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, ...
- Chemistry
- Physics
- 2D images
- 3D Points clouds
- Other geometric and/or irregular shapes


Source: Awesome Graph Classification https://github.com/benedekrozemberczki/awesome-graph-classification
Setting: Attributed Graphs

- In the general case, **nodes and/or edges can carry information**:
  - Edges = existence of some relationship
  - Nodes = Attributes, or Features / Signals

\[ G = (V, E, X) = (A, X) \]

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

- Adjacency matrix

\[
A = \begin{pmatrix}
0 & 0.5 & 0 & 1 \\
0.5 & 0 & 1 & 0 \\
0 & 1 & 0 & 0.5 \\
1 & 0 & 0.5 & 0
\end{pmatrix}, \\
A \in [0, 1]|V| \times |V|
\]

- Attribute matrix

\[
X = \begin{pmatrix}
0.1 & 1 & -0.4 \\
0 & 1 & 0 \\
2 & 2.1 & 0 \\
-1 & 0 & -0.5
\end{pmatrix}, \\
X \in \mathbb{R}^{n \times d}
\]
Many Machine Learning tasks for Data on Graphs

Supervised Tasks

• Learn to classify Nodes

• Learn to classify Graphs

Etiquette rouge

Red labels

Blue labels

Etiquette bleue

Quelle est l’étiquette ?

Which label ?

Blue labels

Etiquette bleue

Which label ?

Quelle est l’étiquette ?

Etiquette rouge

Red labels
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,...)

- For high-dimensional embeddings

Chapter 3

Neighborhood Reconstruction Methods

This part of the book is concerned with methods for learning node embeddings. The goal of these methods is to encode nodes as low-dimensional vectors that summarize their graph position and the structure of their local graph neighborhood. In other words, we want to project nodes into a latent space, where geometric relations in this latent space correspond to relationships (e.g., edges) in the original graph or network [Ho et al., 2002] (Figure 3.1).

In this chapter we will provide an overview of node embedding methods for simple and weighted graphs. Chapter 4 will provide an overview of analogous embedding approaches for multi-relational graphs.

Figure 3.1: Illustration of the node embedding problem. Our goal is to learn an encoder ($\text{enc}$), which maps nodes to a low-dimensional embedding space. These embeddings are optimized so that distances in the embedding space reflect the relative positions of the nodes in the original graph.

From [Tremblay & Borgnat, 2014]

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 $y$
  \[ y = F(x) \]

- Machine Learning in the good all times

  \[ F = F_{\text{decision}} \circ F_{\text{features}}(x) \]

  learnt from data

- Machine Learning with Representation Learning / Deep Learning

  \[ F = F_{\text{decision}} \circ F_{\text{features}} \quad / \quad F = F_{\text{decision}} \circ F_{\text{layer d}} \circ \cdots \circ F_{\text{layer 1}} \]

  All learnt from data

  in multiple layers

[From Pierre Vandergheynst’ talk, 2021]

**Low Level task: Graphs Representation Learning**

- For Graphs, Representation learning can be summarised as:
  - For Collection of Graphs
  - For Nodes in a Graph

- 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!)

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

  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 \( \mathcal{X} = \{ x_i \}_{i=1}^{\lvert \mathcal{X} \rvert} \in \mathbb{R}^{q \times \lvert \mathcal{X} \rvert} \) and \( \mathcal{X}' \) and two distributions on these \( \mu = \sum_{x_i \in \mathcal{X}} a_i \delta_{x_i} \) and \( \nu = \sum_{x_i' \in \mathcal{X}'} b_i \delta_{x_i'} \) with

\[
a_i \geq 0, \ b_i \geq 0 \text{ and } \sum_{i=1}^{n} a_i = 1, \ \sum_{i=1}^{n'} b_i = 1
\]

- Given a cost function \( c : \mathbb{R}^{q} \times \mathbb{R}^{q} \to \mathbb{R}_+ \), one builds the 2-Wasserstein distance \( \mathcal{W}_2 \) as:

\[
\mathcal{W}_2(\mu, \nu) = \inf_{\pi_{i,j} \in \Pi_{a,b}} \left( \sum_{i,j=1}^{n,n'} \pi_{i,j} c(x_i, x_j)^2 \right)^{\frac{1}{2}}
\]

where \( \Pi_{a,b} \) is the set of joint distributions on \( \mathcal{X} \times \mathcal{X}' \)

whose marginals are the distributions \( \mu = \sum_{x_i' \in \mathcal{X}'} \pi(\cdot, x_i') \) and \( \nu = \sum_{x_i \in \mathcal{X}} \pi(x_i, \cdot) \)
Optimal Transport for Graphs

- For Graphs: one has to **associate a distribution to a graph**
  - A first solution: rely on the Weisfeiler-Lehman test

- A 2nd solution: **Comparison through probabilistic models of graph signals**
  - ["Graph Optimal Transport", H. Maretic et al. NeurIPS 2019]
  - for a graph $\mathcal{G}$ with Laplacian $L$, one considers: $x \sim \nu_{\mathcal{G}} = \mathcal{N}(0,L^\dagger)$
  - 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**
  - 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**


• 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)

\[
\begin{align*}
X^s & \xrightarrow{\exp(-\tau^s L^s)} \tilde{X}^s \\
X^t & \xrightarrow{\exp(-\tau^t L^t)} \tilde{X}^t
\end{align*}
\]

\[
\begin{align*}
\bar{M} & \xrightarrow{\min_{\gamma \in \Pi(a,b)}} \{ \langle \gamma, \tilde{M}^p \rangle_F \} \\
& \xrightarrow{\text{DW}_p^p(U^s, U^t)}
\end{align*}
\]

Diagramme du calcul de Diffusion-Wasserstein:

\[
\begin{align*}
\tilde{X}^s & \xrightarrow{\exp(-\tau^s L^s)} \tilde{X}^s \\
\tilde{X}^t & \xrightarrow{\exp(-\tau^t L^t)} \tilde{X}^t
\end{align*}
\]

Avantages :
1. Un seul terme pour attributs et structure
2. $\tau^s$ et $\tau^t$ pour régler le compromis entre les deux modalités
3. diffusion lisse les attributs (filtre passe-bas)
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 $D = \text{diag}(A \cdot \mathbf{1})$, 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”
The GDD is a metric, in the strict mathematical sense, i.e. exponential may be computed by diffusion distance is given by columns of the Laplacian exponential kernels, \( v \) a time-varying vector representing the value of the quantity that is graph Laplacian operator \( [5] \), defined by a straightforward application of (3) and (2) allows computation of different sets of weighted edges. Computing the sum of squared each vertex generated by diffusion up to time \( t \) is precisely the (binary) graphs. We will make frequent use of the (unnormalized) diagonal. Note that we are not restricting ourselves to unweighted FOR WEIGHTED GRAPHS BASED ON THE GRAPH LAPLACIAN EXPONENTIAL KERNEL”

They define a Diffusion distance between graphs having the same number of nodes

\[
\xi(A_1, A_2; t) = \sum_{i,j} ((\exp(-tL_1))_{i,j} - (\exp(-tL_2))_{i,j})^2 \\
= \| \exp(-tL_1) - \exp(-tL_2) \|_F^2 
\] (2)

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

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

**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”  
- You reduced the title text too long!
Optimal Transport and Graph Signal Processing for Attributed Graphs

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

• Then, we define it as:

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

• Theoretically, it has good properties:
  • it is a distance
  • we have bounds for small and large \( \tau \)
  • it’s efficient to be computed, more than Fused GW

\[
\begin{align*}
X^s &\xrightarrow{\exp(-\tau^sL^s)} \tilde{X}^s \\
X^t &\xrightarrow{\exp(-\tau^tL^t)} \tilde{X}^t \\
\tilde{M} &\xrightarrow{\min_{\gamma \in \Pi(a,b)} \left\{ \langle \gamma, \tilde{M}^p \rangle \right\}} DW_p^P(U^s, U^t)
\end{align*}
\]
The Diffusion Wasserstein Distances for Attributed Graphs

\[ DW_p^p(\mu, \nu \mid \tau^s, \tau^t) = \min_{\gamma \in \Pi(a,b)} \langle \gamma, \tilde{M}^p \rangle. \]

- **Experimentally**, it works well: the task for comparison is **Domain Adaptation**
  - 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.!
The Diffusion Wasserstein Distances for Attributed Graphs

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

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

![Domain Adaptation graphs and plots](Image)  

from [Barbe et al., ECML-PKDD 2020]
The Diffusion Wasserstein Distances for Attributed Graphs, in action

- **How to set diffusion parameters** $\tau$? For **unsupervised DA**!
- Use an ER random graph and features as Wasserstein barycenter as an impostor:
  \[
  X^0 = \underset{X \in \mathbb{R}^i \times r}{\arg\min} \left\{ \frac{1}{2} (\mathcal{W}(X^s, X) + \mathcal{W}(X^t, X)) \right\}.
  \]
- And a triplet loss to be optimized for $\tau$:
  \[
  \tau^* = \underset{\tau \geq 0}{\arg\min} \left\{ \mathcal{L}(\tau) \right\}, \quad \text{with}\]
  \[
  \mathcal{L}(\tau) = \mathcal{D}_p(G^s, G^t | \tau) - (\mathcal{D}_p(G^s, G^0 | \tau) + \mathcal{D}_p(G^t, G^0 | \tau)).
  \]
- Avoid the use of Circular Validation for DA

---

from [Barbe et al., ICTAI 2021]
The Diffusion Wasserstein Distances for **Attributed Graphs**, in action

\[
\text{Dw}_p^\tau(\mu, \nu \mid \tau^s, \tau^t) = \min_{\gamma \in \Pi(a, b)} \langle \gamma, \tilde{M}^p \rangle.
\]

- Impostor + Triplet loss = **set the diffusion parameter** \(\tau\)!
- No Circular Validation \(\Rightarrow\) more stability, better perf.

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

\[
\tau^* = \arg\min_{\tau \geq 0} \{ \mathcal{L}(\tau) \}, \quad \mathcal{L}(\tau) = \text{Dw}_p(G^s, G^t \mid \tau) - \left( \text{Dw}_p(G^s, G^0 \mid \tau) + \text{Dw}_p(G^t, G^0 \mid \tau) \right).
\]

---

**Fig. 6:** Median, quartile and decile accuracy of various OT methods on the task of transferring the labels of \(G^s\) to \(G^t\).
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**

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]

\[ \gamma \rightarrow \min_{\phi \in H(a,b)} \{ \langle \gamma, \cdot \rangle \} \]

\[ DW_p^m(\mu, \nu) \]
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

\[ \mathcal{F}(x)_{\text{layer}(l)} = \sigma(W^{(l)}x + b^{(l)}) \]

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

use **Convolutions** for \( W \Rightarrow \text{CNN} \)

- 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 ($L = \text{Laplacian}$)
  \[
  W = P_\Theta(L) \quad \text{Special form, polynomial of shift operator}
  \]
  \[
  \mathcal{F}_{i} = \sigma(w_i^T x + b_i) \quad w_i = [P_\Theta(L)]_i
  \]
  
  same parameters for all nodes

- **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

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

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

- **Trainable Parameters:** $\Theta \in \mathbb{R}^{q \times p}$ with hyper-parameters $p$ and $r$

- **Graph Representation:** $\mathcal{D}_\Theta(\mathcal{G}, \mathbf{X}) = \sum_{i=1}^{n} \frac{1}{n} \delta_{\mathbf{Y}(i,:)}$
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 $\theta$, then average

  \[ \text{Figure: T. Vayer} \]

  \[ \text{Figure: T. Vayer} \]

- 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

\[ \mathcal{W}_2(\mu, \nu) = \inf_{\pi_{i,j} \in \Pi_{\alpha, \beta}} \left( \sum_{i,j=1}^{n,n'} \pi_{i,j} c(x_i, x'_j) \right)^{\frac{1}{2}} \]

- **Thee candidates for fast OT:**
  
  - **Sliced Wasserstein Distance** \( \mathcal{S}\mathcal{W}_2 \) with directions sampled at random, and
    \[ \mathcal{S}\mathcal{W}_2(\mu, \nu)^2 = \int_{\mathcal{S}^{d-1}} \mathcal{W}_2(\mu_\theta, \nu_\theta)^2 d\theta \]

  - **Projected Sliced Wasserstein Distance** \( \mathcal{P}\mathcal{W}_2 \), when \( n = n' \), computing the distance in the original domain
    \[ \mathcal{P}\mathcal{W}_2(\mu, \nu)^2 = \int_{\mathcal{S}^{d-1}} \sum_{i,j=1}^{n,n'} \pi_{i,j}^\theta \| x_i - x'_j \|^2 d\theta \]

  - **Our proposition:** **Restricted Projected Sliced-Wasserstein** \( \mathcal{R}\mathcal{P}\mathcal{W}_2 \): One limits the integral to a spanning set of vectors, conveniently chosen as the canonical basis vectors \( \{ u_k \}_{k=1}^p \), hence:
    \[ \mathcal{R}\mathcal{P}\mathcal{W}_2(\mu, \nu)^2 = \frac{1}{p} \sum_{k=1}^p \sum_{i,j=1}^{n,n} \pi_{i,j}^{u_k \theta} \| x_i - x'_j \|^2 \]

- **Property:** \( \mathcal{R}\mathcal{P}\mathcal{W}_2 \) is a metric.
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,
  - \( \Rightarrow \) Nearest Class Cloud Metric Learning

- Designed to boost k-NN classification
  (remind: no retraining is what we look for)

\[
P^\Theta(e|G) = \frac{\exp \left( \sum_{g_i \in G_x} -d_{SW}(G, G_i)^2 \right)}{\sum_{e' \in E} \exp \left( \sum_{g_i \in G_x} -d_{SW}(G, G_i)^2 \right)}
\]

Probability for the graphs \( G \) to have label \( e \)

\[
\max_\Theta \sum_{G_i \in G_x} \log P^\Theta(E(G_i)|G_i)
\]

Maximize the probability for each graph to have its 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 $d^{RPW_2}_\Theta$.

  **Require:** A dataset of attributed graphs $\mathcal{G}$ and their labeling function $\mathcal{E}$.
  
  for each epoch $e \in \{1, \ldots, E\}$ do
  
  Build a partition: $\bigcup_k B_k = \mathcal{G}$ 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 $d^{RPW_2}_\Theta(\mathcal{G}, \mathcal{G}')$ (Eq. (9))
  
  Compute $-\mathcal{F}^{B_k}_\Theta$ (Eq. (11)) and apply an iteration of Adam descent algorithm.
  
  return all pairwise distance $d^{RPW_2}_\Theta$ in $\mathcal{G}$.

- **Hyper-Parameters:** $p$ and $r$ for the SimpleGCN

- **Complexity** of the method:
  
  - Time complexity in $O(|\mathcal{G}| \tilde{n}(p^2 + \tilde{n}rp) + |\mathcal{G}|^2 p^2 \tilde{n} \log \tilde{n})$
  
  - Space complexity in $O(\tilde{n}^2 p)$
SGML model

Numerical Experiments

- Graph Datasets
  - Task of Supervised Classification
    - either \(k\)-NN classifier
    - or SVM with induced kernel

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

<table>
<thead>
<tr>
<th>Method</th>
<th>MUTAG</th>
<th>NCI1</th>
<th>PROTEINS</th>
<th>ENZYMES</th>
<th>IMDB-M</th>
<th>IMDB-B</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\mathcal{RPW}_2)</td>
<td>90.00 ± 7.60</td>
<td>72.12 ± 1.65</td>
<td>70.18 ± 4.01</td>
<td>49.00 ± 8.17</td>
<td>45.00 ± 5.46</td>
<td>68.90 ± 5.45</td>
</tr>
<tr>
<td>Net-LSD-h</td>
<td>84.90</td>
<td>65.89</td>
<td>64.89</td>
<td>31.99</td>
<td>40.51</td>
<td>68.04</td>
</tr>
<tr>
<td>FGSD</td>
<td>86.47</td>
<td>75.77</td>
<td>65.30</td>
<td>41.58</td>
<td>41.14</td>
<td>69.54</td>
</tr>
<tr>
<td>NetSimile</td>
<td>84.09</td>
<td>66.56</td>
<td>62.45</td>
<td>33.23</td>
<td>40.97</td>
<td>69.20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>k-NN</th>
<th>SVM &amp; GCN</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\mathcal{RPW}_2)</td>
<td>88.95 ± 7.61</td>
<td>74.84 ± 1.81</td>
</tr>
<tr>
<td>WWL</td>
<td>87.27 ± 1.50</td>
<td>85.75 ± 0.25</td>
</tr>
<tr>
<td>(\mathcal{FGW})</td>
<td>83.26 ± 10.30</td>
<td>72.82 ± 1.46</td>
</tr>
<tr>
<td>(\mathcal{FGW-WL})</td>
<td>88.42 ± 5.67</td>
<td>86.42 ± 1.63</td>
</tr>
<tr>
<td>WL-OA</td>
<td>87.15 ± 1.82</td>
<td>86.08 ± 0.27</td>
</tr>
<tr>
<td>PSCN</td>
<td>83.47 ± 10.26</td>
<td>70.65 ± 2.58</td>
</tr>
</tbody>
</table>
SGML model

Numerical Experiments

- Graph Datasets

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

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

<table>
<thead>
<tr>
<th>Method</th>
<th>BZR</th>
<th>COX2</th>
<th>PROTEINS</th>
<th>ENZYMES</th>
<th>CUNEIFORM</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{RPW}_2$ (kNN)</td>
<td>85.61 ± 2.98</td>
<td>79.79 ± 2.18</td>
<td>71.79 ± 4.47</td>
<td>51.66 ± 5.16</td>
<td>54.81 ± 12.26</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>BZR</th>
<th>COX2</th>
<th>PROTEINS</th>
<th>ENZYMES</th>
<th>CUNEIFORM</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{RPW}_2$</td>
<td>84.39 ± 3.81</td>
<td>78.51 ± 0.01</td>
<td>74.29 ± 4.11</td>
<td>48.83 ± 4.78</td>
<td>64.44 ± 10.50</td>
</tr>
<tr>
<td>WWL</td>
<td>84.42 ± 2.03</td>
<td>78.29 ± 0.47</td>
<td>77.91 ± 0.80</td>
<td>73.25 ± 0.87</td>
<td>X</td>
</tr>
<tr>
<td>$\mathcal{FGW}$</td>
<td>85.12 ± 4.15</td>
<td>77.23 ± 4.86</td>
<td>74.55 ± 2.74</td>
<td>71.00 ± 6.76</td>
<td>76.67 ± 7.04</td>
</tr>
<tr>
<td>PROPAC</td>
<td>79.51 ± 5.02</td>
<td>77.66 ± 3.95</td>
<td>61.34 ± 4.38</td>
<td>71.67 ± 5.63</td>
<td>12.59 ± 6.67</td>
</tr>
<tr>
<td>HGGK-SP</td>
<td>76.42 ± 0.72</td>
<td>72.57 ± 1.18</td>
<td>75.78 ± 0.17</td>
<td>66.36 ± 0.37</td>
<td>X</td>
</tr>
<tr>
<td>PSCN [K = 10] (GCN)</td>
<td>80.00 ± 4.47</td>
<td>71.70 ± 3.57</td>
<td>67.95 ± 11.28</td>
<td>26.67 ± 4.77</td>
<td>25.19 ± 7.73</td>
</tr>
</tbody>
</table>
SGML model

Visualisation of a Numerical Experiment

- For MUTAG Dataset

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

• Scalability in running time

Scalability and Ablation study

• Ablative study

<table>
<thead>
<tr>
<th>Dataset</th>
<th>WWL</th>
<th>SGML - $\mathcal{S}W_2$</th>
<th>SGML - NCA</th>
<th>SGML - $\mathcal{P}W_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>BZR</td>
<td>78.05</td>
<td>82.93</td>
<td>83.41</td>
<td>84.39</td>
</tr>
<tr>
<td>COX2</td>
<td>78.51</td>
<td>78.30</td>
<td>77.66</td>
<td>78.94</td>
</tr>
<tr>
<td>MUTAG</td>
<td>83.68</td>
<td>86.84</td>
<td>87.37</td>
<td>90.00</td>
</tr>
<tr>
<td>NCI1</td>
<td>80.43</td>
<td>69.03</td>
<td>69.66</td>
<td>72.90</td>
</tr>
<tr>
<td>PROTEINS</td>
<td>71.60</td>
<td>71.34</td>
<td>71.70</td>
<td>70.54</td>
</tr>
<tr>
<td>IMDB-B</td>
<td>68.20</td>
<td>68.20</td>
<td>67.40</td>
<td>68.80</td>
</tr>
<tr>
<td>IMDB-M</td>
<td>48.73</td>
<td>42.33</td>
<td>42.73</td>
<td>44.13</td>
</tr>
<tr>
<td>ENZYMES</td>
<td>56.00</td>
<td>44.33</td>
<td>55.33</td>
<td>44.83</td>
</tr>
</tbody>
</table>

Acc. is the accuracy. $\Delta$ is the difference in accuracy.

• 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