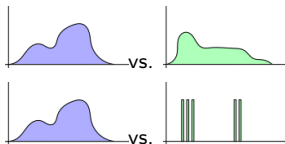
- + Connections to information theory/max-likelihood
- + Unbiased (gradient) estimators via samples from p

p target distribution, q_θ parametrised approximating distribution.

Can compute unbiased gradients $\widehat{\nabla}_\theta KL(q_\theta||p)$ by sampling from q and using score estimator (REINFORCE)/reparametrisation trick.

Motivating problems

How to compare probability distributions?



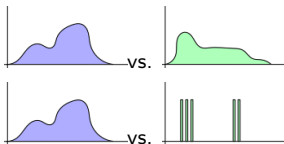
Kullback-Leibler divergence

$$KL(q||p) = \mathbb{E}_{X \sim q} [\log(q(X)/p(X))]$$

- + Connections to information theory/max-likelihood
- + Unbiased (gradient) estimators via samples from p
- Requires that both distributions have known densities*

Motivating problems

How to compare probability distributions?



Kullback-Leibler divergence

$$KL(q||p) = \mathbb{E}_{X \sim q} [\log(q(X)/p(X))]$$

- + Connections to information theory/max-likelihood
- + Unbiased (gradient) estimators via samples from p

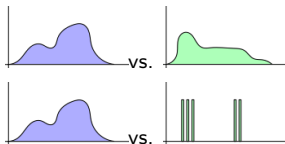
- Requires that both distributions have known densities*

We can't make sense of $KL(\delta_0||N(0, 1))$, for example.

Even if both distributions have densities, we may not be able to write them down (e.g. implicit generative models).

Motivating problems

How to compare probability distributions?



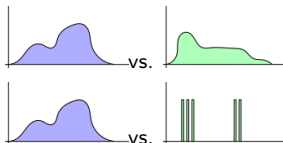
Kullback-Leibler divergence

$$KL(q||p) = \mathbb{E}_{X \sim q} [\log(q(X)/p(X))]$$

- + Connections to information theory/max-likelihood
- + Unbiased (gradient) estimators via samples from p
- Requires that both distributions have known densities*
- No information about metrics in underlying base space used

Motivating problems

How to compare probability distributions?



Kullback-Leibler divergence

$$KL(q||p) = \mathbb{E}_{X \sim q} [\log(q(X)/p(X))]$$

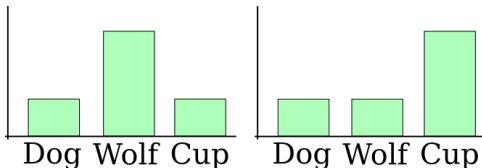
- + Connections to information theory/max-likelihood
- + Unbiased (gradient) estimators via samples from p
- Requires that both distributions have known densities*
- No information about metrics in underlying base space used



Siberian husky



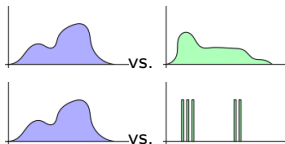
Eskimo dog



[Image credit: *Learning with a Wasserstein Loss*, Frogner et al. 2015]

Motivating problems

How to compare probability distributions?



Kullback-Leibler divergence

$$KL(q||p) = \mathbb{E}_{X \sim q} [\log(q(X)/p(X))]$$

- + Connections to information theory/max-likelihood
- + Unbiased (gradient) estimators via samples from p
- Requires that both distributions have known densities*
- No information about metrics in underlying base space used

Optimal Transport (Wasserstein) distances

$$W_1(p, q) = \inf_{\substack{(X, Y) \sim T \\ X \sim p, Y \sim q}} \mathbb{E} [d(X, Y)]$$

- + Can use information about base space
- + Can compare e.g. discrete and continuous distributions
- Unbiased estimation in intractable cases is very difficult
- Requires solving an optimisation problem to calculate, even in tractable cases

Plan

Part I : Intro to Optimal Transport

- Discrete case

- Generalisation to arbitrary probability measures

- Algorithmic considerations

Part II : Applications to Generative Modelling

- Kantorovich-Rubenstein duality

- Scalable implementaton for deep learning

Disclaimer: there are important theoretical details and conditions for some of the theory we'll talk through. I'll mention these points verbally, but won't dwell on them. Villani's book is great for the precise details.

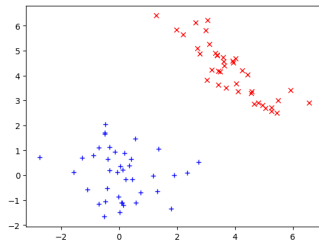
Discrete case

$$p = \sum_{i=1}^M p_i \delta_{x_i}, \quad q = \sum_{j=1}^N q_j \delta_{y_j}$$

Source locations x_1, \dots, x_M , with proportions p_1, \dots, p_M of mass.

Target locations y_1, \dots, y_N , with target proportions q_1, \dots, q_N .

How to move mass from sources to targets in cheapest possible way?



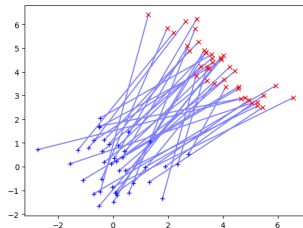
Discrete case

$$p = \sum_{i=1}^M p_i \delta_{x_i}, \quad q = \sum_{j=1}^N q_j \delta_{y_j}$$

Source locations x_1, \dots, x_M , with proportions p_1, \dots, p_M of mass.

Target locations y_1, \dots, y_N , with target proportions q_1, \dots, q_N .

How to move mass from sources to targets in cheapest possible way?



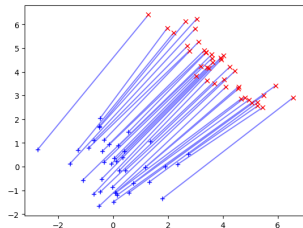
Discrete case

$$p = \sum_{i=1}^M p_i \delta_{x_i}, \quad q = \sum_{j=1}^N q_j \delta_{y_j}$$

Source locations x_1, \dots, x_M , with proportions p_1, \dots, p_N of mass.

Target locations y_1, \dots, y_N , with target proportions q_1, \dots, q_N .

How to move mass from sources to targets in cheapest possible way?



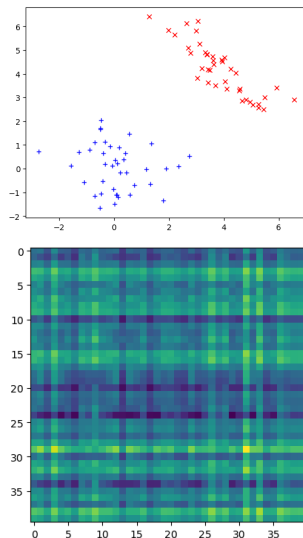
Discrete case

$$p = \sum_{i=1}^M p_i \delta_{x_i}, \quad q = \sum_{j=1}^N q_j \delta_{y_j}$$

Source locations x_1, \dots, x_M , with proportions p_1, \dots, p_M of mass.

Target locations y_1, \dots, y_N , with target proportions q_1, \dots, q_N .

$$C_{ij} = d(x_i, y_j)$$



Discrete case

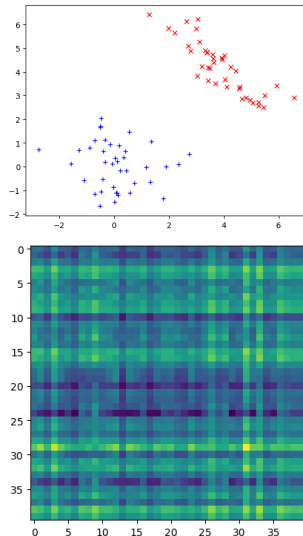
$$p = \sum_{i=1}^M p_i \delta_{x_i}, \quad q = \sum_{j=1}^N q_j \delta_{y_j}$$

Source locations x_1, \dots, x_M , with proportions p_1, \dots, p_M of mass.

Target locations y_1, \dots, y_N , with target proportions q_1, \dots, q_N .

$$C_{ij} = d(x_i, y_j)$$

T_{ij} = mass to be moved from x_i to y_j



Discrete case

$$p = \sum_{i=1}^M p_i \delta_{x_i}, \quad q = \sum_{j=1}^N q_j \delta_{y_j}$$

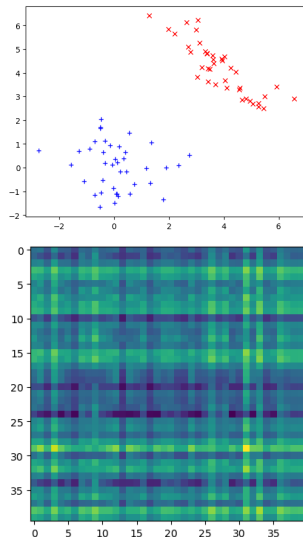
Source locations x_1, \dots, x_M , with proportions p_1, \dots, p_M of mass.

Target locations y_1, \dots, y_N , with target proportions q_1, \dots, q_N .

$$C_{ij} = d(x_i, y_j)$$

T_{ij} = mass to be moved from x_i to y_j

Minimise total cost: $\sum_{i=1}^M \sum_{j=1}^N C_{ij} T_{ij}$.



Discrete case

$$p = \sum_{i=1}^M p_i \delta_{x_i}, \quad q = \sum_{j=1}^N q_j \delta_{y_j}$$

Source locations x_1, \dots, x_M , with proportions p_1, \dots, p_M of mass.

Target locations y_1, \dots, y_N , with target proportions q_1, \dots, q_N .

$$C_{ij} = d(x_i, y_j)$$

T_{ij} = mass to be moved from x_i to y_j

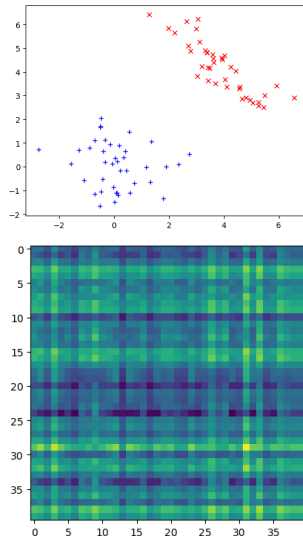
Minimise total cost: $\sum_{i=1}^M \sum_{j=1}^N C_{ij} T_{ij}$.

Such that:

$$T_{ij} \geq 0 \text{ for all } i, j$$

$$\sum_{j=1}^N T_{ij} = p_i \text{ (the correct mass is moved away from } x_i) \quad \forall i$$

$$\sum_{i=1}^M T_{ij} = q_j \text{ (the correct mass is moved towards } y_j) \quad \forall j$$



Discrete case

$$p = \sum_{i=1}^M p_i \delta_{x_i}, \quad q = \sum_{j=1}^N q_j \delta_{y_j}$$

Source locations x_1, \dots, x_M , with proportions p_1, \dots, p_M of mass.

Target locations y_1, \dots, y_N , with target proportions q_1, \dots, q_N .

$$C_{ij} = d(x_i, y_j)$$

T_{ij} = mass to be moved from x_i to y_j

Minimise total cost: $\sum_{i=1}^M \sum_{j=1}^N C_{ij} T_{ij}$.

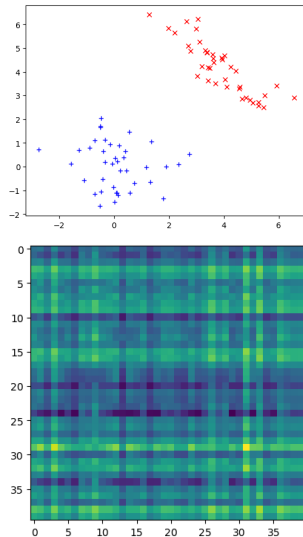
Such that:

$$T_{ij} \geq 0 \text{ for all } i, j$$

$$\sum_{j=1}^N T_{ij} = p_i \text{ (the correct mass is moved away from } x_i) \quad \forall i$$

$$\sum_{i=1}^M T_{ij} = q_j \text{ (the correct mass is moved towards } y_j) \quad \forall j$$

$$W_1(p, q) = \inf_{\substack{T | T_{ij} \geq 0 \forall i, j \\ \sum_{j=1}^N T_{ij} = p_i \forall i \\ \sum_{i=1}^M T_{ij} = q_j \forall j}} \sum_{i=1}^M \sum_{j=1}^N C_{ij} T_{ij}$$



Discrete case

$$p = \sum_{i=1}^M p_i \delta_{x_i}, \quad q = \sum_{j=1}^N q_j \delta_{y_j}$$

Source locations x_1, \dots, x_M , with proportions p_1, \dots, p_M of mass.

Target locations y_1, \dots, y_N , with target proportions q_1, \dots, q_N .

$$C_{ij} = d(x_i, y_j)$$

T_{ij} = mass to be moved from x_i to y_j

Minimise total cost: $\sum_{i=1}^M \sum_{j=1}^N C_{ij} T_{ij}$.

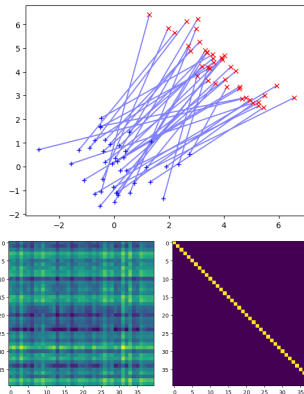
Such that:

$$T_{ij} \geq 0 \text{ for all } i, j$$

$$\sum_{j=1}^N T_{ij} = p_i \text{ (the correct mass is moved away from } x_i) \quad \forall i$$

$$\sum_{i=1}^M T_{ij} = q_j \text{ (the correct mass is moved towards } y_j) \quad \forall j$$

$$W_1(p, q) = \inf_{\substack{T \mid T_{ij} \geq 0 \forall i, j \\ \sum_{j=1}^N T_{ij} = p_i \forall i \\ \sum_{i=1}^M T_{ij} = q_j \forall j}} \sum_{i=1}^M \sum_{j=1}^N C_{ij} T_{ij}$$



Discrete case

$$p = \sum_{i=1}^M p_i \delta_{x_i}, \quad q = \sum_{j=1}^N q_j \delta_{y_j}$$

Source locations x_1, \dots, x_M , with proportions p_1, \dots, p_M of mass.

Target locations y_1, \dots, y_N , with target proportions q_1, \dots, q_N .

$$C_{ij} = d(x_i, y_j)$$

T_{ij} = mass to be moved from x_i to y_j

Minimise total cost: $\sum_{i=1}^M \sum_{j=1}^N C_{ij} T_{ij}$.

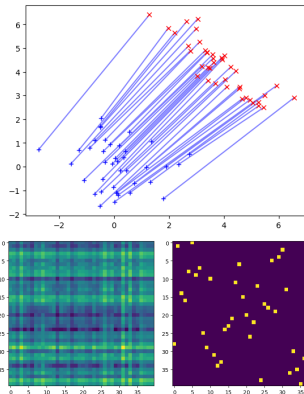
Such that:

$$T_{ij} \geq 0 \text{ for all } i, j$$

$$\sum_{j=1}^N T_{ij} = p_i \text{ (the correct mass is moved away from } x_i) \quad \forall i$$

$$\sum_{i=1}^M T_{ij} = q_j \text{ (the correct mass is moved towards } y_j) \quad \forall j$$

$$W_1(p, q) = \inf_{\substack{T \mid T_{ij} \geq 0 \forall i, j \\ \sum_{j=1}^N T_{ij} = p_i \forall i \\ \sum_{i=1}^M T_{ij} = q_j \forall j}} \sum_{i=1}^M \sum_{j=1}^N C_{ij} T_{ij}$$



Moving to the general case

$$p = \sum_{i=1}^M p_i \delta_{x_i}, \quad q = \sum_{j=1}^N q_j \delta_{y_j}$$

Minimise total cost: $\sum_{i=1}^M \sum_{j=1}^N C_{ij} T_{ij}$.

Such that:

$$T_{ij} \geq 0 \text{ for all } i, j$$

$$\sum_{j=1}^N T_{ij} = p_i \text{ (the correct mass is moved away from } x_i) \quad \forall i$$

$$\sum_{i=1}^M T_{ij} = q_j \text{ (the correct mass is moved towards } y_j) \quad \forall j$$

Moving to the general case

$$p = \sum_{i=1}^M p_i \delta_{x_i}, \quad q = \sum_{j=1}^N q_j \delta_{y_j}$$

Minimise total cost: $\sum_{i=1}^M \sum_{j=1}^N C_{ij} T_{ij}$.

Such that:

$$T_{ij} \geq 0 \text{ for all } i, j$$

$$\sum_{j=1}^N T_{ij} = p_i \text{ (the correct mass is moved away from } x_i) \quad \forall i$$

$$\sum_{i=1}^M T_{ij} = q_j \text{ (the correct mass is moved towards } y_j) \quad \forall j$$

Interpretation:

Think of T as a joint probability distribution:

$$T = \sum_{i=1}^M \sum_{j=1}^N T_{ij} \delta_{(x_i, y_j)}$$

Moving to the general case

$$p = \sum_{i=1}^M p_i \delta_{x_i}, \quad q = \sum_{j=1}^N q_j \delta_{y_j}$$

Minimise total cost: $\sum_{i=1}^M \sum_{j=1}^N C_{ij} T_{ij}$.

Such that:

$$T_{ij} \geq 0 \text{ for all } i, j$$

$$\sum_{j=1}^N T_{ij} = p_i \text{ (the correct mass is moved away from } x_i) \quad \forall i$$

$$\sum_{i=1}^M T_{ij} = q_j \text{ (the correct mass is moved towards } y_j) \quad \forall j$$

Interpretation:

Think of T as a joint probability distribution:

$$T = \sum_{i=1}^M \sum_{j=1}^N T_{ij} \delta_{(x_i, y_j)}$$

The red constraints guarantee that the marginal on the first coordinate is $\sum_{i=1}^M p_i \delta_{x_i}$.

Moving to the general case

$$p = \sum_{i=1}^M p_i \delta_{x_i}, \quad q = \sum_{j=1}^N q_j \delta_{y_j}$$

Minimise total cost: $\sum_{i=1}^M \sum_{j=1}^N C_{ij} T_{ij}$.

Such that:

$$T_{ij} \geq 0 \text{ for all } i, j$$

$$\sum_{j=1}^N T_{ij} = p_i \text{ (the correct mass is moved away from } x_i) \quad \forall i$$

$$\sum_{i=1}^M T_{ij} = q_j \text{ (the correct mass is moved towards } y_j) \quad \forall j$$

Interpretation:

Think of T as a joint probability distribution:

$$T = \sum_{i=1}^M \sum_{j=1}^N T_{ij} \delta_{(x_i, y_j)}$$

The red constraints guarantee that the marginal on the first coordinate is $\sum_{i=1}^M p_i \delta_{x_i}$.

The purple constraints guarantee that the marginal on the second coordinate is $\sum_{j=1}^N q_j \delta_{y_j}$.

Moving to the general case

$$p = \sum_{i=1}^M p_i \delta_{x_i}, \quad q = \sum_{j=1}^N q_j \delta_{y_j}$$

Minimise total cost: $\sum_{i=1}^M \sum_{j=1}^N C_{ij} T_{ij}$.

Such that:

$$T_{ij} \geq 0 \text{ for all } i, j$$

$$\sum_{j=1}^N T_{ij} = p_i \text{ (the correct mass is moved away from } x_i) \quad \forall i$$

$$\sum_{i=1}^M T_{ij} = q_j \text{ (the correct mass is moved towards } y_j) \quad \forall j$$

Interpretation:

Think of T as a joint probability distribution:

$$T = \sum_{i=1}^M \sum_{j=1}^N T_{ij} \delta_{(x_i, y_j)}$$

The red constraints guarantee that the marginal on the first coordinate is $\sum_{i=1}^M p_i \delta_{x_i}$.

The purple constraints guarantee that the marginal on the second coordinate is $\sum_{j=1}^N q_j \delta_{y_j}$. The cost can be interpreted as an expectation:

$$\sum_{i=1}^M \sum_{j=1}^N C_{ij} T_{ij} = \sum_{i=1}^M \sum_{j=1}^N d(x_i, y_j) T_{ij} = \mathbb{E}_{(X, Y) \sim T} [d(X, Y)]$$

Moving to the general case

Linear programming perspective

$$p = \sum_{i=1}^M p_i \delta_{x_i}, \quad q = \sum_{j=1}^N q_j \delta_{y_j}$$

Minimise $\sum_{i=1}^M \sum_{j=1}^N C_{ij} T_{ij}$.

Such that:

$$T_{ij} \geq 0 \text{ for all } i, j$$

$$\sum_{j=1}^N T_{ij} = p_i \quad \forall i$$

$$\sum_{i=1}^M T_{ij} = q_j \quad \forall j$$

General perspective

$$W_1(p, q) = \inf_{(X, Y) \sim T} [d(X, Y)]$$

where the infimum is over all joint probability distributions T with marginals so that $X \sim p$ and $Y \sim q$.

Examples

Distributions over \mathbb{R} , d is the usual metric.

$$W_1(\delta_a, \delta_b) = \inf_{(X,Y) \sim T} [d(X, Y)]$$

where the infimum is over all joint probability distributions T with marginals so that $X \sim \delta_a$ and $Y \sim \delta_b$.

Examples

Distributions over \mathbb{R} , d is the usual metric.

$$W_1(\delta_a, \delta_b) = \inf_{(X,Y) \sim T} \mathbb{E}[d(X, Y)]$$

where the infimum is over all joint probability distributions T with marginals so that $X \sim \delta_a$ and $Y \sim \delta_b$.

There is only one valid joint distribution on \mathbb{R}^2 : $\delta_{(a,b)}$.

So $W_1(\delta_a, \delta_b) = \mathbb{E}_{(X,Y) \sim \delta_{(a,b)}} [d(X, Y)] = d(a, b) = |a - b|$

Examples

Distributions over \mathbb{R} , d is the usual metric.

$$W_1(\delta_a, \delta_b) = \inf_{(X,Y) \sim T} [d(X, Y)]$$

where the infimum is over all joint probability distributions T with marginals so that $X \sim \delta_a$ and $Y \sim \delta_b$.

There is only one valid joint distribution on \mathbb{R}^2 : $\delta_{(a,b)}$.

$$\text{So } W_1(\delta_a, \delta_b) = \mathbb{E}_{(X,Y) \sim \delta_{(a,b)}} [d(X, Y)] = d(a, b) = |a - b|$$

$$W_1(\delta_0, N(0, \sigma^2)) = \inf_{(X,Y) \sim T} [d(X, Y)]$$

where the infimum is over all joint probability distributions T with marginals so that $X \sim \delta_0$ and $Y \sim N(0, \sigma^2)$.

Examples

Distributions over \mathbb{R} , d is the usual metric.

$$W_1(\delta_a, \delta_b) = \inf_{(X,Y) \sim T} [d(X, Y)]$$

where the infimum is over all joint probability distributions T with marginals so that $X \sim \delta_a$ and $Y \sim \delta_b$.

There is only one valid joint distribution on \mathbb{R}^2 : $\delta_{(a,b)}$.

So $W_1(\delta_a, \delta_b) = \mathbb{E}_{(X,Y) \sim \delta_{(a,b)}} [d(X, Y)] = d(a, b) = |a - b|$

$$W_1(\delta_0, N(0, \sigma^2)) = \inf_{(X,Y) \sim T} [d(X, Y)]$$

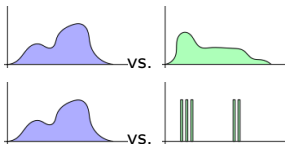
where the infimum is over all joint probability distributions T with marginals so that $X \sim \delta_0$ and $Y \sim N(0, \sigma^2)$.

There is only one valid joint distribution on \mathbb{R}^2 , where the first coordinate is deterministically 0, and the second follows a normal distribution.

So, $W_1(\delta_0, N(0, \sigma^2)) = \mathbb{E}_{(X,Y) \sim \delta_0 \otimes N(0,1)} [d(X, Y)] = \int_{\mathbb{R}} d(0, y) f(y) dy = \dots = \sqrt{\frac{2}{\pi}} \sigma$

Comparing KL with Wasserstein distance

How to compare probability distributions?



Kullback-Leibler divergence

$$KL(q||p) = \mathbb{E}_{X \sim q} [\log(q(X)/p(X))]$$

- + Connections to information theory/max-likelihood
- + Unbiased (gradient) estimators via samples from p
- Requires that both distributions have known densities*
- No information about metrics in underlying base space used

Optimal Transport (Wasserstein) distance

$$W_1(p, q) = \inf_{\substack{(X, Y) \sim T \\ X \sim p, Y \sim q}} \mathbb{E} [d(X, Y)]$$

- + Can use information about base space
- + Can compare e.g. discrete and continuous distributions
- Unbiased estimation in intractable cases is very difficult
- Requires solving an optimisation problem to calculate, even in tractable cases

Algorithmic considerations: How do we actually compute optimal transport distances?

Discrete case

Finitely supported discrete distributions \implies linear program

Interior point polynomial-time algorithms exist, but practically can become very costly.

Important development for practical application in machine learning: Cuturi's entropy regularisation.

Algorithmic considerations: How do we actually compute optimal transport distances?

General case

As for other probability metrics/divergences, in general intractable to compute $W_1(P, Q)$ for general distributions P, Q .

Solutions:

Use samples – much more later on. [Wasserstein Generative Adversarial Networks, Arjovsky et al. 2017]

Problem: hard to get *unbiased* estimators [The Cramer Distance as a Solution to Biased Wasserstein Gradients, Bellemare et al. 2017]

Discretise domain to obtain discrete problem

Problem: curse of dimensionality

Interesting theory that we don't have time for

p -Wasserstein metrics W_p

Existence and uniqueness of optimal transport plans for arbitrary probability measures

Connections with cyclic monotonicity and duality theory

Connections with PDEs and gradient flows

etc.

Find out more

Courses and books

Graduate course on optimal transport (theory and algorithms) at the Maths dept next term, given by Matthew Thorpe.

Optimal Transport: Old and New (Cédric Villani, 2009)

Topics in Optimal Transportation (Cédric Villani, 2003)

Optimal Transport for Applied Mathematicians (Santambrogio, 2015)

Try it yourself

Python Optimal Transport Toolbox: github.com/rflamary/POT

Part II: Plan

Dual form Wasserstein distance and Applications:

- Review: Primal form linear programming

- Dual formulation for discrete case

- Weight clipping Wasserstein GAN

- Gradient penalty Wasserstein GAN

Linear Programming

General primal-form formulation between two probability distributions $\eta_1, \eta_2 \in \mathcal{P}(X)$ with joint probability distribution Π :

$$W(\eta_1, \eta_2) = \inf_{\Pi} \mathbb{E}_{(X,Y) \sim \Pi} [d(X, Y)]$$

For distributions with discrete support:

Discrete Samples $\{X_i\}_{i=1}^N$ and $\{Y_i\}_{i=1}^N$

Joint Probability Matrix \mathbf{T} with corresponding flattened vector \mathbf{t}

Marginal Probability Constraints

$$\sum_{j=1}^N T_{ij} = p_i$$

$$\sum_{i=1}^N T_{ij} = p_j$$

Distance Matrix \mathbf{C} with corresponding flattened vector \mathbf{c}

Linear Programming

Linear Programming Formulation:

$$\begin{aligned} \min_{\mathbf{t}} \quad & \mathbf{t}^T \mathbf{c} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{t} = \mathbf{b} \\ & \mathbf{t} \geq 0 \end{aligned}$$

Linear Programming

Linear Programming Formulation:

$$\begin{aligned} \min_{\mathbf{t}} \quad & \mathbf{t}^T \mathbf{c} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{t} = \mathbf{b} \\ & \mathbf{t} \geq 0 \end{aligned}$$

$\mathbf{b} = \begin{bmatrix} \mathbf{p}_x \\ \mathbf{p}_y \end{bmatrix}$ where \mathbf{p}_x and \mathbf{p}_y contains corresponding discrete probability

Matrix \mathbf{A} contains 0s and 1s to sum corresponding elements in \mathbf{t}

Satisfy the marginalization constraints

Linear Programming

Linear Programming Formulation:

$$\begin{aligned} \min_{\mathbf{t}} \quad & \mathbf{t}^T \mathbf{c} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{t} = \mathbf{b} \\ & \mathbf{t} \geq 0 \end{aligned}$$

$\mathbf{b} = \begin{bmatrix} \mathbf{p}_x \\ \mathbf{p}_y \end{bmatrix}$ where \mathbf{p}_x and \mathbf{p}_y contains corresponding discrete probability

Matrix \mathbf{A} contains 0s and 1s to sum corresponding elements in \mathbf{t}

Satisfy the marginalization constraints

Problems:

- Computationally expensive for real life problems e.g. images

- Lack of smoothness in transportation plan

- Only care about distance not joint probability

Dual formulation for discrete case

Primal form linear programming:

$$\begin{aligned} \min_{\mathbf{t}} \quad & \mathbf{t}^T \mathbf{c} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{t} = \mathbf{b} \\ & \mathbf{t} \geq 0 \end{aligned}$$

Dual form linear programming:

$$\begin{aligned} \max_{\mathbf{f}} \quad & \mathbf{b}^T \mathbf{f} \\ \text{s.t.} \quad & \mathbf{A}^T \mathbf{f} \leq \mathbf{c} \end{aligned}$$

Dual formulation for discrete case

Primal form linear programming:

$$\begin{aligned} \min_{\mathbf{t}} \quad & \mathbf{t}^T \mathbf{c} \\ \text{s.t.} \quad & \mathbf{A} \mathbf{t} = \mathbf{b} \\ & \mathbf{t} \geq 0 \end{aligned}$$

Dual form linear programming:

$$\begin{aligned} \max_{\mathbf{f}} \quad & \mathbf{b}^T \mathbf{f} \\ \text{s.t.} \quad & \mathbf{A}^T \mathbf{f} \leq \mathbf{c} \end{aligned}$$

Weak Duality: If either form of linear programming is feasible, so is the other. The dual solution forms a **lower bound** for primal solution.

Easily check: $\mathbf{t}^T \mathbf{c} \geq \mathbf{t}^T \mathbf{A}^T \mathbf{f} = \mathbf{b}^T \mathbf{f}$

Strong Duality: Primal and dual solutions coincides.

Dual formulation for discrete case

Dual form linear programming:

$$\begin{aligned} \max_{\mathbf{f}} \quad & \mathbf{b}^T \mathbf{f} \\ \text{s.t.} \quad & \mathbf{A}^T \mathbf{f} \leq \mathbf{c} \end{aligned}$$

If $\mathbf{f} = \begin{bmatrix} \mathbf{f}_x \\ \mathbf{f}_y \end{bmatrix}$, we have $\mathbf{b}^T \mathbf{f} = \mathbf{p}_x^T \mathbf{f}_x + \mathbf{p}_y^T \mathbf{f}_y$.

If we regard \mathbf{f}_x and \mathbf{f}_y as function f evaluated at some points \mathbf{x} and \mathbf{y}

If we define discrete distribution $P(x) = \sum_{i=1}^N p_{x_i} \delta_{x_i}$ and $P(y) = \sum_{i=1}^N p_{y_i} \delta_{y_i}$

Dual formulation for discrete case

Dual form linear programming:

$$\begin{aligned} \max_{\mathbf{f}} \quad & \mathbf{b}^T \mathbf{f} \\ \text{s.t.} \quad & \mathbf{A}^T \mathbf{f} \leq \mathbf{c} \end{aligned}$$

If $\mathbf{f} = \begin{bmatrix} \mathbf{f}_x \\ \mathbf{f}_y \end{bmatrix}$, we have $\mathbf{b}^T \mathbf{f} = \mathbf{p}_x^T \mathbf{f}_x + \mathbf{p}_y^T \mathbf{f}_y$.

If we regard \mathbf{f}_x and \mathbf{f}_y as function f evaluated at some points \mathbf{x} and \mathbf{y}

If we define discrete distribution $P(x) = \sum_{i=1}^N p_{x_i} \delta_{x_i}$ and $P(y) = \sum_{i=1}^N p_{y_i} \delta_{y_i}$

We can obtain

$$\mathbf{b}^T \mathbf{f} = \mathbb{E}_{x \sim P(x)}[f(x)] + \mathbb{E}_{y \sim P(y)}[f(y)]$$

Constraints:

$$f(x_i) + f(y_j) \leq d(x_i, y_j)$$

Dual formulation for discrete case

Dual form linear programming:

$$\begin{aligned} \max_{\mathbf{f}} \quad & \mathbf{b}^T \mathbf{f} \\ \text{s.t.} \quad & \mathbf{A}^T \mathbf{f} \leq \mathbf{c} \end{aligned}$$

If $\mathbf{f} = \begin{bmatrix} \mathbf{f}_x \\ \mathbf{f}_y \end{bmatrix}$, we have $\mathbf{b}^T \mathbf{f} = \mathbf{p}_x^T \mathbf{f}_x + \mathbf{p}_y^T \mathbf{f}_y$.

If we regard \mathbf{f}_x and \mathbf{f}_y as function f evaluated at some points \mathbf{x} and \mathbf{y}

If we define discrete distribution $P(x) = \sum_{i=1}^N p_{x_i} \delta_{x_i}$ and $P(y) = \sum_{i=1}^N p_{y_i} \delta_{y_i}$

With bit of work, for discrete case, we can modify the dual formulation objective as:

$$\sup_f \mathbb{E}_{x \sim P(x)}[f(x)] - \mathbb{E}_{y \sim P(y)}[f(y)]$$

Constraints:

$$f(x_i) - f(y_j) \leq d(x_i, y_j)$$

Kantorovich-Rubinstein duality

For continuous case, **assume the distance measure d is Euclidean distance**, we have Kantorovich-Rubinstein duality formulation(proof omitted):

$$W(P_x, P_y) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim P_x}[f(x)] - \mathbb{E}_{y \sim P_y}[f(y)]$$

Kantorovich-Rubinstein duality

For continuous case, **assume the distance measure d is Euclidean distance**, we have Kantorovich-Rubinstein duality formulation(proof omitted):

$$W(P_x, P_y) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim P_x}[f(x)] - \mathbb{E}_{y \sim P_y}[f(y)]$$

f is any function satisfying Lipschitz continuity constraints

$\|f\|_L \leq 1$ represents the function f has to satisfy **Lipschitz continuity** smaller than 1.

For real valued function $f(\cdot)$, for all x, y , it satisfies $\frac{|f(x) - f(y)|}{|x - y|} \leq 1$

Enforce how fast function f can change w.r.t change in input (total variation)

Application in GAN

GAN Objective:

$$\min_{G(\cdot)} \max_{D(\cdot)} \mathbb{E}_{x \sim p(x)} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))]$$

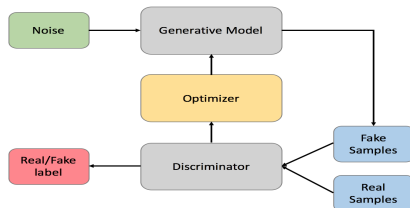
Original GAN training:

- Avoid using likelihood

- Successful in training implicit generative models (Intractable Density)

- Use discriminator to classify real and adversarial samples

- Use Information-Theoretic divergence (Jensen-Shannon Divergence) as objective function



Problems in original GAN

Vanishing Gradient:

- Good discriminator is harmful for generator learning

- Exist perfect discriminator

- Divergence max out

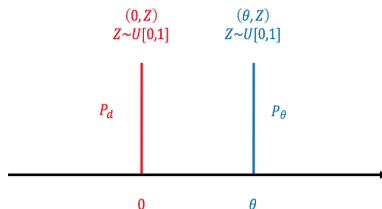
Mode-collapsing:

- Limited variabilities in generated images

Wasserstein GAN

Wasserstein distance can be shown to be a more sensible divergence than traditional KL, JSD or TV when learning distribution with lower dimensional support than actual data dimension.

Illustrating example : Learn parallel lines ¹



$$W(P_d, P_\theta) = |\theta|$$

$$JS(P_d, P_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0. \\ 0 & \text{if } \theta = 0. \end{cases}$$

$$KL(P_d || P_\theta) = \begin{cases} +\infty & \text{if } \theta \neq 0. \\ 0 & \text{if } \theta = 0. \end{cases}$$

¹All the examples and images up to WGAN-GP are from [Arjovsky et al., 2017]

Wasserstein GAN

Wasserstein distance can be shown to be a more sensible divergence than traditional KL, JSD or TV when learning distribution with lower dimensional support than actual data dimension.

What forms of Wasserstein distance we should use?

Wasserstein GAN

Wasserstein distance can be shown to be a more sensible divergence than traditional KL, JSD or TV when learning distribution with lower dimensional support than actual data dimension.

Dual formulation:

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

function $f_w(\cdot)$ is neural network parametrized by weight w

\mathcal{W} is a compact parameter space to ensure $f_w(\cdot)$ is K-Lipschitz continuous

Generator $g_\theta(\cdot)$ is also neural network parametrized by θ

Lipschitz Continuity Constraints

Replace the original constraints $\|f\|_L \leq 1$ with $\|f\|_L \leq K$ associated with the compact parameters space \mathcal{W}

Controlling parameter space \mathcal{W} controls function space

Maximization of new objective yields a calculation of original **dual Wasserstein distance up to a multiplicative constant**

To ensure $w \in \mathcal{W}$, **simply clip the weight** $w \in [-c, c]$ **with small c**

Intuition: Small w implies small variations $f_w(\cdot)$ w.r.t input.

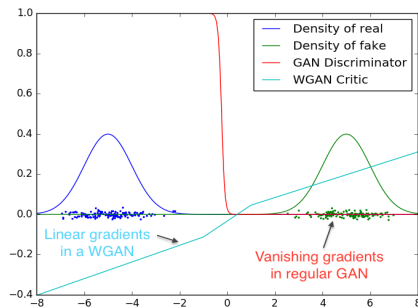
Comparison with Original GAN

Learning:

Analogous discriminator and generator learning: Stochastic gradient descent with reparametrization

Resolve the vanishing gradient problem, improve stability and reduce mode collapsing

Toy Gaussian Example:



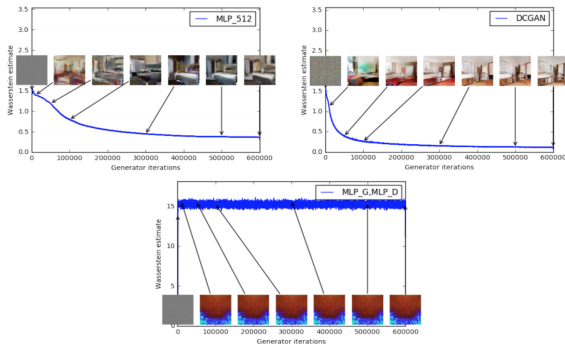
Comparison with Original GAN

Learning:

Analogous discriminator and generator learning: Stochastic gradient descent with reparametrization

Resolve the vanishing gradient problem, improve stability and reduce mode collapsing

Meaningful loss metric:



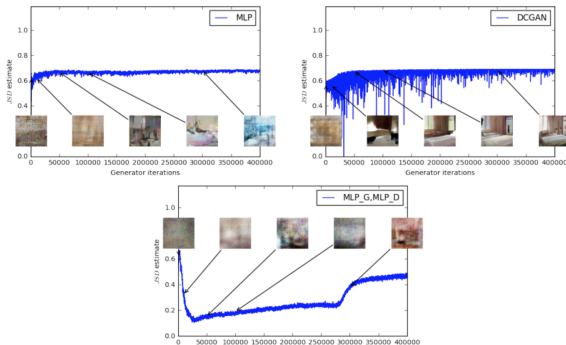
Comparison with Original GAN

Learning:

Analogous discriminator and generator learning: Stochastic gradient descent with reparametrization

Resolve the vanishing gradient problem, improve stability and reduce mode collapsing

Meaningful loss metric:



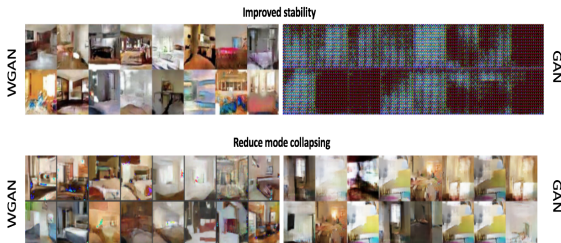
Comparison with Original GAN

Learning:

Analogous discriminator and generator learning: Stochastic gradient descent with reparametrization

Resolve the vanishing gradient problem, improve stability and reduce mode collapsing

Improved stability and reduce mode collapsing:



Reconsider Lipschitz Continuity

Is weight clipping a sensible way to enforce Lipschitz continuity constraints?

Reconsider Lipschitz Continuity

Is weight clipping a sensible way to enforce Lipschitz continuity constraints?

No, clipping weight will damage the modeling power of neural network and require careful tuning of weight boundary

Variations of function $f_w(\cdot)$ can also be achieved using **Gradient Penalty**

Optimal discriminator property ¹:

For sample x from generator, there exists a sample y from true distribution, such that the gradient of discriminator $f_w(\cdot)$ w.r.t any points on the line $x_t = (1 - t)x + ty$ points towards y .

$$\nabla_{x_t} f_w(x_t) = \frac{y - x_t}{\|y - x_t\|}$$

It implies the **norm of discriminator gradient is 1 almost everywhere**

¹Theorem proved in [Gulrajani et al., 2017]

Optimal discriminator property ¹:

For sample x from generator, there exists a sample y from true distribution, such that the gradient of discriminator $f_w(\cdot)$ w.r.t any points on the line $x_t = (1 - t)x + ty$ points towards y .

$$\nabla_{x_t} f_w(x_t) = \frac{y - x_t}{\|y - x_t\|}$$

It implies the **norm of discriminator gradient is 1 almost everywhere**

Instead of clipping weight, we regularize the gradients.

Objective:

$$\max_w \underbrace{\mathbb{E}_{x \sim p(x)}[f_w(x)] - \mathbb{E}_{z \sim p(z)}[f_w(g_\theta(z))]}_{\text{Original dual form loss}} + \underbrace{\lambda \mathbb{E}_{\hat{x} \sim p(\hat{x})}[(\|\nabla_{\hat{x}} f_w(\hat{x})\|_2 - 1)]^2}_{\text{Gradient Penalty}}$$

λ controls relative importance of the constraints

\hat{x} are sampled from the line between generated and real samples.

$\hat{x} = \epsilon x + (1 - \epsilon)g_\theta(z)$ with $\epsilon \sim U(0, 1)$

¹Theorem proved in [Gulrajani et al., 2017]

Modeling Power:

Weight clipping over-simplified the learned discriminator surface

May not provide correct gradient for generator

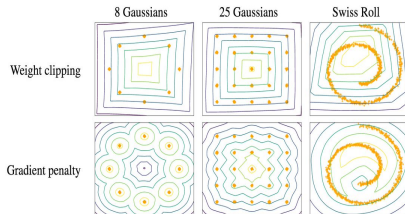


Figure 1: Value surfaces of WGAN critics trained to optimality on toy datasets. Critics trained with weight clipping fail to capture higher moments of the data distribution. The 'generator' is held fixed at the real data plus Gaussian noise.

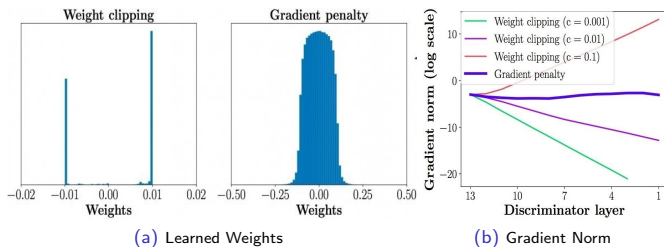
¹All images and examples are from [Gulrajani et al., 2017]

Comparison ¹

Learned discriminator weights:

Weight clipping pushes weights to boundaries.

Require careful tuning of the boundary otherwise exploding or vanishing of gradients.

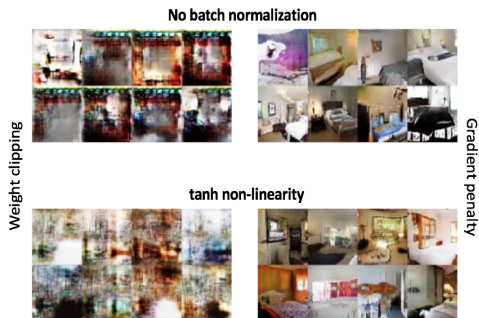


¹All images and examples are from [Gulrajani et al., 2017]

Comparison ¹

LSUN image generation:

WGAN-GP is more robust than weight clipping



¹All images and examples are from [Gulrajani et al., 2017]

Other related forms of WGAN

There are some other forms of GAN which are closely related to Wasserstein GAN but we don't have time for them:

Primal form WGAN:

Sinkhorn GAN [Genevay et al., 2017]: Fast approximation to primal solutions

Wasserstein Auto-Encoders [Tolstikhin et al., 2017]: Auto-encoder based primal formulation

Generalizations:

Relaxed Wasserstein distance [Guo et al., 2017]: Bregman divergence as the distance measure

Unbiased sample gradients:

Cramer GAN [Bellemare et al., 2017]: Cramer distance as replacement of Wasserstein distance.

Bibliography I



Arjovsky, M., Chintala, S., and Bottou, L. (2017).

Wasserstein gan.

arXiv preprint arXiv:1701.07875.



Bellemare, M. G., Danihelka, I., Dabney, W., Mohamed, S., Lakshminarayanan, B., Hoyer, S., and Munos, R. (2017).

The cramer distance as a solution to biased wasserstein gradients.

arXiv preprint arXiv:1705.10743.



Genevay, A., Peyré, G., and Cuturi, M. (2017).

Sinkhorn-autodiff: Tractable wasserstein learning of generative models.

arXiv preprint arXiv:1706.00292.



Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. (2017).

Improved training of wasserstein gans.

arXiv preprint arXiv:1704.00028.

Bibliography II



Guo, X., Hong, J., Lin, T., and Yang, N. (2017).

Relaxed wasserstein with applications to gans.

arXiv preprint arXiv:1705.07164.



Tolstikhin, I., Bousquet, O., Gelly, S., and Schoelkopf, B. (2017).

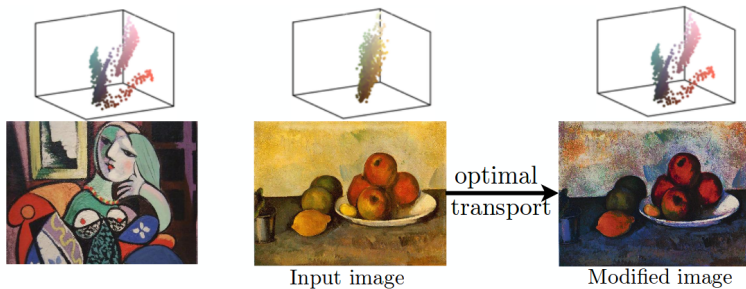
Wasserstein auto-encoders.

arXiv preprint arXiv:1711.01558.

Extra slides

Applications

Image processing (e.g. colour transfer)



[Image credit: *Numerical Optimal Transport and Applications*, Gabriel Peyré]

Applications

Clustering/distillation

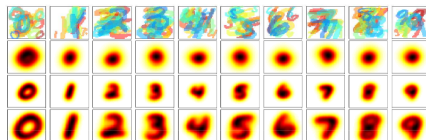
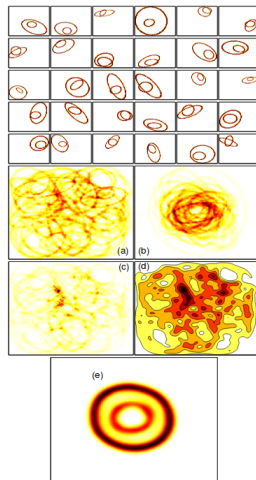


Figure 1. (Top) 30 artificial images of two nested random ellipses. Mean measures using the (a) Euclidean distance (b) Euclidean after re-centering images (c) Jeffrey centroid (Nielsen, 2013) (d) RKHS distance (Gaussian kernel, $\sigma = 0.002$) (e) 2-Wasserstein distance.

[Image credit: *Fast Computation of Wasserstein Barycenters*, Cuturi and Doucet]

Cuturi's Entropic Regularisation

Instead of optimising a linear cost $\sum_{i=1}^N \sum_j = 1^N C_{ij} T_{ij}$, optimise a strongly convex cost $\sum_{i=1}^N \sum_j = 1^N C_{ij} T_{ij} + \gamma E(T)$, where $E(T)$ is the entropy of the joint distribution T .

Algorithm 1 Computation of $d_M^\lambda(r, c)$ using Sinkhorn-Knopp's fixed point iteration

Input M, λ, r, c .

$I = (r > 0)$; $r = r(I)$; $M = M(I, :)$; $K = \exp(-\lambda * M)$

Set $x = \text{ones}(\text{length}(r), \text{size}(c, 2)) / \text{length}(r)$;

while x changes **do**

$x = \text{diag}(1./r) * K * (c .* (1 ./ (K' * (1./x))))$

end while

$u = 1./x$; $v = c .* (1 ./ (K' * u))$

$d_M^\lambda(r, c) = \text{sum}(u .* (K .* M) * v)$

Mathematically equivalent to optimising linear cost over smaller convex region.

