Learning Wasserstein Embeddings

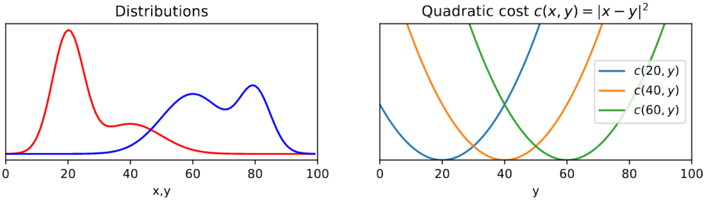
Nicolas Courty[†], Rémi Flamary[‡] and Mélanie Ducoffe[⋆]

[†] *IRISA, Université de Bretagne Sud,Vannes, France*
[‡] *Université Côte d'Azur, Lab. Lagrance UMR CNRS 7293, OCA, Nice, France*
[⋆] *Université Côte d'Azur, I3S, UMR CNRS 7271, Nice France*

Disclaimer

- Awesome Wasserstein library (python): POT
https://github.com/rflamary/POT
- other sources:
https://github.com/rflamary/OTML_Statlearn2018

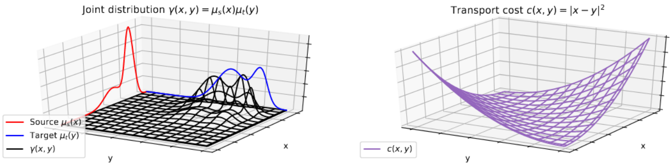
Reminder: Optimal Transport (Monge formulation)



- Probability measures μ_s and μ_t on and a cost function $c : \Omega_s \times \Omega_t \rightarrow \mathbb{R}^+$.
- The Monge formulation [Monge, 1781] aim at finding a mapping $T : \Omega_s \rightarrow \Omega_t$

$$\inf_{T\#\mu_s=\mu_t} \quad \int_{\Omega_s} c(\mathbf{x}, T(\mathbf{x}))\mu_s(\mathbf{x})d\mathbf{x} \tag{1}$$

Reminder: Optimal Transport (Kantorovich formulation)



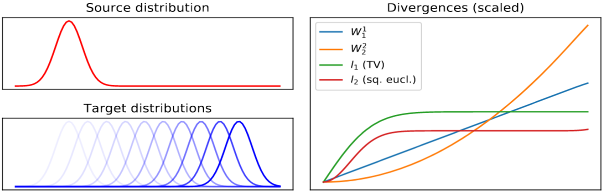
- The Kantorovich formulation [Kantorovich, 1942] seeks for a probabilistic coupling $\gamma \in \mathcal{P}(\Omega_s \times \Omega_t)$ between Ω_s and Ω_t:

$$\gamma_0 = \underset{\gamma}{\operatorname{argmin}} \int_{\Omega_s \times \Omega_t} c(\mathbf{x}, \mathbf{y})\gamma(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y}, \tag{2}$$

$$\text{s.t.} \quad \gamma \in \mathcal{P} = \left\{ \gamma \geq \mathbf{0}, \int_{\boldsymbol{\Omega_t}} \gamma(\mathbf{x}, \mathbf{y}) \mathbf{dy} = \boldsymbol{\mu_s}, \int_{\boldsymbol{\Omega_s}} \gamma(\mathbf{x}, \mathbf{y}) \mathbf{dx} = \boldsymbol{\mu_t} \right\}$$

- γ is a joint probability measure with marginals μ_s and μ_t.
- Linear Program that always have a solution.

Reminder: Wasserstein distance



More formally, let $\boldsymbol{X}$ be a metric space endowed with a metric $\boldsymbol{d_X}$. The p-Wasserstein distance between two measures $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ is defined as:

$$\boldsymbol{W_p(\mu,\nu) = \left(\inf_{\pi\in\Pi(\mu,\nu)} \iint_{X\times X} d(x,y)^p d\pi(x,y)\right)^{\frac{1}{p}}}. \quad (1)$$

Here, $\boldsymbol{\Pi(\mu,\nu)}$ is the set of probabilistic couplings $\boldsymbol{\pi}$ on $\boldsymbol{(\mu,\nu)}$.

Reminder: 3 cases for Optimal Transport

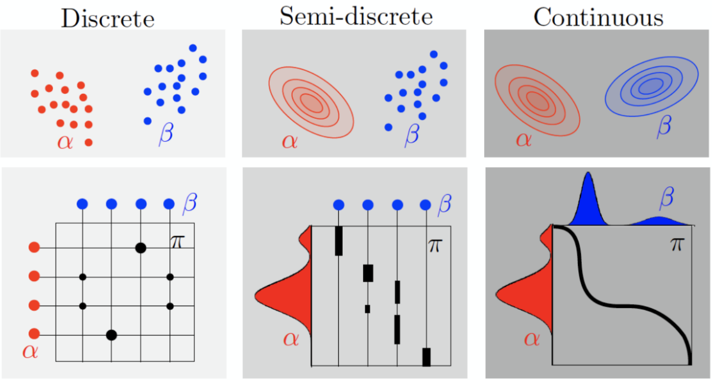
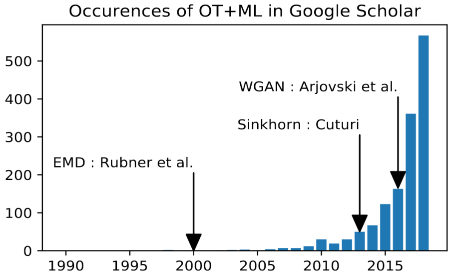


Image from Gabriel Peyré

Optimal Transport for Machine Learning



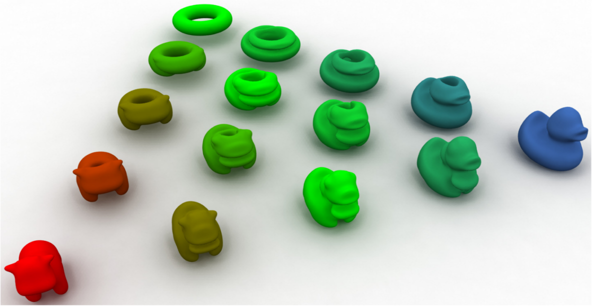
Short history of OT for ML

- Recently introduced to ML (well known in image processing since 2000s)
- Computationnal OT allow numerous applications (regularization)
- Deep learning boost (numerical optimization and GAN)

Context

1. clustering
2. similarity
3. interpolation

Shape interpolation [Solomon et al., 2015]



Problematic

- Discrete distributions: histograms
- Solving the corresponding Linear Program (LP) is super cubical in complexity
- Approximations:
  1. slicing techniques [?]
  2. entropic regularization [?]
  3. stochastic optimization [GCPB16]

computing pairwise Wasserstein distances between a huge number of large distributions (like an image collection) or optimization problems with a lot of Wasserstein distances (e.g. barycenters) is still intractable.

Learning Wasserstein Embeddings

We propose to **learn** a linear embedding where the Wasserstein distance is reproduced by the Euclidean norm.

Once this embedding is found, computing distances or solving problems involving Wasserstein distances can be conducted extremely fast.

We also show that it is possible to **simultaneously learn the inverse mapping** back to the original space.

DWE for Deep Wasserstein Embedding

- learn in a supervised way a new representation of the data
- pre-computed dataset that consists of pairs of histograms $\{x_i^1, x_i^2\}_{i \in 1,\dots,n}$ of dimensionality d and their corresponding W_2^2 Wasserstein distance $\{y_i = W_2^2(x_i^1, x_i^2)\}_{i \in 1,\dots,n}$
- **Siamese architecture** [?]

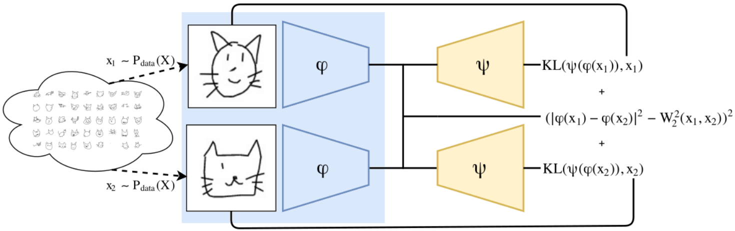


Figure: Architecture of the Wasserstein Deep Learning: two samples are drawn from the data distribution and set as input of the same network (ϕ) that computes the embedding.

DWE: Decoder

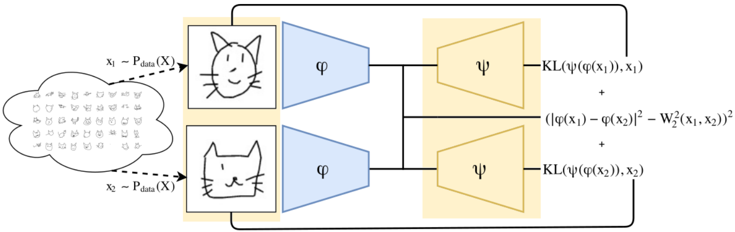


Figure: Autoencoding with a KL regularizer loss

DWE

The global objective function reads

$$\min_{\phi,\psi} \sum_i \left\| \|\phi(x_i^1) - \phi(x_i^2)\|^2 - y_i \right\|^2 + \lambda \sum_i \mathrm{KL}(\psi(\phi(x_i^1)), x_i^1) + \mathrm{KL}(\psi(\phi(x_i^2)), x_i^2) \quad (2)$$

- Decoding helps

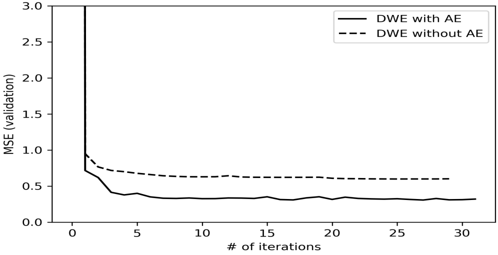


Figure: W_2^2 validation MSE along the number of epochs for the MNIST dataset (DWE).

Discussion

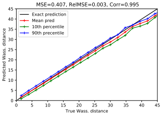
- What is the meaning of embedding the non-flat geometry of the Wasserstein space in a linear space ?
- Embedding Wasserstein space in normed metric space is still a theoretical and open questions [**?**]
- For $\boldsymbol{W_2^2}$, theoretical embeddability results do not exist, but we show that, for a population of locally concentrated measures, a good approximation can be obtained with our technique.
- Many possible extensions and possibilities for data mining tasks based on Wasserstein distances.

Google Doodle datasets

- **28** $\times$ **28** images seen as probability distributions

Performances on MNIST / Google Doodle datasets

- $\mathbf{28 \times 28}$ images seen as probability distributions
- We use a convolutional architectureto reach a size of embedding of $\mathbf{100}$
- 700 000 pairs used for learning, 200 000 for validation and 100 000 for testing



Method	$\boldsymbol{W_2^2}$/sec
LP network flow (1 CPU)	192
DWE Indep. (1 CPU)	3 633
DWE Pairwise (1 CPU)	213 384
DWE Indep. (GPU)	233 981
DWE Pairwise (GPU)	10 477 901

Figure: The test performance are as follows: MSE=0.40, Relative MSE=0.002 and Correlation=0.996. (Table 1) Computational performance of $\boldsymbol{W_2^2}$ and DWE given as average number of $\boldsymbol{W_2^2}$ computation per seconds for different configurations

Cross-Datasets Performance

- The cross-datasets performances indicate that the learnt embedding is **data-dependent**

Learn / Test	CAT	CRAB	FACE	MNIST
CAT	**1.195**	*1.654*	2.069	12.131
CRAB	*2.621*	**0.854**	3.158	10.881
FACE	*5.025*	5.445	**1.254**	50.526
MNIST	9.118	6.698	*4.68*	**0.412**

Table: Cross performances between the DWE embedding learned on each datasets. On each row, we observe the MSE on the test set of each dataset given a DWL (Cat, Crab, Faces and MNIST)

Results: Wasserstein Barycenters

Defined by analogy with Euclidean spaces, the Wasserstein barycenters of a family of measures are defined as minimizers of a weighted sum of $\boldsymbol{W_2^2}$:

$$\bar{x} = \arg\min_{x} \sum_i \alpha_i W(x, x_i) \approx \psi\Big(\sum_i \alpha_i \phi(x_i)\Big), \qquad (3)$$

where $\boldsymbol{x_i}$ are the data samples and the positive weights $\boldsymbol{\alpha_i}$ sum to $\mathbf{1}$.

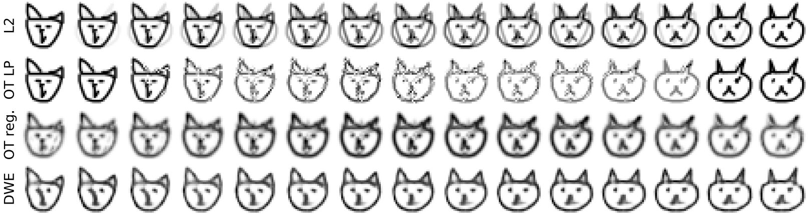


Figure: Comparison of the interpolation with L2 Euclidean distance (top), LP Wasserstein interpolation (top middle) regularized Wasserstein Barycenter (down middle) and DWE (down).

Results: Geodesic Analysis in Wasserstein space

Generalization of PCA in Wasserstein space, expressed as a regular PCA in the embedded space.

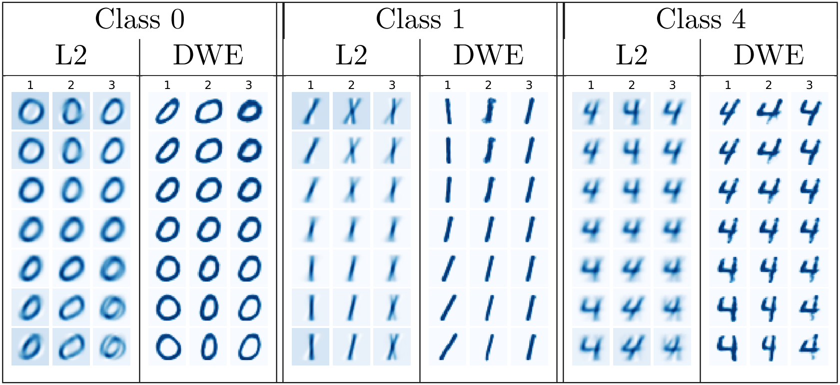


Figure: Principal Geodesic Analysis for classes 0,1 and 4 from the MNIST dataset for squared Euclidean distance (L2) and Wasserstein Deep Learning (DWE). For each class and method we show the variation from the barycenter along one of the first 3 principal modes of variation.

Ongoing work

- High dimensional histograms ?
- Reference: From Word Embeddings to Document Distances
- Word Wasserstein distance : measuring document similarity with word2vec

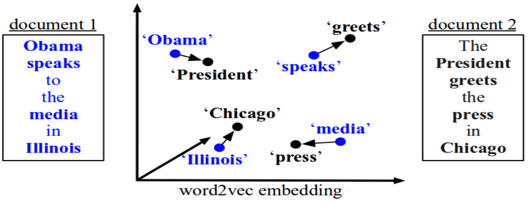


Figure: **An illustration of the word mover's distance**: All noon-stop words (bold) of both documents are embedded into a word2vec space. The distance between two documents is the minimum cumulative distance that all words in document 1 need to travel to exactly match document 2

From Word Embeddings To Document Distances

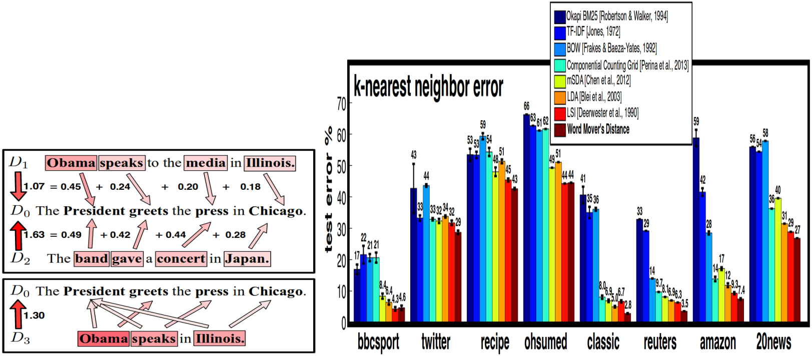


Figure: The kNN test error results on 8 document classification data sets, compared to canonical and state-of-the-art baselines methods

Acknowledgements

- French ANR under reference ANR-17-CE23-0012 (OATMIL project)

Code Release: https://github.com/mducoffe/Learning-Wasserstein-Embeddings

Bibliography

[[GCPB16]] A. Genevay, M. Cuturi, G. Peyré, and F. Bach, *Stochastic optimization for large-scale optimal transport*, NIPS, 2016, pp. 3432–3440.

Bibliography

[[GCPB16]] A. Genevay, M. Cuturi, G. Peyré, and F. Bach, *Stochastic optimization for large-scale optimal transport*, NIPS, 2016, pp. 3432–3440.

Bibliography

[[GCPB16]] A. Genevay, M. Cuturi, G. Peyré, and F. Bach, *Stochastic optimization for large-scale optimal transport*, NIPS, 2016, pp. 3432–3440.

Bibliography

[[GCPB16]] A. Genevay, M. Cuturi, G. Peyré, and F. Bach, *Stochastic optimization for large-scale optimal transport*, NIPS, 2016, pp. 3432–3440.