Learning distances for attributed graphs with optimal transport

Pierre Borgnat (ENS Lyon)

The obtention of good distances (or of metric) between objects is an important step for many Machine Learning methods. For structured data such as graphs, Optimal Transport (OT) has received much attention both from the ML community and from the Graph Signal Processing perspective. This especially allowed to address graph classification or graph transfer learning tasks. From our work, I will discuss two novel propositions that combine graphs (structure) and signals (attributes) using Optimal Transport for two situations: i) a new OT-based distance, called Diffusion Wasserstein distance, that generalizes Wasserstein distance to attributed graphs thanks to the application of a graph filter (e.g. heat diffusion) combining structural and feature information, and that both provides a significant performance boost when solving graph domain adaptation tasks; ii) a Metric Learning algorithm for OT-based distance between attributed graphs. Many Metric Learning algorithms have been developed in recent years, yet none for attributed graphs despite the interest of learning distances for better discriminating between graphs. We designed a novel Simple Graph Metric Learning (SGML) model, built from OT yet scalable and with few trainable parameters, that allows us to learn an appropriate metric from a set of labeled (attributed) graphs, so as to improve classification procedures.

Learning Distances for Attributed Graphs with Optimal Transport

Pierre Borgnat
Chair of Equipe Sisyphe, Laboratoire de Physique, CNRS, ENS de Lyon

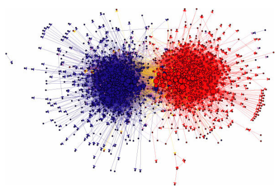
03/2023

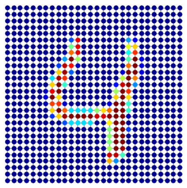
perso.ens-lyon.fr/pierre.borgnat

Work supported by: CNRS, ENS de Lyon, ANR, IDEXLyon, and the **CHIST-Era project: GraphNEX**

Graphs: useful structures for data processing

- Social Networks
- Sensors' data
- Transportations
- Electricity, Water, communications,..



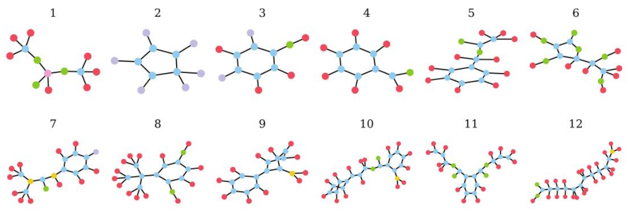


Source: Michael Edwards and Xianghua Xie. **Graph based convolutional neural network.** CoRR, abs/1609.08965, 2016.

- 2D images
- 3D Points clouds
- Other geometric

and/or irregular shapes

- Chemistry
- Physics



Source: Yunsheng Bai, Hao Ding, Yang Qiao, Agustin Marinovic, Ken Gu, Ting Chen,Yizhou Sun, and Wei Wang. **Unsupervised inductive whole-graph embedding by preserving graph proximity.** arXiv preprint arXiv:1904.01098, 2019.

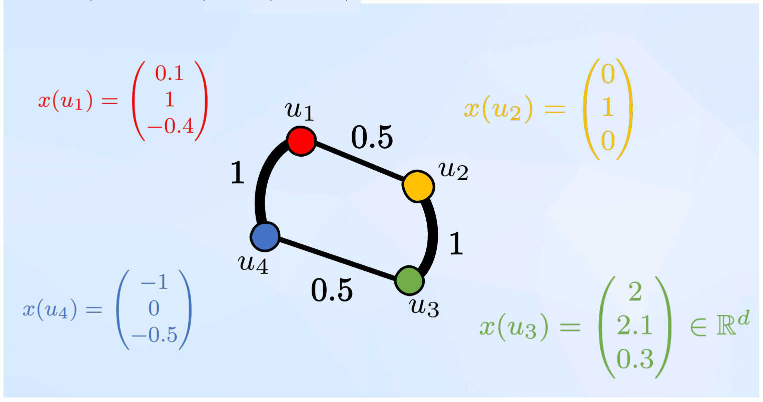
Setting: Attributed Graphs

- In the general case, **nodes and/or edges can carry information**:

❖ Edges = existence of some relationship

❖ Nodes = Attributes, or Features / Signals

$$\mathcal{G} = (V, E, X) = (A, X)$$



$x(u_1) = \begin{pmatrix} 0.1 \\ 1 \\ -0.4 \end{pmatrix}$
$x(u_2) = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$
u_1
0.5
u_2
1
1
u_4
0.5
u_3
$x(u_4) = \begin{pmatrix} -1 \\ 0 \\ -0.5 \end{pmatrix}$
$x(u_3) = \begin{pmatrix} 2 \\ 2.1 \\ 0.3 \end{pmatrix} \in \mathbb{R}^d$

- Adjacency matrix

$$\begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ u_4 \end{array} \begin{array}{c} \begin{array}{cccc} u_1 & u_2 & u_3 & u_4 \end{array} \\ \begin{pmatrix} 0 & 0.5 & 0 & 1 \\ 0.5 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0.5 \\ 1 & 0 & 0.5 & 0 \end{pmatrix} \end{array}$$

$$A \in [0,1]^{|\mathcal{V}| \times |\mathcal{V}|}$$

- Attribute matrix

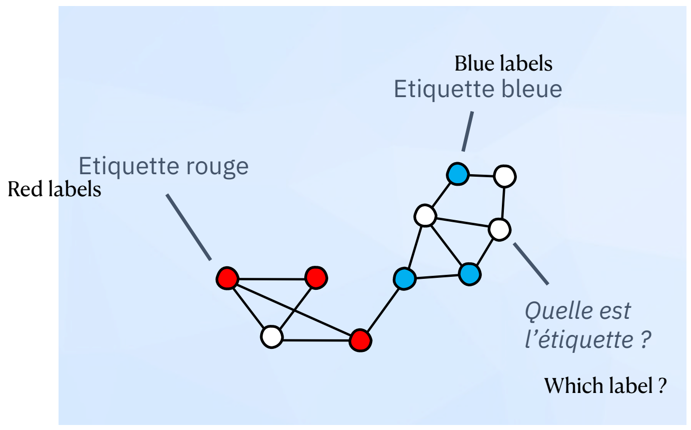
$$\begin{array}{c} u_1 \\ u_2 \\ u_3 \\ u_4 \end{array} \begin{pmatrix} 0.1 & 1 & -0.4 \\ 0 & 1 & 0 \\ 2 & 2.1 & 0 \\ -1 & 0 & -0.5 \end{pmatrix}$$

$$X \in \mathbb{R}^{n \times d}$$

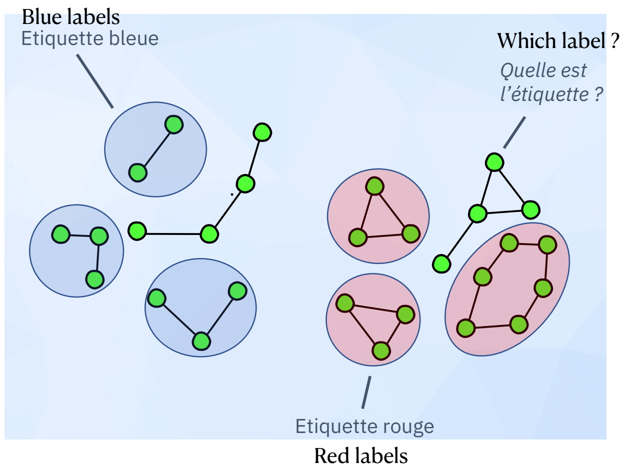
Many Machine Learning tasks for Data on Graphs

Supervised Tasks

- Learn to classify Nodes



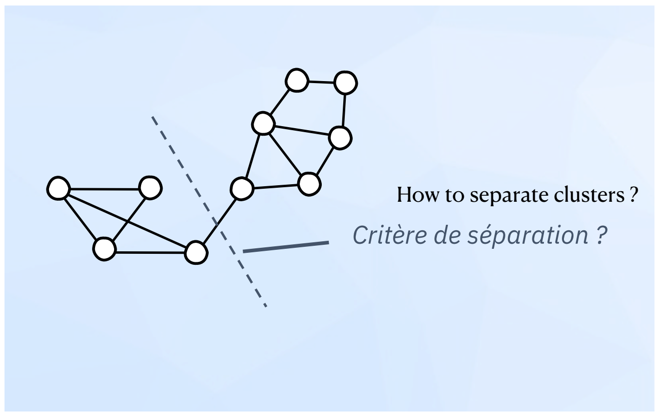
- Learn to classify Graphs



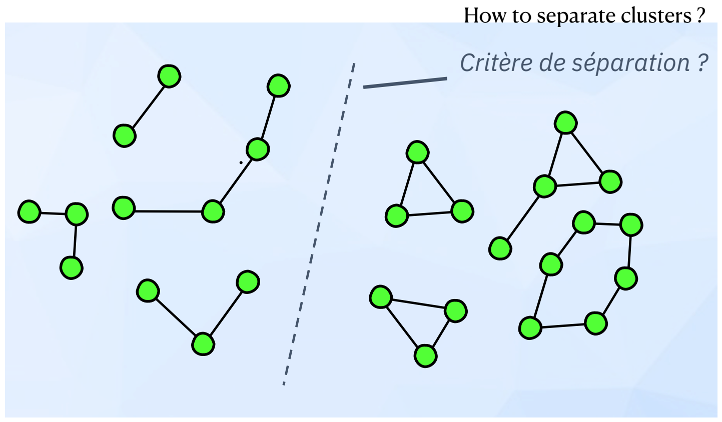
Many Machine Learning tasks for Data on Graphs

Unsupervised Tasks

- Learn to find clusters (or modules, communities,...)



- Learn to cluster collection of graphs



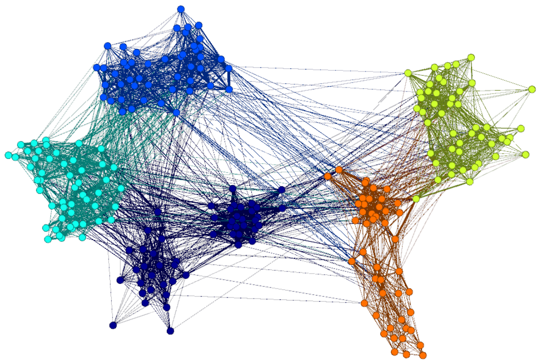
- Note: more general features -> small-world, scale-free, hubs, higher-order interactions...

Many Machine Learning tasks for Data on Graphs

Representation of graphs : Embeddings

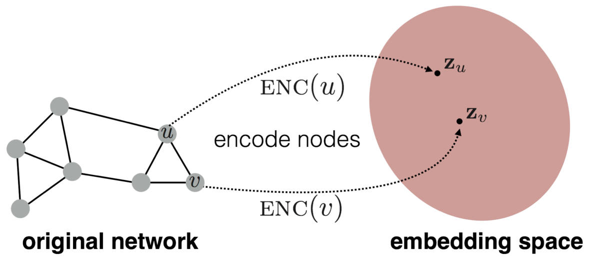
- For Visualisations or low-dim. embeddings

  (Laplacian Maps, LLE, ForceAtlas, t-SNE, UMAP,...)



From [Tremblay & Borgnat, 2014]

- For high-dimensional embeddings



From [Hamilton., "Graph Representation Learning", 2020]

Low Level task: (Graphs) Representation Learning

- **Representation Learning** = discover, or learn, adequate representations for studied data so as to extract information
- Machine Learning in one sentence: build a map from data x to decision y

$$y = \mathscr{F}(x)$$

- Machine Learning in the good all times

hand-crafted using domain knowledge

$$\mathscr{F} = \mathscr{F}_{\text{decision}} \circ \mathscr{F}_{\text{features}}(x)$$

learnt from data

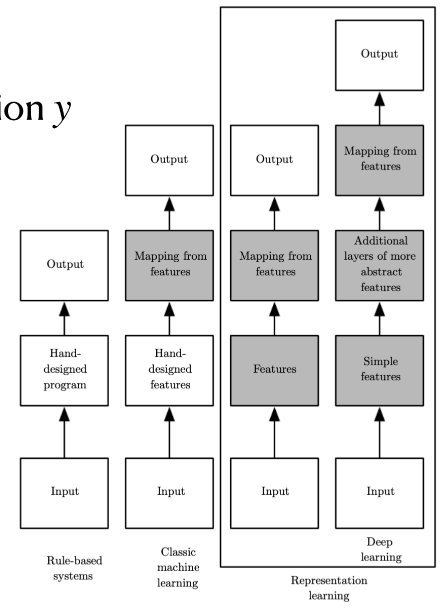
- Machine Learning with Representation Learning / Deep Learning

$$\mathscr{F} = \mathscr{F}_{\text{decision}} \circ \mathscr{F}_{\text{features}} \quad / \quad \mathscr{F} = \mathscr{F}_{\text{decision}} \circ \mathscr{F}_{\text{layer d}} \circ \cdots \mathscr{F}_{\text{layer 1}}$$

All learnt from data

in multiple layers

[From Pierre Vandergheynst' talk, 2021]

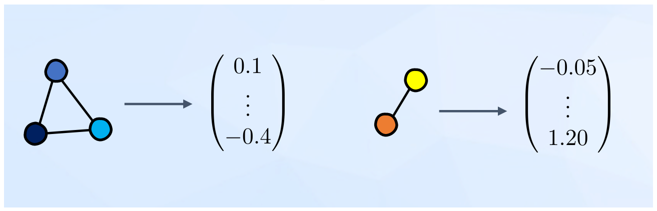


From [Goodfellow et al., "Deep Learning", 2016]

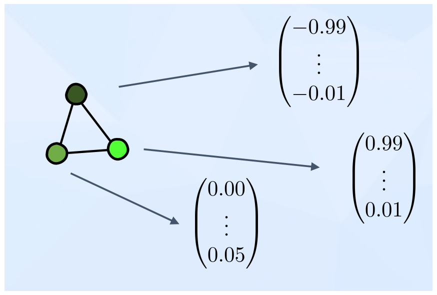
Low Level task: Graphs Representation Learning

- For Graphs, Representation learning can be summarised as:

❖ For Collection of Graphs


$\begin{pmatrix}0.1\\ \vdots\\ -0.4\end{pmatrix}$
$\begin{pmatrix}-0.05\\ \vdots\\ 1.20\end{pmatrix}$

❖ For Nodes in a Graph


$\begin{pmatrix}-0.99\\ \vdots\\ -0.01\end{pmatrix}$
$\begin{pmatrix}0.99\\ \vdots\\ 0.01\end{pmatrix}$
$\begin{pmatrix}0.00\\ \vdots\\ 0.05\end{pmatrix}$

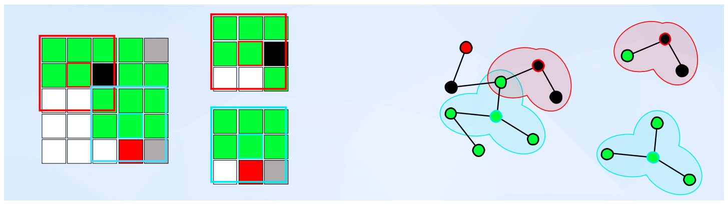
❖ For graphs: often one will agglomerate Nodes representations

- Combine a model of Classification & one of Representation
  - Define a task, a dataset, learn & see
- e.g.: the powerful Graph Neural Networks can do that…

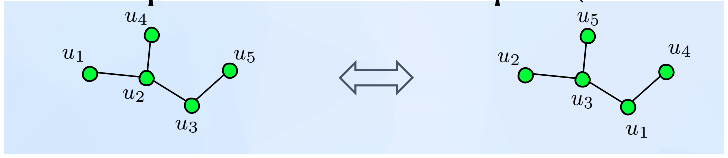
Low Level task: Similarities or distances for Graphs

Some Associated Difficulties

- Node-level: local inhomogeneities in structure => hard to compare two nodes



- Graph-level: possible isomorphism => hard to compare (even to find equality of) two graphs



- Attributed Graphs => how to efficiently **combine structure and attributes** ?

What to do ?

A different Low Level approach: (Dis)-Similarity or Distance-based methods for graphs

- Instead of finding a full representation space, focus on comparing graphs
- Advantages: think of the kernel trick ! **d(x,x')** can be put in many algorithms
  - SVM still have good (better) performance (than representation methods)
  - k-NN are still efficient / scalable approaches (no re-training)
  - …
- Disadvantages:
  - Direct comparisons of Graphs is hard / computationally challenging
  - e.g.: GED (Graph Edit Distance) is NP-hard (or use approximations)
  - …

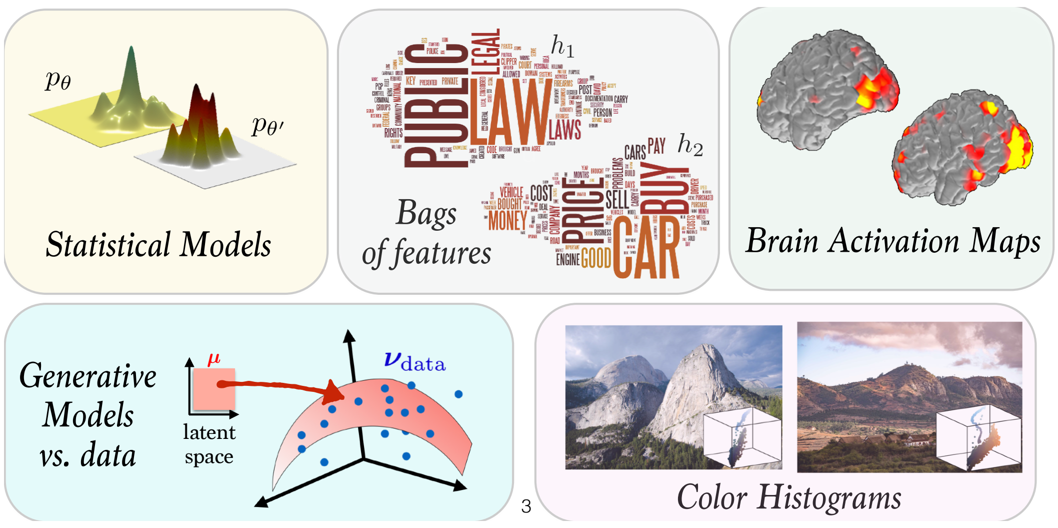
Optimal Transport: a generic tool to probe the geometry of probability measures

- **Optimal Transport**: an approach to **compute a distance between 2 distributions**, while **finding the optimal coupling (or transport plan)** between them
- Put forward in Data Science/Processing & ML since…
  - since ~2010 (at least) ; since ~2000 in image processing (Earth Mover Distance); well before in mathematics (cf. Villani, 2003); in the 70's for the Mallows distance in statistics,…
  - *(see my completely ignored ICASSP paper of 2012: "Using Surrogates and Optimal Transport for Synthesis of Stationary Multivariate Series […]")* (Title way too long!)
- cf. "Computational Optimal Transport" (G. Peyré & M. Cuturi), 2019

  https://arxiv.org/abs/1803.00567v4
- cf. Cuturi & Salomon "A primer on Optimal Transport", NIPS 2017 Tutorial

  https://optimaltransport.github.io/slides/ (and other resources)

Optimal Transport: a generic tool to probe the geometry of probability measures

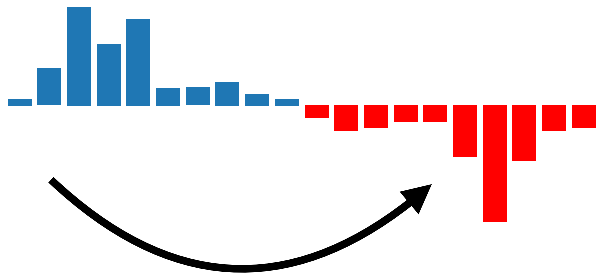


- from Cuturi & Salomon "A primer on Optimal Transport", NIPS 2017 Tutorial

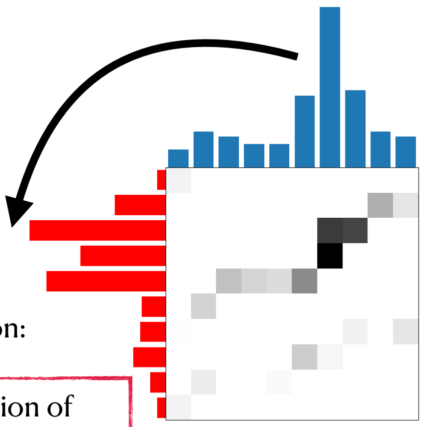
Optimal Transport for distributions

- from "Computational Optimal Transport" (G. Peyré & M. Cuturi), 2019

  https://arxiv.org/abs/1803.00567v4



Problem of Monge : « Mémoire sur la théorie des déblais et des remblais », 1776



One solution:

With relaxation of Kantorovich

Optimal Transport for distributions

- **Optimal Transport:** Consider two finite sets $\mathbb{X} = \{\mathbf{x}_i\}_{i=1}^{|\mathbb{X}|} \in \mathbb{R}^{q \times |\mathbb{X}|}$ and $\mathbb{X}'$ and two distributions on these $\mu = \sum_{\mathbf{x}_i \in \mathbb{X}} a_i \delta_{\mathbf{x}_i}$ and $\nu = \sum_{\mathbf{x}'_i \in \mathbb{X}'} b_i \delta_{\mathbf{x}'_i}$ with $a_i \geq 0, b_i \geq 0$ and $\sum_{i=1}^{n} a_i = 1, \sum_{i=1}^{n'} b_i = 1$
- Given a cost function $c : \mathbb{R}^q \times \mathbb{R}^q \to \mathbb{R}_+$, one builds the **2-Wasserstein distance** $\mathscr{W}_2$ as:

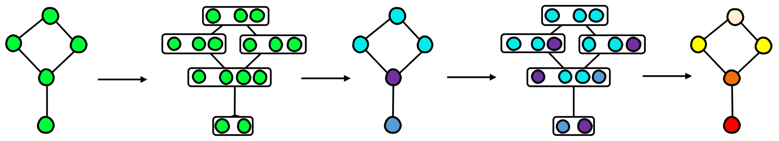
$$\mathscr{W}_2(\mu, \nu) = \inf_{\pi_{i,j} \in \Pi_{a,b}} \left(\sum_{i,j=1}^{n,n'} \pi_{i,j} c(\mathbf{x}_i, \mathbf{x}'_j)^2 \right)^{\frac{1}{2}}$$

where $\Pi_{a,b}$ is the set of joint distributions on $\mathbb{X} \times \mathbb{X}'$

whose marginals are the distributions $\mu = \sum_{\mathbf{x}'_i \in \mathbb{X}'} \pi(\cdot, \mathbf{x}'_i)$ and $\nu = \sum_{\mathbf{x}_i \in \mathbb{X}} \pi(\mathbf{x}_i, \cdot)$

Optimal Transport for Graphs

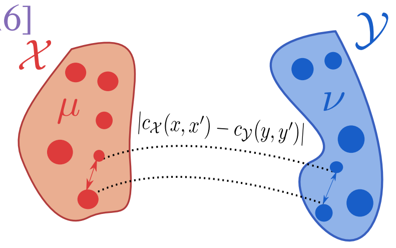
- For Graphs: one has to **Associate a distribution to a graph**
  - A first solution: rely on the the Weisfeiler-Lehman test
  - cf. [Togninalli et al., "Wasserstein Weisfeiler-Lehman graph kernels" NeurIPS 2019]



- A 2nd solution: **Comparison through probabilistic models of graph signals**
  - ["Graph Optimal Transport", H. Maretic et al. NeuRIPS 2019]
  - for a graph $\mathscr{G}$ with Laplacian L, one considers: $x \sim \nu^{\mathscr{G}} = \mathscr{N}(0, L^{\dagger})$
  - then: compute the 2-Wasserstein distance between Gaussian signals
  - allows graph alignment, gives a structurally-meaningful graph distance,...

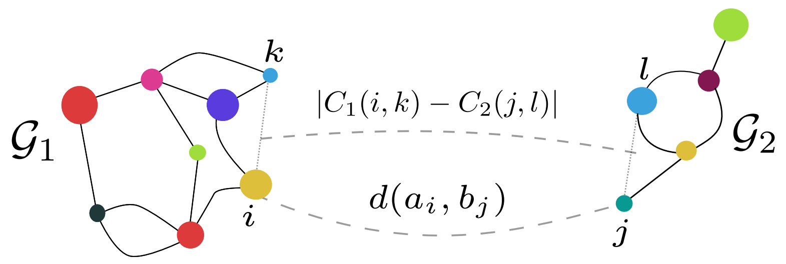
Optimal Transport for ~~Graphs or~~ Attributed Graphs

- A third solution: **The Gromov-Wasserstein distance**
  - [Mémoli, Found. Comp. Math. 2011; Peyré, Cuturi, Solomon, ICML 2016]
  - structures are compared through their pairwise distances
  - cf. also N. Courty, R. Flamary, T. Vayer [PhD 2020]



- One can then **combine Attributes and Gromov-Wasserstein** characterisation of graphs

  "Fused Gromov-Wasserstein distance" [Vayer et al., ICML 2019]

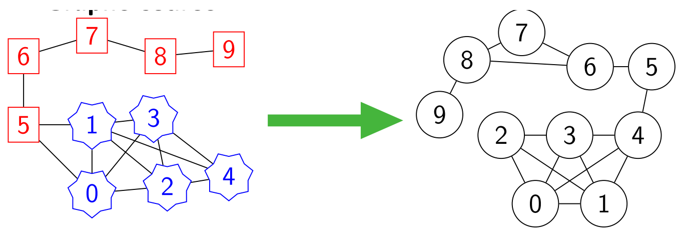


-

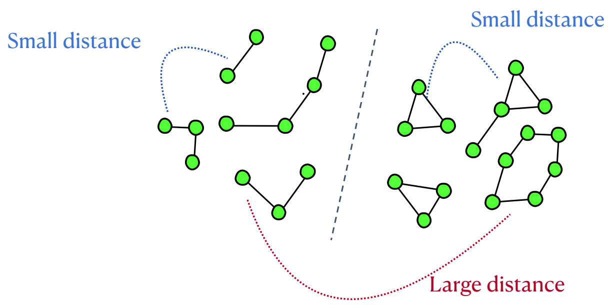
OT-based methods for Attributed Graphs

Some Recent examples from our works

- How to **combine Structures and Attributes to define a distance,** then solve some Domain Adaptation problem ? Our proposition : **Graph Diffusion Wasserstein Distance**
  [A. Barbe, M. Sebban, P. Gonçalves, **P. Borgnat**, R. Gribonval, T. Vayer, ECML-PKDD 2020 ; ICTAI 2021 ; GRETSI 2019]



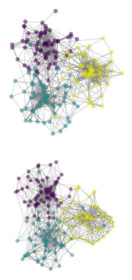
- How to **learn distances** between **Attributed Graphs** ? Our contribution: **Scalable Metric Learning for Graphs**
  [Y. Kaloga, P. Borgnat, A. Habrard, LoG 2022]



Graph Diffusion Wasserstein Distances & Application to Domain Adaptation for Graphs

From Amélie Barbe PhD thesis (12/2021) ; ECML-PKDD 2020 ; ICTAI (2021) ; GRETSI (2019)

Joint work with Marc Sebban (LabHC; Saint-Etienne) ; Rémi Gribonval, Paulo Gonçalves, and Titouan Vayer (LIP, Inria, ENS de Lyon)



$$\begin{tikzcd}
\mathrm{X}^s \arrow[r, "\exp(-\tau^s \mathrm{L}^s)\cdot"] & \tilde{\mathrm{X}}^s \arrow[dr] & & \\
& & \tilde{M} \arrow[r, "\min\limits_{\gamma \in \Pi(a,b)} \left\{ \langle \gamma, \tilde{M}^p \rangle_F \right\}"] & \mathrm{DW}_p^p(U^s, U^t) \\
\mathrm{X}^t \arrow[r, "\exp(-\tau^t \mathrm{L}^t)\cdot"] & \tilde{\mathrm{X}}^t \arrow[ur] & &
\end{tikzcd}$$

Optimal Transport for Attributed Graphs

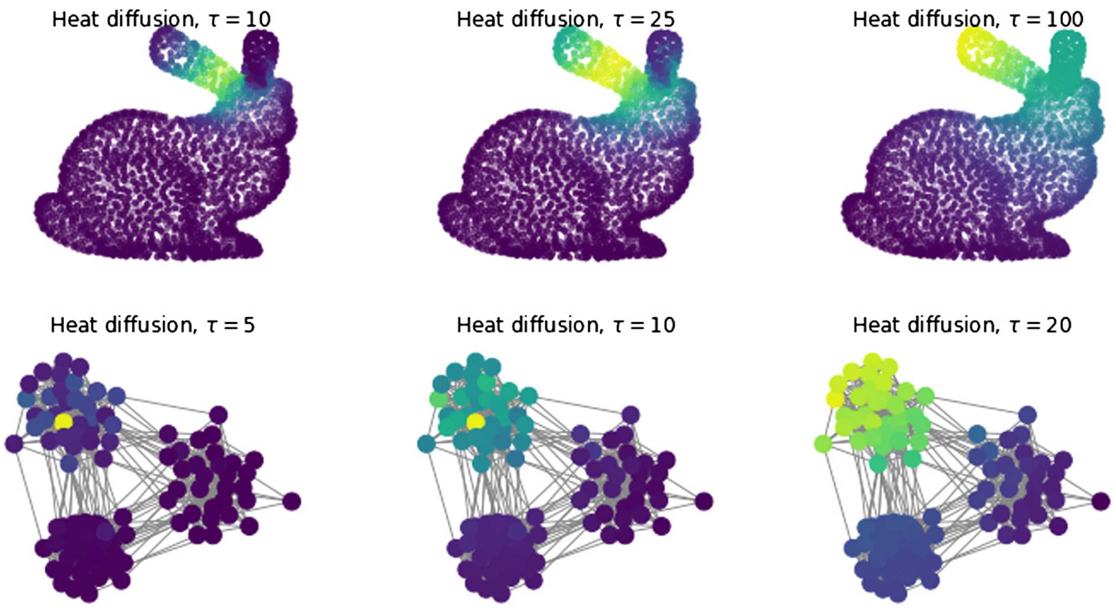
- A different way to **combine Attributes and Structure of Graphs** is to begin first by **processing the Attributes according to the Structure of the graph**

  => This is exactly what **Graph Signal Processing** is studying since ~2010

  see from [Shuman et al., SP Mag 2013] to [Ortega, CUP, 2022]

- More precisely, given a signal x and a graph $\mathcal{G}$:
  - Adjacency matrix A, degree matrix $D = \mathrm{diag}(A \cdot \mathbf{1})$, Laplacian $L = D - A$
  - The "processing" (filtering) of x through $\mathcal{G}$ has the form: $\tilde{x} = f(L) \cdot x$
- Example of useful filter: the **heat diffusion**
  - A good model of graph signals [Thanou, Dong, Kressner, Frossard, 2017]
  - Characterizes some structure of the graphs, e.g. [Ricaud, Borgnat, et al. CR Phys., 2019]

Graph Signal Processing: Heat Diffusion



- from [Ricaud, Borgnat, Tremblay, Gonçalves, Vandergheynst. CR Phys., 2019]
  "Fourier could be a data scientists: from Graph Fourier transform to signal processing on graphs"

Graph Signal Processing: distance from Heat Diffusion

- from [Hammond, Gur, Johnson, GlobalSIP 2013] "**GRAPH DIFFUSION DISTANCE**: A DIFFERENCE MEASURE FOR WEIGHTED GRAPHS BASED ON THE GRAPH LAPLACIAN EXPONENTIAL KERNEL" (Title way too long!)
- They define a **Diffusion distance between graphs** having the same number of nodes

$$\begin{aligned}\xi(A_1, A_2; t) &= \sum_{i,j} ((\exp(-tL_1))_{i,j} - (\exp(-tL_2))_{i,j})^2 \\ &= ||\exp(-tL_1) - \exp(-tL_2)||_F^2 \qquad (2)\end{aligned}$$

$$\boxed{d_{gdd}(A_1, A_2) = \max_t \sqrt{\xi(A_1, A_2; t)}.}$$

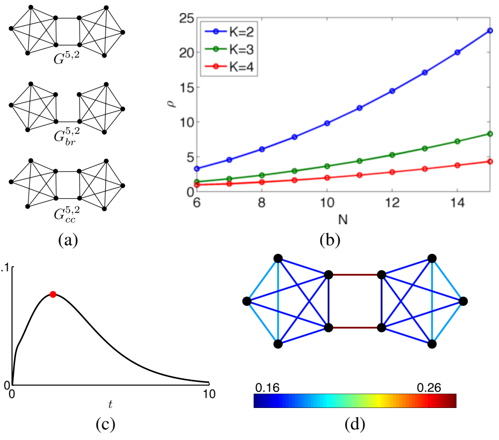
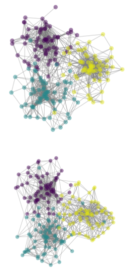


Fig. 1. (a) Barbell graph, and single-edge perturbations, for $N = 5$, $K = 2$. (b) Plot of ratio $d_{gdd}(G^{N,2}, G^{N,2}_{br})/d_{gdd}(G^{N,2}, G^{N,2}_{cc})$ vs N. (c) Plot of $\xi(t)$ for $A_1 = G^{5,2}$, $A_2 = G^{5,2}_{cc}$, red dot indicates maximum, corresponding to $d_{gdd}(A_1, A_2)^2$. (d) Values of normalized edge deletion perturbation, on edges of $G^{5,2}$.

Optimal Transport and Graph Signal Processing for Attributed Graphs

- We can leverage (**combine**) all that: **OptTr ; Diff distance ; GSP** (process signals by L)
- We generalize the previous ideas, and we consider:
  - two graphs of sizes n and m and their associated Laplacians: $\mathbf{L}^s$ and $\mathbf{L}^t$
  - the features of these *source* and *target* graphs: $\mathbf{X} \in \mathbb{R}^{m \times r}$; $\mathbf{Y} \in \mathbb{R}^{n \times r}$
  - a cost function between features: $M(\mathbf{X}, \mathbf{Y}) = [d(x_i, y_j)]$ for any $\mathbf{X} \in \mathbb{R}^{m \times r}$; $\mathbf{Y} \in \mathbb{R}^{n \times r}$
  - the diffused features: $\tilde{\mathbf{X}} = \exp(-\tau^s \mathbf{L}^s) \cdot \mathbf{X}$ and $\tilde{\mathbf{Y}} = \exp(-\tau^t \mathbf{L}^t) \cdot \mathbf{Y}$



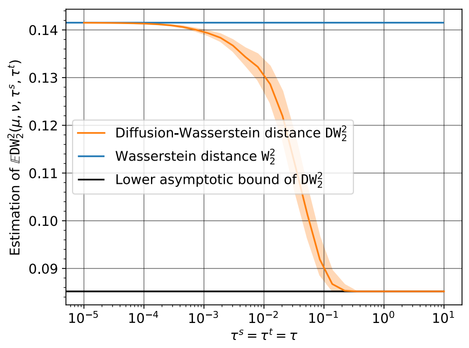
$$\begin{tikzcd}
X^s \arrow[r, "\exp(-\tau^s L^s)\cdot"] & \tilde{X}^s \arrow[dr] & & \\
& & \tilde{M} \arrow[r, "\min_{\gamma \in \Pi(a,b)} \left\{ \langle \gamma, \tilde{M}^p \rangle_F \right\}"] & \mathrm{DW}_p^p(U^s, U^t) \\
X^t \arrow[r, "\exp(-\tau^t L^t)\cdot"] & \tilde{X}^t \arrow[ur] & &
\end{tikzcd}$$

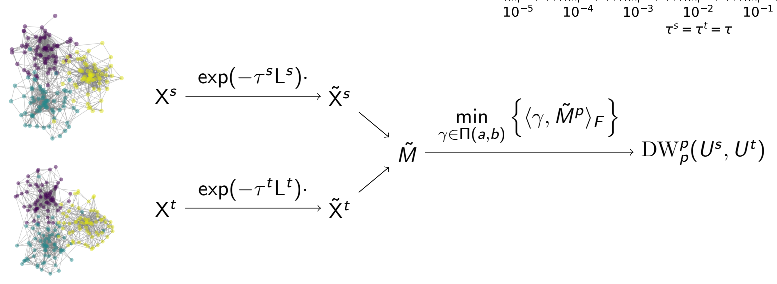
The Diffusion Wasserstein Distances for Attributed Graphs

- Then, we define it as:

$$\boxed{\mathtt{DW}_p^p(\mu, \nu \mid \tau^s, \tau^t) = \min_{\gamma \in \Pi(a,b)} \langle \gamma, \tilde{M}^p \rangle.}$$

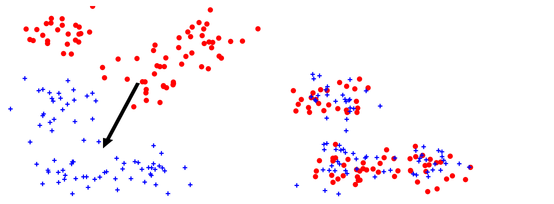
- **Theoretically**, it has good properties:
  - it is a distance
  - we have bounds for small and large τ
  - it's efficient to be computed, more than Fused GW



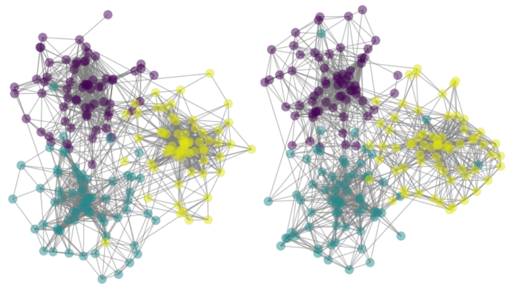

$X^s \xrightarrow{\exp(-\tau^s L^s)\cdot} \tilde{X}^s$
$X^t \xrightarrow{\exp(-\tau^t L^t)\cdot} \tilde{X}^t$
$\tilde{M} \xrightarrow{\min_{\gamma \in \Pi(a,b)} \left\{ \langle \gamma, \tilde{M}^p \rangle_F \right\}} \mathrm{DW}_p^p(U^s, U^t)$

The Diffusion Wasserstein Distances for Attributed Graphs

$$\boxed{\mathtt{DW}_p^p(\mu, \nu \mid \tau^s, \tau^t) = \min_{\gamma \in \Pi(a,b)} \langle \gamma, \tilde{M}^p \rangle.}$$

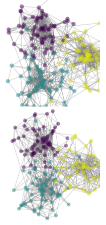

(a) Distributions before alignment. (b) Distributions after alignment.

- **Experimentally**, it works well: the task for comparison is **Domain Adaptation**
  - by itself a cheap way for **DA** on Attr. Graphs



  - can be combined with Fused GW, for an even better

DifFused GW distance, which has best perf. !



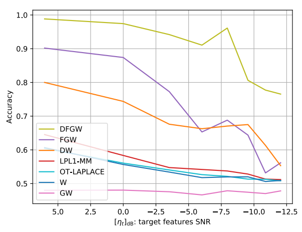
$$\begin{tikzcd} X^s \arrow[r, "\exp(-\tau^s L^s)\cdot"] & \tilde{X}^s \arrow[dr] & & \\ & & \tilde{M} \arrow[r, "\min_{\gamma \in \Pi(a,b)} \left\{ \langle \gamma, \tilde{M}^p \rangle_F \right\}"] & \mathrm{DW}_p^p(U^s, U^t) \\ X^t \arrow[r, "\exp(-\tau^t L^t)\cdot"] & \tilde{X}^t \arrow[ur] & & \end{tikzcd}$$

The Diffusion Wasserstein Distances for Attributed Graphs

$$\boxed{\mathtt{DW}_p^p(\mu,\nu \mid \tau^s,\tau^t) = \min_{\gamma\in\Pi(a,b)} \langle \gamma, \tilde{M}^p\rangle.}$$

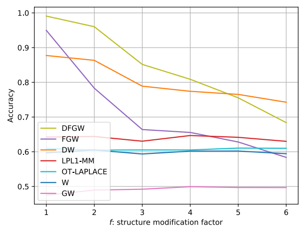
- **Experimentally**, it works well: the task for comparison is **Domain Adaptation**

(a)



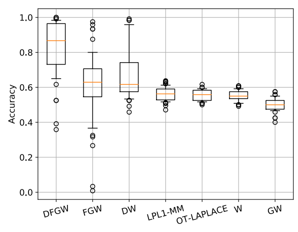
$[\eta^t]_{\text{dB}}$	6	3	0	-3	-6	-9	-12
α (FGW)	0.4	0.6	0.6	0.6	0.7	0.7	0.6
α (DFGW)	0.4	0.6	0.6	0.7	0.7	0.6	0.7

(b)

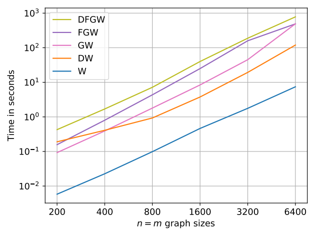


f	1	2	3	4	6
α (FGW)	0.63	0.63	0.45	0.39	0.54
α (DFGW)	0.64	0.56	0.66	0.62	0.46

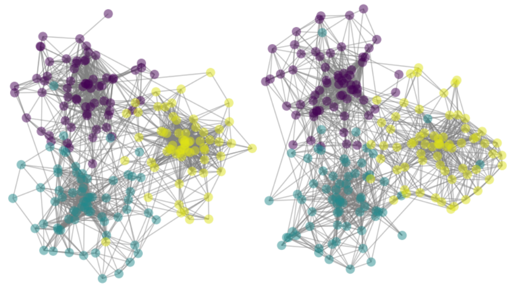
(c)

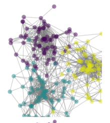


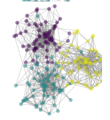
(d)



from [Barbe et al., ECML-PKDD 2020]





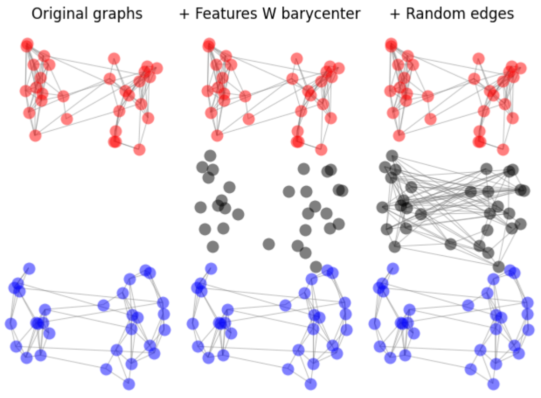


$$\begin{tikzcd}
X^s \arrow[r, "\exp(-\tau^s L^s)\cdot"] & \tilde{X}^s \arrow[dr] & & \\
 & & \tilde{M} \arrow[r, "\min_{\gamma\in\Pi(a,b)} \left\{\langle\gamma, \tilde{M}^p\rangle_F\right\}"] & \mathrm{DW}_p^p(U^s, U^t) \\
X^t \arrow[r, "\exp(-\tau^t L^t)\cdot"] & \tilde{X}^t \arrow[ur] & &
\end{tikzcd}$$

The Diffusion Wasserstein Distances for Attributed Graphs, in action

- How to **set diffusion parameters** τ ? For **unsupervised DA** !
- Use an ER random graph and features as Wasserstein barycenter as an impostor:

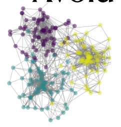
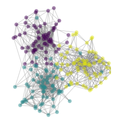
$$X^0 = \underset{X\in\mathbb{R}^{i\times r}}{\operatorname{argmin}} \left\{ \frac{1}{2} \left(\mathtt{W}(X^s, X) + \mathtt{W}(X^t, X) \right) \right\}.$$



- And a triplet loss to be optimized for τ:

$$\boxed{\begin{aligned} \tau^* &= \underset{\tau\geq 0}{\operatorname{argmin}} \left\{ \mathcal{L}(\tau) \right\}, \quad \text{with} && (10)\\ \mathcal{L}(\tau) &= \mathtt{DW}_p(\mathcal{G}^s, \mathcal{G}^t \mid \tau) - \left(\mathtt{DW}_p(\mathcal{G}^s, \mathcal{G}^0 \mid \tau) + \mathtt{DW}_p(\mathcal{G}^t, \mathcal{G}^0 \mid \tau) \right). && (11) \end{aligned}}$$

- Avoid the use of Circular Validation for DA

$$\begin{tikzcd} X^s \arrow[r, "\exp(-\tau^s L^s)\cdot"] & \tilde{X}^s \arrow[dr] & & \\ & & \tilde{M} \arrow[r, "\underset{\gamma\in\Pi(a,b)}{\min}\left\{\langle\gamma, \tilde{M}^p\rangle_F\right\}"] & \mathrm{DW}_p^p(U^s, U^t) \\ X^t \arrow[r, "\exp(-\tau^t L^t)\cdot"] & \tilde{X}^t \arrow[ur] & & \end{tikzcd}$$

from [Barbe et al., ICTAI 2021]

The Diffusion Wasserstein Distances for Attributed Graphs, in action

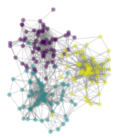
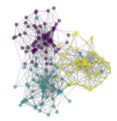
$$\mathtt{DW}_p^p(\mu, \nu \mid \tau^s, \tau^t) = \min_{\gamma \in \Pi(a,b)} \langle \gamma, \tilde{M}^p \rangle.$$

$$\tau^* = \operatorname*{argmin}_{\tau \geq 0} \{\mathcal{L}(\tau)\}, \text{ with} \tag{10}$$

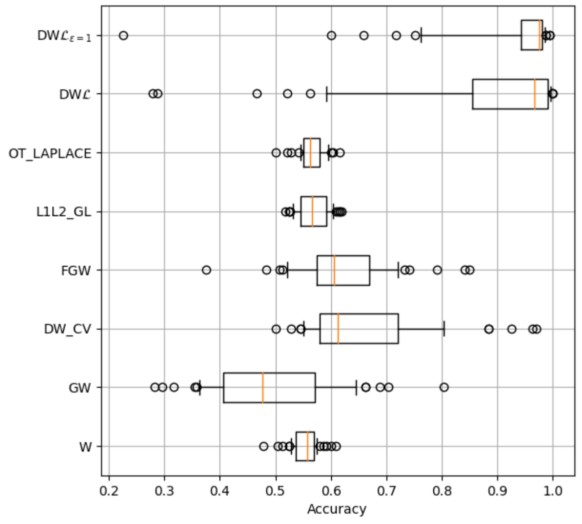
$$\mathcal{L}(\tau) = \mathtt{DW}_p(\mathcal{G}^s, \mathcal{G}^t \mid \tau) - \left(\mathtt{DW}_p(\mathcal{G}^s, \mathcal{G}^0 \mid \tau) + \mathtt{DW}_p(\mathcal{G}^t, \mathcal{G}^0 \mid \tau)\right). \tag{11}$$

- Impostor + Triplet loss = **set the diffusion parameter τ** !
- No Circular Validation => more stability, better perf.

- **Take-Home message :** GSP + OT works very well
  - or even : GSP + ML rocks for graphs learning!

$$\begin{tikzcd} X^s \arrow[r, "\exp(-\tau^s L^s)\cdot"] & \tilde{X}^s \arrow[dr] & & \\ & & \tilde{M} \arrow[r, "\min_{\gamma \in \Pi(a,b)} \left\{\langle \gamma, \tilde{M}^p \rangle_F\right\}"] & \mathrm{DW}_p^p(U^s, U^t) \\ X^t \arrow[r, "\exp(-\tau^t L^t)\cdot"] & \tilde{X}^t \arrow[ur] & & \end{tikzcd}$$



from [Barbe et al., ICTAI 2021] Fig. 6: Median, quartile and decile accuracy of various OT methods on the task of transferring the labels of $\mathcal{G}^s$ to $\mathcal{G}^t$.

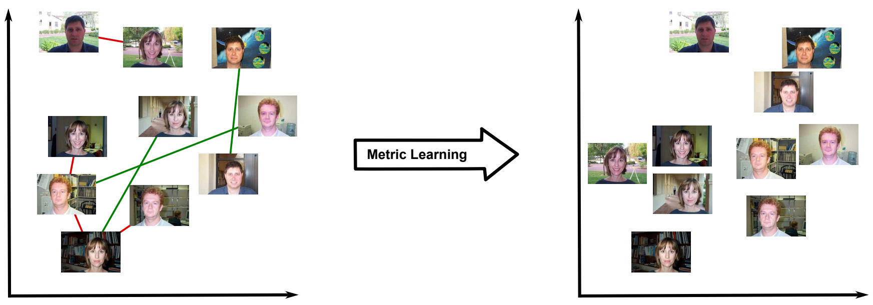
Another take at the low-level task: compute distances

- Why ? Distances are at the input of many (many!) methods

  "Real" distances between graphs are often hard to compute (G. Edit Distance),

  or can ignore some aspects (e.g. spectral distances),

  and usually forget about attributes
- What for ? Parametric distances allow for **Metric Learning**
- cf. Tutorial on Metric Learning (A. Bellet), 2013 & https://arxiv.org/abs/1306.6709



Metric Learning for Attributed Graphs = Leveraging the structure

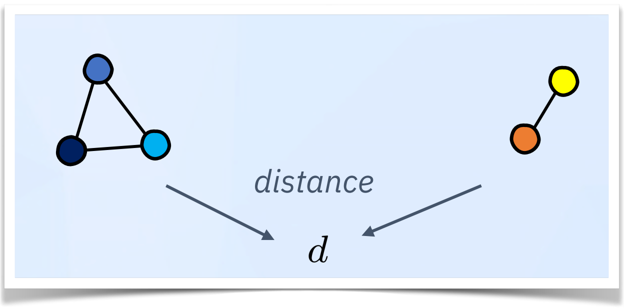
A Review of some existing works to compare attributed graphs

- The main objective is to **jointly code for topologies & attributes**
- Some Existing Solutions :
  - ❖ In ML: low scalability when methods rely of GED (Graph Edit Distance)
  - ❖ In ML with kernels: usually nonparametric (exception multiple kernel learning)
  - ❖ in ML: the fruitful change of point-of-view: use **Optimal Transport between distributions representing graphs** so as to compare graphs+ attributes => Fused Gromov-Wasserstein
  - ❖ In GSP, as quoted, works using OT where signals on G allows comparisons / alignements of graphs
  - ❖ In GSP, notions of distances between graphs
  - ❖ In ML+GSP : ways to propose distances between Attributed Graphs, and parametric them

A Simple Way to Learn Metrics Between Attributed Graphs

From Yacouba Kaloga PhD thesis (12/2021) ; LoG 2022 ; arXiv:2209.12727 (2022)

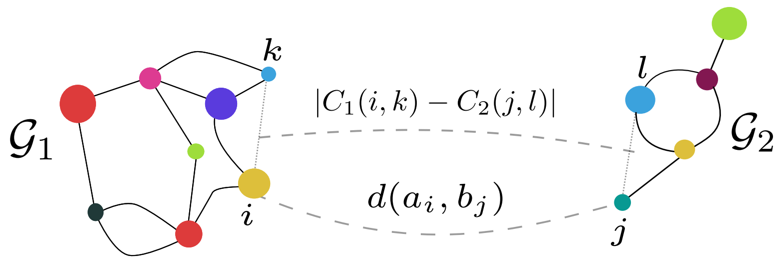
Joint work with Amaury Habrard (LabHC; Saint-Etienne)



Optimal Transport for ~~Graphs or~~ Attributed Graphs

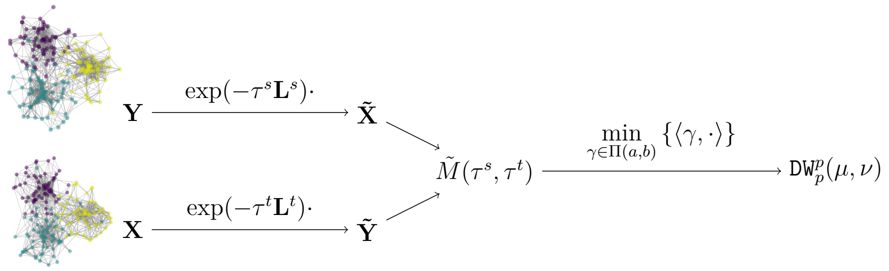
- One can **combine Attributes and Gromov-Wasserstein** characterisation of graphs

  “Fused Gromov-Wasserstein distance“ [Vayer et al., ICML 2019]



- If you have followed up to now: **The Diffusion Wasserstein distance**

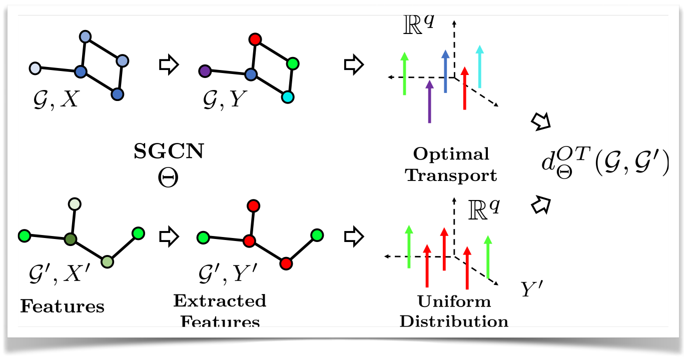
[Barbe et al., ECML 2020; ICTAI 2021]


$\mathbf{Y}$
$\exp(-\tau^s \mathbf{L}^s)\cdot$
$\tilde{\mathbf{X}}$
$\mathbf{X}$
$\exp(-\tau^t \mathbf{L}^t)\cdot$
$\tilde{\mathbf{Y}}$
$\tilde{M}(\tau^s, \tau^t)$
$\min_{\gamma \in \Pi(a,b)} \{\langle \gamma, \cdot \rangle\}$
$\mathtt{DW}_p^p(\mu, \nu)$

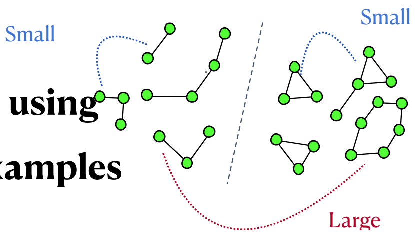
Optimal Transport for Attributed Graphs, with Metric Learning

- Our proposition: 1) **parametrize the (graphs+attributes) representation through a GCN**

  2) **compute distance** between them **by optimal transport**



3) (semi-)supervised **training of the distance using positive (close) and negative (far) sets of examples**



Optimal Transport for Attributed Graphs, with Metric Learning

=> The **Simple (& Scalable) Graph Metric Learning** model

- Our constraints :
- Be able to deal with graphs of **different sizes**, attributes of **various natures**
- Keep a **reasonable number of parameters** (to avoid overfitting)
- Keep the computational load acceptable, as the training will call the distance function many times
- Focus on the scalability of the method
- Focus on a method which has not be trained anew if one is given new instances of data
- Motivation : **frugal Machine Learning!**

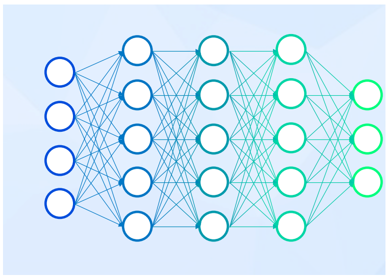
1) Trainable Learning and Graphs: Graph Neural Networks

- From ~2015 on: an ever growing interest to adapt Deep Learning to Graph Structures

$$\mathscr{F}(x)_{\text{layer (l)}} = \sigma(W^{(l)}x + b^{(l)})$$

weighted average of input + bias/offset

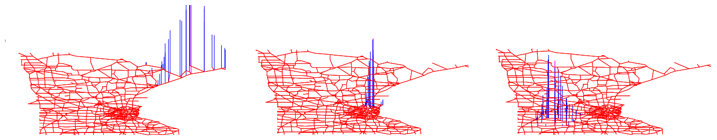
non-linear activation function



then Stack them => multilayer (or deep) neural network

use **Convolutions** for W => CNN

- For Graphs: One needs to **combine** information from **irregular neighbourhoods**.
- Thanks to **Graph Signal Processing**, one knows about **convolutions in graphs**.



[See Shuman et al., SP Mag 2013]

1) Trainable Learning and Graphs: Graph Neural Networks

- **Convolutional GNNs:** convolutions are defined in the Spectral domain (L = Laplacian)

$W = P_\Theta(L)$ Special form, polynomial of shift operator

$\mathcal{F}_{i^{\text{th}} \text{ node}}(x) = \sigma({w_i}^T x + b_i)$ $w_i = [P_\Theta(L)]_i$

same parameters for all nodes

[Defferrard et al., 2016]

GCN [Kipf & Welling, 2017]

- **What we will not do:** propose a new GNN architecture
- Many exist, with various limits associated to GNNs /GCNs, and well studied
  - S. Luan et al., "Break the ceiling: Stronger multi-scale deep graph convolutional networks." NeurIPS 2019
  - K. Xu et al. "How powerful are Graph Neural Networks », ICLR 2019
  - A. Loukas et al. "What graph neural networks cannot learn: deepth vs. width" ICLR 2020
  - Z. We et al. "A comprehensive survey on graph neural networks." IEEE Trans. NNL 2020

and still counting...

Learning and Graphs: Graph Neural Networks

- GNN = Gives a trend to powerful methods:
  - Whatever the flavor (filters ; attention-based ; message passing)

- **Strong applications :**
  - Drug Discovery ChemProp [Cell 2020];
  - Alphafold2 and Transformers use graphs
  - Drug repurposing [see S. Chepuri, 2020: Dr-COVID: graph neural networks for SARS-CoV-2 drug repurposing]
  - OpenCatalyst: discover new molecules that are catalysts for Chemistry (e.g., for fuel conversion)

- Some smart (and nice) people working on the subject
- Insights from Graph Signal Processing are useful for GCN/GNN/…

1) Trainable Learning and Graphs: Graph Neural Networks

- **What we will do:** think of GNNs/GCNs as a way to **obtain a Graph Representation**
- Extract Features for Attributed Graphs: we use **Simple GCN [2019]**

  - Amounts to **Graph Filtering** (Feature Propagation) then standard **Non-Linear Activation** fct

Initial attributes $\mathbf{X} \in \mathbb{R}^{n \times q}$; Modified Adjacency matrix: $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_n$

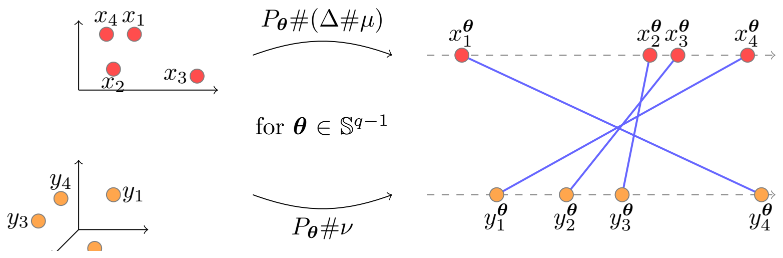
Features $\mathbf{Y} \in \mathbb{R}^{n \times p}$ are generated as

$$\mathbf{Y} = \mathrm{ReLU}(\widetilde{\mathbf{A}}^{r}\, \mathbf{X}\, \boldsymbol{\Theta})$$

- **Trainable Parameters:** $\boldsymbol{\Theta} \in \mathbb{R}^{q \times p}$ with hyper-parameters p and r
- **Graph Representation:** $\mathscr{D}_{\Theta}(\mathscr{G}, \mathbf{X}) = \sum_{i=1}^{n} \frac{1}{n}\, \delta_{\mathbf{Y}(i,:)}$

2) Optimal Transport with a Reduced Computational Load

- For Optimal Transport: Use the **Sliced methods**
  - **[N. Bonneel et al., "Sliced and Radon Wasserstein barycenters of measures", JMIV 2015]**
  - One projects the distribution (in $\mathbb{R}^p$) onto various 1-D directions θ, then average


x_4
x_1
x_2
x_3
$P_{\boldsymbol{\theta}}\#(\Delta\#\mu)$
for $\boldsymbol{\theta} \in \mathbb{S}^{q-1}$
$P_{\boldsymbol{\theta}}\#\nu$
y_4
y_1
y_3
$x_1^{\boldsymbol{\theta}}$
$x_2^{\boldsymbol{\theta}}$
$x_3^{\boldsymbol{\theta}}$
$x_4^{\boldsymbol{\theta}}$
$y_1^{\boldsymbol{\theta}}$
$y_2^{\boldsymbol{\theta}}$
$y_3^{\boldsymbol{\theta}}$
$y_4^{\boldsymbol{\theta}}$
Figure: T. Vayer

- The main advantage is that **1D optimal transport** is easily computed **by sorting**
- Property: one can show that it is a metric (excepted specific conditions)

2) Optimal Transport with a Reduced Computational Load

$$\mathscr{W}_2(\mu,\nu) = \inf_{\pi_{i,j}\in\Pi_{a,b}} \left(\sum_{i,j=1}^{n,n'} \pi_{i,j} c(\mathbf{x}_i, \mathbf{x}'_j)^2\right)^{\frac{1}{2}}$$

- Thee candidates for fast OT:

- **Sliced Wasserstein Distance** $\mathscr{SW}_2$ with directions sampled at random, and

$$\mathscr{SW}_2(\mu,\nu)^2 = \int_{\mathbb{S}^{q-1}} \mathscr{W}_2(\mu_\theta,\nu_\theta)^2 d\theta$$

- **Projected Sliced Wasserstein Distance** $\mathscr{PW}_2$, when $n = n'$, computing the distance in the original domain

[Rowland et al. AISTATS 2019]

$$\mathscr{PW}_2(\mu,\nu)^2 = \int_{\mathbb{S}^{q-1}} \sum_{i,j=1}^{n,n'} \pi_{i,j}^{\theta,*} \|x_i - x'_j\|_2^2 d\theta$$

- Our proposition: **Restricted Projected Sliced-Wasserstein** $\mathscr{RPW}_2$: One limits the integral to a spanning set fo vectors, conveniently chosen as the canonical basis vectors $\{u_k\}_{k=1}^p$, hence:

$$\mathscr{RPW}_2(\mu,\nu)^2 = \frac{1}{p}\sum_{k=1}^{p}\sum_{i,j=1}^{n,n'} \pi_{i,j}^{u_k,*} \|x_i - x'_j\|_2^2$$

- Property: $\mathscr{RPW}_2$ is a metric.

$$\boxed{d_\Theta^{\mathscr{RPW}_2}(\mathscr{G},\mathscr{G}') = \mathscr{RPW}_2(\mathscr{D}_\Theta(\mathscr{G},\mathbf{X}),\mathscr{D}_\Theta(\mathscr{G}',\mathbf{X}'))}$$

3) Loss for training the distance:

the Nearest Class Cloud Metric Learning

- **Objective function ?**
- Go back to tutorial of Bellet et al.
- Here: a variant of NCA,
- => **Nearest Class Cloud Metric Learning**

- Designed to boost k-NN classification

  (remind : no retraining is what we look for)

$\mathcal{G}_i \in \mathbb{G}_x$ $\qquad$ $e \in \mathbb{L}$

Attributed graph $\qquad$ *label*

$$p^{\boldsymbol{\Theta}}(e|\mathcal{G}) = \frac{\exp\left(\sum_{\substack{\mathcal{G}_i \in \mathbb{G}_x \\ \mathcal{E}(\mathcal{G}_i)=e}} -d_{\boldsymbol{\Theta}}^{\mathcal{SW}}(\mathcal{G},\mathcal{G}_i)^2\right)}{\sum_{e' \in \mathbb{E}} \exp\left(\sum_{\substack{\mathcal{G}_i \in \mathbb{G}_x \\ \mathcal{E}(\mathcal{G}_i)=e'}} -d_{\boldsymbol{\Theta}}^{\mathcal{SW}}(\mathcal{G},\mathcal{G}_i)^2\right)}$$

Probability for the graphs G to have label e

$$\downarrow$$

$$\boxed{\max_{\boldsymbol{\Theta}} \sum_{\mathcal{G}_i \in \mathbb{G}_x} \log p^{\boldsymbol{\Theta}}(\mathcal{E}(\mathcal{G}_i)|\mathcal{G}_i)}$$

Maximize the probability for each graph to have is own label

Some elements on this **Simple Graph Metric Learning** model

- Training of the SGML model in a nutshell:

Algorithm 1 SGML: High-level algorithm to build $d_{\mathbf{\Theta}^*}^{\mathcal{RPW}_2}$.

```
Require: A dataset of attributed graphs $\mathbb{G}$ and their labeling function $\mathcal{E}$.
  for each epoch $e \in \{1, \dots, E\}$ do
    Build a partition: $\cup_k B_k = \mathbb{G}$ such that $B_k \cap B_{k'} = \emptyset$.
    for each batch $B_k$ do
      for each graph pair $(\mathcal{G}, \mathcal{G}') \in B_k \times B_k$ do
        Compute distance $d_{\mathbf{\Theta}}^{\mathcal{RPW}_2}(\mathcal{G}, \mathcal{G}')$ (Eq. (9))
      Compute $-\mathcal{F}_{\mathbf{\Theta}}^{B_k}$ (Eq. (11)) and apply an iteration of Adam descent algorithm.
  return all pairwise distance $d_{\mathbf{\Theta}^*}^{\mathcal{RPW}_2}$ in $\mathbb{G}$.
```

- Hyper-Parameters: p and r for the SimpleGCN
- **Complexity** of the method:
  - Time complexity in $O(|\mathbb{G}|\tilde{n}(p^2 + \tilde{n}rp) + |\mathbb{G}|^2 p^2 \tilde{n} \log \tilde{n})$
  - Space complexity in $O(\tilde{n}^2 p)$

SGML model

Numerical Experiments

- Graph Datasets

Datasets	BZR	COX2	PROTEINS	ENZYMES	MUTAG	NCI1	IMDB-B	IMDB-M	CUNEIFORM
#Graphs	405	467	1113	600	188	4110	1000	1500	267
#Nodes	35.75	41.22	39.06	32.63	17.93	29.97	19.77	13	21.27
Node attributes	cont.	cont.	cont. / lab.	cont. / lab.	deg.	lab.	deg.	deg.	cont. / lab.
q	3	3	1 / 3	18 /3	4	38	135	88	3 / 3

- Task of Supervised Classification
- either *k*-NN classifier
- or SVM with induced kernel

Method	MUTAG	NCI1	PROTEINS	ENZYMES	IMDB-M	IMDB-B
			k-NN			
$\mathcal{RPW}_2$	$\mathbf{90.00 \pm 7.60}$	72.12 ± 1.65	$\mathbf{70.18 \pm 4.01}$	49.00 ± 8.17	45.00 ± 5.46	68.90 ± 5.45
Net-LSD-h	84.90	65.89	64.89	31.99	40.51	68.04
FGSD	86.47	75.77	65.30	41.58	41.14	69.54
NetSimile	84.09	66.56	62.45	33.23	40.97	69.20
			SVM & **GCN**			
$\mathcal{RPW}_2$	$\mathbf{88.95 \pm 7.61}$	74.84 ± 1.81	74.55 ± 4.19	54.00 ± 7.07	51.00 ± 5.44	72.00 ± 3.16
WWL	87.27 ± 1.50	85.75 ± 0.25	74.28 ± 0.56	$\mathbf{59.13 \pm 0.80}$	✗	✗
$\mathcal{FGW}$	83.26 ± 10.30	72.82 ± 1.46	✗	✗	48.00 ± 3.22	63.80 ± 3.49
$\mathcal{FGW}$-WL	88.42 ± 5.67	$\mathbf{86.42 \pm 1.63}$	✗	✗	✗	✗
WL-OA	87.15 ± 1.82	86.08 ± 0.27	$\mathbf{76.37 \pm 0.30}$	58.97 ± 0.82	✗	✗
PSCN	83.47 ± 10.26	70.65 ± 2.58	58.34 ± 7.71	✗	✗	✗

SGML model

Numerical Experiments

- Graph Datasets

Datasets	BZR	COX2	PROTEINS	ENZYMES	MUTAG	NCI1	IMDB-B	IMDB-M	CUNEIFORM
#Graphs	405	467	1113	600	188	4110	1000	1500	267
#Nodes	35.75	41.22	39.06	32.63	17.93	29.97	19.77	13	21.27
Node attributes	cont.	cont.	cont. / lab.	cont. / lab.	deg.	lab.	deg.	deg.	cont. / lab.
q	3	3	1 / 3	18 /3	4	38	135	88	3 / 3

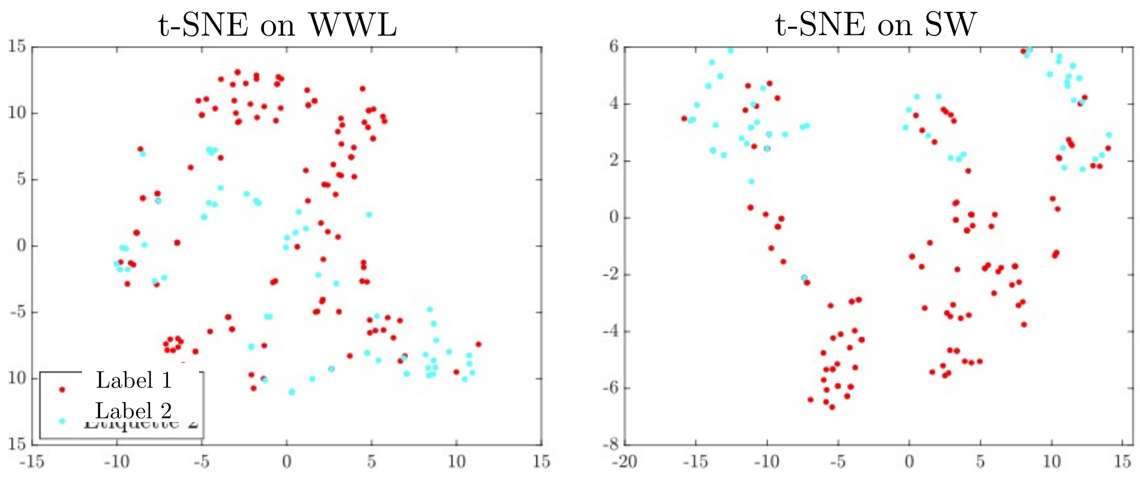
- Task of Supervised Classification
- either k-NN classifier
- or SVM with induced kernel

Method	BZR	COX2	PROTEINS	ENZYMES	CUNEIFORM
$\mathcal{RPW}_2$ (kNN)	$\mathbf{85.61 \pm 2.98}$	$\mathbf{79.79 \pm 2.18}$	71.79 ± 4.47	51.66 ± 5.16	54.81 ± 12.26
			SVM & GCN		
$\mathcal{RPW}_2$	84.39 ± 3.81	$\mathbf{78.51 \pm 0.01}$	74.29 ± 4.11	48.83 ± 4.78	64.44 ± 10.50
WWL	84.42 ± 2.03	78.29 ± 0.47	$\mathbf{77.91 \pm 0.80}$	73.25 ± 0.87	✗
$\mathcal{FGW}$	$\mathbf{85.12 \pm 4.15}$	77.23 ± 4.86	74.55 ± 2.74	71.00 ± 6.76	76.67 ± 7.04
PROPAK	79.51 ± 5.02	77.66 ± 3.95	61.34 ± 4.38	71.67 ± 5.63	12.59 ± 6.67
HGK-SP	76.42 ± 0.72	72.57 ± 1.18	75.78 ± 0.17	66.36 ± 0.37	✗
PSCN [K = 10] (GCN)	80.00 ± 4.47	71.70 ± 3.57	67.95 ± 11.28	26.67 ± 4.77	25.19 ± 7.73

SGML model

Visualisation of a Numerical Experiment

- For MUTAG Dataset

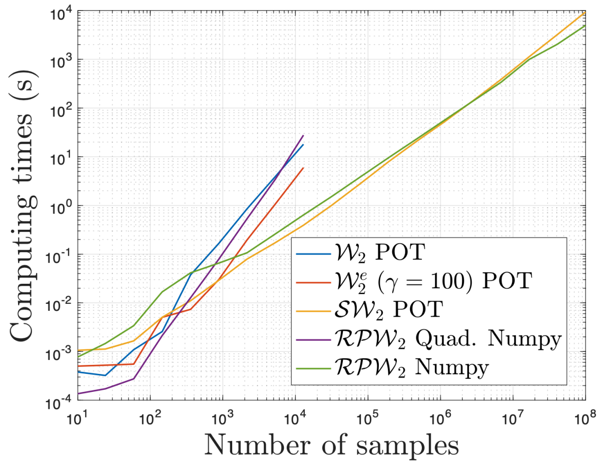


Embedding in 2D with t-SNE, comparing WWL and SGML

SGML model

Scalability and Ablation study

- Scalability in running time



- Ablative study

Dataset	**WWL**		**SGML** - $\mathcal{SW}_2$		**SGML** - NCA		**SGML** - $\mathcal{PW}_2$	
Method	**Acc.**	**Δ**	**Acc.**	**Δ**	**Acc.**	**Δ**	**Acc.**	**Δ**
BZR	78.05	- 7.56	82.93	- 2.68	83.41	- 2.20	84.39	- 1.22
COX2	78.51	-1.26	78.30	- 1.49	77.66	- 2.13	78.94	- 0.85
MUTAG	83.68	- 6.32	86.84	- 3.16	87.37	- 2.63	90.00	0.00
NCI1	80.43	5.31	69.03	- 3.09	69.66	- 2.46	72.90	0.78
PROTEINS	71.60	1.42	71.34	1.16	71.70	1.52	70.54	0.36
IMDB-B	68.20	- 0.7	68.20	-0.70	67.40	-1.5	68.80	- 0.10
IMDB-M	48.73	3.73	42.33	-2.67	42.73	-2.27	44.13	- 0.87
ENZYMES	56.00	7.00	44.33	- 4.67	55.33	6.33	44.83	-4.17

- **Message**: it's scalable, perf are ok, with some theoretical insights!

Now is the time to conclude

- **2)** A scalable & simple model to **Learn Distances between Attributed Graphs**
- **->** **SGML**: a simple, motivated, scalable and efficient, method for (semi-supervised) metric learning between attributed graphs
- **1)** A novel way to **combine structure and attributes by Diffusion** + **OT**
- **->** **Diffusion Wasserstein distance:** a powerful method, for unsupervised graph domain adaptation tasks

- We favor **simple methods**, with a specific objectives and reduced computational costs (& waste)

- Our **way forward:**
  - 1) improve feature extraction thanks to **insights from GSP**
  - 2) **more explainability** for these graph-based ML methods (see our GraphNEx project)

Contact: Pierre BORGNAT, CNRS, LP ENS de Lyon
perso.ens-lyon.fr/pierre.borgnat