Gromov-Wasserstein Learning for Structured Data Modeling

Hongteng Xu

Gaoling School of Artificial Intelligence, Renmin University of China
Beijing Key Laboratory of Big Data Management and Analysis Methods

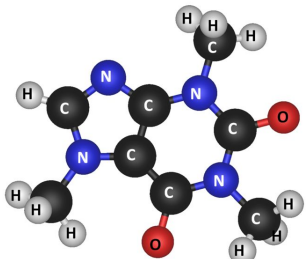
February 23, 2022

中國人民大學
RENMIN UNIVERSITY OF CHINA

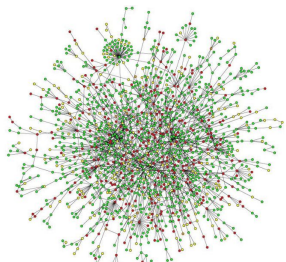
高瓴人工智能学院
Gaoling School of Artificial Intelligence

Outline

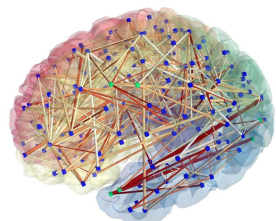
Real-world Structured Data



Molecule

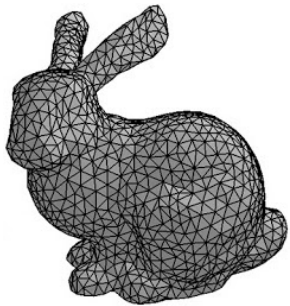


PPI Network

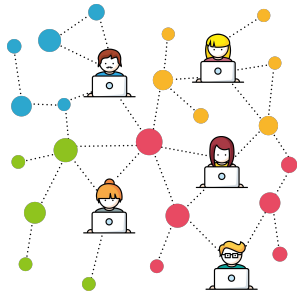


Brain Connectivity

Real-world Structured Data



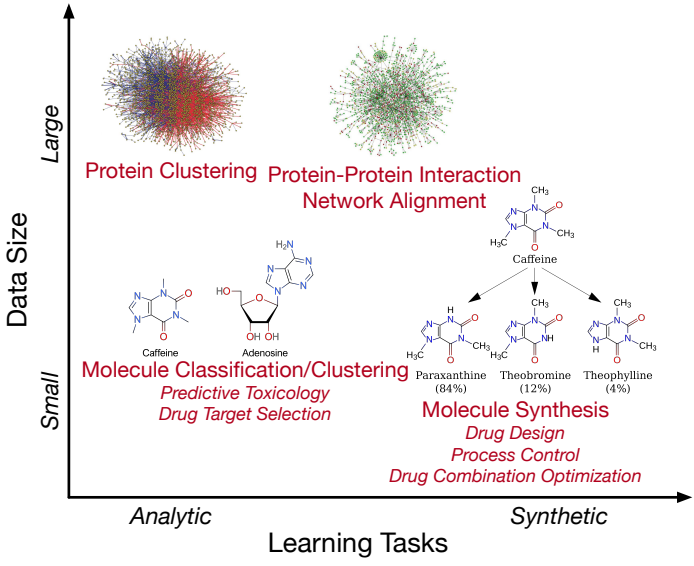
3D mesh



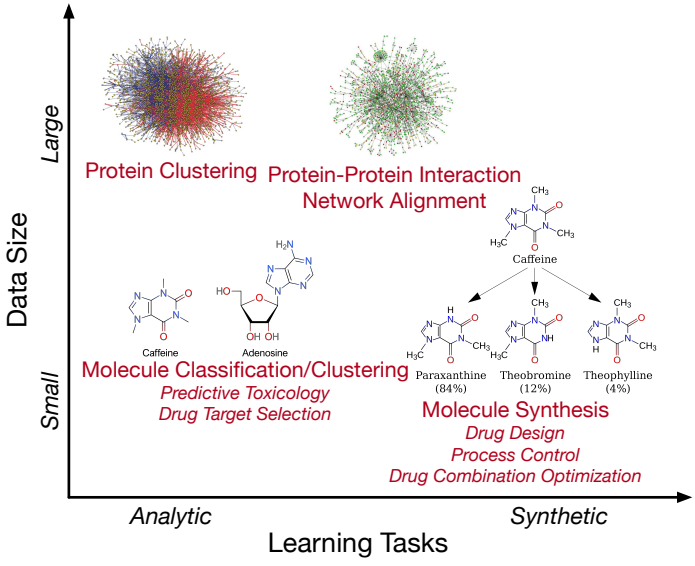
Social Network

Extract and predict relations among entities effectively and efficiently.

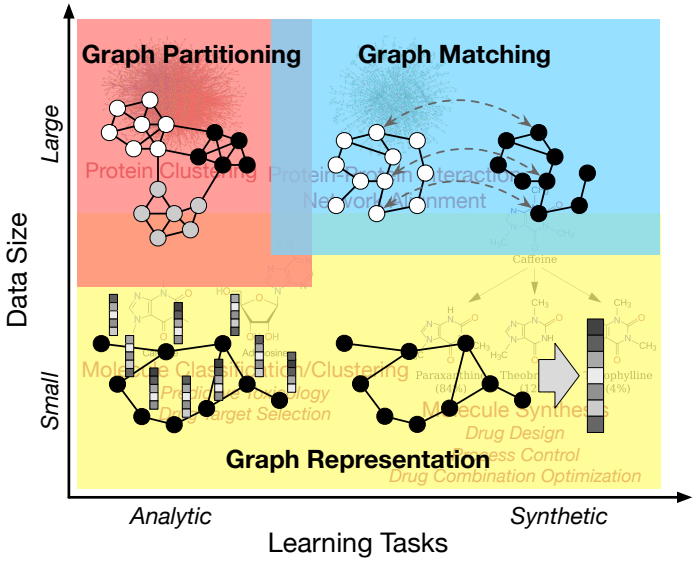
Typical (Biochemical) Problems of Structured Data

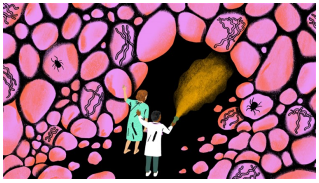


Typical (Biochemical) Problems of Structured Data



Typical (Biochemical) Problems of Structured Data



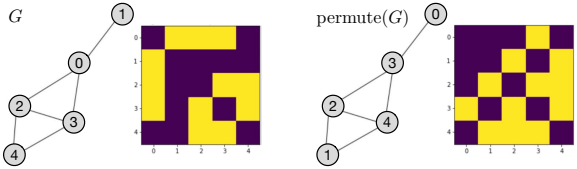


The Challenges of The Problems

- NP-completeness
  - Approximation algorithms with high stability and scalability

The Challenges of The Problems

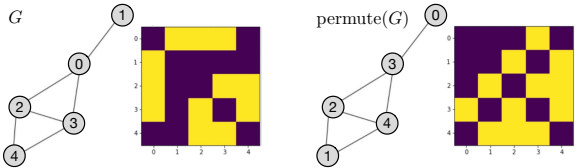
- NP-completeness
  - Approximation algorithms with high stability and scalability
- Permutation invariance



  - A permutation-invariant metric d: $d(G_X, G_Y) = d(G_X, \text{permute}(G_Y))$
  - A permutation-invariant representation model f: $f(G) = f(\text{permute}(G))$

The Challenges of The Problems

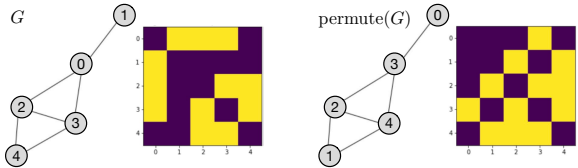
- NP-completeness
  - Approximation algorithms with high stability and scalability
- Permutation invariance



  - A permutation-invariant metric d: $d(G_X, G_Y) = d(G_X, \text{permute}(G_Y))$
  - A permutation-invariant representation model f: $f(G) = f(\text{permute}(G))$
- (Often) No labels
  - Unsupervised or semi-supervised learning

The Challenges of The Problems

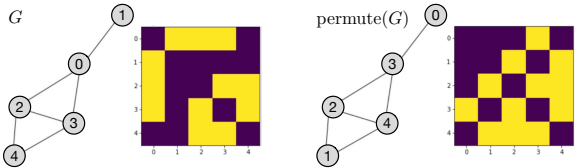
- NP-completeness
  - Approximation algorithms with high stability and scalability
- Permutation invariance



  - A permutation-invariant metric d: $d(G_X, G_Y) = d(G_X, \text{permute}(G_Y))$
  - A permutation-invariant representation model f: $f(G) = f(\text{permute}(G))$
- (Often) No labels
  - Unsupervised or semi-supervised learning
- Structured data generation
  - Generative model with generalizability and transferability

The Challenges of The Problems

- NP-completeness
  - Approximation algorithms with high stability and scalability
- Permutation invariance



  - A permutation-invariant metric d: $d(G_X, G_Y) = d(G_X, \text{permute}(G_Y))$
  - A permutation-invariant representation model f: $f(G) = f(\text{permute}(G))$
- (Often) No labels
  - Unsupervised or semi-supervised learning
- Structured data generation
  - Generative model with generalizability and transferability

Gromov-Wasserstein Learning (GWL): An optimal transport-based machine learning framework to solve the problems.

What Is GWL and Where Is It From?

Landmarks in History (A very personal and non-theoretical viewpoint)

What Is GWL and Where Is It From?

Landmarks in History (A very personal and non-theoretical viewpoint)

- **1781, Monge-form** of optimal transport problem (Gaspard Monge)

What Is GWL and Where Is It From?

Landmarks in History (A very personal and non-theoretical viewpoint)

- **1781, Monge-form** of optimal transport problem (Gaspard Monge)
- **1939, Kantorovich-form** of optimal transport problem (Wasserstein distance) and its dual problem (Leonid Kantorovich)

What Is GWL and Where Is It From?

Landmarks in History (A very personal and non-theoretical viewpoint)

- **1781, Monge-form** of optimal transport problem (Gaspard Monge)
- **1939, Kantorovich-form** of optimal transport problem (Wasserstein distance) and its dual problem (Leonid Kantorovich)
- **1967, Sinkhorn-Knopp algorithm** for doubly-stochastic matrix (Richard Sinkhorn & Paul Knopp)

What Is GWL and Where Is It From?

Landmarks in History (A very personal and non-theoretical viewpoint)

- **1781, Monge-form** of optimal transport problem (Gaspard Monge)
- **1939, Kantorovich-form** of optimal transport problem (Wasserstein distance) and its dual problem (Leonid Kantorovich)
- **1967, Sinkhorn-Knopp algorithm** for doubly-stochastic matrix (Richard Sinkhorn & Paul Knopp)
- **1981, Gromovization of metric**, e.g., Gromov-Hausdorff (Mikhail Gromov)

What Is GWL and Where Is It From?

Landmarks in History (A very personal and non-theoretical viewpoint)

- **1781, Monge-form** of optimal transport problem (Gaspard Monge)
- **1939, Kantorovich-form** of optimal transport problem (Wasserstein distance) and its dual problem (Leonid Kantorovich)
- **1967, Sinkhorn-Knopp algorithm** for doubly-stochastic matrix (Richard Sinkhorn & Paul Knopp)
- **1981, Gromovization of metric**, e.g., Gromov-Hausdorff (Mikhail Gromov)
- **2006 - 2007, Gromov-Wasserstein distance** for geometric problems like shape matching (Karl-Theodor Sturm, Facundo Mémoli)

What Is GWL and Where Is It From?

Landmarks in History (A very personal and non-theoretical viewpoint)

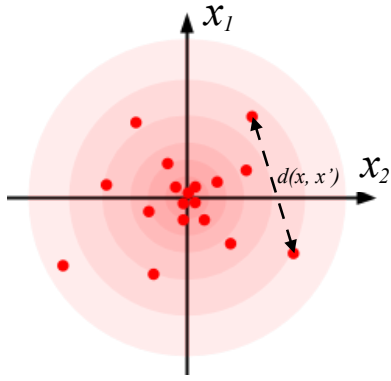
- **1781, Monge-form** of optimal transport problem (Gaspard Monge)
- **1939, Kantorovich-form** of optimal transport problem (Wasserstein distance) and its dual problem (Leonid Kantorovich)
- **1967, Sinkhorn-Knopp algorithm** for doubly-stochastic matrix (Richard Sinkhorn & Paul Knopp)
- **1981, Gromovization of metric**, e.g., Gromov-Hausdorff (Mikhail Gromov)
- **2006 - 2007, Gromov-Wasserstein distance** for geometric problems like shape matching (Karl-Theodor Sturm, Facundo Mémoli)
- **2013 - Present, Progress on computational optimal transport**
  - Entropic optimal transport and Sinkhorn scaling (2013, Marco Cuturi)
  - Variants of OT problems and GW distance, and Python OT package (Gabriel Peyré, Justin Solomon, Rémi Flamary, Nicolas Courty, ...)

What Is GWL and Where Is It From?

Landmarks in History (A very personal and non-theoretical viewpoint)

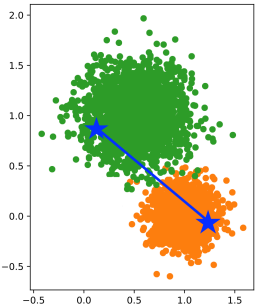
- **1781, Monge-form** of optimal transport problem (Gaspard Monge)
- **1939, Kantorovich-form** of optimal transport problem (Wasserstein distance) and its dual problem (Leonid Kantorovich)
- **1967, Sinkhorn-Knopp algorithm** for doubly-stochastic matrix (Richard Sinkhorn & Paul Knopp)
- **1981, Gromovization of metric**, e.g., Gromov-Hausdorff (Mikhail Gromov)
- **2006 - 2007, Gromov-Wasserstein distance** for geometric problems like shape matching (Karl-Theodor Sturm, Facundo Mémoli)
- **2013 - Present, Progress on computational optimal transport**
  - Entropic optimal transport and Sinkhorn scaling (2013, Marco Cuturi)
  - Variants of OT problems and GW distance, and Python OT package (Gabriel Peyré, Justin Solomon, Rémi Flamary, Nicolas Courty, ...)
- **2016 - Present, Blooming in machine learning and data science**
  - Wasserstein GAN (2017, Arjovsky, Martin, Soumith Chintala, and Léon Bottou)
  - Gromov-Wasserstein learning for graph modeling and analysis (2019, Hongteng Xu, Titouan Vayer, Makoto Yamata, Tam Le, Samir Chowdhury, Tom Needham, ...)

Metric-Measure Space (MM-Space)



- $\mathcal{X}_{d,\mu} := (\mathcal{X}, d, \mu)$: A metric-measure space, where $x \in \mathcal{X}$ is a sample in the space.
- d: A distance metric of samples (e.g., Euclidean distance).
- $\mathbb{P}$: A space of (probabilistic) measures defined on $\mathcal{X}$.
- $\mu \in \mathbb{P}$: a measure on $\mathcal{X}$.

Optimal Transport between the Measures in the Same Space

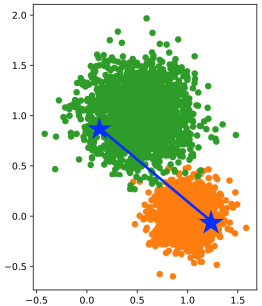


Kantorovich-form of optimal transport problem:

$$d_w(\mu,\eta) := \inf_{\pi\in\Pi(\mu,\eta)} \mathbb{E}_{x,y\sim\pi}[d(x,y)] = \inf_{\pi\in\Pi(\mu,\eta)} \iint_{(x,y)\in\mathcal{X}^2} d(x,y)\pi(x,y)dxdy, \qquad (1)$$

where $\Pi(\mu,\eta) = \{\pi > 0 | \int_x \pi(x,y)dx = \eta(y), \int_y \pi(x,y)dy = \mu(x)\}$, the optimum π^* is called optimal transport (plan).

Optimal Transport between the Measures in the Same Space

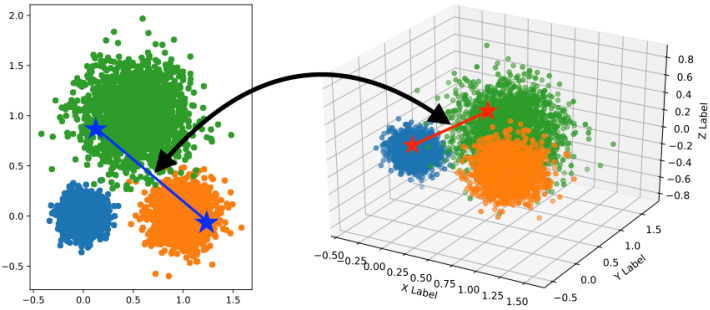


Given finite numbers of samples and their empirical distributions, i.e., $\{x_m\}_{m=1}^M$, $\{y_n\}_{n=1}^N$, $\boldsymbol{\mu} \in \Delta^{N-1}$ and $\boldsymbol{\eta} \in \Delta^{M-1}$, we have:

$$\hat{d}_w(\boldsymbol{\mu}, \boldsymbol{\eta}) = \min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\eta})} \langle \boldsymbol{D}, \boldsymbol{T} \rangle = \min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\eta})} \sum_{m=1}^{M} \sum_{n=1}^{N} d(x_m, y_n) T_{mn}, \tag{2}$$

where $\boldsymbol{T} = [T_{mn}]$, $\Pi(\boldsymbol{\mu}, \boldsymbol{\eta}) = \{\boldsymbol{T} > \mathbf{0} | \boldsymbol{T}\mathbf{1}_M = \boldsymbol{\mu}, \boldsymbol{T}^\top \mathbf{1}_N = \boldsymbol{\eta}\}$, the optimum $\boldsymbol{T}^*$ is called optimal transport matrix.

Gromov-Wasserstein Distance for MM-Spaces

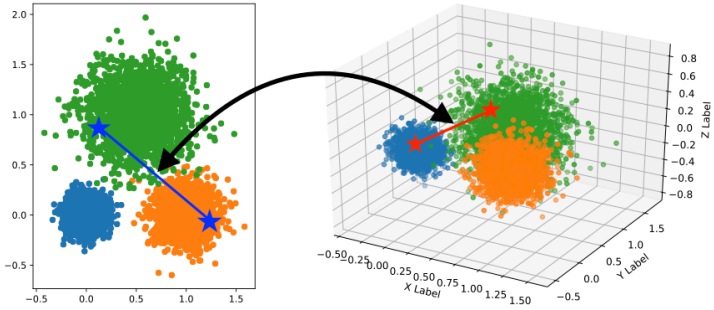


The Gromov-Wasserstein distance between $\mathcal{X}_{d_X,\mu_X}$ and $\mathcal{Y}_{d_Y,\mu_Y}$:

$$d_{gw}(\mathcal{X}_{d_X,\mu_X},\mathcal{Y}_{d_Y,\mu_Y}) := \inf_{\pi\in\Pi(\mu_X,\mu_Y)} \mathbb{E}_{x,x',y,y'\sim\pi\times\pi}[r(x,x',y,y')]$$
$$= \inf_{\pi\in\Pi(\mu_X,\mu_Y)} \iint_{(x,x')\in\mathcal{X}^2} \iint_{(y,y')\in\mathcal{Y}^2} r(x,x',y,y')\pi(x,y)\pi(x',y')dxdx'dydy', \quad (3)$$

where $r(x,x',y,y') = d(d_X(x,x'),d_Y(y,y'))$ is called relational distance.

Gromov-Wasserstein Distance for Structured Data (Point Clouds)

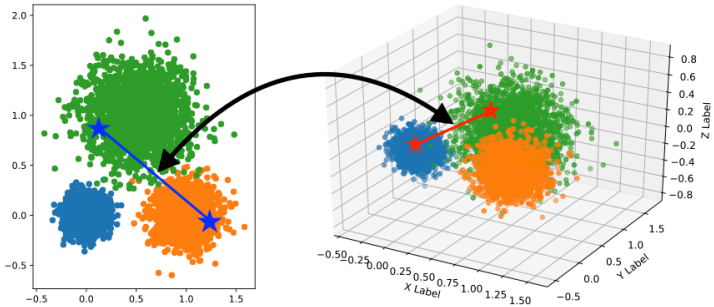


Given finite numbers of samples and their empirical distributions, i.e., $\boldsymbol{X} = \{x_m\}_{m=1}^{M}$, $\boldsymbol{Y} = \{y_n\}_{n=1}^{N}$, $\boldsymbol{\mu}_X \in \Delta^{N-1}$ and $\boldsymbol{\mu}_Y \in \Delta^{M-1}$, we have:

$$\begin{aligned} \hat{d}_{gw}(\boldsymbol{X}, \boldsymbol{Y}) &= \min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_X, \boldsymbol{\mu}_Y)} \langle \boldsymbol{R}, \boldsymbol{T} \otimes \boldsymbol{T} \rangle \\ &= \min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_X, \boldsymbol{\mu}_Y)} \sum_{m,m'=1}^{M} \sum_{n,n'=1}^{N} r(x_m, x_{m'}, y_n, y_{n'}) T_{mn} T_{m'n'}, \end{aligned} \tag{4}$$

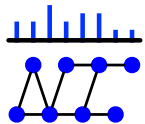
where $\boldsymbol{R} = [r(x_m, x_{m'}, y_n, y_{n'})] \in \mathbb{R}^{M^2 \times N^2}$.

Gromov-Wasserstein Distance for Structured Data (Point Clouds)



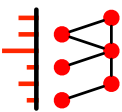
- Commonly-used relational distance:
  $r(x_m, x_{m'}, y_n, y_{n'}) = |d_X(x_m, x_{m'}) - d_Y(y_n, y_{n'})|^2$, where d_X and d_Y are often Euclidean.
- Useful properties:
  - Translation-, rotation-, and permutation-invariance

Gromov-Wasserstein Distance for Structured Data (Graphs)



$G(\mathcal{V}_Y, \mu_Y, \mathbf{D}_Y)$

$G(\mathcal{V}_X, \mu_X, \mathbf{D}_X)$

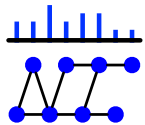


From a mm-space $\mathcal{X}_{d,\mu}$ to a measure graph $G_{\boldsymbol{D},\boldsymbol{\mu}} := G(\mathcal{V}, \boldsymbol{\mu}, \boldsymbol{D})$ [Chowdhury, et al., 2019]

- $\mathcal{V}$: A node set
- $\boldsymbol{\mu}$: a predefined distribution of nodes
- $\boldsymbol{D} = [d_{ii'}]$: a matrix recording the relations among nodes.

[Chowdhury, et al., 2019] Chowdhury, S., & Mémoli, F. The gromovwasserstein distance between networks and stable network invariants. Information and Inference: A Journal of the IMA, 2019.

Gromov-Wasserstein Distance for Structured Data (Graphs)

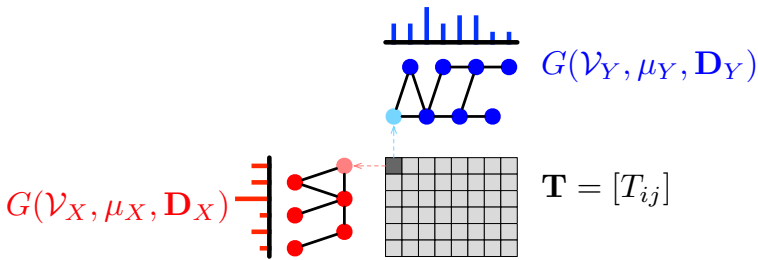


$G(\mathcal{V}_Y, \mu_Y, \mathbf{D}_Y)$

$G(\mathcal{V}_X, \mu_X, \mathbf{D}_X)$

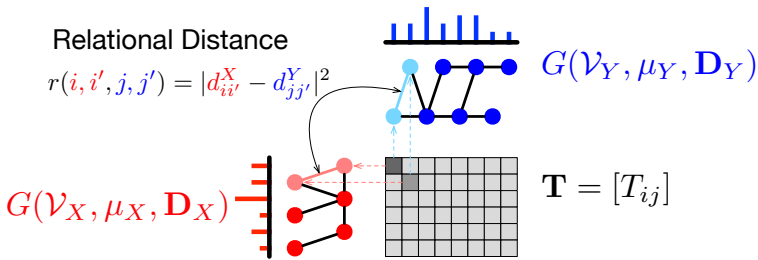
- Options of $\boldsymbol{\mu}$:
  - Uniform distribution: $\boldsymbol{\mu} = \frac{1}{|\mathcal{V}|}\mathbf{1}_{|\mathcal{V}|}$.
  - Degree distribution [Xu, et al. NeurIPS 2019]: $\boldsymbol{\mu} = \frac{1}{\mathbf{1}^T\boldsymbol{p}}\boldsymbol{p}$, $p_i = (deg(v_i) + a)^b$, $a, b \geq 0$.
- Options of $\boldsymbol{D} = [d_{ii'}]$:
  - Adjacency matrix $\boldsymbol{A} = [a_{ii'}]$ or Laplacian matrix $\boldsymbol{L} = diag(\boldsymbol{A}\mathbf{1}) - \boldsymbol{A}$.
  - Distance matrix $\boldsymbol{D} = [d_{ii'}]$, where $d_{ii'}$ is the shortest path distance.
  - Kernel matrix $\boldsymbol{K}$, e.g., the t-th order heat kernel $\boldsymbol{K}^t = \exp(-t\boldsymbol{L})$.

Gromov-Wasserstein Distance for Structured Data (Graphs)



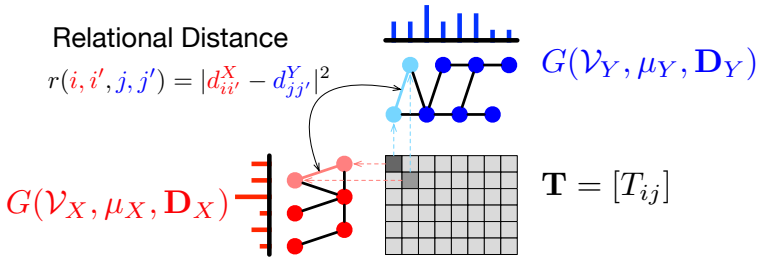
- $\boldsymbol{T} = [T_{ij}]$: a joint distribution of nodes
- $\boldsymbol{T} \in \Pi(\boldsymbol{u}_X, \boldsymbol{u}_Y) = \{\boldsymbol{T} \geq \mathbf{0} \mid \boldsymbol{T}\mathbf{1} = \boldsymbol{u}_X,\ \boldsymbol{T}^\top \mathbf{1} = \boldsymbol{u}_Y\}$
- The pair of nodes $(i \in \mathcal{V}_X, j \in \mathcal{V}_j) \sim \boldsymbol{T}$.

Gromov-Wasserstein Distance for Structured Data (Graphs)



- The pair of relations $(d^X_{ii'}, d^Y_{jj'}) \sim \boldsymbol{T} \times \boldsymbol{T}$.
- $r(i,i',j,j')$: the relational distance between node pairs.

Gromov-Wasserstein Distance for Structured Data (Graphs)



- The pair of relations $(d^X_{ii'}, d^Y_{jj'}) \sim \boldsymbol{T} \times \boldsymbol{T}$.
- $r(i,i',j,j')$: the relational distance between node pairs.

The GWD is **the minimum expectation of the relational distance:**

$$d_{gw}(G_X, G_Y) := \min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_X, \boldsymbol{\mu}_Y)} \mathbb{E}_{(i,i',j,j') \sim \boldsymbol{T} \times \boldsymbol{T}}[r(i,i',j,j')] = \min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_X, \boldsymbol{\mu}_Y)} \langle \boldsymbol{R}, \boldsymbol{T} \otimes \boldsymbol{T} \rangle. \quad (5)$$

Simplification Under Specific Relational Distance

- Commonly used relational distance:
  - Euclidean distance (MSE): $r(i,i',j,j') = |d^X_{ii'} - d^Y_{jj'}|^2$
  - KL-divergence: $r(i,i',j,j') = d^X_{ii'} \log \frac{d^X_{ii'}}{d^Y_{jj'}} - d^X_{ii'} + d^Y_{jj'}$.
  - Cross-entropy loss: $r(i,i',j,j') = -d^X_{ii'} \log d^Y_{jj'} - (1 - d^X_{ii'}) \log(1 - d^Y_{jj'})$.

Simplification Under Specific Relational Distance

- Commonly used relational distance:
  - Euclidean distance (MSE): $r(i,i',j,j') = |d^X_{ii'} - d^Y_{jj'}|^2$
  - KL-divergence: $r(i,i',j,j') = d^X_{ii'} \log \frac{d^X_{ii'}}{d^Y_{jj'}} - d^X_{ii'} + d^Y_{jj'}$.
  - Cross-entropy loss: $r(i,i',j,j') = -d^X_{ii'} \log d^Y_{jj'} - (1 - d^X_{ii'}) \log(1 - d^Y_{jj'})$.
- A special form $r(i,i',j,j') = f_1(d^X_{ii'}) + f_2(d^Y_{jj'}) - h_1(d^X_{ii'})h_2(d^Y_{jj'})$:
  - Euclidean distance:
    $f_1(d^X_{ii'}) = (d^X_{ii'})^2$, $f_2(d^Y_{jj'}) = (d^Y_{jj'})^2$, $h_1(d^X_{ii'}) = d^X_{ii'}$, and $h_2(d^Y_{jj'}) = 2d^Y_{jj'}$.

Simplification Under Specific Relational Distance

- Commonly used relational distance:
  - Euclidean distance (MSE): $r(i,i',j,j') = |d^X_{ii'} - d^Y_{jj'}|^2$
  - KL-divergence: $r(i,i',j,j') = d^X_{ii'} \log \frac{d^X_{ii'}}{d^Y_{jj'}} - d^X_{ii'} + d^Y_{jj'}$.
  - Cross-entropy loss: $r(i,i',j,j') = -d^X_{ii'} \log d^Y_{jj'} - (1 - d^X_{ii'}) \log(1 - d^Y_{jj'})$.
- A special form $r(i,i',j,j') = f_1(d^X_{ii'}) + f_2(d^Y_{jj'}) - h_1(d^X_{ii'}) h_2(d^Y_{jj'})$:
  - Euclidean distance:
    $f_1(d^X_{ii'}) = (d^X_{ii'})^2$, $f_2(d^Y_{jj'}) = (d^Y_{jj'})^2$, $h_1(d^X_{ii'}) = d^X_{ii'}$, and $h_2(d^Y_{jj'}) = 2d^Y_{jj'}$.
  - KL-divergence:
    $f_1(d^X_{ii'}) = d^X_{ii'}(\log d^X_{ii'} - 1)$, $f_2(d^Y_{jj'}) = d^Y_{jj'}$, $h_1(d^X_{ii'}) = d^X_{ii'}$, and $h_2(d^Y_{jj'}) = \log d^Y_{jj'}$.

Simplification Under Specific Relational Distance

- Commonly used relational distance:
  - Euclidean distance (MSE): $r(i,i',j,j') = |d^X_{ii'} - d^Y_{jj'}|^2$
  - KL-divergence: $r(i,i',j,j') = d^X_{ii'} \log \frac{d^X_{ii'}}{d^Y_{jj'}} - d^X_{ii'} + d^Y_{jj'}$.
  - Cross-entropy loss: $r(i,i',j,j') = -d^X_{ii'} \log d^Y_{jj'} - (1 - d^X_{ii'}) \log(1 - d^Y_{jj'})$.
- A special form $r(i,i',j,j') = f_1(d^X_{ii'}) + f_2(d^Y_{jj'}) - h_1(d^X_{ii'})h_2(d^Y_{jj'})$:
  - Euclidean distance:
    $f_1(d^X_{ii'}) = (d^X_{ii'})^2$, $f_2(d^Y_{jj'}) = (d^Y_{jj'})^2$, $h_1(d^X_{ii'}) = d^X_{ii'}$, and $h_2(d^Y_{jj'}) = 2d^Y_{jj'}$.
  - KL-divergence:
    $f_1(d^X_{ii'}) = d^X_{ii'}(\log d^X_{ii'} - 1)$, $f_2(d^Y_{jj'}) = d^Y_{jj'}$, $h_1(d^X_{ii'}) = d^X_{ii'}$, and $h_2(d^Y_{jj'}) = \log d^Y_{jj'}$.
  - Cross-entropy loss:
    $f_1(d^X_{ii'}) = 0$, $f_2(d^Y_{jj'}) = -\log(1 - d^Y_{jj'})$, $h_1(d^X_{ii'}) = d^X_{ii'}$, and $h_2(d^Y_{jj'}) = \log \frac{d^Y_{jj'}}{1 - d^Y_{jj'}}$.

Simplification Under Specific Relational Distance

- Proposition 1 in [Peyré, et al., ICML 2016]

$$\langle \boldsymbol{R}, \boldsymbol{T} \otimes \boldsymbol{T} \rangle = \langle f_1(\boldsymbol{D}_X)\boldsymbol{\mu}_X \mathbf{1}_M^\top + \mathbf{1}_N \boldsymbol{\mu}_Y^\top f_2(\boldsymbol{D}_Y^\top) - h_1(\boldsymbol{D}_X)\boldsymbol{T} h_2(\boldsymbol{D}_Y^\top), \boldsymbol{T} \rangle. \quad (6)$$

Simplification Under Specific Relational Distance

- Proposition 1 in [Peyré, et al., ICML 2016]

$$\langle \boldsymbol{R}, \boldsymbol{T} \otimes \boldsymbol{T} \rangle = \langle f_1(\boldsymbol{D}_X)\boldsymbol{\mu}_X \mathbf{1}_M^\top + \mathbf{1}_N \boldsymbol{\mu}_Y^\top f_2(\boldsymbol{D}_Y^\top) - h_1(\boldsymbol{D}_X)\boldsymbol{T} h_2(\boldsymbol{D}_Y^\top), \boldsymbol{T} \rangle. \quad (6)$$

- Furthermore, for $\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_X, \boldsymbol{\mu}_Y)$, we have

$$\begin{aligned} &\langle f_1(\boldsymbol{D}_X)\boldsymbol{\mu}_X \mathbf{1}_M^\top + \mathbf{1}_N \boldsymbol{\mu}_Y^\top f_2(\boldsymbol{D}_Y^\top), \boldsymbol{T} \rangle \\ =&tr(f_1(\boldsymbol{D}_X)\boldsymbol{\mu}_X \underbrace{\mathbf{1}_M^\top \boldsymbol{T}^\top}_{\boldsymbol{\mu}_X^\top}) + tr(\underbrace{\boldsymbol{T}^\top \mathbf{1}_N}_{\boldsymbol{\mu}^Y} \boldsymbol{\mu}_Y^\top f_2(\boldsymbol{D}_Y^\top). \end{aligned} \quad (7)$$

Simplification Under Specific Relational Distance

- Proposition 1 in [Peyré, et al., ICML 2016]

$$\langle \boldsymbol{R}, \boldsymbol{T} \otimes \boldsymbol{T} \rangle = \langle f_1(\boldsymbol{D}_X)\boldsymbol{\mu}_X \mathbf{1}_M^\top + \mathbf{1}_N \boldsymbol{\mu}_Y^\top f_2(\boldsymbol{D}_Y^\top) - h_1(\boldsymbol{D}_X)\boldsymbol{T} h_2(\boldsymbol{D}_Y^\top), \boldsymbol{T} \rangle. \quad (6)$$

- Furthermore, for $\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_X, \boldsymbol{\mu}_Y)$, we have

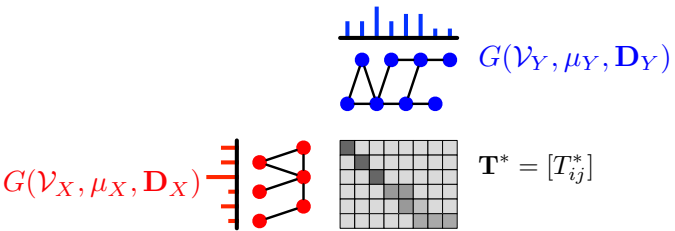
$$\begin{aligned} &\langle f_1(\boldsymbol{D}_X)\boldsymbol{\mu}_X \mathbf{1}_M^\top + \mathbf{1}_N \boldsymbol{\mu}_Y^\top f_2(\boldsymbol{D}_Y^\top), \boldsymbol{T} \rangle \\ =& tr(f_1(\boldsymbol{D}_X)\boldsymbol{\mu}_X \underbrace{\mathbf{1}_M^\top \boldsymbol{T}^\top}_{\boldsymbol{\mu}_X^\top}) + tr(\underbrace{\boldsymbol{T}^\top \mathbf{1}_N}_{\boldsymbol{\mu}^Y} \boldsymbol{\mu}_Y^\top f_2(\boldsymbol{D}_Y^\top). \end{aligned} \quad (7)$$

- Accordingly, we can simplify the computation of GWD from $\mathcal{O}(M^2N^2)$ to $\mathcal{O}(M^2N + N^2M)$ (or $\mathcal{O}(N^3)$ when $M \approx N$):

$$\begin{aligned} \boldsymbol{T}^* &= \arg\min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_X, \boldsymbol{\mu}_Y)} -\langle h_1(\boldsymbol{D}_X)\boldsymbol{T} h_2(\boldsymbol{D}_Y^\top), \boldsymbol{T} \rangle. \\ \xrightarrow{\text{Euclidean } r} \boldsymbol{T}^* &= \arg\min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_X, \boldsymbol{\mu}_Y)} -\langle \boldsymbol{D}_X \boldsymbol{T} \boldsymbol{D}_Y^\top, \boldsymbol{T} \rangle. \end{aligned} \quad (8)$$

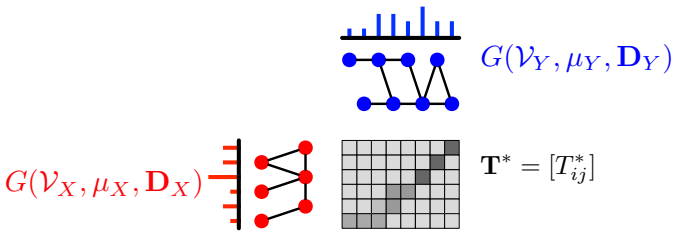
[Peyré, et al., ICML 2016] Peyré, G., Cuturi, M., & Solomon, J. Gromov-Wasserstein averaging of kernel and distance matrices. ICML, 2016.

Advantages of GWD



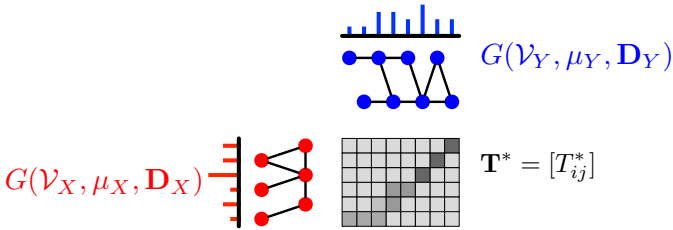
- The optimal joint distribution $\boldsymbol{T}^*$ (or called "**optimal transport**" matrix) indicates the correspondence between the two graphs/point clouds.

Advantages of GWD



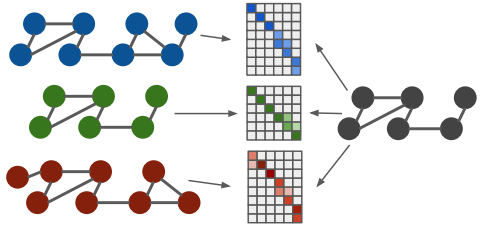
- The optimal joint distribution $\boldsymbol{T}^*$ (or called "**optimal transport**" matrix) indicates the correspondence between the two graphs.
- A (pseudo) metric with symmetry and following invariance properties
  - **Translation+Rotation-invariance for point clouds:** $d_{gw}(\boldsymbol{X}, \boldsymbol{Y}) = d_{gw}(\boldsymbol{X}, \mathsf{Rot}{+}\mathsf{Trans}(\boldsymbol{Y}))$
  - **Permutation-invariance for graphs:** $d_{gw}(G_X, G_Y) = d_{gw}(G_X, \mathsf{Permute}(G_Y))$

Advantages of GWD



- The optimal transport $\boldsymbol{T}^*$ is a joint distribution indicating the correspondence between the two graphs.
- A (pseudo) metric with symmetry and following invariance properties
  - **Translation+Rotation-invariance for point clouds:** $d_{gw}(\boldsymbol{X}, \boldsymbol{Y}) = d_{gw}(\boldsymbol{X}, \mathsf{Rot{+}Trans}(\boldsymbol{Y}))$
  - **Permutation-invariance for graphs:** $d_{gw}(G_X, G_Y) = d_{gw}(G_X, \mathsf{Permute}(G_Y))$
- Applicable to the graphs with different sizes, $i.e.$, $|\mathcal{V}_X| \neq |\mathcal{V}_Y|$.

From GW Distance to a (Weighted) GW Barycenter

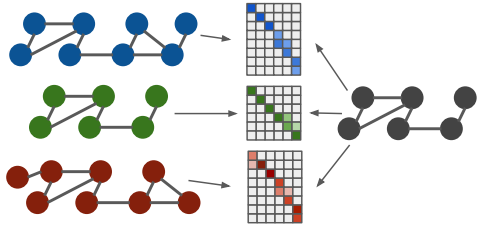


Given $\{G_k\}_{k=1}^K$, $K \geq 2$, their (weighted) GW barycenter [Peyré, et al., ICML 2016] is

$$\underbrace{\bar{G}(\bar{\mathcal{V}}, \bar{\boldsymbol{\mu}}, \boldsymbol{B}^*)}_{\text{Barycenter graph}}, \underbrace{\{\boldsymbol{T}_k^*\}_{k=1}^K}_{\text{OT matrices}} := \arg\min_G \sum_{k=1}^K \lambda_k d_{gw}(G_k, G) \tag{9}$$

$$= \arg\min_B \min_{\{\boldsymbol{T}_k \in \Pi(\boldsymbol{\mu}_k, \bar{\boldsymbol{\mu}})\}_{k=1}^K} -\sum_{k=1}^K \lambda_k \langle \boldsymbol{D}_k \boldsymbol{T}_k \boldsymbol{B}^\top, \boldsymbol{T}_k \rangle$$

From GW Distance to a (Weighted) GW Barycenter



Given $\{G_k\}_{k=1}^K$, $K \geq 2$, their (weighted) GW barycenter [Peyré, et al., ICML 2016] is

$$\underbrace{\bar{G}(\bar{\mathcal{V}}, \bar{\boldsymbol{\mu}}, \boldsymbol{B}^*)}_{\text{Barycenter graph}}, \underbrace{\{\boldsymbol{T}_k^*\}_{k=1}^K}_{\text{OT matrices}} := \arg\min_G \sum_{k=1}^K \lambda_k d_{gw}(G_k, G) \tag{9}$$

$$= \arg\min_B \min_{\{\boldsymbol{T}_k \in \Pi(\boldsymbol{\mu}_k, \bar{\boldsymbol{\mu}})\}_{k=1}^K} -\sum_{k=1}^K \lambda_k \langle \boldsymbol{D}_k \boldsymbol{T}_k \boldsymbol{B}^\top, \boldsymbol{T}_k \rangle$$

Permutation-invariance: If $\bar{G}$ is a GW barycenter of $\{G_k\}_{k=1}^K$, then permute($\bar{G}$) is a valid GW barycenter as well.

Implementation of GW Barycenter

- Approximation of $\bar{\boldsymbol{\mu}}$ [Xu, et al., NeurIPS 2019]:

$$\bar{\boldsymbol{\mu}} = \sum_{k=1}^{K} \lambda_k \text{interpolate}_{|\bar{\mathcal{V}}|}(\text{sort}(\boldsymbol{\mu}_k)), \tag{10}$$

  - $\text{sort}(\cdot)$ sorts the elements of the input vector in descending order.
  - $\text{interpolate}_{|\bar{\mathcal{V}}|}(\cdot)$ samples $|\bar{\mathcal{V}}|$ values from the input vector via a certain interpolation method (e.g., bilinear or cubic interpolation).

Implementation of GW Barycenter

- Approximation of $\bar{\boldsymbol{\mu}}$ [Xu, et al., NeurIPS 2019]:

$$\bar{\boldsymbol{\mu}} = \sum_{k=1}^{K} \lambda_k \text{interpolate}_{|\bar{\mathcal{V}}|}(\text{sort}(\boldsymbol{\mu}_k)), \tag{10}$$

  - $\text{sort}(\cdot)$ sorts the elements of the input vector in descending order.
  - $\text{interpolate}_{|\bar{\mathcal{V}}|}(\cdot)$ samples $|\bar{\mathcal{V}}|$ values from the input vector via a certain interpolation method (e.g., bilinear or cubic interpolation).

- Alternating optimization strategy:
  - Obtain $\boldsymbol{T}_k = \arg\min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_k, \bar{\boldsymbol{\mu}})} -\langle \boldsymbol{D}_k \boldsymbol{T} \boldsymbol{B}^\top, \boldsymbol{T} \rangle$ for $k = 1, ..., K$.
  - Update barycenter in a closed form (the first-order optimality condition):

$$\boldsymbol{B}^* = \frac{1}{\bar{\boldsymbol{\mu}} \bar{\boldsymbol{\mu}}^\top} \sum_{k=1}^{K} \lambda_k \boldsymbol{T}_k^\top \boldsymbol{D}_k \boldsymbol{T}_k. \tag{11}$$

[Xu, et al. NeurIPS, 2019] Xu, H., Luo, D., & Carin, L. Scalable Gromov-Wasserstein learning for graph partitioning and matching. NeurIPS, 2019.

Graph matching and partitioning

Apply the Gromov-Wasserstein distance/barycenter as objective functions

Graph Matching via Calculating GW Distance

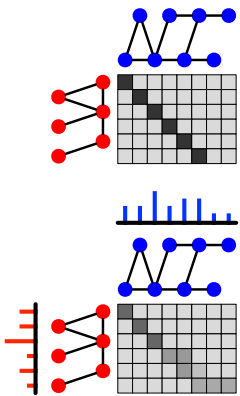
Quadratic assignment problem (QAP):

$$\max_{\boldsymbol{P}\in\mathcal{P}} \langle \boldsymbol{D}_X \boldsymbol{P} \boldsymbol{D}_Y^\top, \boldsymbol{P}\rangle,$$
$$\mathcal{P} = \{\boldsymbol{P} \in \{0,1\}^{|\mathcal{V}_X|\times|\mathcal{V}_Y|} \mid \boldsymbol{P}\mathbf{1} = \mathbf{1}, \boldsymbol{P}^\top\mathbf{1} \le \mathbf{1}\}.$$

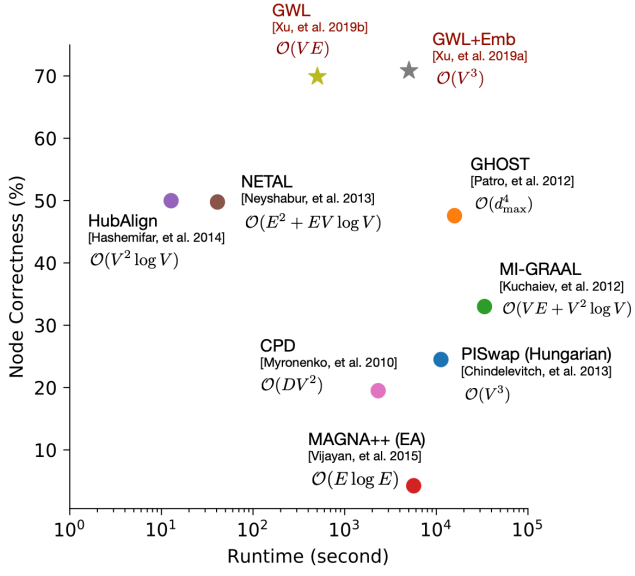
Gromov-Wasserstein distance (with Euclidean relational distance):

$$\min_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu}_X,\boldsymbol{\mu}_Y)} -\langle \boldsymbol{D}_X \boldsymbol{T} \boldsymbol{D}_Y^\top, \boldsymbol{T}\rangle,$$
$$\Pi(\boldsymbol{\mu}_X, \boldsymbol{\mu}_Y) = \{\boldsymbol{T} \ge \mathbf{0} \mid \boldsymbol{T}\mathbf{1} = \boldsymbol{\mu}_X,\ \boldsymbol{T}^\top\mathbf{1} = \boldsymbol{\mu}_Y\}$$

- Conditional distribution: $\boldsymbol{P}^* = \frac{\boldsymbol{T}^*}{\boldsymbol{\mu}_X \mathbf{1}^\top}$.
- For each node $i \in \mathcal{V}_X$, $j^* = \arg\max_j P^*_{ij}$.

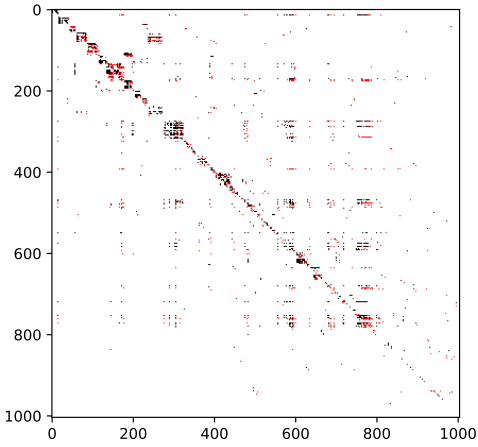


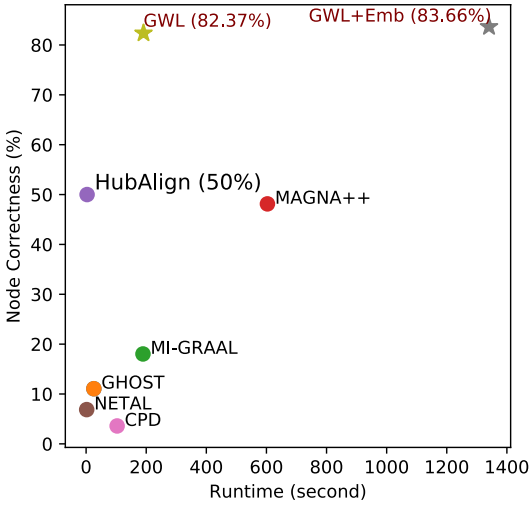
Matching Synthetic Graphs



Real-world PPI Network Alignment

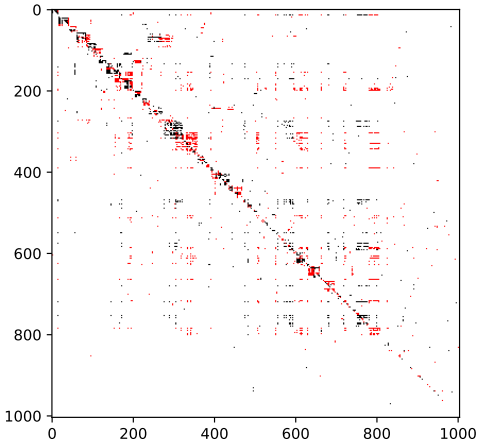
Yeast PPI $\leftrightarrow$ Yeast PPI + 5% LC edges

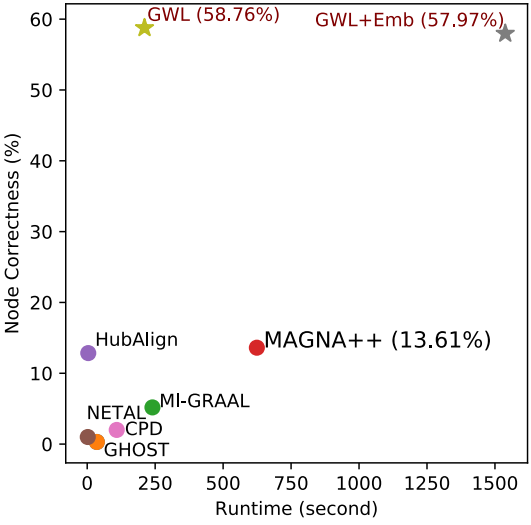




Real-world PPI Network Alignment

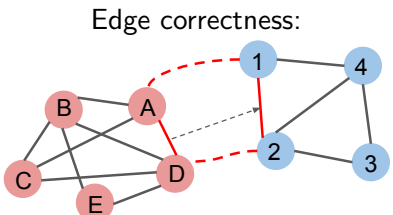
Yeast PPI $\leftrightarrow$ Yeast PPI + **25% LC edges**





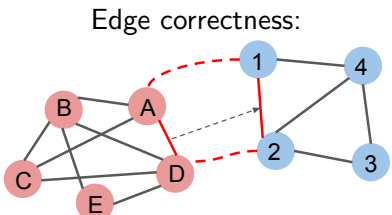
Real-world PPI Network Alignment

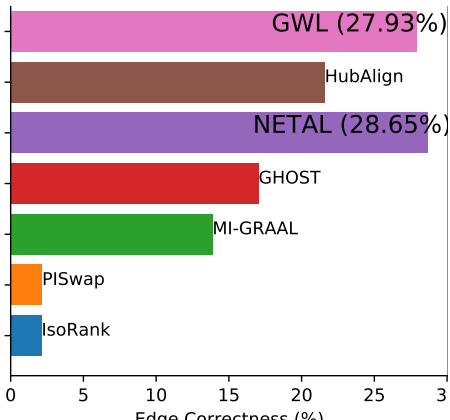
Matching the PPI network of yeast (2,340 proteins) to that of human (9,141 proteins)



Real-world PPI Network Alignment

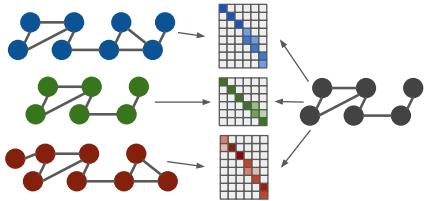
Matching the PPI network of yeast (2,340 proteins) to that of human (9,141 proteins)





Multi-Graph Matching via Calculating GW Barycenters

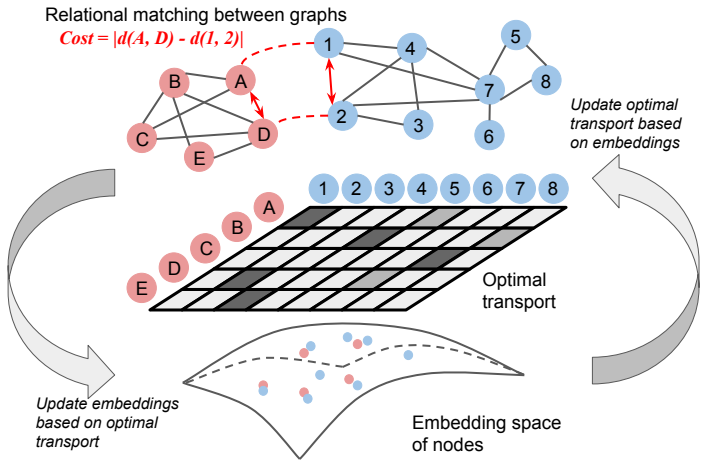
$$\bar{G}, \{\boldsymbol{T}_k^*\}_{k=1}^K = \arg\min \sum_{k=1}^K \lambda_k d_{gw}(G_k, \bar{G}),$$



For each node v in the barycenter graph $\bar{G}$, its correspondences in each graph, $i.e.$, $\{v_k\}_{k=1}^K$, where $v_k = \arg\max_{v'} T_k^*(v', v)$ are matched with each other [Xu, et al. NeurIPS, 2019].

Method	3 graphs		4 graphs		5 graphs		6 graphs	
	NC@1	NC@all	NC@1	NC@all	NC@1	NC@all	NC@1	NC@all
MultiAlign	62.97	45.19	—	—	—	—	—	—
GWL	**63.84**	**46.22**	**68.73**	**39.14**	71.61	31.57	76.49	28.39

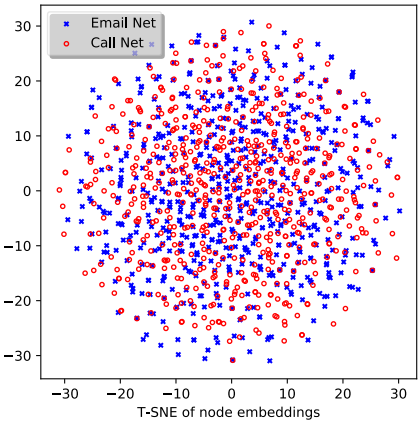
Joint Graph Matching and Node Embedding



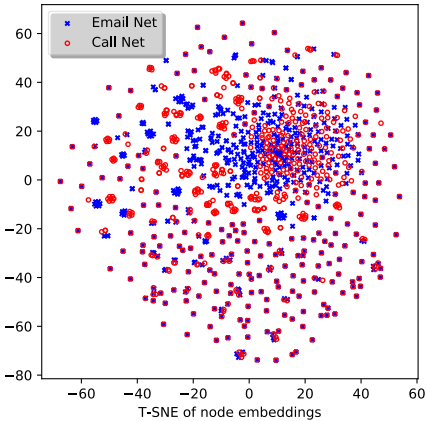
[Xu, et al. ICML 2019] Xu, H., Luo, D., Zha, H., & Carin, L. Gromov-wasserstein learning for graph matching and node embedding. ICML, 2019.

Joint Graph Matching and Node Embedding

Binding accounts across communication networks



(a) Epoch 0

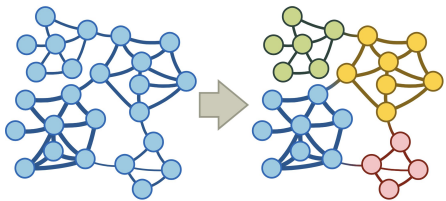


(b) Epoch 30

Graph Partitioning via Calculating GW Distance

Modularity maximization principle

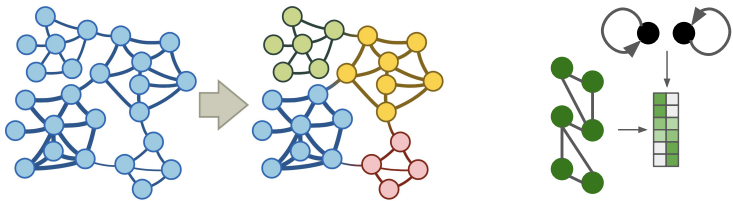
- Dense internal edges + sparse external edges.



Graph Partitioning via Calculating GW Distance

Modularity maximization principle

- Dense internal edges + sparse external edges.

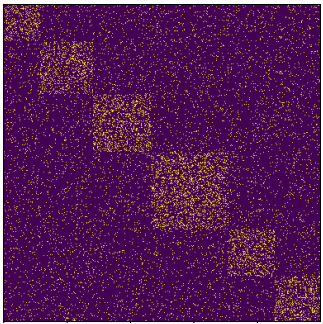


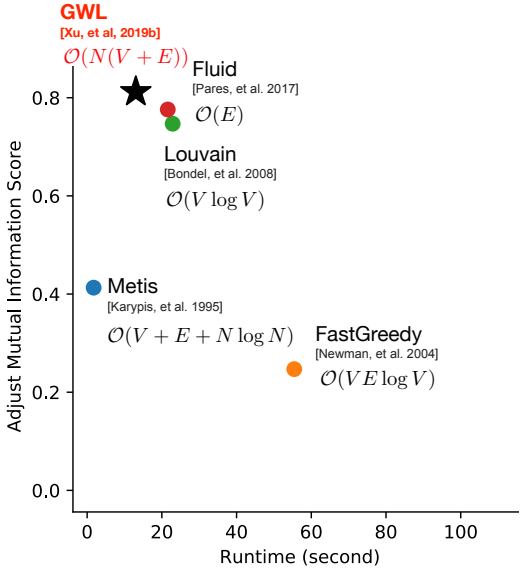
A GWD-based solution [Xu, et al. NeurIPS, 2019]:

- $\boldsymbol{T}^* \in \mathbb{R}^{|\mathcal{V}| \times N} \leftarrow d_{gw}(G, G_{iso})$
- $G_{iso}(\mathcal{V}_{iso}, \boldsymbol{\mu} \in \Delta^{N-1}, \text{diag}(\boldsymbol{\mu}))$
- For each node $i \in G$, its cluster is $j^* = \arg\max_j T^*_{ij}$.

Partitioning Synthetic Graphs

$V = 4,000$
$p_{\text{within}} = 0.2$
$p_{\text{across}} = 0.05$



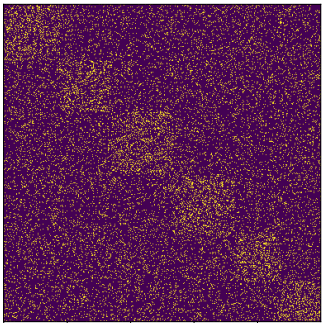


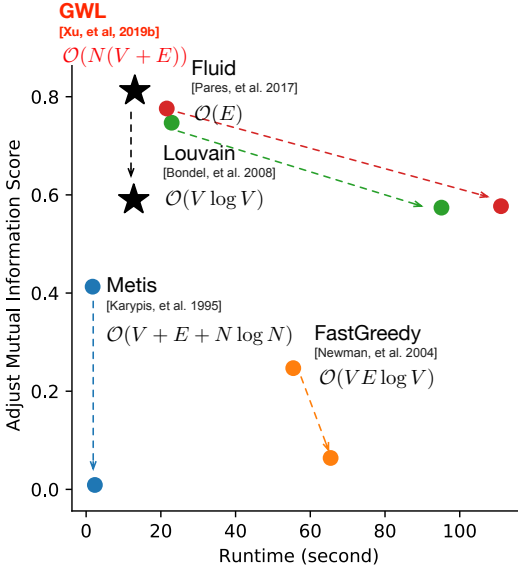
Partitioning Synthetic Graphs

$V = 4,000$

$p_{\text{within}} = 0.2$

$p_{\text{across}} = 0.1$





Graph Partitioning via Calculating (Spectral) GW Distance

- The work in [Chowdhury, et al. AISTATS, 2021] applies t-th order heat kernels, rather than adjacency matrices, when implementing $d_{gw}(G, G_{iso})$, which improves both partitioning accuracy and efficiency.

Table 2: Node Correctness, Mean ± St. Dev. (Time).

Dataset	Adj	Spec^{10}
Proteins	.68 ± .22 (31.9)	**.78** ± .22 (5.1)
Enzymes	.70 ± .18 (8.9)	**.79** ± .17 (1.4)
Reddit	.29 ± .21 (3941.7)	**.50** ± .11 (206.1)
Collab	**.50** ± .27 (4.3)	**.50** ± .27 (5.6)

[Chowdhury, et al. AISTATS, 2021] Chowdhury, S., & Needham, T. Generalized spectral clustering via Gromov-Wasserstein learning. AISTATS, 2021.

Graph Partitioning via Calculating (Spectral) GW Distance

- Assumptions of a graph G:
  - The multiplicity of the smallest positive eigenvalue of the graph Laplacian $\boldsymbol{L}$ is 1.
  - Uniform node probability distribution $\boldsymbol{\mu} = \frac{1}{|\mathcal{V}|}\mathbf{1}_{|\mathcal{V}|}$.

Graph Partitioning via Calculating (Spectral) GW Distance

- Assumptions of a graph G:
  - The multiplicity of the smallest positive eigenvalue of the graph Laplacian $\boldsymbol{L}$ is 1.
  - Uniform node probability distribution $\boldsymbol{\mu} = \frac{1}{|\mathcal{V}|}\mathbf{1}_{|\mathcal{V}|}$.
- **Fiedler vector of G:** The eigenvector corresponding to the smallest positive eigenvalue of graph Laplacian $\boldsymbol{L}$.

Graph Partitioning via Calculating (Spectral) GW Distance

- Assumptions of a graph G:
  - The multiplicity of the smallest positive eigenvalue of the graph Laplacian $\boldsymbol{L}$ is 1.
  - Uniform node probability distribution $\boldsymbol{\mu} = \frac{1}{|\mathcal{V}|}\mathbf{1}_{|\mathcal{V}|}$.
- **Fiedler vector of G:** The eigenvector corresponding to the smallest positive eigenvalue of graph Laplacian $\boldsymbol{L}$.
- **Fiedler partition of G:** The nodes of G are partitioned according to the sign of their entry in the Fiedler vector.

Graph Partitioning via Calculating (Spectral) GW Distance

- Assumptions of a graph G:
  - The multiplicity of the smallest positive eigenvalue of the graph Laplacian $\boldsymbol{L}$ is 1.
  - Uniform node probability distribution $\boldsymbol{\mu} = \frac{1}{|\mathcal{V}|}\mathbf{1}_{|\mathcal{V}|}$.
- **Fiedler vector of G:** The eigenvector corresponding to the smallest positive eigenvalue of graph Laplacian $\boldsymbol{L}$.
- **Fiedler partition of G:** The nodes of G are partitioned according to the sign of their entry in the Fiedler vector.
- Connect optimal transport with classic spectral clustering:
  - **Theorem 3** in [Chowdhury, et al. AISTATS, 2021]: *The 2-way partition of such G derived from the spectral version of $d_{gw}(G, G_{iso})$ agrees with the* **Fiedler partitioning** *when t is sufficiently large.*

Summary

- The GW distance works as a metric of mm-spaces, which can be used as a (pseudo) metric of point clouds / graphs.
- The GW barycenter provides a way to obtain the average of point clouds / graphs under GW distance.
- The optimal transport associated with the GW distance achieves a soft assignment of graph nodes.
- Based on GW distance, we actually achieves a unified algorithmic framework for graph matching, partitioning, and node embedding.
- The GW-based framework has the potential to many real-world problems, e.g., PPI network alignment, community detection.

Summary

- The GW distance works as a metric of mm-spaces, which can be used as a (pseudo) metric of point clouds / graphs.
- The GW barycenter provides a way to obtain the average of point clouds / graphs under GW distance.
- The optimal transport associated with the GW distance achieves a soft assignment of graph nodes.
- Based on GW distance, we actually achieves a unified algorithmic framework for graph matching, partitioning, and node embedding.
- The GW-based framework has the potential to many real-world problems, e.g., PPI network alignment, community detection.

Next,

- **How to compute/approximate GW distance efficiently?**
- **How to extend its applications based on its variants?**

5-min break for Q & A

Outline

The Optimization Problem Corresponding to GW Distance

As aforementioned, $d_{gw}(G_X, G_Y)$ leads to a **non-convex non-smooth** optimization problem:

$$\min_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu}_X,\boldsymbol{\mu}_Y)} -\langle \boldsymbol{D}_X \boldsymbol{T} \boldsymbol{D}_Y^\top,\ \boldsymbol{T}\rangle \tag{12}$$

- Iterative optimization is required.
- Low complexity, guaranteed convergence, stable approximation.

Typical Method: Entropic Regularization + Projected Gradient Descent

Sinkhorn Distance (Entropic OT Problem) [Cuturi, NeurIPS 2013]

$$\min_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu},\boldsymbol{\eta})} \langle \boldsymbol{D}, \boldsymbol{T}\rangle - \gamma \underbrace{\mathsf{H}(\boldsymbol{T})}_{\text{Entropy}},$$

$$\mathsf{H}(\boldsymbol{T}) = -\langle \log \boldsymbol{T} - \mathbf{1}, \boldsymbol{T}\rangle.$$

Typical Method: Entropic Regularization + Projected Gradient Descent

Sinkhorn Distance (Entropic OT Problem) [Cuturi, NeurIPS 2013]

$$\begin{aligned} &\min_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu},\boldsymbol{\eta})} \langle \boldsymbol{D}, \boldsymbol{T}\rangle - \gamma \underbrace{\mathsf{H}(\boldsymbol{T})}_{\text{Entropy}}, \\ &\mathsf{H}(\boldsymbol{T}) = -\langle \log \boldsymbol{T} - \mathbf{1}, \boldsymbol{T}\rangle. \end{aligned} \tag{13}$$

Entropic Gromov-Wasserstein Distance [Peyré, et al., ICML 2016]

$$\min_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu}_X,\boldsymbol{\mu}_Y)} \underbrace{-\langle \boldsymbol{D}_X \boldsymbol{T} \boldsymbol{D}_Y^\top, \boldsymbol{T}\rangle}_{\text{GW term}} - \gamma \underbrace{\mathsf{H}(\boldsymbol{T})}_{\text{Entropy}}. \tag{14}$$

[Cuturi, NeurIPS 2013] Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. NeurIPS, 2013.

Typical Method: Entropic Regularization + Projected Gradient Descent

In each iteration

- Fix one optimal transport and obtain an entropic OT problem:

$$\boldsymbol{T}^{(m+1)} = \arg\min_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu}_X,\boldsymbol{\mu}_Y)} -\langle \underbrace{\boldsymbol{D}_X \boldsymbol{T}^{(m)} \boldsymbol{D}_Y^\top}_{\text{Constant}}, \boldsymbol{T}\rangle - \gamma \mathsf{H}(\boldsymbol{T}) \tag{15}$$

Typical Method: Entropic Regularization + Projected Gradient Descent

In each iteration

- Fix one optimal transport and obtain an entropic OT problem:

$$\boldsymbol{T}^{(m+1)} = \arg\min_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu}_X,\boldsymbol{\mu}_Y)} -\langle \underbrace{\boldsymbol{D}_X\boldsymbol{T}^{(m)}\boldsymbol{D}_Y^\top}_{\text{Constant}}, \boldsymbol{T}\rangle - \gamma \mathsf{H}(\boldsymbol{T}) \tag{15}$$

- Sinkhorn-Knopp algorithm: Define $\boldsymbol{\Phi} = \exp(-\frac{\boldsymbol{D}_X\boldsymbol{T}^{(m)}\boldsymbol{D}_Y^\top}{\gamma})$, $\boldsymbol{a} = \boldsymbol{\mu}_X$, and
  1. Repeat $\boldsymbol{b} = \frac{\boldsymbol{\mu}_Y}{\boldsymbol{\Phi}^\top \boldsymbol{a}}$ and $\boldsymbol{a} = \frac{\boldsymbol{\mu}_X}{\boldsymbol{\Phi}\boldsymbol{b}}$ until convergence
  2. $\boldsymbol{T}^{(m+1)} = \mathsf{diag}(\boldsymbol{a})\boldsymbol{\Phi}\mathsf{diag}(\boldsymbol{b})$.

Typical Method: Proximal Gradient

Motivation:

- The entropic regularizer might be unnecessary for the GW distance.

Typical Method: Proximal Gradient

Motivation:

- The entropic regularizer might be unnecessary for the GW distance.

Principle:

- In each iteration, consider the penalty between the optimal transport and its previous approximation [Xu, et al., ICML 2019]

$$\min_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu}_X,\boldsymbol{\mu}_Y)} -\langle \boldsymbol{D}_X \boldsymbol{T}^{(m)} \boldsymbol{D}_Y^\top, \boldsymbol{T}\rangle + \gamma \underbrace{\mathsf{KL}(\boldsymbol{T}\|\boldsymbol{T}^{(m)})}_{\text{Proximal term}}$$

Typical Method: Proximal Gradient

Motivation:

- The entropic regularizer might be unnecessary for the GW distance.

Principle:

- In each iteration, consider the penalty between the optimal transport and its previous approximation [Xu, et al., ICML 2019]

$$\begin{aligned} &\min_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu}_X,\boldsymbol{\mu}_Y)} -\langle \boldsymbol{D}_X \boldsymbol{T}^{(m)} \boldsymbol{D}_Y^\top, \boldsymbol{T}\rangle + \gamma \underbrace{\mathsf{KL}(\boldsymbol{T}\|\boldsymbol{T}^{(m)})}_{\text{Proximal term}} \\ \Rightarrow &\min_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu}_X,\boldsymbol{\mu}_Y)} -\langle \underbrace{\boldsymbol{D}_X \boldsymbol{T}^{(m)} \boldsymbol{D}_Y^\top + \gamma \log \boldsymbol{T}^{(m)}}_{\text{Constant}}, \boldsymbol{T}\rangle - \gamma \mathsf{H}(\boldsymbol{T}). \end{aligned} \tag{16}$$

Typical Method: Proximal Gradient

Motivation:

- The entropic regularizer might be unnecessary for the GW distance.

Principle:

- In each iteration, consider the penalty between the optimal transport and its previous approximation [Xu, et al., ICML 2019]

$$\begin{aligned} &\min_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu}_X,\boldsymbol{\mu}_Y)} -\langle \boldsymbol{D}_X \boldsymbol{T}^{(m)} \boldsymbol{D}_Y^\top, \boldsymbol{T}\rangle + \gamma \underbrace{\mathsf{KL}(\boldsymbol{T}\|\boldsymbol{T}^{(m)})}_{\text{Proximal term}} \\ \Rightarrow &\min_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu}_X,\boldsymbol{\mu}_Y)} -\langle \underbrace{\boldsymbol{D}_X \boldsymbol{T}^{(m)} \boldsymbol{D}_Y^\top + \gamma \log \boldsymbol{T}^{(m)}}_{\text{Constant}}, \boldsymbol{T}\rangle - \gamma \mathsf{H}(\boldsymbol{T}). \end{aligned} \tag{16}$$

- Similar to the proximal point method of OT problem in [Xie, et al., UAI 2020]

[Xie, et al., UAI 2020] Xie, Y., Wang, X., Wang, R., & Zha, H. A fast proximal point method for computing exact Wasserstein distance. UAI 2020.

Typical Method: Proximal Gradient

$\min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_X, \boldsymbol{\mu}_Y)} -\langle \boldsymbol{D}_X \boldsymbol{T} \boldsymbol{D}_Y^\top, \boldsymbol{T} \rangle$. (Without any modification or relaxation)

Typical Method: Proximal Gradient

$\min_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu}_X,\boldsymbol{\mu}_Y)} -\langle \boldsymbol{D}_X \boldsymbol{T} \boldsymbol{D}_Y^\top, \boldsymbol{T}\rangle$. (Without any modification or relaxation)

- Initialize $\boldsymbol{T}^{(0)} = \boldsymbol{\mu}_X \boldsymbol{\mu}_Y^\top$ and in each iteration

$$\min_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu}_X,\boldsymbol{\mu}_Y)} -\langle \boldsymbol{D}_X \boldsymbol{T}^{(m)} \boldsymbol{D}_Y^\top + \gamma \log \boldsymbol{T}^{(m)}, \boldsymbol{T}\rangle - \gamma \mathsf{H}(\boldsymbol{T}). \tag{17}$$

Typical Method: Proximal Gradient

$\min_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu}_X,\boldsymbol{\mu}_Y)} -\langle \boldsymbol{D}_X\boldsymbol{T}\boldsymbol{D}_Y^\top, \boldsymbol{T}\rangle$. (Without any modification or relaxation)

- Initialize $\boldsymbol{T}^{(0)} = \boldsymbol{\mu}_X\boldsymbol{\mu}_Y^\top$ and in each iteration

$$\min_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu}_X,\boldsymbol{\mu}_Y)} -\langle \boldsymbol{D}_X\boldsymbol{T}^{(m)}\boldsymbol{D}_Y^\top + \gamma \log \boldsymbol{T}^{(m)}, \boldsymbol{T}\rangle - \gamma \mathsf{H}(\boldsymbol{T}). \tag{17}$$

1. Compute $\boldsymbol{\Phi} = \exp(-\frac{\boldsymbol{D}_X\boldsymbol{T}^{(m)}\boldsymbol{D}_Y^\top}{\gamma}) \odot \boldsymbol{T}^{(m)}$, $\boldsymbol{a} = \boldsymbol{\mu}_X$.
2. Sinkhorn-Knopp Iterations: Repeat $\boldsymbol{b} = \frac{\boldsymbol{\mu}_Y}{\boldsymbol{\Phi}^\top \boldsymbol{a}}$ and $\boldsymbol{a} = \frac{\boldsymbol{\mu}_X}{\boldsymbol{\Phi}\boldsymbol{b}}$ until convergence.
3. $\boldsymbol{T}^{(m+1)} = \mathsf{diag}(\boldsymbol{a})\boldsymbol{\Phi}\mathsf{diag}(\boldsymbol{b})$.

Typical Method: Conditional Gradient (CG)

ProxGrad is a special case of the conditional gradient (CG) used in [Titouan, et al. ICML 2019]

- Initialization: $\boldsymbol{T}^0 = \boldsymbol{\mu}_X \boldsymbol{\mu}_Y^\top$.
- In each iteration:
  1. Compute the gradient w.r.t. $\boldsymbol{T}^{(m)}$: $\nabla_{\boldsymbol{T}^{(m)}} L = -\boldsymbol{D}_X \boldsymbol{T}^{(m)} \boldsymbol{D}_Y^\top$.
  2. Solve an OT problem: $\tilde{\boldsymbol{T}}^{(m+1)} = \arg\min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_X, \boldsymbol{\mu}_Y)} \langle \nabla_{\boldsymbol{T}^{(m)}} L, \boldsymbol{T} \rangle$.
  3. Line-search to check whether the objective function decreases given $\tilde{\boldsymbol{T}}^{(m+1)}$ and determine a momentum $\tau \in [0, 1]$.
  4. $\boldsymbol{T}^{(m+1)} = (1 - \tau)\boldsymbol{T}^{(m)} + \tau \tilde{\boldsymbol{T}}^{(m+1)}$.

Typical Method: Conditional Gradient (CG)

ProxGrad is a special case of the conditional gradient (CG) used in [Titouan, et al. ICML 2019]

- Initialization: $\boldsymbol{T}^0 = \boldsymbol{\mu}_X \boldsymbol{\mu}_Y^\top$.
- In each iteration:
  1. Compute the gradient w.r.t. $\boldsymbol{T}^{(m)}$: $\nabla_{\boldsymbol{T}^{(m)}} L = -\boldsymbol{D}_X \boldsymbol{T}^{(m)} \boldsymbol{D}_Y^\top$.
  2. Solve an OT problem: $\tilde{\boldsymbol{T}}^{(m+1)} = \arg\min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_X, \boldsymbol{\mu}_Y)} \langle \nabla_{\boldsymbol{T}^{(m)}} L, \boldsymbol{T} \rangle$.
  3. Line-search to check whether the objective function decreases given $\tilde{\boldsymbol{T}}^{(m+1)}$ and determine a momentum $\tau \in [0, 1]$.
  4. $\boldsymbol{T}^{(m+1)} = (1 - \tau)\boldsymbol{T}^{(m)} + \tau \tilde{\boldsymbol{T}}^{(m+1)}$.

- The Step 1+2 corresponds to solve Eq. (5) (The sub-problem of ProxGrad).
- Without line-search, the ProxGrad always sets $\tau = 1$.

[Titouan, et al. ICML 2019] Titouan, V., Courty, N., Tavenard, R., & Flamary, R. Optimal transport for structured data with application on graphs. ICML, 2019.

Typical Method: Bregman ADMM

Motivation:

- Linear Programming is expensive, while Sinkhorn-Knopp algorithm often suffers from numerical instability issue.
- Logarithmic stabilization [Chizat, et al., 2018] is not suitable to GW distance.

Typical Method: Bregman ADMM

Motivation:

- Linear Programming is expensive, while Sinkhorn-Knopp algorithm often suffers from numerical instability issue.
- Logarithmic stabilization [Chizat, et al., 2018] is not suitable to GW distance.

Principle:

- Bregman ADMM for OT problem [Wang, et al. NeurIPS 2014]:

$$\min_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu},\cdot),\boldsymbol{S}\in\Pi(\cdot,\boldsymbol{\eta}),\boldsymbol{T}=\boldsymbol{S}} \langle \boldsymbol{D}, \boldsymbol{T}\rangle$$

Typical Method: Bregman ADMM

Motivation:

- Linear Programming is expensive, while Sinkhorn-Knopp algorithm often suffers from numerical instability issue.
- Logarithmic stabilization [Chizat, et al., 2018] is not suitable to GW distance.

Principle:

- Bregman ADMM for OT problem [Wang, et al. NeurIPS 2014]:

$$\begin{aligned} &\min_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu},\cdot),\boldsymbol{S}\in\Pi(\cdot,\boldsymbol{\eta}),\boldsymbol{T}=\boldsymbol{S}} \langle \boldsymbol{D},\boldsymbol{T}\rangle \\ \Rightarrow &\min_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu},\cdot),\boldsymbol{S}\in\Pi(\cdot,\boldsymbol{\eta}),\boldsymbol{Z}} \langle \boldsymbol{D},\boldsymbol{T}\rangle + \underbrace{\langle \boldsymbol{Z},\boldsymbol{T}-\boldsymbol{S}\rangle + \overbrace{\gamma \mathsf{B}_\phi(\boldsymbol{T},\boldsymbol{S})}^{\text{Bregman Div.}}}_{\text{Augmented Lagrangian}} \end{aligned} \tag{18}$$

[Chizat, et al., 2018] Chizat, L., Peyré, G., Schmitzer, B., & Vialard, F. X. Scaling algorithms for unbalanced optimal transport problems. Mathematics of Computation, 2018.
[Wang, et al. NeurIPS 2014] Wang, H., & Banerjee, A. Bregman alternating direction method of multipliers. NeurIPS 2014.

Typical Method: Bregman ADMM

Bregman Divergence: Given a differentiable and strictly convex function ϕ,

$$B_\phi(x, y) = \phi(x) - \phi(y) - \langle \nabla\phi(y), x - y \rangle. \tag{19}$$

Typical Method: Bregman ADMM

Bregman Divergence: Given a differentiable and strictly convex function ϕ,

$$B_\phi(x, y) = \phi(x) - \phi(y) - \langle \nabla\phi(y), x - y \rangle. \tag{19}$$

Commonly-used Bregman divergence:

- $\phi(x) = \frac{1}{2}x^2$: Euclidean distance $B_\phi(x, y) = \frac{1}{2}\|x - y\|^2$.
- $\phi(x) = x \log x - x$: KL-divergence $B_\phi(x, y) = \mathrm{KL}(x\|y) = x \log \frac{x}{y} - x + y\rangle$.

Typical Method: Bregman ADMM

Bregman Divergence: Given a differentiable and strictly convex function ϕ,

$$B_\phi(x, y) = \phi(x) - \phi(y) - \langle \nabla\phi(y), x - y\rangle. \tag{19}$$

Commonly-used Bregman divergence:

- $\phi(x) = \frac{1}{2}x^2$: Euclidean distance $B_\phi(x, y) = \frac{1}{2}\|x - y\|^2$.
- $\phi(x) = x \log x - x$: KL-divergence $B_\phi(x, y) = \mathsf{KL}(x\|y) = x \log \frac{x}{y} - x + y\rangle$.

Apply the Bregman ADMM to compute GW distance [Xu, AAAI 2020]:

$$\begin{aligned} &\min_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu}_X,\cdot),\boldsymbol{S}\in\Pi(\cdot,\boldsymbol{\mu}_Y),\boldsymbol{T}=\boldsymbol{S}} -\langle \boldsymbol{D}_X \boldsymbol{S} \boldsymbol{D}_Y^\top, \boldsymbol{T}\rangle \\ \Rightarrow &\min_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu}_X,\cdot),\boldsymbol{S}\in\Pi(\cdot,\boldsymbol{\mu}_Y),\boldsymbol{Z}} -\langle \boldsymbol{D}_X \boldsymbol{S} \boldsymbol{D}_Y^\top, \boldsymbol{T}\rangle + \langle \boldsymbol{Z}, \boldsymbol{T} - \boldsymbol{S}\rangle + \gamma \mathsf{B}(\boldsymbol{T}, \boldsymbol{S}) \end{aligned} \tag{20}$$

[Xu, AAAI 2020] Xu, H. Gromov-Wasserstein factorization models for graph clustering. AAAI 2020.

Typical Method: Bregman ADMM (B-ADMM)

Initialize $\boldsymbol{Z}^{(0)} = \mathbf{0}$, $\boldsymbol{T}^{(0)} = \boldsymbol{S}^{(0)} = \boldsymbol{\mu}_X \boldsymbol{\mu}_Y^\top$:

Typical Method: Bregman ADMM (B-ADMM)

Initialize $\boldsymbol{Z}^{(0)} = \mathbf{0}$, $\boldsymbol{T}^{(0)} = \boldsymbol{S}^{(0)} = \boldsymbol{\mu}_X \boldsymbol{\mu}_Y^\top$:

$$\boldsymbol{T}^{(m+1)} = \arg\min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_X, \cdot)} -\langle \boldsymbol{D}_X \boldsymbol{S}^{(m)} \boldsymbol{D}_Y^\top, \boldsymbol{T} \rangle + \langle \boldsymbol{Z}^{(m)}, \boldsymbol{T} \rangle + \gamma \mathsf{KL}(\boldsymbol{T} \| \boldsymbol{S}^{(m)})$$

$$\xrightarrow{\text{First-order optimality}} \boldsymbol{T}^{(m+1)} = \exp\left(\frac{\gamma \log \boldsymbol{S}^{(m)} + \boldsymbol{D}_X \boldsymbol{S}^{(m)} \boldsymbol{D}_Y^\top - \boldsymbol{Z}^{(m)}}{\gamma} \right) \tag{21}$$

$$\xrightarrow{\text{Project to } \Pi(\boldsymbol{\mu}^{(k)}, \cdot)} \boldsymbol{T}^{(m+1)} = \mathsf{diag}(\boldsymbol{\mu}_X) \sigma_{row} \left(\frac{\gamma \log \boldsymbol{S}^{(m)} + \boldsymbol{D}_X \boldsymbol{S}^{(m)} \boldsymbol{D}_Y^\top - \boldsymbol{Z}^{(m)}}{\gamma} \right)$$

Typical Method: Bregman ADMM (B-ADMM)

Initialize $\boldsymbol{Z}^{(0)} = \mathbf{0}$, $\boldsymbol{T}^{(0)} = \boldsymbol{S}^{(0)} = \boldsymbol{\mu}_X \boldsymbol{\mu}_Y^\top$:

$$\boldsymbol{T}^{(m+1)} = \arg\min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_X, \cdot)} -\langle \boldsymbol{D}_X \boldsymbol{S}^{(m)} \boldsymbol{D}_Y^\top, \boldsymbol{T} \rangle + \langle \boldsymbol{Z}^{(m)}, \boldsymbol{T} \rangle + \gamma \mathsf{KL}(\boldsymbol{T} \| \boldsymbol{S}^{(m)})$$

$$\xrightarrow{\text{First-order optimality}} \boldsymbol{T}^{(m+1)} = \exp\left(\frac{\gamma \log \boldsymbol{S}^{(m)} + \boldsymbol{D}_X \boldsymbol{S}^{(m)} \boldsymbol{D}_Y^\top - \boldsymbol{Z}^{(m)}}{\gamma} \right) \tag{21}$$

$$\xrightarrow{\text{Project to } \Pi(\boldsymbol{\mu}^{(k)}, \cdot)} \boldsymbol{T}^{(m+1)} = \mathsf{diag}(\boldsymbol{\mu}_X) \sigma_{row} \left(\frac{\gamma \log \boldsymbol{S}^{(m)} + \boldsymbol{D}_X \boldsymbol{S}^{(m)} \boldsymbol{D}_Y^\top - \boldsymbol{Z}^{(m)}}{\gamma} \right)$$

$$\boldsymbol{S}^{(m+1)} = \arg\min_{\boldsymbol{S} \in \Pi(\cdot, \boldsymbol{\mu}_Y)} -\langle \boldsymbol{D}_X^\top \boldsymbol{T}^{(m+1)} \boldsymbol{D}_Y, \boldsymbol{S} \rangle - \langle \boldsymbol{Z}^{(m)}, \boldsymbol{S} \rangle + \gamma \mathsf{KL}(\boldsymbol{S} \| \boldsymbol{T}^{(m+1)})$$

$$= \sigma_{col} \left(\frac{\gamma \log \boldsymbol{T}^{(m+1)} + \boldsymbol{D}_X^\top \boldsymbol{T}^{(m+1)} \boldsymbol{D}_Y + \boldsymbol{Z}^{(m)}}{\gamma} \right) \mathsf{diag}(\boldsymbol{\mu}_Y) \tag{22}$$

Typical Method: Bregman ADMM (B-ADMM)

Initialize $\boldsymbol{Z}^{(0)} = \mathbf{0}$, $\boldsymbol{T}^{(0)} = \boldsymbol{S}^{(0)} = \boldsymbol{\mu}_X \boldsymbol{\mu}_Y^\top$:

$$\boldsymbol{T}^{(m+1)} = \arg\min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_X, \cdot)} -\langle \boldsymbol{D}_X \boldsymbol{S}^{(m)} \boldsymbol{D}_Y^\top, \boldsymbol{T} \rangle + \langle \boldsymbol{Z}^{(m)}, \boldsymbol{T} \rangle + \gamma \mathsf{KL}(\boldsymbol{T} \| \boldsymbol{S}^{(m)})$$

$$\xrightarrow{\text{First-order optimality}} \boldsymbol{T}^{(m+1)} = \exp\left(\frac{\gamma \log \boldsymbol{S}^{(m)} + \boldsymbol{D}_X \boldsymbol{S}^{(m)} \boldsymbol{D}_Y^\top - \boldsymbol{Z}^{(m)}}{\gamma} \right) \tag{21}$$

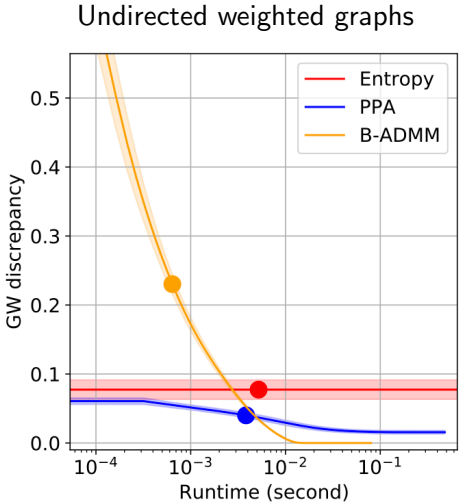
$$\xrightarrow{\text{Project to } \Pi(\boldsymbol{\mu}^{(k)}, \cdot)} \boldsymbol{T}^{(m+1)} = \mathsf{diag}(\boldsymbol{\mu}_X) \sigma_{row} \left(\frac{\gamma \log \boldsymbol{S}^{(m)} + \boldsymbol{D}_X \boldsymbol{S}^{(m)} \boldsymbol{D}_Y^\top - \boldsymbol{Z}^{(m)}}{\gamma} \right)$$

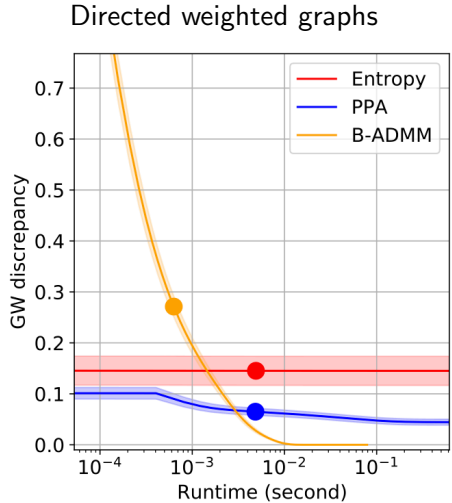
$$\boldsymbol{S}^{(m+1)} = \arg\min_{\boldsymbol{S} \in \Pi(\cdot, \boldsymbol{\mu}_Y)} -\langle \boldsymbol{D}_X^\top \boldsymbol{T}^{(m+1)} \boldsymbol{D}_Y, \boldsymbol{S} \rangle - \langle \boldsymbol{Z}^{(m)}, \boldsymbol{S} \rangle + \gamma \mathsf{KL}(\boldsymbol{S} \| \boldsymbol{T}^{(m+1)})$$

$$= \sigma_{col} \left(\frac{\gamma \log \boldsymbol{T}^{(m+1)} + \boldsymbol{D}_X^\top \boldsymbol{T}^{(m+1)} \boldsymbol{D}_Y + \boldsymbol{Z}^{(m)}}{\gamma} \right) \mathsf{diag}(\boldsymbol{\mu}_Y) \tag{22}$$

$$\boldsymbol{Z}^{(m+1)} = \boldsymbol{Z}^{(m)} + \gamma(\boldsymbol{T}^{(m+1)} - \boldsymbol{S}^{(m+1)}). \tag{23}$$

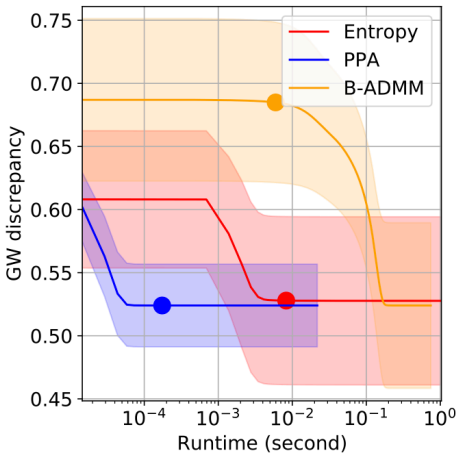
Comparisons on Weighted Graphs



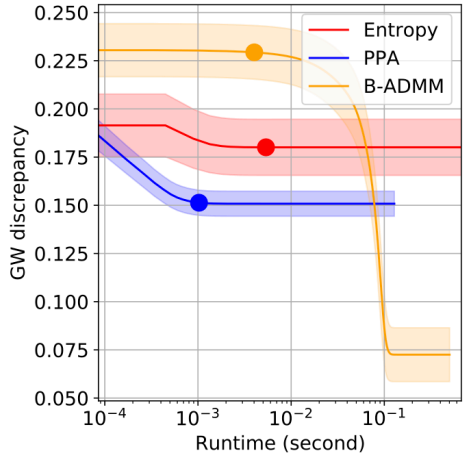


Comparisons on Unweighted Graphs

Undirected unweighted graph



Directed unweighted graph



Convergence and Complexity

Convergence

- $\lim_{m\to\infty} \boldsymbol{T}^{(m)}$ is a stationary point [Xu, et al., ICML 2019].
- Convergence: Both Sinkhorn-based and Bregman-ADMM methods own $\mathcal{O}(1/\epsilon^2)$.

Convergence and Complexity

Convergence

- $\lim_{m\to\infty} \boldsymbol{T}^{(m)}$ is a stationary point [Xu, et al., ICML 2019].
- Convergence: Both Sinkhorn-based and Bregman-ADMM methods own $\mathcal{O}(1/\epsilon^2)$.

Computational complexity per iteration: $\boldsymbol{D}_X \boldsymbol{T} \boldsymbol{D}_Y^\top$

- $\mathcal{O}(V^3)$ for dense distance/kernel matrix $(V \times V)$

Convergence and Complexity

Convergence

- $\lim_{m\to\infty} \boldsymbol{T}^{(m)}$ is a stationary point [Xu, et al., ICML 2019].
- Convergence: Both Sinkhorn-based and Bregman-ADMM methods own $\mathcal{O}(1/\epsilon^2)$.

Computational complexity per iteration: $\boldsymbol{D}_X \boldsymbol{T} \boldsymbol{D}_Y^\top$

- $\mathcal{O}(V^3)$ for dense distance/kernel matrix ($V \times V$)
- $\mathcal{O}(VE)$ for the graph with V nodes and E edges.

Convergence and Complexity

Convergence

- $\lim_{m\to\infty} \boldsymbol{T}^{(m)}$ is a stationary point [Xu, et al., ICML 2019].
- Convergence: Both Sinkhorn-based and Bregman-ADMM methods own $\mathcal{O}(1/\epsilon^2)$.

Computational complexity per iteration: $\boldsymbol{D}_X \boldsymbol{T} \boldsymbol{D}_Y^\top$

- $\mathcal{O}(V^3)$ for dense distance/kernel matrix ($V \times V$)
- $\mathcal{O}(VE)$ for the graph with V nodes and E edges.
- **Can we further accelerate the process?**

Acceleration Strategy 1: Divide and Conquer

Motivation:

- When $V = |\mathcal{V}_X| \gg |\mathcal{V}_Y| = N$ (graph partitioning), $\mathcal{O}(N(E+V))$.

Acceleration Strategy 1: Divide and Conquer

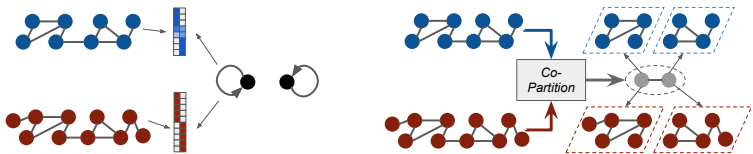
Motivation:

- When $V = |\mathcal{V}_X| \gg |\mathcal{V}_Y| = N$ (graph partitioning), $\mathcal{O}(N(E+V))$.

Co-partition two graphs:

$$\bar{\boldsymbol{D}}, \boldsymbol{T}_X, \boldsymbol{T}_Y = \arg\min \frac{|\mathcal{V}_X|}{|\mathcal{V}_X| + |\mathcal{V}_Y|} d_{gw}(B, G_X) + \frac{|\mathcal{V}_Y|}{|\mathcal{V}_X| + |\mathcal{V}_Y|} d_{gw}(B, G_Y)$$

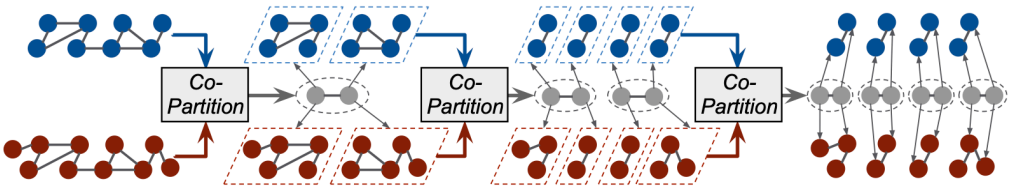
Initialize the barycenter graph by a disconnected graph.



Computational complexity: $\mathcal{O}(2(V+E))$

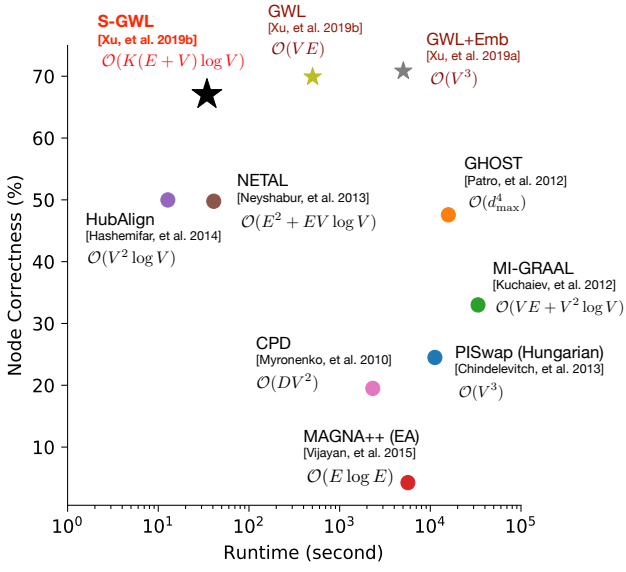
Acceleration Strategy 1: Divide and Conquer

A "Divide and Conquer" strategy based on recursive co-partitioning



Computational complexity: $\mathcal{O}((E+V)\log V)$ (Proposition 3.1 in [Xu, et al., NeurIPS 2019])

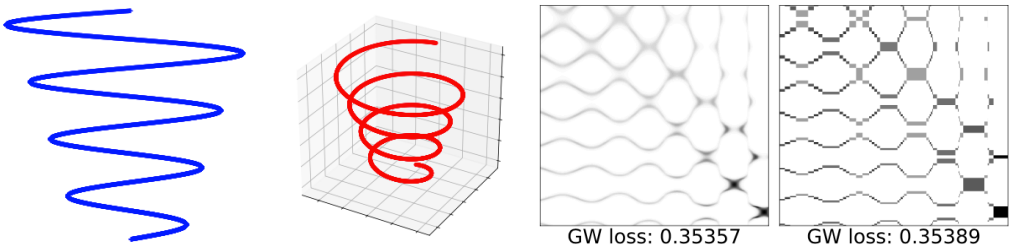
Acceleration Strategy 1: Divide and Conquer



Acceleration Strategy 2: Low-rank Structured GW

Motivation:

- The optimal transport matrix often has a low-rank approximation.



An illustration of low-rank structured GW shown in [Scetbon, et al., 2021].

[Scetbon, et al., 2021] Scetbon, M., Peyré, G., & Cuturi, M. Linear-time Gromov-Wasserstein distances using low rank couplings and costs. arXiv 2021.

Acceleration Strategy 2: Low-rank Structured GW

Principle:

- A low-rank factorization of proposed optimal transport matrix

$$\boldsymbol{T} = \boldsymbol{Q}\mathrm{diag}(1/\boldsymbol{g})\boldsymbol{R}^{\top}, \text{ where}$$
$$\underbrace{\boldsymbol{Q} \in \Pi(\boldsymbol{\mu}_X, \boldsymbol{g}), \quad \boldsymbol{R} \in \Pi(\boldsymbol{\mu}_Y, \boldsymbol{g}), \quad \boldsymbol{g} \in \Delta^{r-1}}_{\mathcal{C}(\boldsymbol{\mu}_X, \boldsymbol{\mu}_Y, r)}, \quad r \ll M, N. \tag{24}$$

Acceleration Strategy 2: Low-rank Structured GW

Principle:

- A low-rank factorization of proposed optimal transport matrix

$$\boldsymbol{T} = \boldsymbol{Q}\text{diag}(1/\boldsymbol{g})\boldsymbol{R}^\top, \text{ where}$$
$$\underbrace{\boldsymbol{Q} \in \Pi(\boldsymbol{\mu}_X, \boldsymbol{g}), \quad \boldsymbol{R} \in \Pi(\boldsymbol{\mu}_Y, \boldsymbol{g}), \quad \boldsymbol{g} \in \Delta^{r-1}}_{\mathcal{C}(\boldsymbol{\mu}_X, \boldsymbol{\mu}_Y, r)}, \quad r \ll M, N. \tag{24}$$

- In each iteration, solving the following entropic low-rank GW problem via the Dykstra's Algorithm shown in [Scetbon, et al., ICML 2021]

$$\min_{\boldsymbol{T}} -\langle \boldsymbol{D}_X \underbrace{\boldsymbol{T}^{(m)}}_{\text{Constant}} \boldsymbol{D}_Y^\top, \boldsymbol{T} \rangle - \gamma \underbrace{H(\boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{g})}_{\text{Entropy}}$$
$$s.t.\ \boldsymbol{T} = \boldsymbol{Q}\text{diag}(1/\boldsymbol{g})\boldsymbol{R}^\top, \text{ and } \boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{g} \in \mathcal{C}(\boldsymbol{\mu}_X, \boldsymbol{\mu}_Y, r). \tag{25}$$

- The complexity per iteration becomes $\mathcal{O}(M^2 r + N^2 r)$.

[Scetbon, et al., ICML 2021] Scetbon, M., Cuturi, M., & Peyré, G. Low-rank Sinkhorn factorization. ICML 2021.

Acceleration Strategy 2: Low-rank Structured GW

- In each step, the convergence to a unique solution is guaranteed (because of the convergence of Dykstra's algorithm).

Acceleration Strategy 2: Low-rank Structured GW

- In each step, the convergence to a unique solution is guaranteed (because of the convergence of Dykstra's algorithm).
- Furthermore, when the matrices $\boldsymbol{D}_X$ and $\boldsymbol{D}_Y$ own low-rank structures, i.e.,

$$\boldsymbol{D}_X = \boldsymbol{X}\boldsymbol{X}^T, \quad \boldsymbol{D}_Y = \boldsymbol{Y}\boldsymbol{Y}^T, \text{ where } \boldsymbol{X} \in \mathbb{R}^{M\times d}, \quad \boldsymbol{Y} \in \mathbb{R}^{N\times d'}, \tag{26}$$

the complexity can be further reduced to $\mathcal{O}(Mr(r+d) + Nr(r+d'))$ (generally, $d \ll M$ and $d' \ll N$).

Acceleration Strategy 3: Tree Structured GW

Motivation:

- In practice, graphs or point clouds often own tree structures.
- The tree structures are also beneficial for accelerating GW distance.

Acceleration Strategy 3: Tree Structured GW

Motivation:

- In practice, graphs or point clouds often own tree structures.
- The tree structures are also beneficial for accelerating GW distance.

Principle:

- Given $\boldsymbol{X} = \{x_i\}_{i=1}^{M}$ and $\boldsymbol{Y} = \{y_j\}_{j=1}^{N}$, define a **flow-based alignment discrepancy** [Le, et al., AISTATS 2021]:

$$d_A(\boldsymbol{X}, \boldsymbol{Y}) = \underbrace{\min_{r_x, r_z, \boldsymbol{T} \in \Pi(\boldsymbol{\mu}_X, \boldsymbol{\mu}_Y)} \sum\nolimits_{i,j} |d_{T_X}(r_x, x_i) - d_{T_Y}(r_y, y_j)|^2 T_{ij}}_{d_w(T_X(r_x, \boldsymbol{\mu}_X),\ T_Y(r_y, \boldsymbol{\mu}_Y))}, \tag{27}$$

  where explore optimal transport under optimal root pair (r_x, r_y).

[Le, et al., AISTATS 2021] Le, T., Ho, N., & Yamada, M. Flow-based alignment approaches for probability measures in different spaces. AISTATS 2021.

Acceleration Strategy 3: Tree Structured GW

$$d_A(\boldsymbol{X}, \boldsymbol{Y}) = \min_{r_x, r_z, \boldsymbol{T} \in \Pi(\boldsymbol{\mu}_X, \boldsymbol{\mu}_Y)} \sum_{i,j} |d_{T_X}(r_x, x_i) - d_{T_Y}(r_y, y_j)|^2 T_{ij},$$

- The roots and tree structures can be generated by exploring the hierarchical clustering structures within $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively.
- Based on the tree structure, the distance from a root to a node (i.e., $d_{T_X}(r_x, x_i)$) can be calculated in a recursive manner.
- Complexity becomes $\mathcal{O}(MN)$.

Acceleration Strategy 3: Tree Structured GW

$$d_A(\boldsymbol{X}, \boldsymbol{Y}) = \min_{r_x, r_z, \boldsymbol{T} \in \Pi(\boldsymbol{\mu}_X, \boldsymbol{\mu}_Y)} \sum_{i,j} |d_{T_X}(r_x, x_i) - d_{T_Y}(r_y, y_j)|^2 T_{ij},$$

- The roots and tree structures can be generated by exploring the hierarchical clustering structures within $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively.
- Based on the tree structure, the distance from a root to a node (i.e., $d_{T_X}(r_x, x_i)$) can be calculated in a recursive manner.
- Complexity becomes $\mathcal{O}(MN)$.
- **Theorem 1** in [Le, et al., AISTATS 2021]: d_A is a pseudo-distance satisfying symmetry and the triangle inequality, and $d_{gw} \leq 2d_A$ for the trees with $depth \leq 2$.

Acceleration Strategy 4: Sliced Gromov-Wasserstein

Motivation:

- The optimal transport between one-dimensional distributions is relatively easy to solve.

Acceleration Strategy 4: Sliced Gromov-Wasserstein

Motivation:

- The optimal transport between one-dimensional distributions is relatively easy to solve.

Sliced-Wasserstein distance [Kolouri, et al., NeurIPS 2019]:

- Given two distributions μ and η defined on a metric space $(\mathcal{X} \subset \mathbb{R}^d, d_X)$, projecting $\forall x \in \mathcal{X}$ through a linear projection $\theta \in \mathcal{S}^{d-1}$, *i.e.*, $R_\theta(x) = \langle x, \theta \rangle$, leads to
  - A one-dimensional metric space $(R_\theta(\mathcal{X}), d_{R_\theta(\mathcal{X})})$.
  - The one-dimensional distributions after projection $R_\theta \# \mu$ and $R_\theta \# \eta$.

Acceleration Strategy 4: Sliced Gromov-Wasserstein

Motivation:

- The optimal transport between one-dimensional distributions is relatively easy to solve.

Sliced-Wasserstein distance [Kolouri, et al., NeurIPS 2019]:

- Given two distributions μ and η defined on a metric space $(\mathcal{X} \subset \mathbb{R}^d, d_X)$, projecting $\forall x \in \mathcal{X}$ through a linear projection $\theta \in \mathcal{S}^{d-1}$, *i.e.*, $R_\theta(x) = \langle x, \theta \rangle$, leads to
  - A one-dimensional metric space $(R_\theta(\mathcal{X}), d_{R_\theta(\mathcal{X})})$.
  - The one-dimensional distributions after projection $R_\theta \# \mu$ and $R_\theta \# \eta$.

$$d_{sw}(\mu, \eta) = \mathbb{E}_{\theta \sim p_{\mathcal{S}^{d-1}}}[d_w(R_\theta \# \mu, R_\theta \# \eta)] = \int_{\theta \in \mathcal{S}^{d-1}} p(\theta) d_w(R_\theta \# \mu, R_\theta \# \eta) d\theta \quad (28)$$

[Kolouri, et al., NeurIPS 2019] Kolouri, S., Nadjahi, K., Simsekli, U., Badeau, R., & Rohde, G. Generalized sliced wasserstein distances. NeurIPS, 2019.

Acceleration Strategy 4: Sliced Gromov-Wasserstein

- Practical implementation:
  - Samples $\boldsymbol{X} = \{x_i\}_{i=1}^N \sim \mu$ and $\boldsymbol{Y} = \{y_i\}_{i=1}^N \sim \eta$ are provided.
  - Finite number of projections are sampled based on a distribution $\{\theta_l\}_{l=1}^L \sim p_{\mathcal{S}^{d-1}}$.

Acceleration Strategy 4: Sliced Gromov-Wasserstein

- Practical implementation:
  - Samples $\boldsymbol{X} = \{x_i\}_{i=1}^N \sim \mu$ and $\boldsymbol{Y} = \{y_i\}_{i=1}^N \sim \eta$ are provided.
  - Finite number of projections are sampled based on a distribution $\{\theta_l\}_{l=1}^L \sim p_{\mathcal{S}^{d-1}}$.
- Sample-based sliced Wasserstein distance:

$$\hat{d}_{sw}(\mu, \eta) = \frac{1}{L} \sum_{l=1}^{L} \left(\min_{\boldsymbol{T} \in \Pi(\frac{1}{N}\mathbf{1}_N, \frac{1}{N}\mathbf{1}_N)} \sum_{i,j=1}^{N} |\theta_l^\top x_i - \theta_l^\top y_j|^2 T_{ij} \right)$$

Acceleration Strategy 4: Sliced Gromov-Wasserstein

- Practical implementation:
  - Samples $\boldsymbol{X} = \{x_i\}_{i=1}^N \sim \mu$ and $\boldsymbol{Y} = \{y_i\}_{i=1}^N \sim \eta$ are provided.
  - Finite number of projections are sampled based on a distribution $\{\theta_l\}_{l=1}^L \sim p_{\mathcal{S}^{d-1}}$.
- Sample-based sliced Wasserstein distance:

$$\begin{aligned}\hat{d}_{sw}(\mu,\eta) &= \frac{1}{L}\sum_{l=1}^{L}\left(\min_{\boldsymbol{T}\in\Pi(\frac{1}{N}\mathbf{1}_N,\frac{1}{N}\mathbf{1}_N)}\sum_{i,j=1}^{N}|\theta_l^\top x_i - \theta_l^\top y_j|^2 T_{ij}\right)\\ &= \frac{1}{L}\sum_{l=1}^{L}\left(\frac{1}{N}\min_{\sigma\in\mathcal{P}_N}\sum_{i=1}^{N}|\theta_l^\top x_i - \theta_l^\top y_{\sigma(i)}|^2\right)\end{aligned} \tag{29}$$

Theorem: For one dimensional $x_1 \leq ... \leq x_N$ and $y_1 \leq ... \leq y_N$, identity permutation ($\sigma(i) = i$ for $i = 1, ..., N$) leads to the optimal transport between them.

Acceleration Strategy 4: Sliced Gromov-Wasserstein

- Similarly, given $(\mathcal{X} \in \mathbb{R}^d, d_X, \mu_X)$ and $(\mathcal{Y} \in \mathbb{R}^d, d_Y, \mu_Y)$, we can project the mm-spaces to one-dimensional mm-spaces by a linear projection $\theta \in \mathcal{S}^{d-1}$:

$$\begin{aligned} \mathcal{X}_\theta &:= R_\theta(\mathcal{X})_{d_{R_\theta(X)}, R_\theta \# \mu_X}, \\ \mathcal{Y}_\theta &:= R_\theta(\mathcal{Y})_{d_{R_\theta(Y)}, R_\theta \# \mu_Y}. \end{aligned} \tag{30}$$

- When $dim(\mathcal{X}) \neq dim(\mathcal{Y})$, padding zeros to the samples in the lower-dimensional space.

Acceleration Strategy 4: Sliced Gromov-Wasserstein

- Similarly, given $(\mathcal{X} \in \mathbb{R}^d, d_X, \mu_X)$ and $(\mathcal{Y} \in \mathbb{R}^d, d_Y, \mu_Y)$, we can project the mm-spaces to one-dimensional mm-spaces by a linear projection $\theta \in \mathcal{S}^{d-1}$:

$$\begin{aligned} \mathcal{X}_\theta &:= R_\theta(\mathcal{X})_{d_{R_\theta(X)}, R_\theta \# \mu_X}, \\ \mathcal{Y}_\theta &:= R_\theta(\mathcal{Y})_{d_{R_\theta(Y)}, R_\theta \# \mu_Y}. \end{aligned} \tag{30}$$

- When $dim(\mathcal{X}) \neq dim(\mathcal{Y})$, padding zeros to the samples in the lower-dimensional space.
- **Sliced Gromov-Wasserstein distance** [Titouan, et al., NeurIPS 2019]:

$$d_{sgw}(\mathcal{X}_{d_X, \mu_X}, \mathcal{Y}_{d_Y, \mu_Y}) = \mathbb{E}_{\theta \sim p_{\mathcal{S}^{d-1}}}[d_{gw}(\mathcal{X}_\theta, \mathcal{Y}_\theta)]. \tag{31}$$

[Titouan, et al., NeurIPS 2019] Titouan, V., Flamary, R., Courty, N., Tavenard, R., & Chapel, L. Sliced gromov-wasserstein. NeurIPS, 2019.

Acceleration Strategy 4: Sliced Gromov-Wasserstein

- Practical implementation:
  - Samples $\boldsymbol{X} = \{x_i\}_{i=1}^N \sim \mu_X$ and $\boldsymbol{Y} = \{y_i\}_{i=1}^N \sim \mu_Y$ are provided.
  - Finite number of projections are sampled based on a distribution $\{\theta_l\}_{l=1}^L \sim p_{\mathcal{S}^{d-1}}$.
- **Sample-based sliced Gromov-Wasserstein distance:**

$$\hat{d}_{sgw}(\boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{L} \sum_{l=1}^{L} \left(\min_{\boldsymbol{T} \in \Pi(\frac{1}{N}\mathbf{1}_N, \frac{1}{N}\mathbf{1}_N)} \sum_{i,i,j,j'=1}^{N} (|\theta_l^\top x_i - \theta_l^\top x_{i'}|^2 - |\theta_l^\top y_j - \theta_l^\top y_{j'}|^2)^2 T_{ij} T_{i'j'} \right)$$

Acceleration Strategy 4: Sliced Gromov-Wasserstein

- Practical implementation:
  - Samples $\boldsymbol{X} = \{x_i\}_{i=1}^N \sim \mu_X$ and $\boldsymbol{Y} = \{y_i\}_{i=1}^N \sim \mu_Y$ are provided.
  - Finite number of projections are sampled based on a distribution $\{\theta_l\}_{l=1}^L \sim p_{\mathcal{S}^{d-1}}$.
- **Sample-based sliced Gromov-Wasserstein distance:**

$$
\begin{aligned}
&\hat{d}_{sgw}(\boldsymbol{X}, \boldsymbol{Y}) \\
&= \frac{1}{L} \sum_{l=1}^{L} \left(\min_{\boldsymbol{T} \in \Pi(\frac{1}{N}\mathbf{1}_N, \frac{1}{N}\mathbf{1}_N)} \sum_{i,i,j,j'=1}^{N} (|\theta_l^\top x_i - \theta_l^\top x_{i'}|^2 - |\theta_l^\top y_j - \theta_l^\top y_{j'}|^2)^2 T_{ij} T_{i'j'} \right) \quad (32) \\
&= \frac{1}{L} \sum_{l=1}^{L} \left(\frac{1}{N^2} \min_{\sigma \in \mathcal{P}_N} \sum_{i,i'=1}^{N} (|\theta_l^\top x_i - \theta_l^\top x_{i'}|^2 - |\theta_l^\top y_{\sigma(i)} - \theta_l^\top y_{\sigma(i')}|^2)^2 \right)
\end{aligned}
$$

Theorem 3.2 in [Titouan, et al., NeurIPS 2019]: If $x_1 < ... < x_n$ and $y_1 < ... < y_n$, this result is achieved either by the identity or the anti-identity permutation.

Facing to More Complicated Scenarios

Challenges:

- The nodes of graphs might be attributed.
- The graphs might be have clustering structures.
- The node distributions might be unavailable or unreliable.

Facing to More Complicated Scenarios

Challenges:

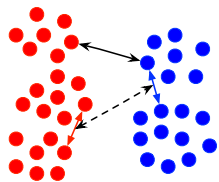
- The nodes of graphs might be attributed.
- The graphs might be have clustering structures.
- The node distributions might be unavailable or unreliable.

More variants of GW:

- Consider node attributes $\Rightarrow$ Fused GW Distance
- Consider clustering structures $\Rightarrow$ Hierarchical (Fused) GW Distance
- Relax doubly-stochastic constraint $\Rightarrow$ Unbalance/Partial GW Distance

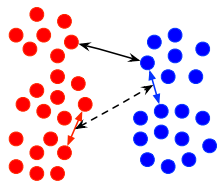
Fused Gromov-Wasserstein Distance

For graphs/point clouds, consider the GW distance between their topological structure/nodes' relations and the Wasserstein distance between their node attributions jointly [Titouan, et al., ICML 2019].



Fused Gromov-Wasserstein Distance

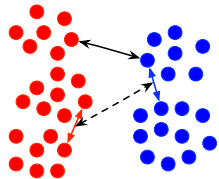
For graphs/point clouds, consider the GW distance between their topological structure/nodes' relations and the Wasserstein distance between their node attributions jointly [Titouan, et al., ICML 2019].



$$d_{fgw}(\mathcal{X}_{d_X,\mu_X}, \mathcal{Y}_{d_Y,\mu_Y}; \beta) = \inf_{\pi \in \Pi(\mu_X,\mu_Y)} \Big((1-\beta) \underbrace{\int_{\mathcal{X}\times\mathcal{Y}} d_{x,y} \pi(x,y) \mathrm{d}x\mathrm{d}y}_{\text{Wasserstein term}} + \beta \underbrace{\int_{\mathcal{X}\times\mathcal{Y}} \int_{\mathcal{X}\times\mathcal{Y}} r_{x,y,x',y'} \pi(x,y)\pi(x',y') \mathrm{d}x\mathrm{d}x'\mathrm{d}y\mathrm{d}y'}_{\text{Gromov-Wasserstein term}} \Big). \tag{33}$$

[Titouan, et al., ICML 2019] Titouan, V., Courty, N., Tavenard, R., & Flamary, R. Optimal transport for structured data with application on graphs. ICML 2019.

Fused Gromov-Wasserstein Distance

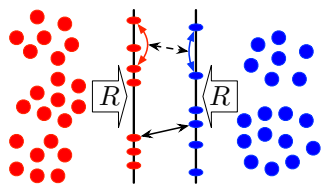


In practice, given $\boldsymbol{X} = \{x_i\}_{i=1}^{M}$ and $\boldsymbol{Y} = \{y_j\}_{j=1}^{N}$, we have

$$\begin{aligned}\hat{d}_{fgw}(\boldsymbol{X}, \boldsymbol{Y}; \beta) = & \min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_X, \boldsymbol{\mu}_Y)} \Big((1-\beta) \sum\nolimits_{i=1}^{M} \sum\nolimits_{j=1}^{N} d(x_i, y_j) T_{ij} \\ & + \beta \sum\nolimits_{i,i'=1}^{M} \sum\nolimits_{j,j'=1}^{N} r(x_i, x_{i'}, y_j, y_{j'}) T_{ij} T_{i'j'} \Big) \\ = & \min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_X, \boldsymbol{\mu}_Y)} \langle \boldsymbol{D}_{XY} - 2 \boldsymbol{D}_X \boldsymbol{T} \boldsymbol{D}_Y^{\top}, \boldsymbol{T} \rangle \end{aligned} \tag{34}$$

Sliced Fused Gromov-Wasserstein Distance

The Sliced FGW distance in [Xu, et al., ICML 2020] considers the expectation of the FGW distance between one-dimensional mm-spaces derived by random projections:



$$d_{sfgw}(\mathcal{X}_{d_X,\mu_X}, \mathcal{Y}_{d_Y,\mu_Y}; \beta) = \mathbb{E}_{\theta \sim p_{\mathcal{S}^{d-1}}}[d_{fgw}(\mathcal{X}_\theta, \mathcal{Y}_\theta; \beta)]. \tag{35}$$

[Xu, et al., ICML 2020] Xu, H., Luo, D., Henao, R., Shah, S., & Carin, L. Learning autoencoders with relational regularization. ICML 2020.

Sliced Fused Gromov-Wasserstein Distance

Given $\boldsymbol{X} = \{x_i\}_{i=1}^N$ and $\boldsymbol{Y} = \{y_j\}_{j=1}^N$, and L linear projections $\{\theta_l \in \mathcal{S}^{d-1}\}$, we have

$$\hat{d}_{sfgw}(\boldsymbol{X}, \boldsymbol{Y}; \beta) = \frac{1}{L} \sum_{l=1}^{L} \Big(\min_{\sigma \in \mathcal{P}_N} (1-\beta) \sum_{i=1}^{N} |\theta_l^\top x_i - \theta_l^\top y_{\sigma(i)}|^2 + \beta \sum_{i,i'=1}^{N} (|\theta_l^\top x_i - \theta_l^\top x_{i'}|^2 - |\theta_l^\top y_{\sigma(i)} - \theta_l^\top y_{\sigma(i')}|^2)^2 \Big). \tag{36}$$

Sliced Fused Gromov-Wasserstein Distance

Given $\boldsymbol{X} = \{x_i\}_{i=1}^N$ and $\boldsymbol{Y} = \{y_j\}_{j=1}^N$, and L linear projections $\{\theta_l \in \mathcal{S}^{d-1}\}$, we have

$$\hat{d}_{sfgw}(\boldsymbol{X}, \boldsymbol{Y}; \beta) = \frac{1}{L} \sum_{l=1}^{L} \Big(\min_{\sigma \in \mathcal{P}_N} (1-\beta) \sum_{i=1}^{N} |\theta_l^\top x_i - \theta_l^\top y_{\sigma(i)}|^2 + \beta \sum_{i,i'=1}^{N} (|\theta_l^\top x_i - \theta_l^\top x_{i'}|^2 - |\theta_l^\top y_{\sigma(i)} - \theta_l^\top y_{\sigma(i')}|^2)^2 \Big). \tag{36}$$

- **Theorem 3.4** in [Xu, et al., ICML 2020]: For one-dimensional samples $x_1 < ... < x_n$ and $y_1 < ... < y_n$, we denote their zero-mean translations as $x' = \{x'_i\}$ and $y' = \{y'_i\}$, respectively.
  1. When $(\sum_i x'_i y'_i + \frac{1-\beta}{8\beta})^2 \geq (\sum_i x'_i y'_{n+1-i} + \frac{1-\beta}{8\beta})^2$, the solution is the identity permutation $\sigma(i) = i$.
  2. Otherwise, the solution is the anti-identity permutation $\sigma(i) = n + 1 - i$.

Sliced Fused Gromov-Wasserstein Distance

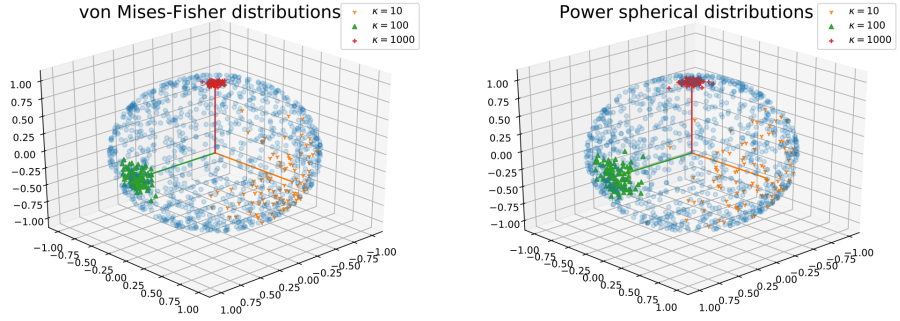
Given $\boldsymbol{X} = \{x_i\}_{i=1}^N$ and $\boldsymbol{Y} = \{y_j\}_{j=1}^N$, and L linear projections $\{\theta_l \in \mathcal{S}^{d-1}\}$, we have

$$\hat{d}_{sfgw}(\boldsymbol{X}, \boldsymbol{Y}; \beta) = \frac{1}{L} \sum_{l=1}^{L} \Big(\min_{\sigma \in \mathcal{P}_N} (1-\beta) \sum_{i=1}^{N} |\theta_l^\top x_i - \theta_l^\top y_{\sigma(i)}|^2 + \beta \sum_{i,i'=1}^{N} (|\theta_l^\top x_i - \theta_l^\top x_{i'}|^2 - |\theta_l^\top y_{\sigma(i)} - \theta_l^\top y_{\sigma(i')}|^2)^2 \Big). \tag{36}$$

- **Theorem 3.4** in [Xu, et al., ICML 2020]: For one-dimensional samples $x_1 < ... < x_n$ and $y_1 < ... < y_n$, we denote their zero-mean translations as $x' = \{x'_i\}$ and $y' = \{y'_i\}$, respectively.
  1. When $(\sum_i x'_i y'_i + \frac{1-\beta}{8\beta})^2 \geq (\sum_i x'_i y'_{n+1-i} + \frac{1-\beta}{8\beta})^2$, the solution is the identity permutation $\sigma(i) = i$.
  2. Otherwise, the solution is the anti-identity permutation $\sigma(i) = n + 1 - i$.
- Obviously, when $\beta = 1$, we obtain sliced GW distance. Theorem 3.4 extends the Theorem 3.2 in [Titouan, et al, NeurIPS 2019], which provides an explicit condition to compute the sliced GW distance.

Sampling Strategies for Sliced (F)GW Distance

- Uniform sampling: $p_{\mathcal{S}^{d-1}} = \text{Uniform}(\mathcal{S}^{d-1})$.
- Spherical distributed sampling [Nguyen, et al., ICLR 2020]



The von Mises-Fisher distribution and the power spherical distribution illustrated in [Nguyen, et al., ICLR 2020].

[Nguyen, et al., ICLR 2020] Nguyen, K., Nguyen, S., Ho, N., Pham, T., & Bui, H. Improving Relational Regularized Autoencoders with Spherical Sliced Fused Gromov Wasserstein. ICLR 2020.

Sampling Strategies for Sliced (F)GW Distance

- Projection pursuit driven by diversity [Meng, et al., NeurIPS 2019]
  - Given current projections, finding the new projection confusing a discriminator.
  - Connect with the idea of GAN — the projections aim at cheating the discriminator classifying point sources.

[Meng, et al., NeurIPS 2019] Meng, C., Ke, Y., Zhang, J., Zhang, M., Zhong, W., & Ma, P. Large-scale optimal transport map estimation using projection pursuit. NeurIPS 2019.

Hierarchical Fused Gromov-Wasserstein Distance

For $(\mathcal{X}, d_X, \mu_X)$ and $(\mathcal{Y}, d_Y, \mu_Y)$, suppose that

$$\begin{aligned} \mu_X &= \sum_{i=1}^{I} a_i p_i, \quad \boldsymbol{a} \in \Delta^{I-1},\ p_i = \mathcal{N}(\boldsymbol{u}_i, \boldsymbol{\Sigma}_i), \\ \mu_Y &= \sum_{j=1}^{J} b_j q_j, \quad \boldsymbol{b} \in \Delta^{J-1},\ q_j = \mathcal{N}(\boldsymbol{v}_j, \boldsymbol{\Phi}_j). \end{aligned} \tag{37}$$

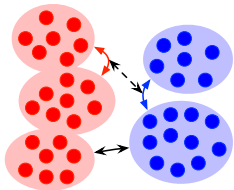
Hierarchical Fused Gromov-Wasserstein Distance

For $(\mathcal{X}, d_X, \mu_X)$ and $(\mathcal{Y}, d_Y, \mu_Y)$, suppose that

$$\begin{aligned} \mu_X &= \sum\nolimits_{i=1}^{I} a_i p_i, \quad \boldsymbol{a} \in \Delta^{I-1},\ p_i = \mathcal{N}(\boldsymbol{u}_i, \boldsymbol{\Sigma}_i), \\ \mu_Y &= \sum\nolimits_{j=1}^{J} b_j q_j, \quad \boldsymbol{b} \in \Delta^{J-1},\ q_j = \mathcal{N}(\boldsymbol{v}_j, \boldsymbol{\Phi}_j). \end{aligned} \tag{37}$$

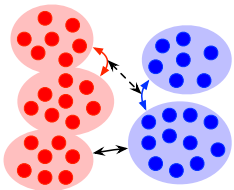
Principle:

- Consider the fused GW distance between the GMM components in different mm-spaces.



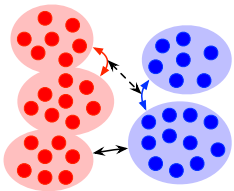
[Xu, et al., ICML 2020] Xu, H., Luo, D., Henao, R., Shah, S., & Carin, L. Learning autoencoders with relational regularization. ICML 2020.

Hierarchical Fused Gromov-Wasserstein Distance



$$
\begin{aligned}
& d_{hfgw}(\mathcal{X}_{d_X,\mu_X}, \mathcal{Y}_{d_Y,\mu_Y}; \beta) \\
=& d_{\text{fgw}}(\boldsymbol{a}, \boldsymbol{b}; \beta) \text{ with Wasserstein ground distance} \\
=& \min_{\boldsymbol{T} \in \Pi(\boldsymbol{a},\boldsymbol{b})} (1-\beta) \sum\nolimits_{i,j} d_w(p_i, q_j) T_{ij} + \\
& \beta \sum\nolimits_{i,i',j,j'} |d_w(p_i, p_{i'}) - d_w(q_j, q_{j'})|^2 T_{ij} T_{i'j'}.
\end{aligned}
\tag{38}
$$

Hierarchical Fused Gromov-Wasserstein Distance



$$
\begin{aligned}
& d_{hfgw}(\mathcal{X}_{d_X,\mu_X}, \mathcal{Y}_{d_Y,\mu_Y}; \beta) \\
=& d_{\text{fgw}}(\boldsymbol{a}, \boldsymbol{b}; \beta) \text{ with Wasserstein ground distance} \\
=& \min_{\boldsymbol{T} \in \Pi(\boldsymbol{a},\boldsymbol{b})} (1-\beta) \sum\nolimits_{i,j} d_w(p_i, q_j) T_{ij} + \\
& \beta \sum\nolimits_{i,i',j,j'} |d_w(p_i, p_{i'}) - d_w(q_j, q_{j'})|^2 T_{ij} T_{i'j'}.
\end{aligned} \tag{38}
$$

- Closed form: $d_w(p,q) = \|\boldsymbol{u}_p - \boldsymbol{u}_q\|_2^2 + \text{trace}(\boldsymbol{\Sigma}_p + \boldsymbol{\Sigma}_q - 2(\boldsymbol{\Sigma}_p^{\frac{1}{2}} \boldsymbol{\Sigma}_q \boldsymbol{\Sigma}_p^{\frac{1}{2}})^{\frac{1}{2}})$ [Takatsu, 2011].
- $\boldsymbol{T} = [T_{ij}]$ is the optimal transport between the components of the two GMMs.
- Complexity: $\mathcal{O}(I^2 J + J^2 I)$.

[Takatsu, 2011] Takatsu, A. Wasserstein geometry of Gaussian measures. Osaka Journal of Mathematics, 48(4), 1005-1026, 2011.

Partial and Unbalanced Gromov-Wasserstein Distance

Motivation:

- For $(\mathcal{X}, d_X, \mu_X)$ and $(\mathcal{Y}, d_Y, \mu_Y)$, the μ_X and μ_Y are prior distributions.

Partial and Unbalanced Gromov-Wasserstein Distance

Motivation:

- For $(\mathcal{X}, d_X, \mu_X)$ and $(\mathcal{Y}, d_Y, \mu_Y)$, the μ_X and μ_Y are prior distributions.

Given $\boldsymbol{X} \subset \mathcal{X}$ and $\boldsymbol{Y} \subset \mathcal{Y}$, partial Gromov-Wasserstein Distance [Chapel, et al., NeurIPS 2020]:

$$d_{pgw}(\boldsymbol{X}, \boldsymbol{Y}) = \min_{\boldsymbol{T} \in \Pi_s(\boldsymbol{\mu}_X, \boldsymbol{\mu}_Y)} \sum_{i,i'=1}^{M} \sum_{j,j'=1}^{N} r(x_i, x_{i'}, y_j, y_{j'}) T_{ij} T_{i'j'} \tag{39}$$

where $\Pi_s(\boldsymbol{\mu}_X, \boldsymbol{\mu}_Y) = \{\boldsymbol{T} > \mathbf{0} | \boldsymbol{T}\mathbf{1}_N \leq \boldsymbol{\mu}_X, \boldsymbol{T}^\top \mathbf{1}_M \leq \boldsymbol{\mu}_Y, \mathbf{1}_M^\top \boldsymbol{T} \mathbf{1}_N = s\}$.

Partial and Unbalanced Gromov-Wasserstein Distance

Motivation:

- For $(\mathcal{X}, d_X, \mu_X)$ and $(\mathcal{Y}, d_Y, \mu_Y)$, the μ_X and μ_Y are prior distributions.

Given $\boldsymbol{X} \subset \mathcal{X}$ and $\boldsymbol{Y} \subset \mathcal{Y}$, partial Gromov-Wasserstein Distance [Chapel, et al., NeurIPS 2020]:

$$
\begin{aligned}
& d_{pgw}(\boldsymbol{X}, \boldsymbol{Y}) = \min_{\boldsymbol{T} \in \Pi_s(\boldsymbol{\mu}_X, \boldsymbol{\mu}_Y)} \sum_{i,i'=1}^{M} \sum_{j,j'=1}^{N} r(x_i, x_{i'}, y_j, y_{j'}) T_{ij} T_{i'j'} \\
& \text{where } \Pi_s(\boldsymbol{\mu}_X, \boldsymbol{\mu}_Y) = \{\boldsymbol{T} > \mathbf{0} | \boldsymbol{T}\mathbf{1}_N \leq \boldsymbol{\mu}_X, \boldsymbol{T}^\top \mathbf{1}_M \leq \boldsymbol{\mu}_Y, \mathbf{1}_M^\top \boldsymbol{T} \mathbf{1}_N = s\}.
\end{aligned} \tag{39}
$$

- Introduce virtual points $\tilde{\boldsymbol{X}} = \boldsymbol{X} \cup x_v$ and $\tilde{\boldsymbol{Y}} = \boldsymbol{Y} \cup y_v$, and reformulate the problem to a classic GW problem:

$$
\tilde{\boldsymbol{T}} \in \Pi(\tilde{\boldsymbol{\mu}}_X, \tilde{\boldsymbol{\mu}}_Y), \quad \text{where } \tilde{\boldsymbol{\mu}}_X = [\boldsymbol{\mu}_X; 1 - \|\boldsymbol{\mu}_X\|_1], \quad \tilde{\boldsymbol{\mu}}_Y = [\boldsymbol{\mu}_Y; 1 - \|\boldsymbol{\mu}_Y\|_1]. \tag{40}
$$

[Chapel, et al., NeurIPS 2020] Chapel, L., Alaya, M. Z., & Gasso, G. Partial optimal tranport with applications on positive-unlabeled learning. NeurIPS 2020.

Partial and Unbalanced Gromov-Wasserstein Distance

Unbalanced Gromov-Wasserstein Distance — Regularize the marginal distributions of optimal transport directly [Séjourné, et al., NeurIPS 2021]:

$$d_{ugw}(\mathcal{X}_{d_X,\mu_X}, \mathcal{Y}_{d_Y,\mu_Y}) = \inf_{\pi} \int_{(x,x',y,y')\in\mathcal{X}^2\times\mathcal{Y}^2} r(x,x',y,y')\pi(x,y)\pi(x',y')dxdx'dydy' + D_\psi(\pi_X \otimes \pi_X \| \mu_X \otimes \mu_X) + D_\psi(\pi_Y \otimes \pi_Y \| \mu_Y \otimes \mu_Y). \quad (41)$$

- π_X and π_Y are marginals of π.
- The quadratic ψ-divergence (which can be specialized as KL-divergence, Total-variance) makes d_{ugw} 2-homogeneous.
- When ψ-divergence = TV, $d_{ugw} \Rightarrow d_{pgw}$.
- An alternating Sinkhorn algorithm can be used to solve this problem.

[Séjourné, et al., NeurIPS 2021] Séjourné, T., Vialard, F. X., & Peyré, G. The unbalanced Gromov Wasserstein distance: Conic formulation and relaxation. NeurIPS 2021.

Application Scenarios of GWD and Its Variances

- **Input:** Relation matrices and marginal distributions;
- **Output:** OT matrix
  - (Fused) GWD (CG, ProxGrad, BADMM, ...)
  - Acceleration: S-GWL (Divide and conquer), Low-rank structured GW.
  - Unreliable distribution: PGW or UGW

Application Scenarios of GWD and Its Variances

- **Input:** Relation matrices and marginal distributions;
- **Output:** OT matrix
  - (Fused) GWD (CG, ProxGrad, BADMM, ...)
  - Acceleration: S-GWL (Divide and conquer), Low-rank structured GW.
  - Unreliable distribution: PGW or UGW
- **Input:** Point clouds and marginal distributions
- **Output:** (Fused) GW distance or its surrogates.
  - (Fused) GWD (CG, ProxGrad, BADMM, ...), Sliced (Fused) GWD.
  - Acceleration: S-GWL (Divide and conquer), Low-rank structured GW.
  - With clustering structure: Tree-structured GWD, Hierarchical (Fused) GWD.
  - Unreliable distribution: PGW or UGW

Summary

- GW distance can be calculated through various iterative optimization methods.
- Many strategies are applicable to accelerate and/or approximate the GW distance.
- With simple extensions, GW distance becomes applicable to complicated scenarios, e.g., attributed graph, point clouds with clustering structures, etc.

Next,

- **Beyond graph matching and partitioning, apply GW distance and its variants to more machine learning tasks.**
- **Discuss ongoing research and open problems.**

5-min break for Q & A

Outline

Overview of GWL and Its Applications

Methodology

- Take GW distance and/or its variants as objective functions or regularizers.
- Leverage the optimization of GW distance to construct computational modules of learning systems.

Overview of GWL and Its Applications

Methodology

- Take GW distance and/or its variants as objective functions or regularizers.
- Leverage the optimization of GW distance to construct computational modules of learning systems.

Applications

- Graph matching and partitioning (See Part 1)
- Generative modeling: Wasserstein Autoencoder with Relational Regularization
- Graph representation: GW factorization model
- Graph generation: GW barycenter-based Graphon estimation

Generative Modeling: Relationally-Regularized Wasserstein Autoencoder

Wasserstein autoencoder (WAE) [Tolstikhin, et al., ICLR 2018] fits the model distribution p_G by minimizing its Wasserstein distance to the data distribution p_x:

$$\min D_{\mathsf{w}}(p_x, p_G) \Longrightarrow \min_{G,Q} \underbrace{\mathbb{E}_{p_x}\mathbb{E}_{q_{z|x;Q}}[d(x, G(z))]}_{\text{reconstruction loss}} + \underbrace{\gamma D(\overbrace{\mathbb{E}_{p_x}[q_{z|x;Q}]}^{q_{z;Q}}, p_z)}_{\text{distance(posterior, prior)}}, \tag{42}$$

- $G : \mathcal{Z} \mapsto \mathcal{X}$ is the target generative model (**decoder**).
- $q_{z|x;Q}$ is the posterior of z given x, parameterized by an **encoder** $Q : \mathcal{X} \mapsto \mathcal{Z}$.
- $q_{z;Q} = \mathbb{E}_{p_x}[q_{z|x;Q}]$ is the expectation of the posterior distributions.
- p_z is the prior of z.

[Tolstikhin, et al., ICLR 2018] Tolstikhin, I., Bousquet, O., Gelly, S., & Schoelkopf, B. (2018, February). Wasserstein Auto-Encoders. ICLR 2018.

Autoencoders with Predefined or Learnable Prior

$$\min_{G,Q} \underbrace{\mathbb{E}_{p_x}\mathbb{E}_{q_{z|x;Q}}[d(x, G(z))]}_{\text{reconstruction loss}} + \underbrace{\gamma D(\overbrace{\mathbb{E}_{p_x}[q_{z|x;Q}]}^{q_{z;Q}}, p_z)}_{\text{distance(posterior, prior)}}, \tag{43}$$

Predefined prior, $e.g.$, $p_z = \mathcal{N}(z; 0, I)$.

- Over-regularization
- Difficult for conditional generation

Autoencoders with Predefined or Learnable Prior

$$\min_{G,Q} \underbrace{\mathbb{E}_{p_x}\mathbb{E}_{q_{z|x;Q}}[d(x, G(z))]}_{\text{reconstruction loss}} + \underbrace{\gamma D(\overbrace{\mathbb{E}_{p_x}[q_{z|x;Q}]}^{q_{z;Q}}, p_z)}_{\text{distance(posterior, prior)}}, \tag{43}$$

Predefined prior, $e.g.$, $p_z = \mathcal{N}(z; 0, I)$.

- Over-regularization
- Difficult for conditional generation

Learnable prior, $e.g.$, $p_z = \frac{1}{K}\sum_{k=1}^{K}\mathcal{N}(z; u_k, \Sigma_k)$.

- Under-regularization
- Sensitive to hyperparameters and initialization

Proposed Method: Learning with A Relational Regularization

Introduce a relational regularizer based on the Gromov-Wasserstein distance between distributions:

$$\min_{G,Q,p_z\in\mathcal{P}} \ \mathbb{E}_{p_x}\mathbb{E}_{q_{z|x;Q}}[d(x,G(z))] + \gamma(\underbrace{(1-\beta)D(q_{z;Q},p_z)}_{\text{direct comparison}} + \underbrace{\beta D_{\text{gw}}(q_{z;Q},p_z)}_{\text{relational comparison}}). \quad (44)$$

Proposed Method: Learning with A Relational Regularization

Introduce a relational regularizer based on the Gromov-Wasserstein distance between distributions:

$$\min_{G,Q,p_z\in\mathcal{P}} \ \mathbb{E}_{p_x}\mathbb{E}_{q_{z|x;Q}}[d(x,G(z))] + \gamma(\underbrace{(1-\beta)D(q_{z;Q},p_z)}_{\text{direct comparison}} + \underbrace{\beta D_{\text{gw}}(q_{z;Q},p_z)}_{\text{relational comparison}}). \quad (44)$$

We consider the direct comparison $D(q_{z;Q},p_z)$ as the Wasserstein distance, and reformulate the regularizers as **fused Gromov-Wasserstein (FGW)** distance [Titouan, et al., ICML 2019].

$$\min_{G,Q,p_z\in\mathcal{P}} \ \mathbb{E}_{p_x}\mathbb{E}_{q_{z|x;Q}}[d(x,G(z))] + \gamma d_{fgw}(q_{z;Q},p_z;\beta). \quad (45)$$

Proposed Method: Learning with A Relational Regularization

Introduce a relational regularizer based on the Gromov-Wasserstein distance between distributions:

$$\min_{G,Q,p_z\in\mathcal{P}} \ \mathbb{E}_{p_x}\mathbb{E}_{q_{z|x;Q}}[d(x,G(z))] + \gamma(\underbrace{(1-\beta)D(q_{z;Q},p_z)}_{\text{direct comparison}} + \underbrace{\beta D_{\text{gw}}(q_{z;Q},p_z)}_{\text{relational comparison}}). \quad (44)$$

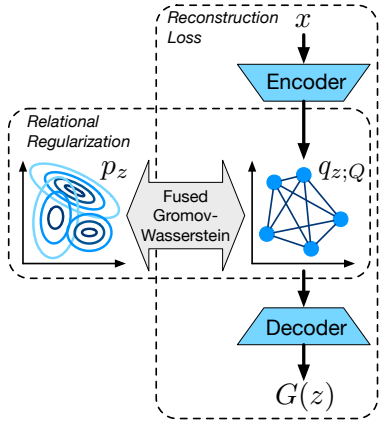
We consider the direct comparison $D(q_{z;Q},p_z)$ as the Wasserstein distance, and reformulate the regularizers as **fused Gromov-Wasserstein (FGW)** distance [Titouan, et al., ICML 2019].

$$\min_{G,Q,p_z\in\mathcal{P}} \ \mathbb{E}_{p_x}\mathbb{E}_{q_{z|x;Q}}[d(x,G(z))] + \gamma d_{fgw}(q_{z;Q},p_z;\beta). \quad (45)$$

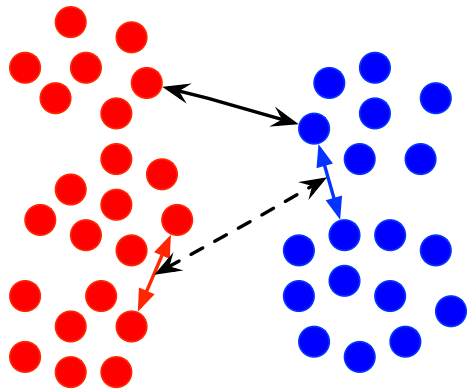
- A shared optimal transport π encourages the consistency of the relations between $q_{z;Q}$ and p_z under different criteria.
- Minimization of an upper bound.

Relationally-Regularized Wasserstein Autoencoder

$$\min_{G,Q,p_z} \mathbb{E}_{p_x}\mathbb{E}_{q_{z|x;Q}}[d(x, G(z))] + \gamma d_{\mathsf{fgw}}(q_{z;Q}, p_z; \beta), \text{ where } p_z = \frac{1}{K}\sum_{k=1}^{K}\mathcal{N}(z; u_k, \Sigma_k). \quad (46)$$



The scheme of the method



The principle of fused GW distance.

Two Computationally-friendly Implementations

Probabilistic Autoencoder + Hierarchical FGW Distance:

- Recall that the prior p_z is a GMM.
- The encoder Q outputs the mean and the log(variance) of $q_{z|x;Q}$ given a sample x. Given a batch of x's, the expected posterior $q_{z;Q}$ becomes a GMM as well.

Two Computationally-friendly Implementations

Probabilistic Autoencoder + Hierarchical FGW Distance:

- Recall that the prior p_z is a GMM.
- The encoder Q outputs the mean and the log(variance) of $q_{z|x;Q}$ given a sample x. Given a batch of x's, the expected posterior $q_{z;Q}$ becomes a GMM as well.

$$d_{hfgw}(p, q; \beta) = \min_{T \in \Pi(a,b)} \langle D_{pq} - 2\beta D_p T D_q^\top, T \rangle, \tag{47}$$

- We solve (47) efficiently by the proximal gradient method in [Xu, et al., ICML 2019].
- The computational complexity is $\mathcal{O}(N^2K + K^2N)$.
- For each batch, we first learn the T, and then update autoencoders with fixed T by backpropagation.

Two Computationally-friendly Implementations

Deterministic Autoencoder + Sliced FGW Distance:

- The encoder Q outputs N samples of $q_{z;Q}$ directly.
- For the GMM prior p_z, we can generate N (differentiable) samples by reparameterization trick.

Two Computationally-friendly Implementations

Deterministic Autoencoder + Sliced FGW Distance:

- The encoder Q outputs N samples of $q_{z;Q}$ directly.
- For the GMM prior p_z, we can generate N (differentiable) samples by reparameterization trick.

$$\hat{d}_{\text{sfgw}}(p_x, p_y; \beta) = \frac{1}{L}\sum\nolimits_{l=1}^{L} \min_{\sigma\in\mathcal{P}_N} \Big(\frac{1-\beta}{N}\sum\nolimits_{i=1}^{N} (x_{i,\theta_l} - y_{\sigma(i),\theta_l})^2 + \frac{\beta}{N}\sum\nolimits_{i,j=1}^{N} ((x_{i,\theta_l} - x_{j,\theta_l})^2 - (y_{\sigma(i),\theta_l} - y_{\sigma(j),\theta_l})^2)^2 \Big). \tag{48}$$

- For projected latent codes, sort them and compute the 1D FGW distance by their identity permutation or anti-identity permutation.
- The average of all the 1D FGW distances leads to the SFGW distance.
- The computational complexity is $\mathcal{O}(L(\underbrace{Nd}_{\text{proj.}} + \underbrace{N\log N}_{\text{sort}} + \underbrace{N^2}_{\text{1D fgw}}))$.

Comparisons with Other Autoencoders

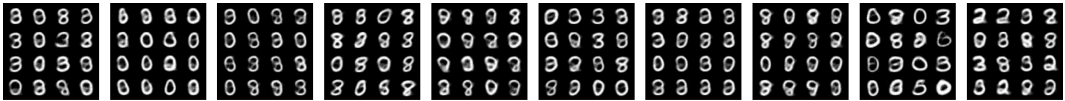
Table: Comparisons for different autoencoders

Method	$Q : \mathcal{X} \mapsto \mathcal{Z}$	p_z	$d(q_{z;Q}, p_z)$
VAE	Probabilistic	$\mathcal{N}(z; 0, I)$	KL
WAE	Deterministic	$\mathcal{N}(z; 0, I)$	MMD/GAN
SWAE	Deterministic	$\mathcal{N}(z; 0, I)$	d_{w}
GMVAE	Probabilistic	$\frac{1}{K}\sum_k \mathcal{N}(z; u_k, \Sigma_k)$	KL
VampPrior	Probabilistic	$\frac{1}{K}\sum_k \mathcal{N}(z; Q(x_k))$	KL
Our RAE	Probabilistic	$\frac{1}{K}\sum_k \mathcal{N}(z; u_k, \Sigma_k)$	d_{hfgw}
	Deterministic		$\hat{d}_{\mathsf{sfgw}}$

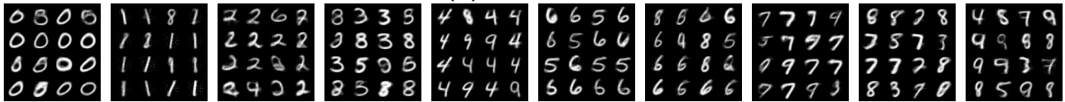
Experiments: Random Image Generation

Figure: WAE v.s. RAE

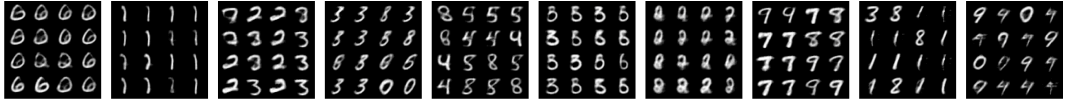
Experiments: Conditional Image Generation



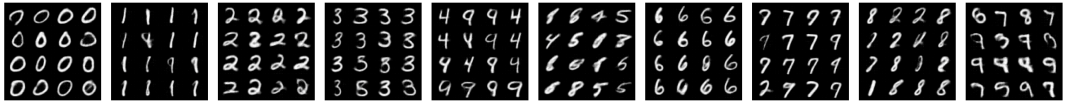
(a) VampPrior



(b) GMVAE



(c) Probabilistic RAE



(d) Deterministic RAE

Figure: Comparisons on conditional digit generation.

Experiments: Conditional Image Generation

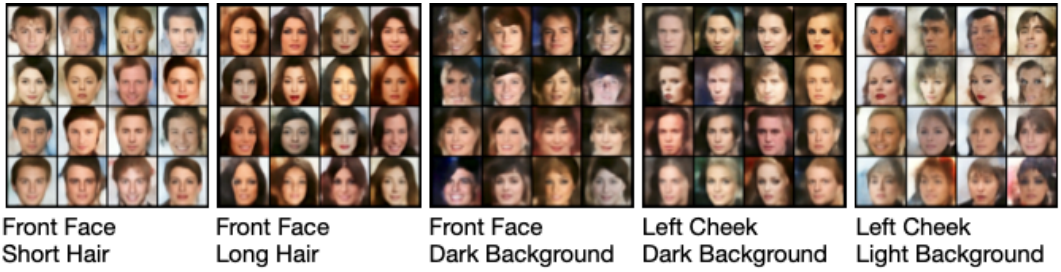
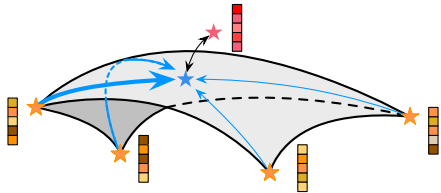


Figure: Deterministic RAE for conditional face generation.

GW-based Graph Representation: GW Factorization

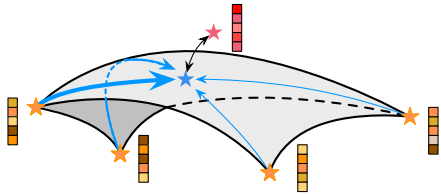


Traditional Factorization Models:

- Given $\{\boldsymbol{y}_1, ..., \boldsymbol{y}_I\}$, we would like to learn the basis $\boldsymbol{A} = [\boldsymbol{a}_1, ..., \boldsymbol{a}_K]$ and the coefficient vector (representation) $\boldsymbol{\lambda}_i$ for each $\boldsymbol{y}_i$.

$$\min_{\{\boldsymbol{A}, \boldsymbol{\lambda}_{1:I}\} \in \Omega} \sum_{i=1}^{I} d_{\text{loss}}(\boldsymbol{A}\boldsymbol{\lambda}_i, \boldsymbol{y}_i). \tag{49}$$

GW-based Graph Representation: GW Factorization



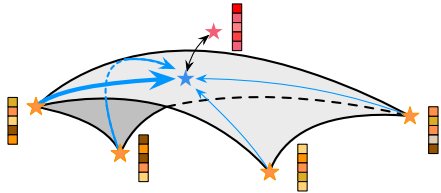
Traditional Factorization Models:

- Given $\{\boldsymbol{y}_1, ..., \boldsymbol{y}_I\}$, we would like to learn the basis $\boldsymbol{A} = [\boldsymbol{a}_1, ..., \boldsymbol{a}_K]$ and the coefficient vector (representation) $\boldsymbol{\lambda}_i$ for each $\boldsymbol{y}_i$.

$$\min_{\{\boldsymbol{A}, \boldsymbol{\lambda}_{1:I}\} \in \Omega} \sum_{i=1}^{I} d_{\text{loss}}(\boldsymbol{A}\boldsymbol{\lambda}_i, \boldsymbol{y}_i). \tag{49}$$

  - PCA: $d_{\text{loss}} = \ell_2$. (MSE)
  - Robust PCA: $d_{\text{loss}} = \ell_1$. (MAE)
  - NMF: $d_{\text{loss}} = \ell_2$, $\Omega =$ Nonnegativeness.
  - LDA: $d_{\text{loss}} =$ KL, $\Omega =$ Simplex.
  - Wasserstein dictionary learning: $d_{\text{loss}} =$ Wasserstein, $\Omega =$ Simplex.

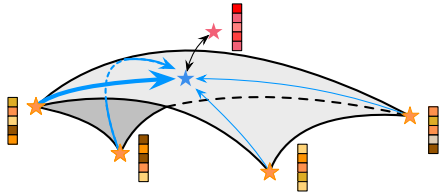
GW-based Graph Representation: GW Factorization



- When $\Omega = \text{Simplex}$, we have

$$\boldsymbol{A}\boldsymbol{\lambda} = \underbrace{\arg\min_{\boldsymbol{y}} \sum_{k=1}^{K} \lambda_k \|\boldsymbol{y} - \boldsymbol{a}_k\|_2^2}_{\text{Euclidean barycenter}} = \boldsymbol{b}(\boldsymbol{A}, \boldsymbol{\lambda}; \ell_2). \tag{50}$$

GW-based Graph Representation: GW Factorization



- When $\Omega = \text{Simplex}$, we have

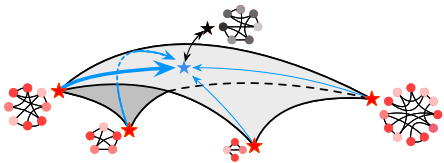
$$\boldsymbol{A}\boldsymbol{\lambda} = \underbrace{\arg\min_{\boldsymbol{y}} \sum_{k=1}^{K} \lambda_k \|\boldsymbol{y} - \boldsymbol{a}_k\|_2^2}_{\text{Euclidean barycenter}} = \boldsymbol{b}(\boldsymbol{A}, \boldsymbol{\lambda}; \ell_2). \tag{50}$$

- Accordingly, a generalized factorization model can be written as

$$\min_{\{\boldsymbol{A}, \boldsymbol{\lambda}_{1:I}\} \in \Omega} \sum_{i=1}^{I} d_{\text{loss}}(\boldsymbol{b}(\boldsymbol{A}, \boldsymbol{\lambda}_i; \underbrace{d_{\text{b}}}_{\text{b's metric}}), \boldsymbol{y}_i). \tag{51}$$

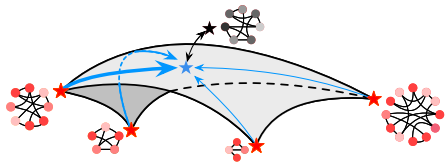
GW-based Graph Representation: GW Factorization

GW factorization [Xu, AAAI 2020]: Learning interpretable factorization model to represent unaligned graphs.



GW-based Graph Representation: GW Factorization

GW factorization [Xu, AAAI 2020]: Learning interpretable factorization model to represent unaligned graphs.



- Estimate each graph by a GW barycenter graph:

$$B_{gw}(\boldsymbol{U}_{1:K}, \boldsymbol{\lambda}) := \arg\min_{B} \sum\nolimits_{k=1}^{K} \lambda_k d_{gw}(B, G_k(\boldsymbol{U}_k)). \tag{52}$$

- $\{G_k(\boldsymbol{U}_k)\}_{k=1}^K$: a set of graph factors.
- $\boldsymbol{\lambda}_{1:I} = \{\boldsymbol{\lambda}_i \in \Delta^{K-1}\}_{i=1}^I$: the coefficients of the graph factors corresponding to $\{G_i\}_{i=1}^I$ (The representations of the observed graphs).

[Xu, AAAI 2020] Xu, H. Gromov-Wasserstein factorization models for graph clustering. AAAI 2020.

GW-based Graph Representation: GW Factorization

Learning task:

$$\min_{\mathbf{1}\geq \boldsymbol{U}_{1:K}\geq \mathbf{0},\ \boldsymbol{\lambda}_{1:I}\in\Delta^{K-1}} \sum_{i=1}^{I} d_{\mathsf{loss}}(B_{gw}(\boldsymbol{U}_{1:K}, \boldsymbol{\lambda}_i), G_i). \tag{53}$$

GW-based Graph Representation: GW Factorization

Learning task:

$$\min_{\mathbf{1}\geq \boldsymbol{U}_{1:K}\geq \mathbf{0},\ \boldsymbol{\lambda}_{1:I}\in\Delta^{K-1}} \sum_{i=1}^{I} d_{\mathsf{loss}}(B_{gw}(\boldsymbol{U}_{1:K}, \boldsymbol{\lambda}_i), G_i). \tag{53}$$

Reparameterize the problem to an unconstrained optimization problem:

$$\min_{\boldsymbol{V}_{1:K},\ \boldsymbol{z}_{1:I}} \sum_{i=1}^{I} d_{\mathsf{loss}}(B_{gw}(\sigma(\boldsymbol{V}_{1:K}), \mathsf{softmax}(\boldsymbol{z}_i)), G_i). \tag{54}$$

GW-based Graph Representation: GW Factorization

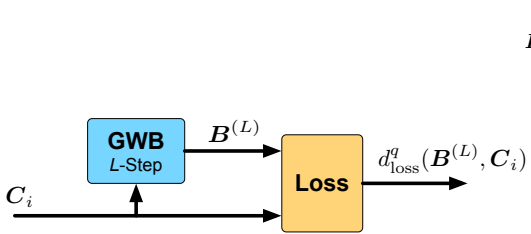
Learning task:

$$\min_{\mathbf{1}\geq \boldsymbol{U}_{1:K}\geq \mathbf{0},\ \boldsymbol{\lambda}_{1:I}\in\Delta^{K-1}} \sum_{i=1}^{I} d_{\text{loss}}(B_{gw}(\boldsymbol{U}_{1:K}, \boldsymbol{\lambda}_i), G_i). \tag{53}$$

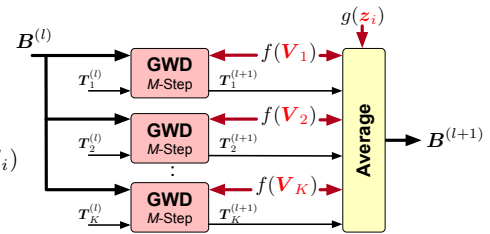
Reparameterize the problem to an unconstrained optimization problem:

$$\min_{\boldsymbol{V}_{1:K},\ \boldsymbol{z}_{1:I}} \sum_{i=1}^{I} d_{\text{loss}}(B_{gw}(\sigma(\boldsymbol{V}_{1:K}), \text{softmax}(\boldsymbol{z}_i)), G_i). \tag{54}$$

(1) Compute optimal transport matrices by GWD modules; (2) Fix the OT matrices and learn the model parameters via SGD:



Overall scheme



The GWB module

GWF v.s. Graph Dictionary Learning (GDL)

GW Factorization

- Learn arbitrarily-sized graph factors, and reconstruct graphs via the weighted GW barycenter of the factors (Nonlinear)
- Unsupervised learning under arbitrary loss functions (MSE, Cross-Entropy, GWD, ...) because of the permutation invariance of GW barycenter.
- High computational complexity

GWF v.s. Graph Dictionary Learning (GDL)

GW Factorization

- Learn arbitrarily-sized graph factors, and reconstruct graphs via the weighted GW barycenter of the factors (Nonlinear)
- Unsupervised learning under arbitrary loss functions (MSE, Cross-Entropy, GWD, ...) because of the permutation invariance of GW barycenter.
- High computational complexity

Graph Dictionary Learning [Vincent, et al., ICML 2021]

- Learn graph factors with fixed sizes, and reconstruct graphs via a linear model:

$$\hat{G} = \sum_{k=1}^{K} \lambda_k \boldsymbol{U}_k \tag{55}$$

- Unsupervised learning, but the loss function has to be GW distance:

$$\min_{\boldsymbol{U}_{1:K},\ \boldsymbol{\lambda}_{1:I}} \sum_{i=1}^{I} d_{gw}(\hat{G}_i, G_i). \tag{56}$$

- Low computational complexity

[Vincent, et al., ICML 2021] Vincent-Cuaz, C., Vayer, T., Flamary, R., Corneli, M., & Courty, N. Online graph dictionary learning. ICML 2021.

GWF v.s. Random-Walk Neural Network (RWNN)

GW Factorization

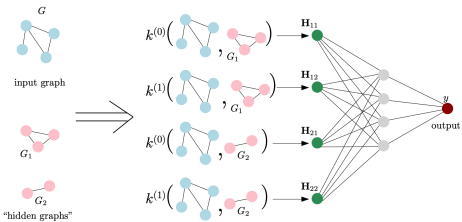
- Consider the GWD between graph factors and observed graphs, and learn the factors unsupervisedly via minimizing reconstruction errors.

GWF v.s. Random-Walk Neural Network (RWNN)

GW Factorization

- Consider the GWD between graph factors and observed graphs, and learn the factors unsupervisedly via minimizing reconstruction errors.

Random-Walk Neural Network in [Nikolentzos, et al., NeurIPS 2020]



The scheme of RWNN shown in [Nikolentzos, et al., NeurIPS 2020]

- Consider the Random-Walk kernel between graph factors and observed graphs, and learn the factors supervisedly driven by graph classification labels.

[Nikolentzos, et al., NeurIPS 2020] Nikolentzos, G., & Vazirgiannis, M. Random walk graph neural networks. NeurIPS 2020.

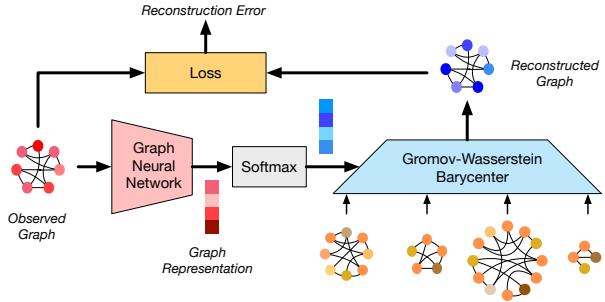
An Inductive Graph Representation Combining GWF with GNN

Achieve an **inductive** GW factorization model by introducing graph neural networks (i.e., GWF-GNN) [Xu, et al, 2022]:

$$\min_{\boldsymbol{V}_{1:K},\, \boldsymbol{\theta}} \sum_{i=1}^{I} d_{gw}(B_{gw}(\sigma(\boldsymbol{V}_{1:K}), \mathsf{softmax}(\mathsf{GNN}_{\theta}(G_i))), G_i). \tag{57}$$

[Xu, et al, 2022] Representing graphs via Gromov-Wasserstein factorization. IEEE TPAMI, 2022 (Accepted)

GWF-GNN v.s. VGAE



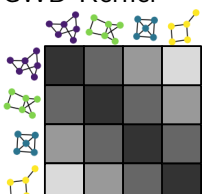
- **GWF-GNN:** A new auto-encoding architecture for learning graph-level representations.
- **VGAE** in [Kipf, et al., 2016]: An auto-encoding architecture for node-level representations.

[Kipf, et al., 2016] Kipf, T. N., & Welling, M. Variational graph auto-encoders. arXiv preprint arXiv:1611.07308, 2016.

Experiments on Graph Clustering

- AIDS: 2,000 compounds active/inactive to anti-HIV
- PROTEIN: 1,113 enzymatic/non-enzymatic proteins

GWD Kernel



GWD+Kmeans

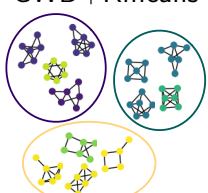
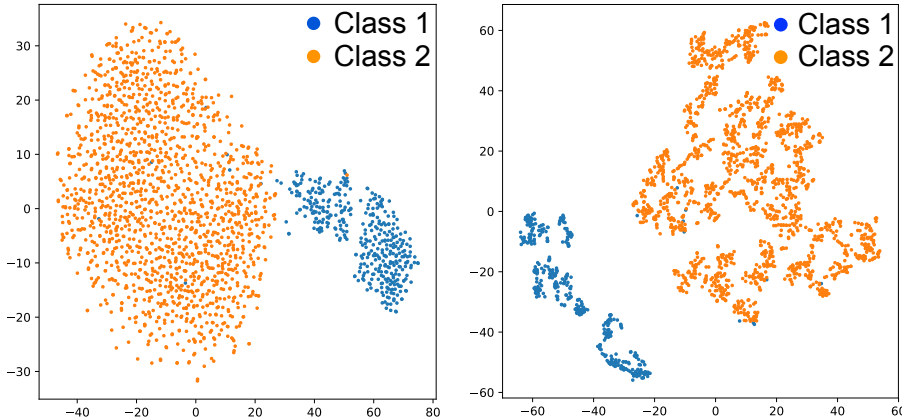


Table: Comparisons on clustering accuracy (%)

Method	AIDS	PROTEIN
GWD Kernel + SC	91.0±0.7	66.4±0.8
GWD + Kmeans	95.2±0.9	64.7±1.1
GWF + Kmeans	**99.5±0.4**	**70.7±0.7**

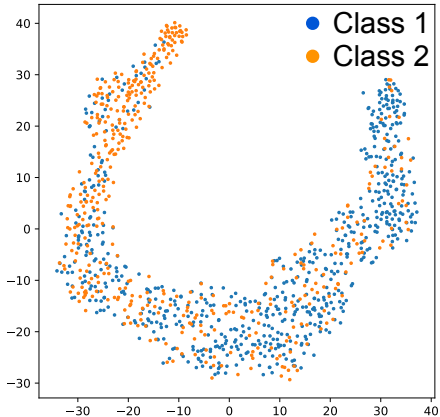
Experiments on Graph Clustering



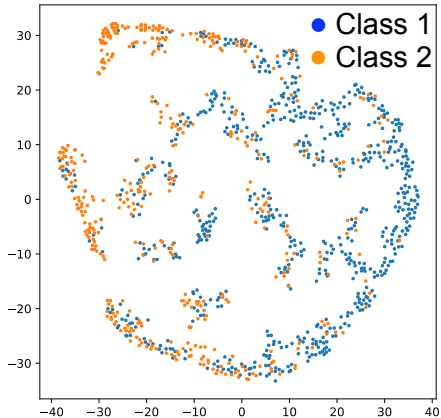
(a) AIDS (GWF) (b) AIDS (GWF-GNN)

Figure: The t-SNE visualizations of $\boldsymbol{z}_{1:I}$ of AIDS dataset derived by the GWF and GWF-GNN.

Experiments on Graph Clustering



(a) PROTEIN (GWF)



(b) PROTEIN (GWF-GNN)

Figure: The t-SNE visualizations of $z_{1:I}$ of Protein dataset derived by the GWF and GWF-GNN.

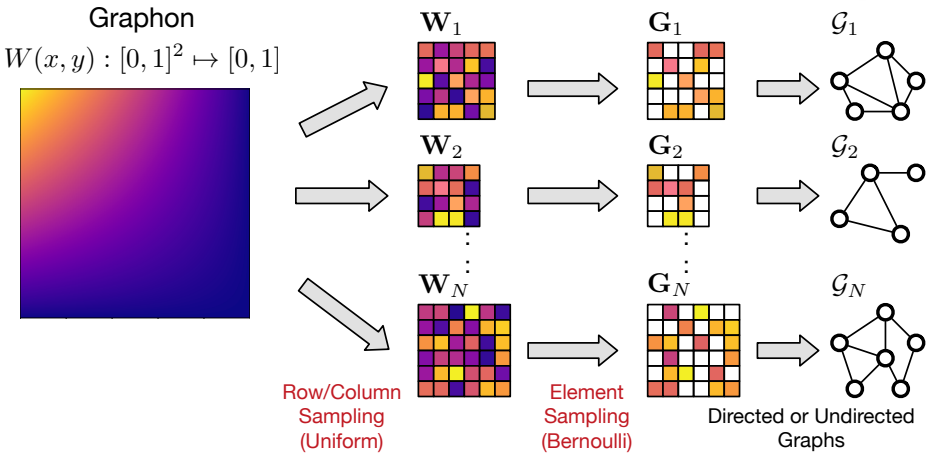
Experiments on Graph Classification

Table: Comparison on classification accuracy (%).

Method Category	Dataset	MUTAG	PTC-MR	IMDB-B	IMDB-M
	# Graphs	188	344	1000	1500
	# Classes	2	2	2	3
	Avg. # Nodes	17.93	14.29	19.77	13
	Avg. # Edges	19.79	14.69	96.53	65.94
Graph kernel dim=512	RW	83.71	57.85	50.68	34.65
	SP	85.22	58.24	55.60	37.99
	GK	81.66	57.26	65.87	43.89
	WL	80.72	57.97	72.30	46.95
	DGK	87.44	60.08	66.96	44.55
	MLG	87.94	63.26	66.55	41.17
Recent Embedding dim=512	node2vec	72.63	58.58	-	-
	sub2vec	61.05	59.99	55.26	36.67
	graph2vec	83.15	60.17	71.1	**50.44**
GWL **dim=15**	GWF	82.93	62.57	63.1	40.67
	GWF-GNN	**90.11**	**63.58**	**73.50**	49.40

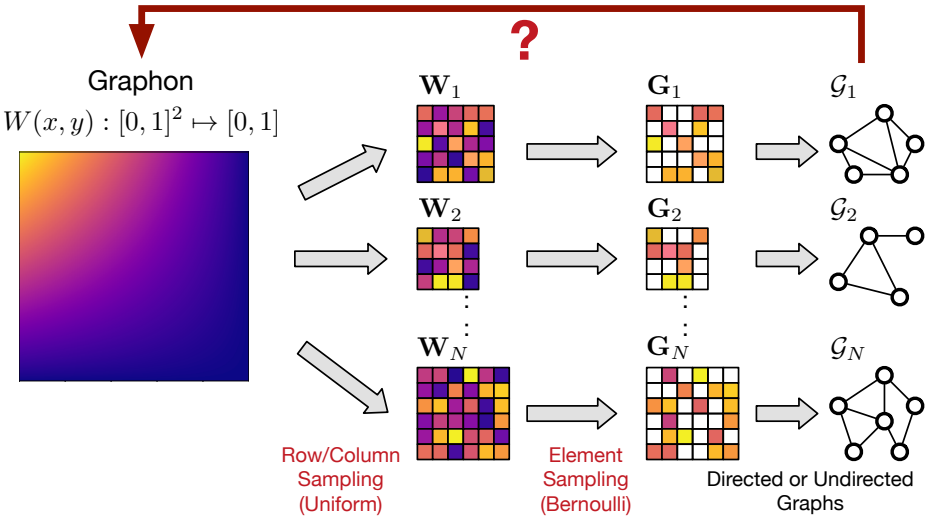
GW-based Graph Generation: GWB-based Graphon Estimation

Graphon: A Nonparametric Graph Generative Model



GW-based Graph Generation: GWB-based Graphon Estimation

Graphon: A Nonparametric Graph Generative Model



GW-based Graph Generation: GWB-based Graphon Estimation

Two learning paradigms:

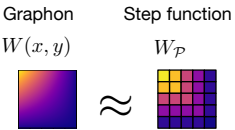
- **Traditional paradigm:** Given a single large-scale graph, estimate a graphon by a step-function and make it as smooth as possible.
  - Collecting and processing a large-scale graph are challenging.
  - The estimation and its smoothness depend on the sorting of nodes (according to their degrees).
- **The proposed paradigm:** Given a set of unaligned but small graphs, estimate a graphon by solving a GW barycenter problem [Xu, et al., AAAI 2021].
  - Reduce the difficulty on data collection and processing.
  - Robust to the challenging cases where the graph nodes are hard to sort.

[Xu, et al., AAAI 2021] Xu, H., Luo, D., Carin, L., & Zha, H. Learning Graphons via Structured Gromov-Wasserstein Barycenters. AAAI 2021.

Oracle Estimator

$$\textbf{Cut norm: } \|W\|_{\square} := \sup_{\mathcal{X},\mathcal{Y}\subset\Omega} \left| \int_{\mathcal{X}\times\mathcal{Y}} W(x,y)dxdy \right|,$$

$$\textbf{Cut distance: } \delta_{\square}(W_1, W_2) := \inf \underbrace{\phi \in \mathcal{S}_{\Omega}}_{\text{Measure-preserved map}} \| \underbrace{W_1 - W_2^{\phi}}_{\text{Residual}} \|_{\square}. \tag{58}$$



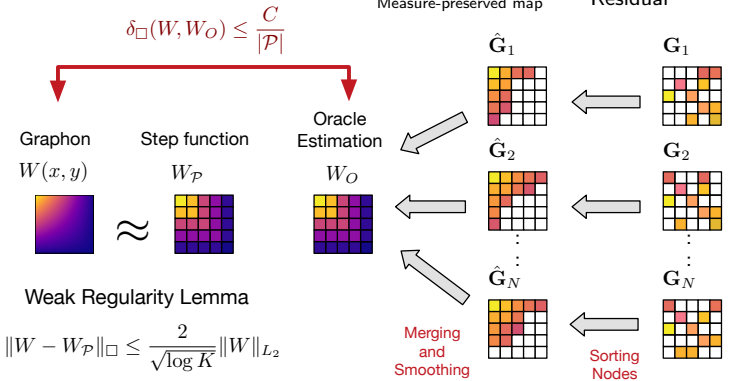
Weak Regularity Lemma

$$\|W - W_{\mathcal{P}}\|_{\square} \leq \frac{2}{\sqrt{\log K}} \|W\|_{L_2}$$

Oracle Estimator

$$\textbf{Cut norm: } \|W\|_{\square} := \sup_{\mathcal{X},\mathcal{Y}\subset\Omega} \left| \int_{\mathcal{X}\times\mathcal{Y}} W(x,y)dxdy \right|,$$

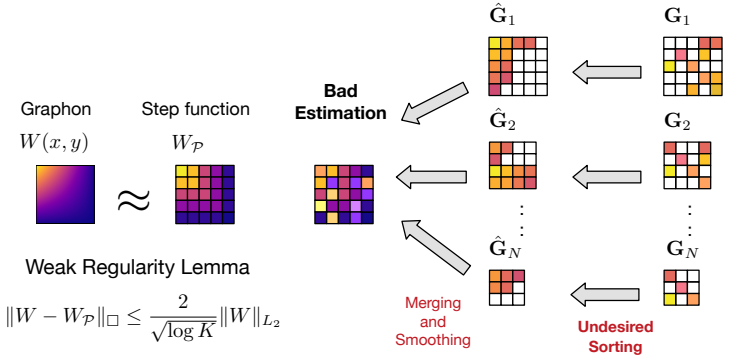
$$\textbf{Cut distance: } \delta_{\square}(W_1, W_2) := \inf_{\underbrace{\phi \in \mathcal{S}_{\Omega}}_{\text{Measure-preserved map}}} \| \underbrace{W_1 - W_2^{\phi}}_{\text{Residual}} \|_{\square}. \tag{58}$$



Oracle Estimator

$$\textbf{Cut norm: } \|W\|_{\square} := \sup_{\mathcal{X},\mathcal{Y}\subset\Omega} \left| \int_{\mathcal{X}\times\mathcal{Y}} W(x,y)dxdy \right|,$$

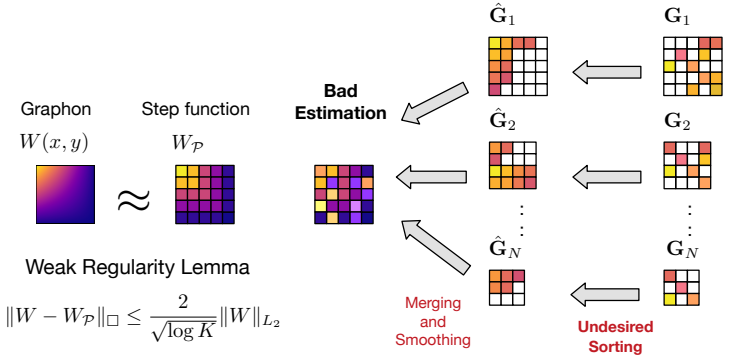
$$\textbf{Cut distance: } \delta_{\square}(W_1, W_2) := \inf_{\underbrace{\phi \in \mathcal{S}_{\Omega}}_{\text{Measure-preserved map}}} \| \underbrace{W_1 - W_2^{\phi}}_{\text{Residual}} \|_{\square}. \tag{58}$$



Oracle Estimator

$$\textbf{Cut norm: } \|W\|_{\square} := \sup_{\mathcal{X},\mathcal{Y}\subset\Omega} \left| \int_{\mathcal{X}\times\mathcal{Y}} W(x,y)dxdy \right|,$$

$$\textbf{Cut distance: } \delta_{\square}(W_1, W_2) := \inf_{\underbrace{\phi \in \mathcal{S}_{\Omega}}_{\text{Measure-preserved map}}} \| \underbrace{W_1 - W_2^{\phi}}_{\text{Residual}} \|_{\square}. \tag{58}$$



Can we achieve graph alignment and graphon learning jointly?

Optimizing An Upper Bound of Estimation Error

$$\begin{aligned}\delta_{\square}(W, W_{\mathcal{P}}) \leq & \delta_{\square}(W, W_O) + \delta_{\square}(W_O, W_{\mathcal{P}}) \qquad \text{(Triangle Inequality)} \\ = & \delta_{\square}(W, W_O) + \delta_{\square}(\frac{1}{M}\sum\nolimits_{m=1}^{M} \hat{G}_m, W_{\mathcal{P}})\end{aligned}$$

Optimizing An Upper Bound of Estimation Error

$$
\begin{aligned}
\delta_{\square}(W, W_{\mathcal{P}}) \leq & \delta_{\square}(W, W_O) + \delta_{\square}(W_O, W_{\mathcal{P}}) && \text{(Triangle Inequality)} \\
= & \delta_{\square}(W, W_O) + \delta_{\square}(\frac{1}{M}\sum\nolimits_{m=1}^{M} \hat{G}_m, W_{\mathcal{P}}) \\
\leq & \delta_{\square}(W, W_O) + \frac{1}{M}\sum\nolimits_{m=1}^{M} \delta_{\square}(\hat{G}_m, W_{\mathcal{P}}) && \text{(Triangle Inequality)}
\end{aligned}
$$

Optimizing An Upper Bound of Estimation Error

$$
\begin{aligned}
\delta_{\square}(W, W_{\mathcal{P}}) \leq & \delta_{\square}(W, W_O) + \delta_{\square}(W_O, W_{\mathcal{P}}) && \text{(Triangle Inequality)} \\
= & \delta_{\square}(W, W_O) + \delta_{\square}(\frac{1}{M}\sum\nolimits_{m=1}^{M} \hat{G}_m, W_{\mathcal{P}}) \\
\leq & \delta_{\square}(W, W_O) + \frac{1}{M}\sum\nolimits_{m=1}^{M} \delta_{\square}(\hat{G}_m, W_{\mathcal{P}}) && \text{(Triangle Inequality)} \\
= & \delta_{\square}(W, W_O) + \frac{1}{M}\sum\nolimits_{m=1}^{M} \delta_{\square}(G_m, W_{\mathcal{P}}) && (\delta_{\square}(G_m, \hat{G}_m) = 0)
\end{aligned}
$$

Optimizing An Upper Bound of Estimation Error

$$
\begin{aligned}
\delta_{\square}(W, W_{\mathcal{P}}) \leq & \delta_{\square}(W, W_O) + \delta_{\square}(W_O, W_{\mathcal{P}}) && \text{(Triangle Inequality)} \\
= & \delta_{\square}(W, W_O) + \delta_{\square}(\frac{1}{M}\sum\nolimits_{m=1}^{M} \hat{G}_m, W_{\mathcal{P}}) \\
\leq & \delta_{\square}(W, W_O) + \frac{1}{M}\sum\nolimits_{m=1}^{M} \delta_{\square}(\hat{G}_m, W_{\mathcal{P}}) && \text{(Triangle Inequality)} \\
= & \delta_{\square}(W, W_O) + \frac{1}{M}\sum\nolimits_{m=1}^{M} \delta_{\square}(G_m, W_{\mathcal{P}}) && (\delta_{\square}(G_m, \hat{G}_m) = 0) \\
\leq & \delta_{\square}(W, W_O) + \frac{1}{M}\sum\nolimits_{m=1}^{M} \delta_1(G_m, W_{\mathcal{P}}). && (\delta_{\square}(W, W') \leq \delta_1(W, W'))
\end{aligned}
\tag{59}
$$

- δ_1 **distance:** $\delta_1(W_1, W_2) := \inf_{\phi \in \mathcal{S}_\Omega} \|W_1 - W_2^\phi\|_{L_1}$.
- **Task:** $\min_{W_{\mathcal{P}} \in [0,1]^{K \times K}} \sum_{m=1}^{M} \delta_1(G_m, W_{\mathcal{P}})$.

Learning Graphons as Gromov-Wasserstein Barycenters

$$\underbrace{\delta_1(W_1, W_2) = d_{\mathbf{gw},\ell_1}(W_1, W_2)}_{\text{[Janson, 2013]}}$$

Learning Graphons as Gromov-Wasserstein Barycenters

$$\underbrace{\delta_1(W_1, W_2) = d_{\mathsf{gw},\ell_1}(W_1, W_2)}_{\text{[Janson, 2013]}} \underbrace{\cong d_{\mathsf{gw},\ell_2}(W_1, W_2)}_{\text{[Mémoli, 2011]}}. \tag{60}$$

Learning Graphons as Gromov-Wasserstein Barycenters

$$\underbrace{\delta_1(W_1, W_2) = d_{\text{gw},\ell_1}(W_1, W_2)}_{\text{[Janson, 2013]}} \underbrace{\cong d_{\text{gw},\ell_2}(W_1, W_2)}_{\text{[Mémoli, 2011]}}. \tag{60}$$

The task becomes a **Gromov-Wasserstein barycenter** (**GWB**) problem:

$$\min_{W_{\mathcal{P}} \in [0,1]^{K \times K}} \sum_{m=1}^{M} d_{\text{gw},\ell_2}(G_m, W_{\mathcal{P}}). \tag{61}$$

where

$$d_{\text{gw},\ell_2}(G_m, W_{\mathcal{P}}) = \min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2)} \underbrace{\sum_{i,i',j,j'} r_{iji'j'} \overbrace{T_{ii'} T_{jj'}}^{p(r)}}_{\mathbb{E}[r]} = \min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2)} -\langle \boldsymbol{A}_1 \boldsymbol{T} \boldsymbol{A}_2^\top, \boldsymbol{T} \rangle. \tag{62}$$

[Janson, 2013] Janson, S. Graphons, cut norm and distance, couplings and rearrangements. New York journal of mathematics, 2013.

[Mémoli, 2011] Mémoli, F. GromovWasserstein distances and the metric approach to object matching. Foundations of computational mathematics, 2011.

Two Variants for Structured GWBs

Can we achieve a smoothed graphon?

Two Variants for Structured GWBs

Can we achieve a smoothed graphon?

- **Smoothed GW Barycenter (SGWB)** [Xu, et al., AAAI 2021]:

$$\min_{W_{\mathcal{P}} \in [0,1]^{K \times K}} \sum_{m=1}^{M} d^2_{\mathsf{gw},2}(G_m, W_{\mathcal{P}}) + \alpha \|\Delta W_{\mathcal{P}}\|_F^2. \tag{63}$$

Two Variants for Structured GWBs

Can we achieve a smoothed graphon?

- **Smoothed GW Barycenter (SGWB)** [Xu, et al., AAAI 2021]:

$$\min_{W_{\mathcal{P}} \in [0,1]^{K \times K}} \sum_{m=1}^{M} d_{\mathsf{gw},2}^2(G_m, W_{\mathcal{P}}) + \alpha \|\Delta W_{\mathcal{P}}\|_F^2. \tag{63}$$

- Similar to classic GWB problem, an alternating optimization strategy works well.

Two Variants for Structured GWBs

Can we achieve a smoothed graphon?

- **Smoothed GW Barycenter (SGWB)** [Xu, et al., AAAI 2021]:

$$\min_{W_{\mathcal{P}} \in [0,1]^{K \times K}} \sum_{m=1}^{M} d^2_{\mathsf{gw},2}(G_m, W_{\mathcal{P}}) + \alpha \|\Delta W_{\mathcal{P}}\|_F^2. \tag{63}$$

- Similar to classic GWB problem, an alternating optimization strategy works well.

When the graphs are from multiple graphons, how to learn the model?

Two Variants for Structured GWBs

Can we achieve a smoothed graphon?

- **Smoothed GW Barycenter (SGWB)** [Xu, et al., AAAI 2021]:

$$\min_{W_{\mathcal{P}} \in [0,1]^{K \times K}} \sum_{m=1}^{M} d^2_{\mathsf{gw},2}(G_m, W_{\mathcal{P}}) + \alpha \|\Delta W_{\mathcal{P}}\|_F^2. \tag{63}$$

- Similar to classic GWB problem, an alternating optimization strategy works well.

When the graphs are from multiple graphons, how to learn the model?

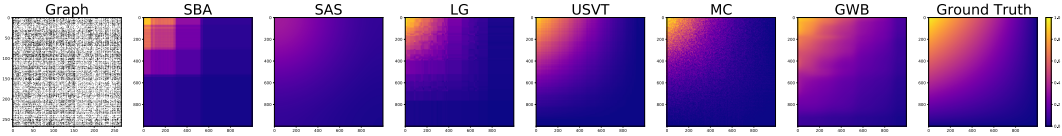
- **A Mixture of GWBs** [Xu, et al., AAAI 2021]:

$$\min_{\{W_c\}_{c=1}^{C}, \boldsymbol{P} \in \Pi(\frac{1}{C}\mathbf{1}_C, \frac{1}{M}\mathbf{1}_M)} \underbrace{\sum_{c=1}^{C} \sum_{m=1}^{M} p_{cm} d^2_{\mathsf{gw},2}(G_m, W_c)}_{\langle \boldsymbol{P},\ \boldsymbol{D}_{\mathsf{gw}} \rangle} \tag{64}$$

- p_{cm}: the probability of generating the m-th graph from the c-th graphon.
- Learn a graphon set to minimize its hierarchical GW distance to the observed graphs.

Experiments

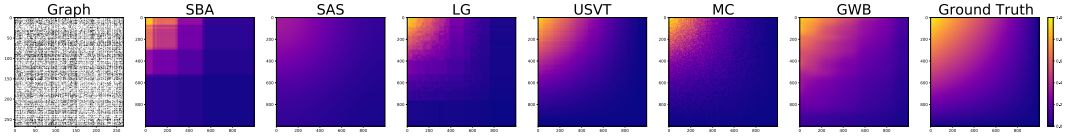
Easy Case: The node degrees provide strong evidence for sorting nodes.



(a) $W(x, y) = xy$ and $0 \leq x, y \leq 1$

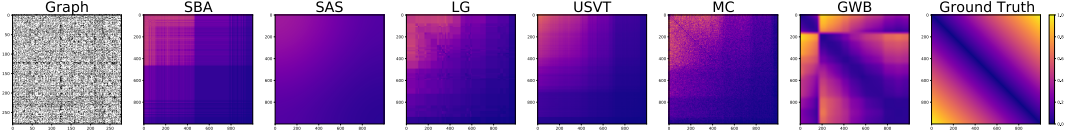
Experiments

Easy Case: The node degrees provide strong evidence for sorting nodes.



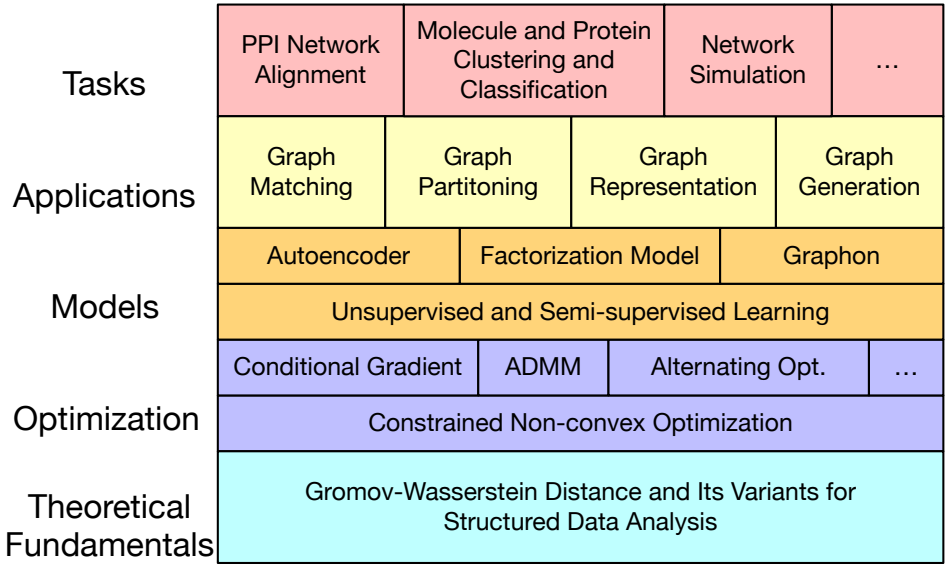
(a) $W(x, y) = xy$ and $0 \leq x, y \leq 1$

Hard Case: The nodes of each graph have comparable degrees.



(b) $W(x, y) = |x - y|$ and $0 \leq x, y \leq 1$

Summary of the GWL framework

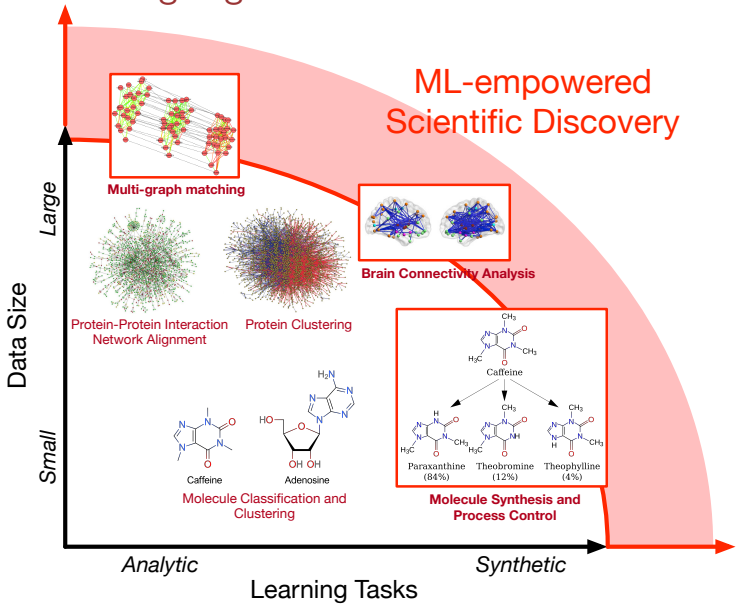


Open Problems and Ongoing Research

- **Gromov-Monge Formulation and Its Applications**
  - Inverse Problems for MM-Spaces [Mémoli, et al., 2021].
  - Reversible Gromov-Monge Sampler [Hur, et al., 2021].
- **Connections to Stochastic Control and Dynamic Systems**
  - OT problems and algorithms are relevant to the Schrödinger bridge problem [Léonard, 2014] (A diffusion model predicting and tracking particles over time.)
  - How to connect GW distance and its algorithms to Stochastic Control?
- **Large-scale Heterogeneous Matching and Learning**
  - Algorithms with lower complexity, higher stability, better approximation.
  - Parallel and distributed learning.

[Mémoli, et al., 2021] Mémoli, F., & Needham, T. Distance distributions and inverse problems for metric measure spaces. arXiv 2021.
[Hur, et al., 2021] Hur, Y., Guo, W., & Liang, T. Reversible Gromov-Monge Sampler for Simulation-Based Inference. arXiv 2021.
[Léonard, 2014] Léonard, C. A survey of the Schrdinger problem and some of its connections with optimal transport. Discrete and Continuous Dynamical Systems-Series A, 2014.

Open Problems and Ongoing Research



Thank you!

https://hongtengxu.github.io

https://github.com/HongtengXu

hongtengxu@ruc.edu.cn

The 1st Workshop on Optimal Transport and Structured Data Modeling (OT-SDM)
https://ot-sdm.github.io (Feb. 28, co-organized with AAAI)