Numerical Optimal Transport

http://optimaltransport.github.io

Gromov Wasserstein

Gabriel Peyré

www.numerical-tours.com

CNRS
ENS
ÉCOLE NORMALE SUPÉRIEURE

Overview

- **Gromov Wasserstein**
- Entropic Regularization
- Applications
- GW Barycenters

Gromov-Wasserstein

Inputs: {(similarity/kernel matrix, histogram)}

$(d, \mu) \qquad \mu = \sum_i \mu_i \delta_{x_i} \qquad d_{i,i'} = d(x_i, x_{i'})$

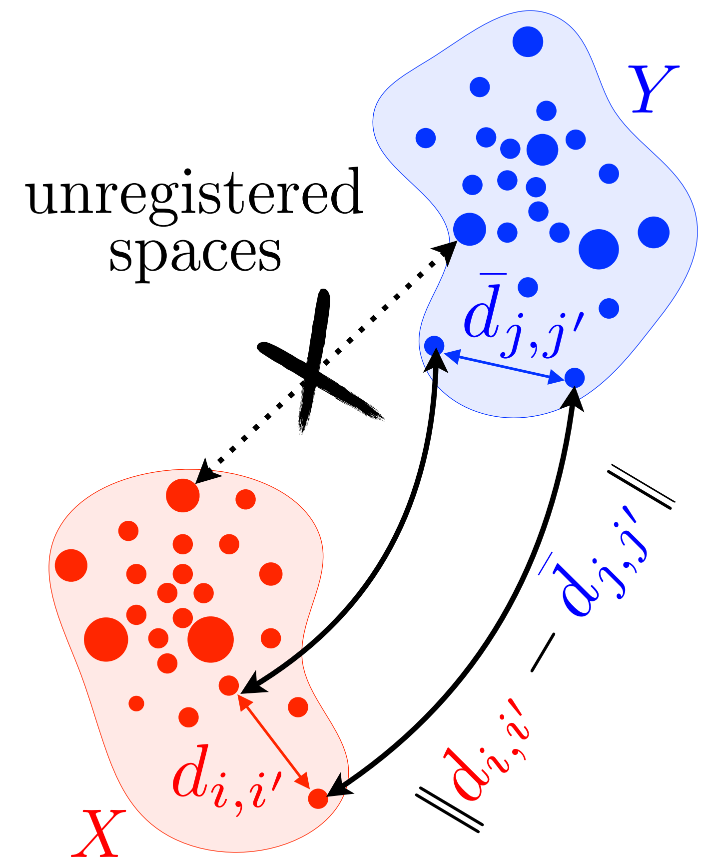
$(\bar{d}, \nu) \qquad \nu = \sum_j \nu_j \delta_{y_j} \qquad \bar{d}_{j,j'} = \bar{d}(y_j, y_{j'})$

Def. *Gromov-Wasserstein distance:*

$$\mathrm{GW}_p^p(d, \mu, \bar{d}, \nu) \stackrel{\text{def.}}{=} \min_{T \in C_{\mu,\nu}} \mathcal{E}^p_{d,\bar{d}}(T)$$

$$\mathcal{E}^p_{d,\bar{d}}(T) \stackrel{\text{def.}}{=} \sum_{i,i',j,j'} |d_{i,i'} - \bar{d}_{j,j'}|^p \, T_{i,j} T_{i',j'}$$

[Memoli 2011]
[Sturm 2012]



Computation of GW is a QAP:

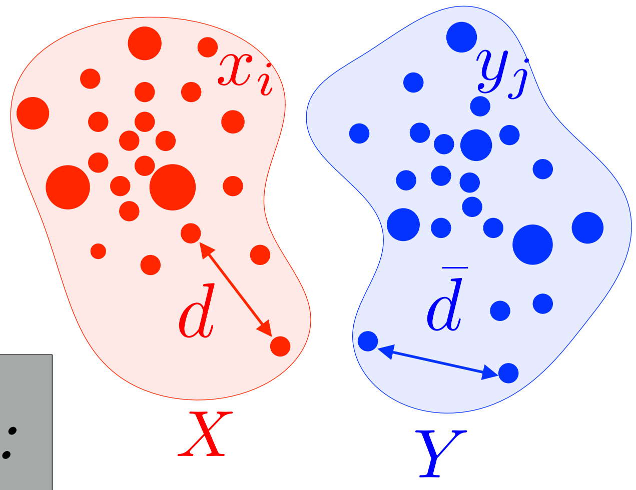
→ NP-hard in general.

→ need for a fast approximate solver.

Gromov-Wasserstein as a Metric

$\mu = \sum_i \mu_i \delta_{x_i} \in \mathcal{M}_+^1(X)$ $\quad d_{i,i'} = d(x_i, x_{i'})$

$\nu = \sum_j \nu_j \delta_{y_j} \in \mathcal{M}_+^1(Y)$ $\quad \bar{d}_{j,j'} = \bar{d}(y_j, y_{j'})$

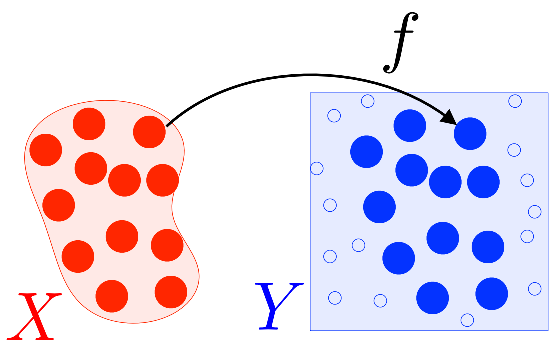


Def. *Metric-measured spaces* $(X, \mu, d) \in \mathbb{M}$:

$\mu \in \mathcal{M}_+^1(X)$ and d is a distance on X

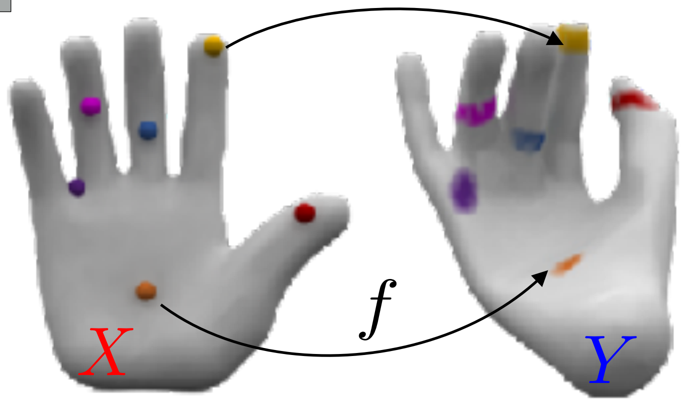
Def. *Isometries on* $\mathbb{M}$: $\quad (\mu, d) \sim (\nu, \bar{d})$

$$\iff \exists f : X \to Y, \begin{cases} f_\sharp \mu = \nu, \\ d(x, x') = \bar{d}(f(x), f(x')). \end{cases}$$



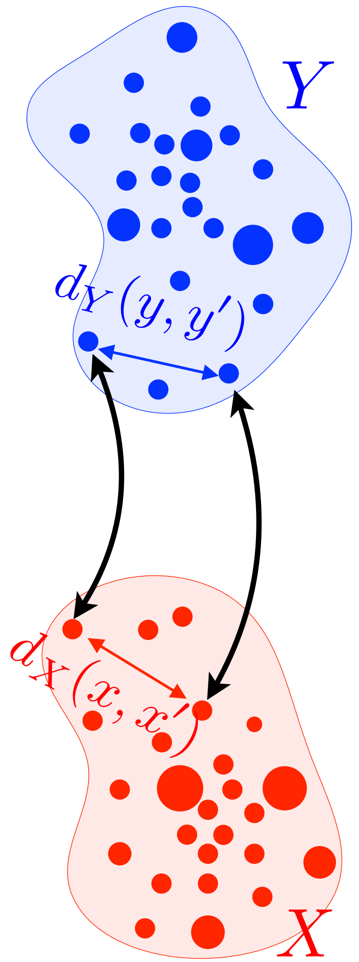
Prop. GW defines a distance on $\mathbb{M}/\sim$.
[Memoli 2011]

$\longrightarrow$ "bending-invariant" objects recognition.



For Arbitrary Spaces

Metric-measure spaces (X, Y): $(d_X, \mu), (d_Y, \nu)$



Def. *Gromov-Wasserstein distance:*

$$\mathrm{GW}_2^2(d_X, \mu, d_Y, \nu) \overset{\text{def.}}{=} \min_{\pi \in \Pi(\mu,\nu)} \int_{X^2 \times Y^2} |d_X(x, x') - d_Y(y, y')|^2 \mathrm{d}\pi(x, y) \mathrm{d}\pi(x', y')$$

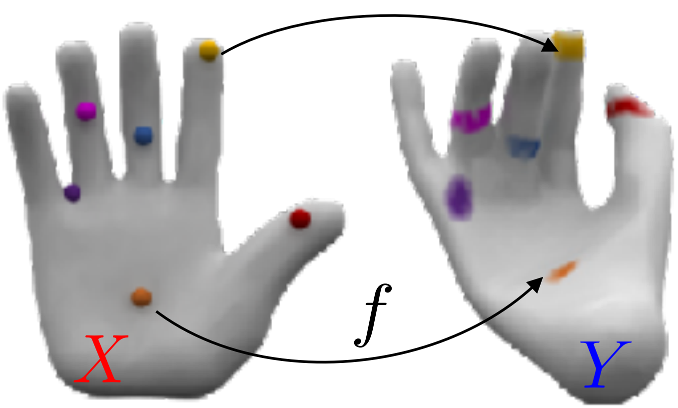
[Sturm 2012] [Memoli 2011]

Prop. GW is a distance on mm-spaces/isometries.

→ "bending-invariant" objects recognition.

→ QAP: NP-hard in general.

→ need for a fast approximate solver.



Overview

- Gromov Wasserstein
- **Entropic Regularization**
- Applications
- GW Barycenters

Entropic Gromov Wasserstein

Def. *Entropic Gromov-Wasserstein*

$$\mathrm{GW}^p_{p,\varepsilon}(d, \mu, \bar{d}, \nu) \overset{\text{def.}}{=} \min_{T \in C_{\mu,\nu}} \mathcal{E}^p_{d,\bar{d}}(T) - \varepsilon H(T)$$

Def. *Projected mirror descent:*

$$T \leftarrow \mathrm{Proj}^{\mathrm{KL}}_{\mathcal{C}_{\mu,\nu}} \left(T \odot e^{-\tau(-\nabla \mathcal{E}^p_{d,\bar{d}}(T) - \varepsilon \nabla H(T))} \right)$$

where $\mathrm{Proj}^{\mathrm{KL}}_{\mathcal{C}_{\mu,\nu}}(K) \overset{\text{def.}}{=} \mathrm{argmin}_T \{\mathrm{KL}(T|K) \; ; \; T \in \mathcal{C}_{\mu,\nu}\}$

Prop. for $\tau = 1/\varepsilon$, the iteration reads

$$T \leftarrow \texttt{Sinkhorn}(\mu, \nu, -d \times T \times \bar{d})$$

Prop. T converges to a stationary point.

func $T = \texttt{GW}(C, \bar{C}, p, q)$
initialization:
  $T \leftarrow \mu\nu^\top$
repeat:
  $D \leftarrow -d \times T \times \bar{d}$
  $T \leftarrow \texttt{Sinkhorn}(\mu, \nu, D)$
until convergence.
return T

Overview

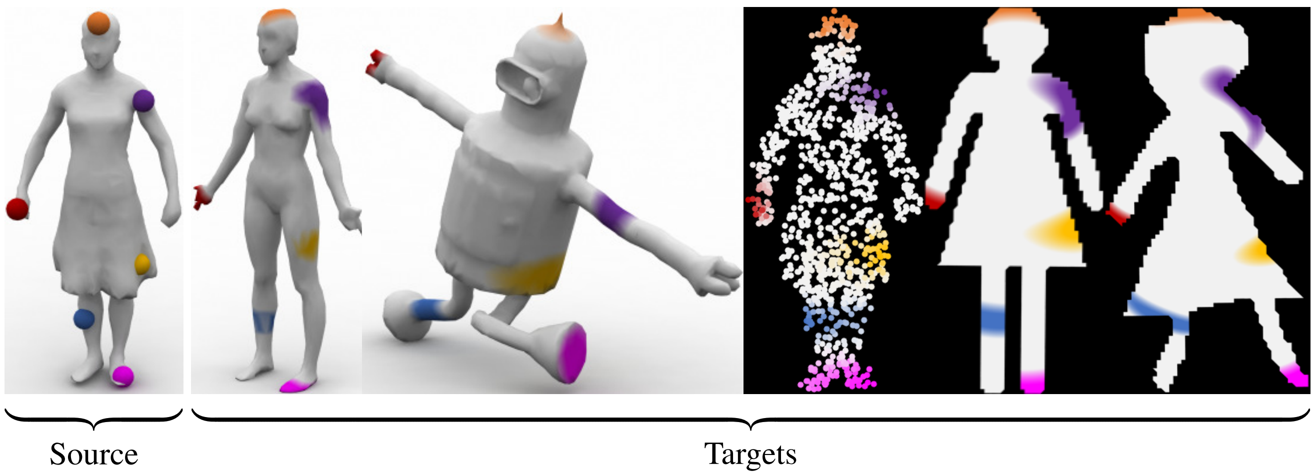
- Gromov Wasserstein
- Entropic Regularization
- **Applications**
- GW Barycenters

Applications of GW: Shapes Analysis

Use T to define registration between:

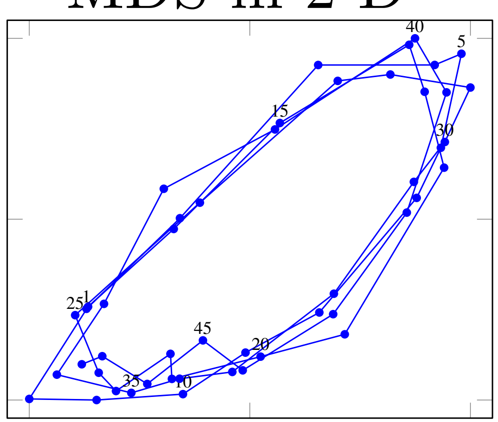
Shape ⟷ Shape

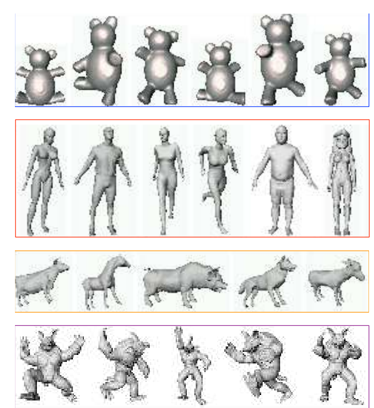
Colors distribution ⟷ Shape



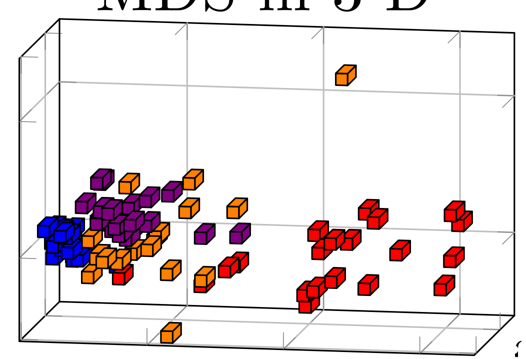
Shapes $(X_s)_s$ → Geodesic distances $d_s = (D_{X_s}(x_i, x_{i'}))_{i,i'}$ → GW distances $(\mathrm{GW}_\varepsilon(d_s, d_{s'}))_{s,s'}$ → MDS Vizualization

MDS in 2-D





MDS in 3-D



Applications of GW: Quantum Chemistry

Input: Molecules with positions and charges $\mu = \sum_i \mu_i \delta_{x_i}$.

Regression problem: approximate ground state energy $\mu \mapsto f(\mu)$.

$\rightarrow$ f by solving DFT approximation is too costly.

Coulomb matrices $d = d(\mu)$:

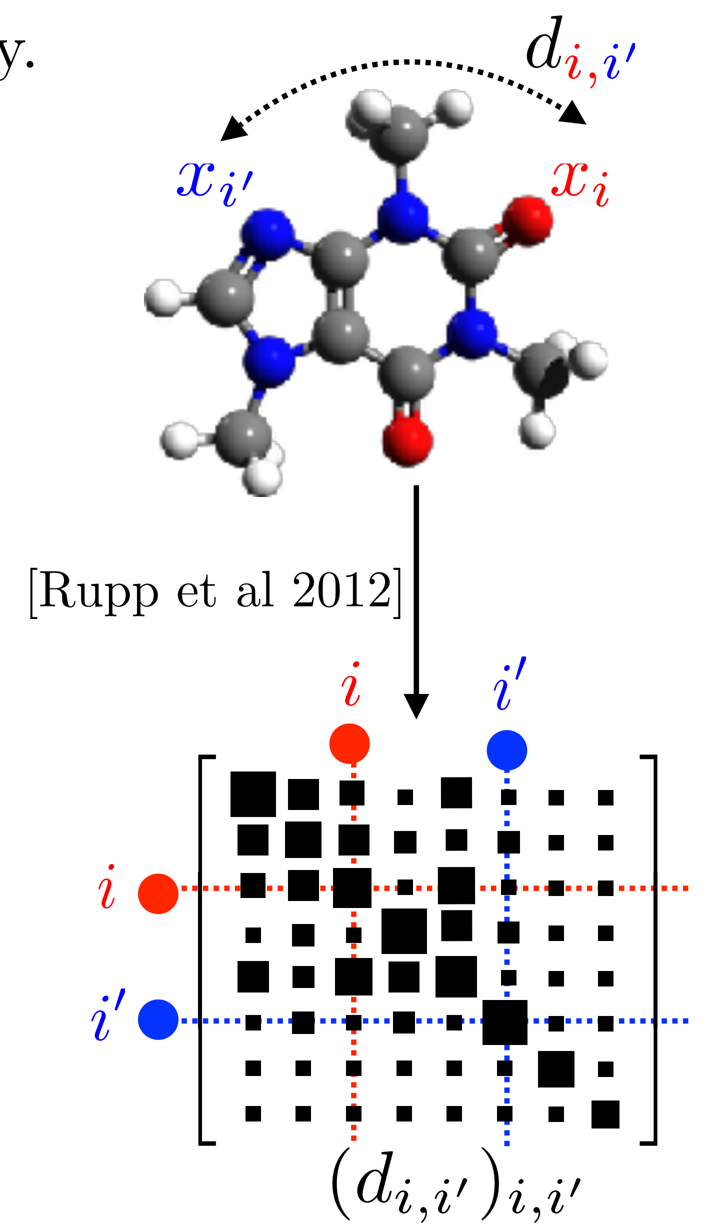
$$d_{i,i'} \overset{\text{def.}}{=} \begin{cases} \frac{\mu_i \mu_{i'}}{\|x_i - x_{i'}\|} & \text{for } (i \neq i') \\ \frac{1}{2}\mu_i^{2.4} & \text{for } (i = i'). \end{cases}$$

Learning: $(\mu_s, f(\mu_s))_s \rightarrow$ approximation $\tilde{f}$.

GW-interpolation: $\tilde{f}(\mu) = f(\mu_{s^\star})$

$$s^\star = \text{argmin}_s \, \text{GW}(d(\mu), d(\mu_s))$$

Algorithm	$\|f - \tilde{f}\|_1$
k-nearest neighbors	71.54
Linear regression	20.72
Gaussian kernel ridge regression	8.57
Laplacian kernel ridge regression (8)	3.07
Multilayer Neural Network (1000)	3.51
GW 3-nearest neighbors	10.83



Overview

- Gromov Wasserstein
- Entropic Regularization
- Applications
- **GW Barycenters**

Gromov-Wasserstein Geodesics

Def. *Gromov-Wasserstein Geodesic*

$$(\mu_t, d_t) \in \underset{(\mu,d)\in\mathbb{X}}{\operatorname{argmin}}\ (1-t)\mathrm{GW}_2^2(\mu_0, d_0, \mu, d) + t\mathrm{GW}_2^2(\mu_1, d_1, \mu, d)$$

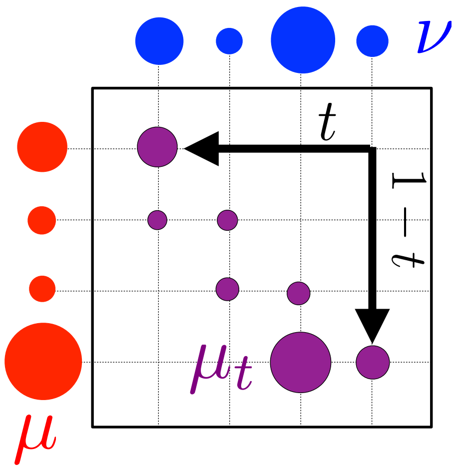
Optimal coupling $T^\star$: $\mathrm{GW}_2^2(d_0, \mu_0, d_1, \mu_1) \stackrel{\text{def.}}{=} \mathcal{E}^2_{d_0, d_1}(T^\star)$

Prop. One can define (μ_t, d_t) on $X \times Y$ as

$$\mu_t = \sum_{i,j} T^\star_{i,j} \delta_{x_i, y_j}$$

$$d_t((x,y),(x',y')) = (1-t)d_0(x,x') + t d_1(y,y')$$

[Sturm 2012]

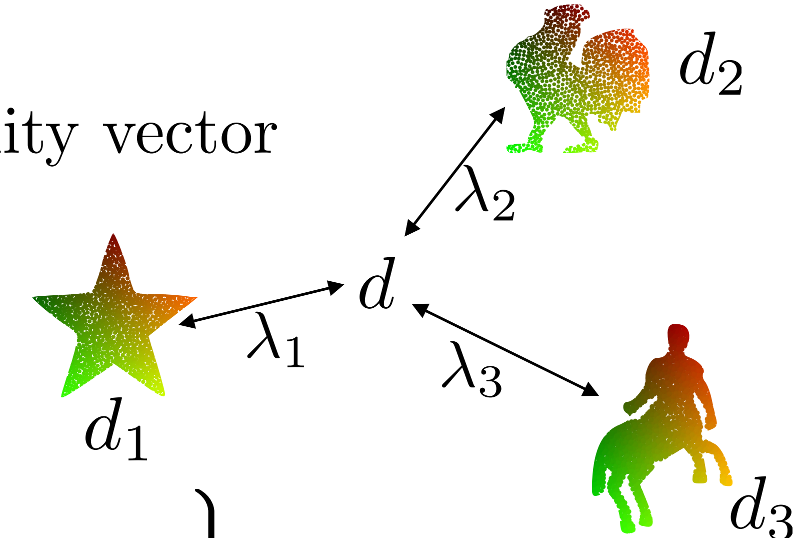


→ $X \times Y$ is not practical for most applications.
(need to fix the size of the geodesic embedding space)

→ Extension to more than 2 input spaces?

Gromov-Wasserstein Barycenters

Input: | Measures $(\mu_s)_s$, matrices $(d_s)_s$
Weights λ, size N, $\mu \in \mathbb{R}_+^N$ probability vector



Def. GW Barycenters

$$\min_{d \in \mathbb{R}^{N \times N}} \sum_s \lambda_s \mathrm{GW}^2_{2,\varepsilon}(d_s, \mu_s, d, \mu)$$

$$\min_{d,(T_s)_s} \left\{ \sum_s \lambda_s \left(\mathcal{E}^2_{d,d_s}(T_s) - \varepsilon H(T_s) \right) \; ; \; \forall\, s, T_s \in \mathcal{C}_{\mu,\mu_s} \right\}$$

Alternating minimization:

func C = GW-bary$(d_s, \mu_s, \mu)_s$

initialization: $C \leftarrow C_0$

repeat:

  for $s = 1$ **to** S **do**

On T_s → $T_s \leftarrow =$ GW(d, μ, d_s, μ_s)

On d → $d \leftarrow \frac{1}{\mu\mu^\top} \sum_s \lambda_s T_s^\top d_s T_s$

until convergence.

return C