Representation Learning for Attributed Graphs

Pierre Borgnat
Equipe Sisyphe, Laboratoire de Physique, CNRS, ENS de Lyon
work: Yacouba Kaloga (LP ENSL) and A. Habrard (Lab. Hubert Curien, Saint-Etienne) et al.

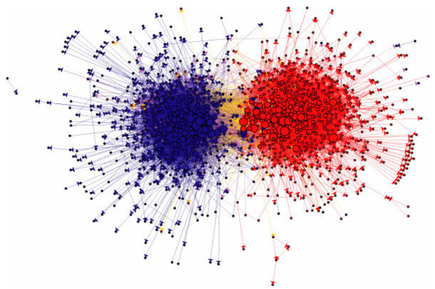
GDR ISIS 08/03/2022

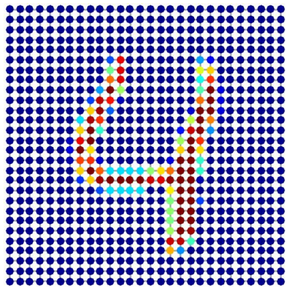
perso.ens-lyon.fr/pierre.borgnat

Work supported by: CNRS, ENSL, ANR, IDEXLyon, CHIST-Era project

Graphs: useful structures for data processing

- Social Networks
- Sensors' data
- Transportations

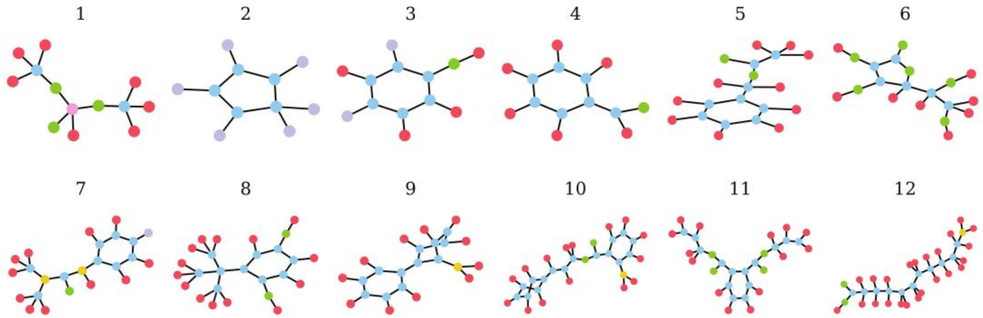




Source: Michael Edwards and Xianghua Xie. **Graph based convolutional neural network.** CoRR, abs/1609.08965, 2016.

- 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

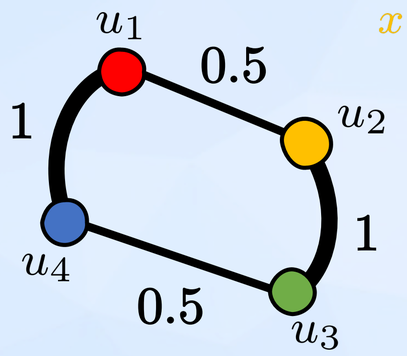
- In the general case, **nodes and/or edges can carry information**:

❖ Edges = existence of some relationship

❖ Nodes = Attributes, or Features / Signals

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

$$x(u_1) = \begin{pmatrix} 0.1 \\ 1 \\ -0.4 \end{pmatrix} \quad x(u_2) = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$



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

- 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

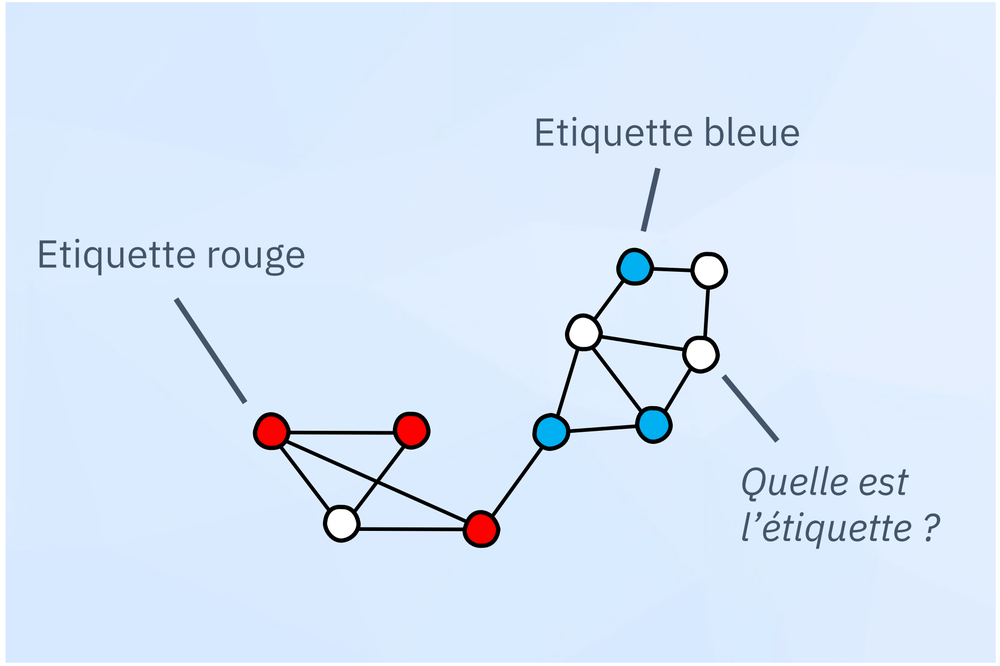
$$\begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ u_4 \end{array} \begin{array}{c} \\ \begin{pmatrix} 0.1 & 1 & -0.4 \\ 0 & 1 & 0 \\ 2 & 2.1 & 0 \\ -1 & 0 & -0.5 \end{pmatrix} \end{array}$$

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

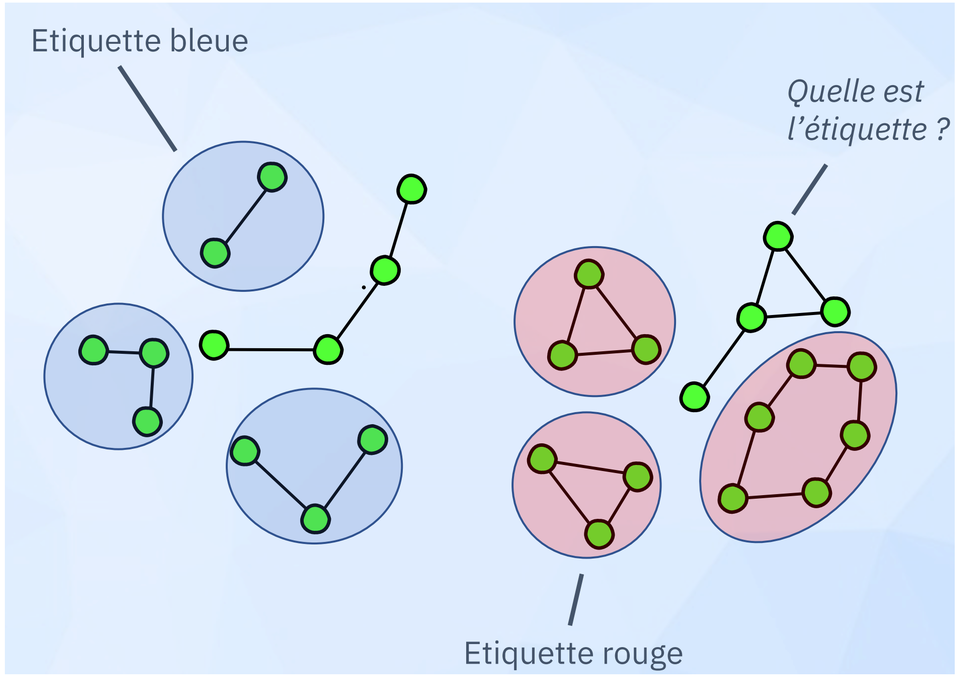
Many Machine Learning tasks for Data on Graphs

Supervised Tasks

- Learn to classify Nodes



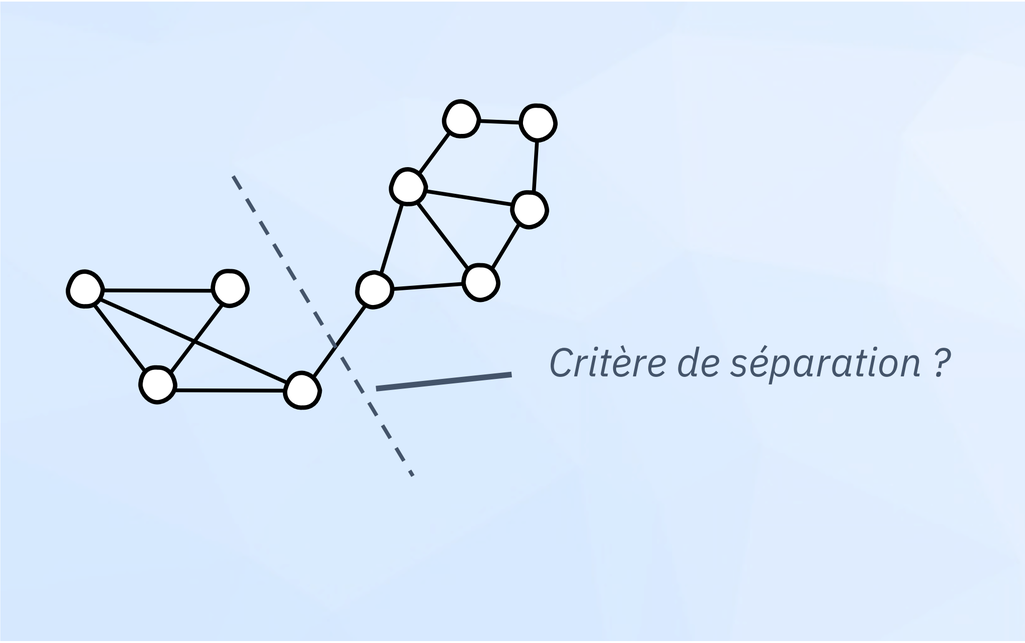
- Learn to classify Graphs



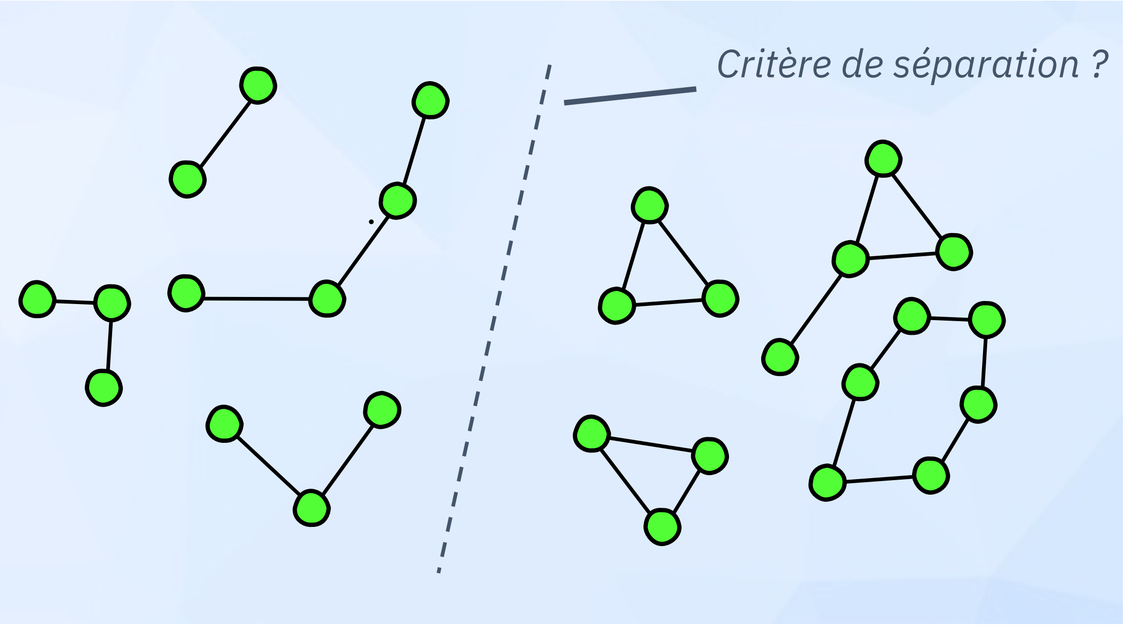
Many Machine Learning tasks for Data on Graphs

Unsupervised Tasks

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



- Learn to cluster collection of graphs



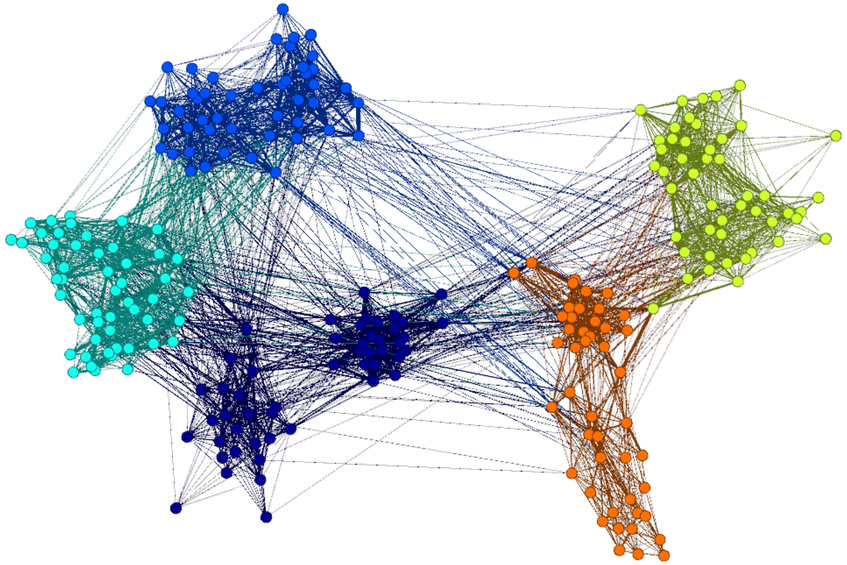
- Note: more general features -> small-world, scale-free,... [see Complex Networks]

Many Machine Learning tasks for Data on Graphs

Representation of graphs : Embeddings

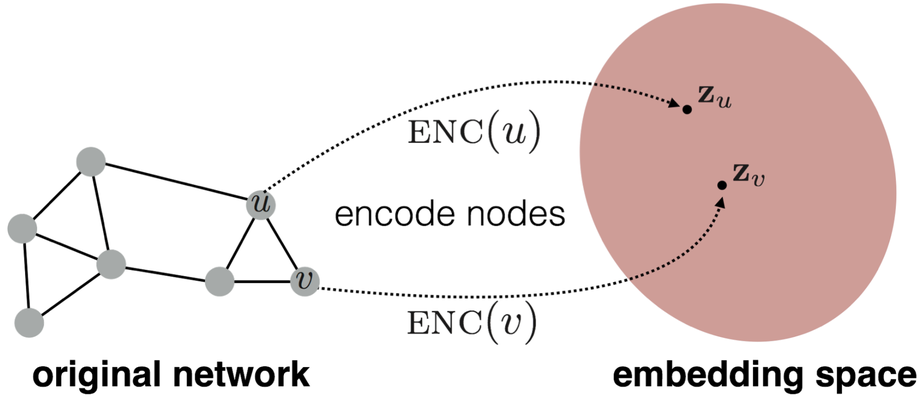
- For Visualisations or low-dim embeddings

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



From [Tremblay & Borgnat, 2015]

- 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

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

- Machine Learning in the good all times

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

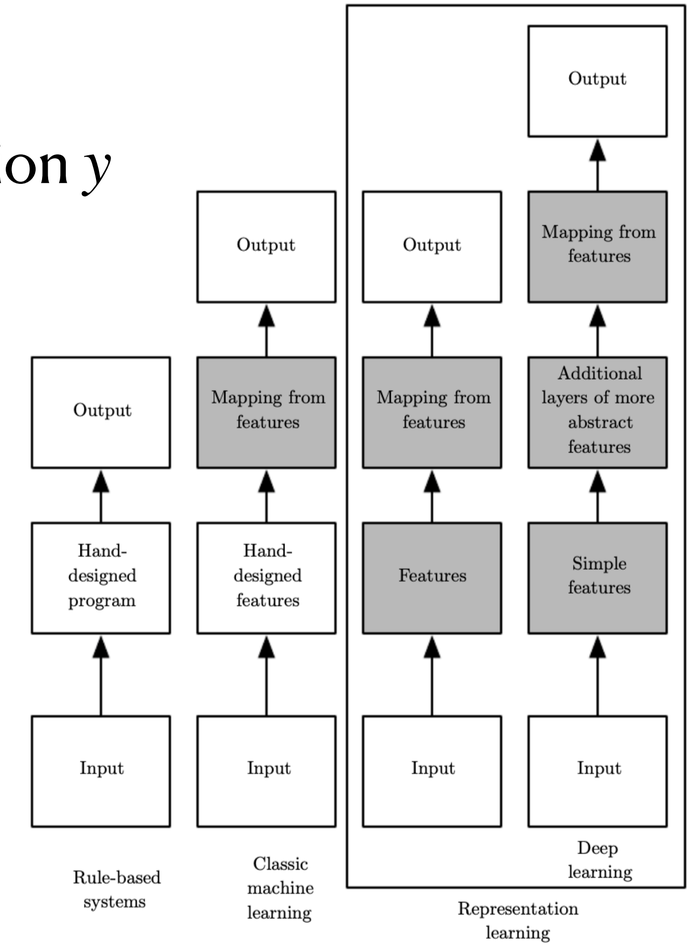
learnt from data

hand-crafted using domain knowledge

- Machine Learning with Representation Learning / Deep Learning

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

All learnt from data



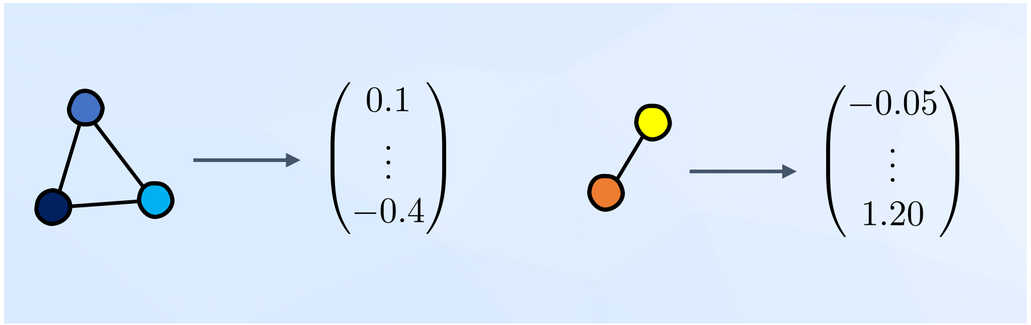
[From Pierre Vandergheynst' talk]

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

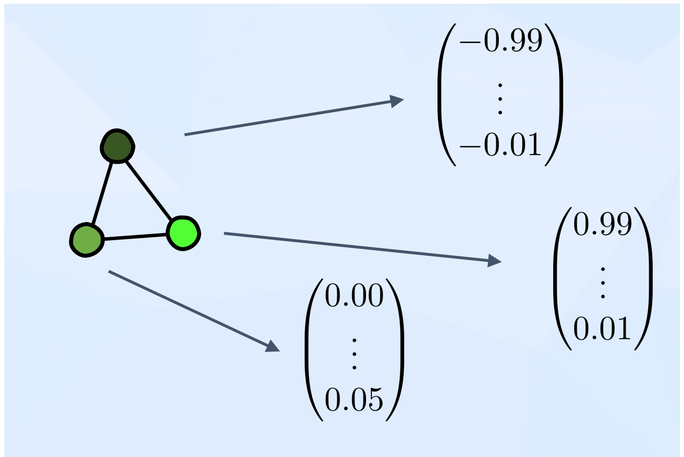
Low Level task: Graphs Representation Learning

- For Graphs, Representation learning can be summarised as:

❖ For Collection of Graphs



❖ For Nodes in a Graph



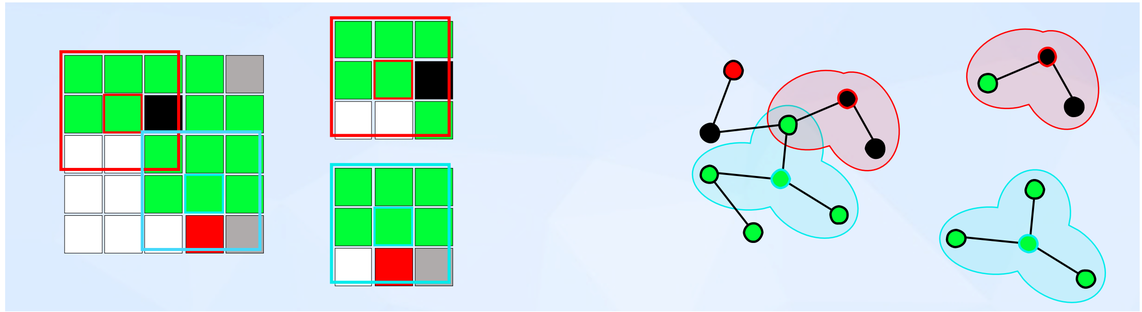
❖ Often for graphs: agglomerate Nodes representations

Low Level task: Graphs Representation Learning

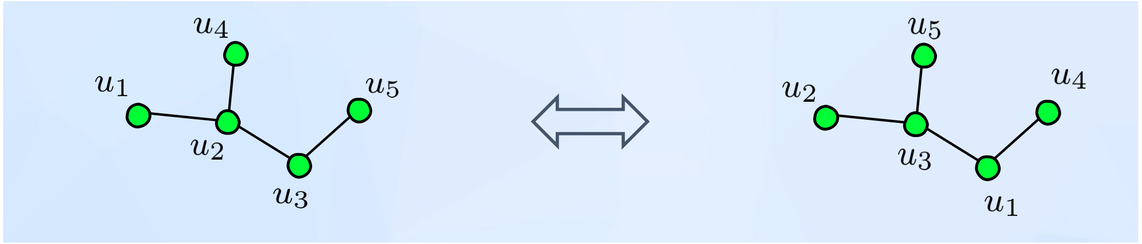
Some Associated Difficulties

- Direct comparisons of Graphs is hard / computationally challenging (e.g.; GED)

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



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



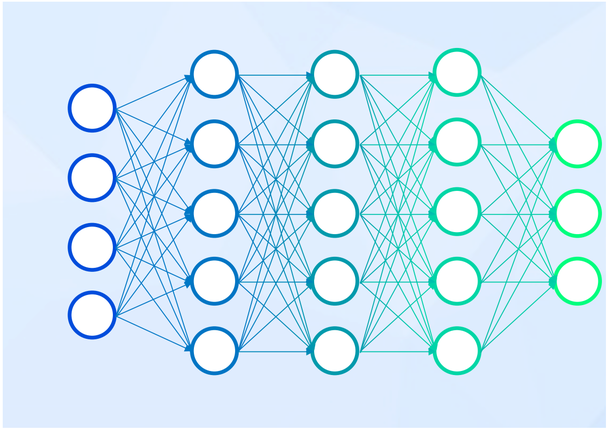
Learning and Graphs: Graph Neural Networks

- From ~2015 on: an ever growing interest to Deep models for Graph Structures

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

weighted average of input + bias/offset

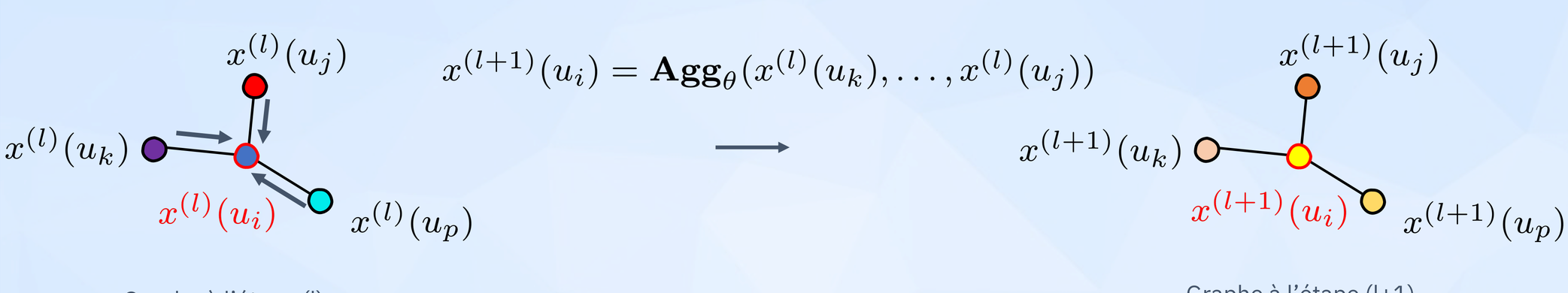
non-linear activation function



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

- For Graphs: One needs to combine information from irregular neighbourhoods

$$x^{(l+1)}(u_i) = \mathbf{Agg}_\theta(x^{(l)}(u_k), \ldots, x^{(l)}(u_j))$$

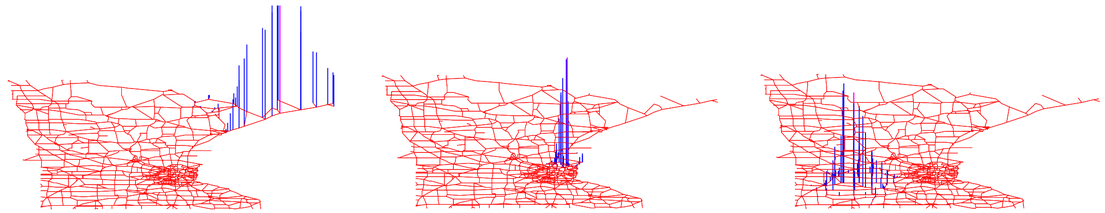


Learning and Graphs: Graph Neural Networks

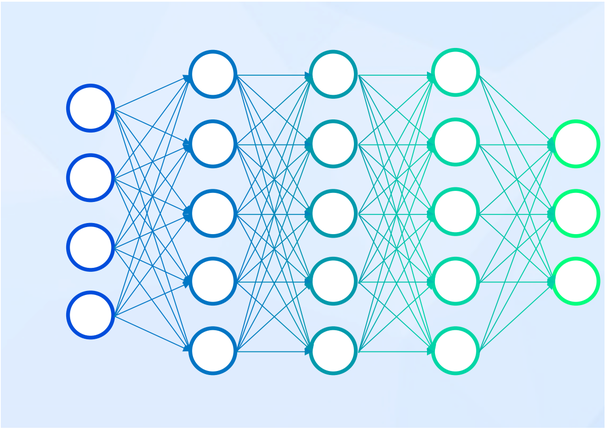
- A big success in deep learning:reduction of parameters thanks to **convolutions**

  (the same weights are re-used in a translation invariant way)

- For Graphs ? Use **translations from Graph Signal Processing**



[See Shuman et al., SP Mag 2013]



- **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

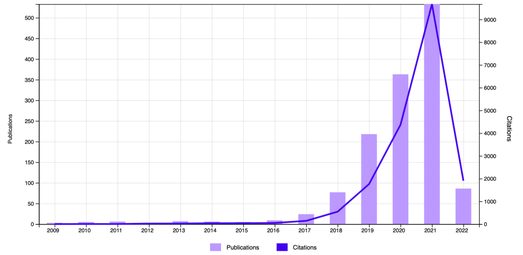
[From Pierre Vandergheynst' talk]

[Defferrard et al., 2015]

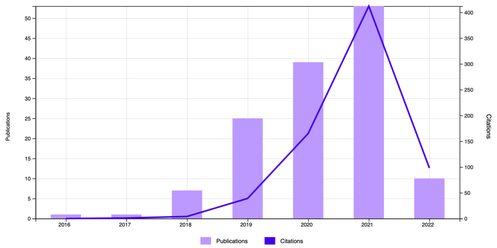
Learning and Graphs: Graph Neural Networks

- Why care ? Gives a trend to powerful methods

Citations of The Graph Neural Network Model [Scarselli et al., 2009]]



Citations of GCN [Kipf & Welling, 2017]

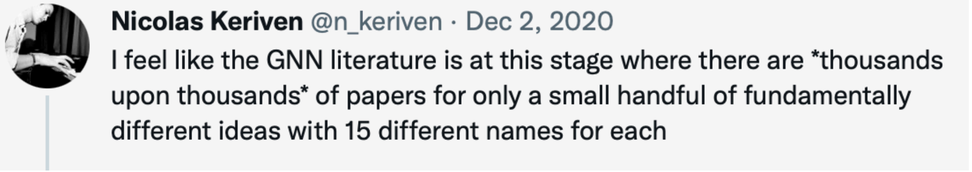


- **Strong applications :**
  - Drug Discovery ChemProp [Cell 2020];
  - Alphafold2 and Transformers use graphs
  - Drug repurposing [see S. Chepuri, 2020: Dr-COVID: graph neural networks for SARS-CoV-2 drug repurposing]
  - OpenCatalyst: discover new molecules that are catalysts for Chemistry (e.g., for fuel conversion)
- **What we will not do:** propose a new GNN architecture

Learning and Graphs: Graph Neural Networks

- **What we will not do:** propose a new GNN architecture
- (too) Many exist and there are limits associated to GNNs / GCNs, and already 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...



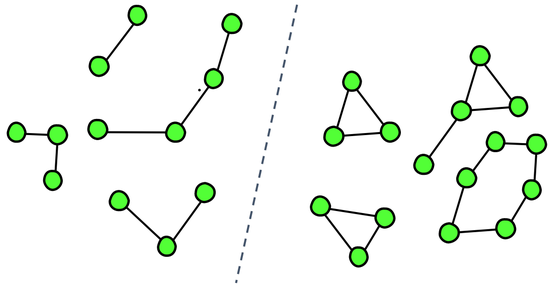
- **What we will do:** think of GNNs/GCNs as element to model a Graph Representation

The Many Sides of Graph Representation Learning

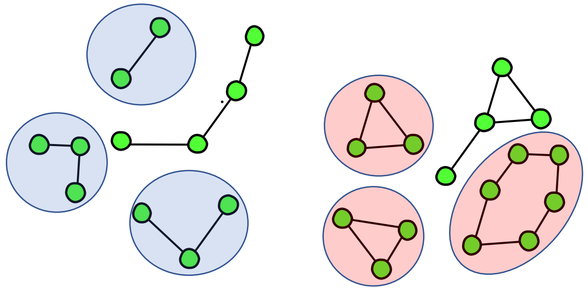
Some Recent examples from our works

- Is it possible to embed attributed graphs with an inductive & trainable representation ?

[L.. Béthune, Y. Kaloga et al, Algorithms 2020/08]



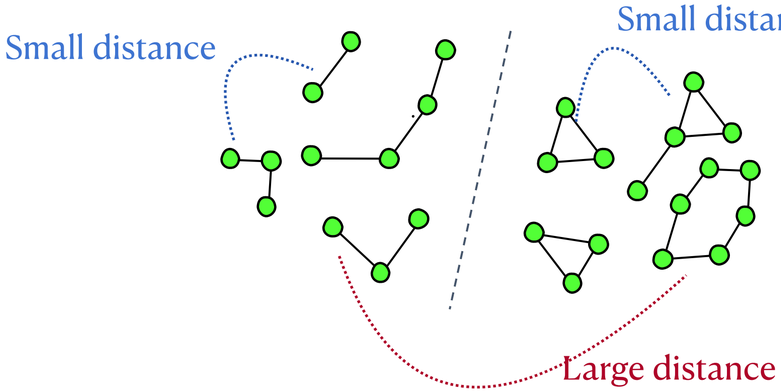
For clustering



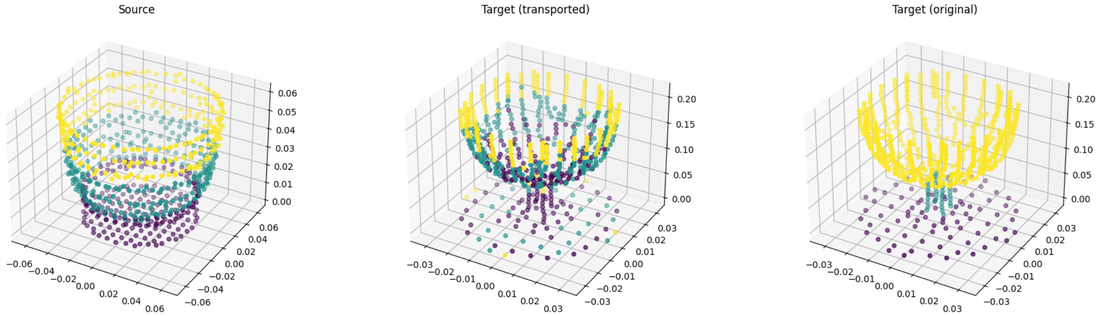
For classification

- How to learn distances between Attributed Graphs ?

[Y. Kaloga, A. Habrard, P. Borgnat, 2022]



- Is it possible to align an attributed graph to another ? e.g., so as to classify the 2nd from known classes of the 1st



[A. Barbe et al, ECML 2020/09]

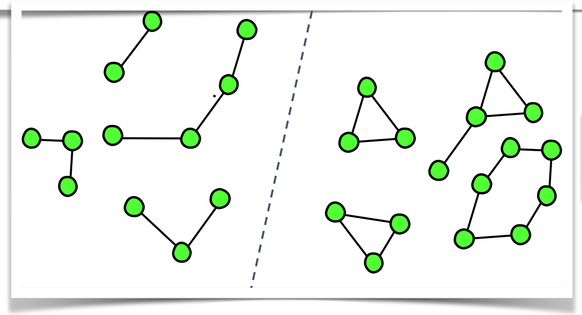
Joint work with M. Sebban, R. Gribonval, P. Gonçalves

The Many Sides of Graph Representation Learning

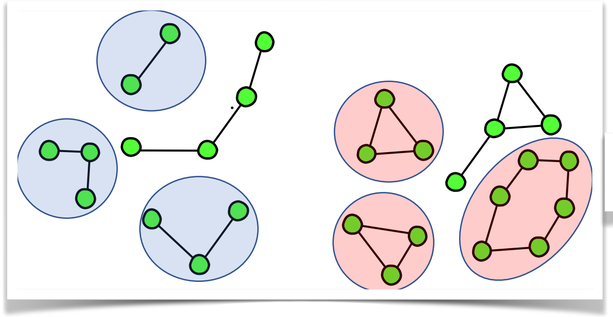
Some Recent examples from our works

- Is it possible to embed attributed graphs with an inductive & trainable representation ?

[L.. Béthune, Y. Kaloga et al, Algorithms 2020/08]



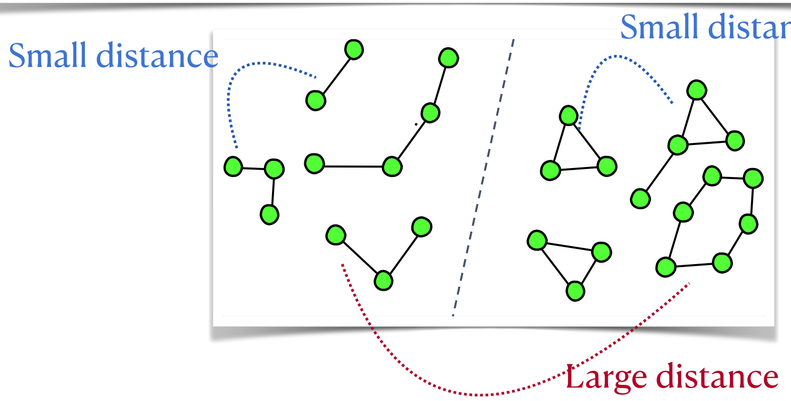
For clustering



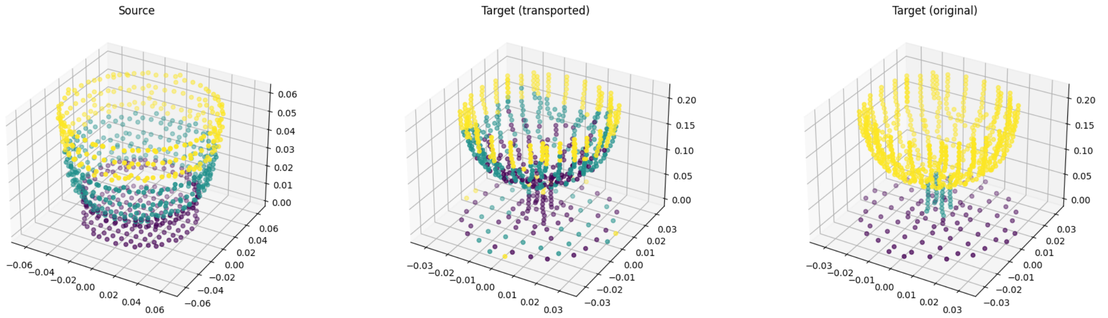
For classification

- How to learn distances between Attributed Graphs ?

[Y. Kaloga, A. Habrard, P. Borgnat, 2022]



- Is it possible to align an attributed graph to another ? e.g., so as to classify the 2nd from known classes of the 1st



[A. Barbe et al, ECML 2020/09]
Joint work with M. Sebban, R. Gribonval, P. Gonçalves

Multiscale Representation of Graphs: the Hierarchical Graph2Vec Algorithm

algorithms

MDPI

Article

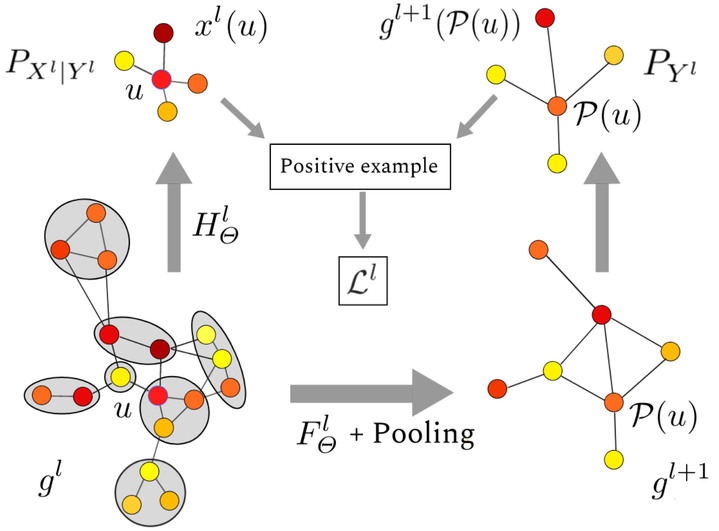
Hierarchical and Unsupervised Graph Representation Learning with Loukas's Coarsening

Louis Béthune [1,*,†], Yacouba Kaloga [1,†], Pierre Borgnat [1], Aurélien Garivier [2] and Amaury Habrard [3]

1 ENS de Lyon, UCB Lyon 1, CNRS, Laboratoire de Physique, UMR 5672, 69342 Lyon, France; yacouba.kaloga@ens-lyon.fr (Y.K.); pierre.borgnat@ens-lyon.fr (P.B.)
2 ENS de Lyon, UCB Lyon 1, CNRS, UMPA UMR 5669 and LIP UMR 5668, 69342 Lyon, France; aurelien.garivier@ens-lyon.fr
3 Laboratoire Hubert Curien, UJM-Saint-Etienne, UMR 5516, 42100 Saint-Étienne, France; amaury.habrard@univ-st-etienne.fr
* Correspondence: louis.bethune@ens-lyon.fr
† These authors contributed equally to this work.

Received: 6 July 2020; Accepted: 17 August 2020; Published: 21 August 2020

check for updates



Multiscale Representation of Graphs: the Hierarchical Graph2Vec Algorithm

The Main Objectives ; Why another method ?

- Several Methods for 1 graph: Node Embeddings; Node2Vec; GraphWave,...

- Several Methods for a collection of graphs: Concatenation of Node Embeddings; **Graph2Vec**; DeepWalk; Kernels; Infograph (2020) (close to HG2V)

Multiscale Representation of Graphs: the Hierarchical Graph2Vec Algorithm

The Main Objectives ; Why another method ?

The properties of a graph may depend on several of its scales

- ❖The representation function must take into account these different scales

Graphs close structurally and/or in terms of attributes should have close representation

- ❖The representation function must be "continuous" in structure and attributes

Labels are expensive to acquire

- ❖The representation function should operate even without a label: unsupervised learning

More and more data and more and more energy-intensive training procedures

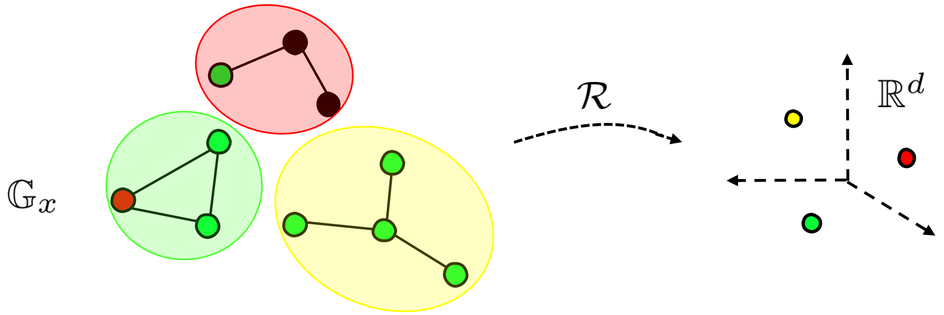
- ❖It must be possible to analyse a large number of graphs
- ❖The representation function must be able to represent graphs never seen before: inductivity

able In addition: able to specifically focus on certain aspects for the representation

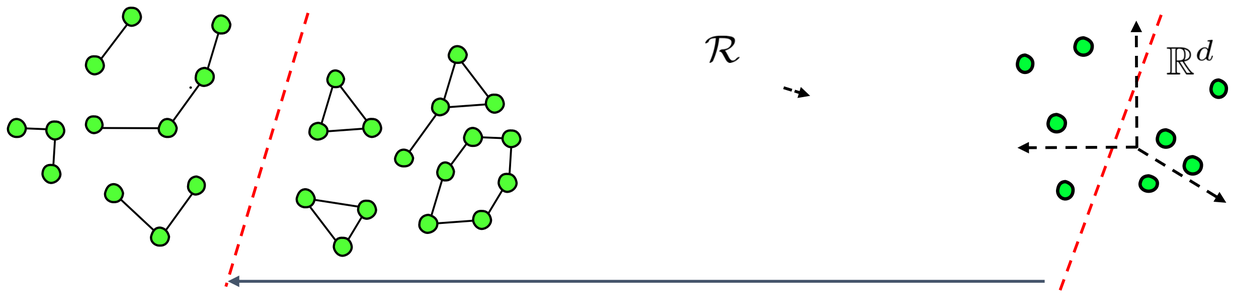
- ❖The function must be end-to-end differentiable

HG2V: a Classical General Framework

- From Attributed Networks to Euclidean Vectors



- Obtained Representations can be used for learning



HG2V: State of the art (partial view)

- **In Machine Learning / Computer Science**
  - ✦ Heuristic methods measuring similarities between graphs
  - ✦ Indirect representations through scalar products / kernel trick
  - ✦ Quadratic Methods for kernels

- **Approaches in Deep Learning**
  - ✦ Neural networks for encode graph features
  - ✦ Auto-Encoders
  - ✦ Negative sampling

➡ **Obtained Representations can be used for learning**

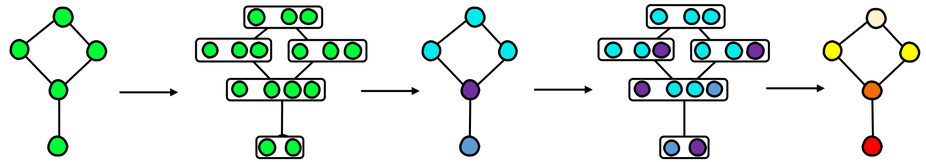
For HG2vec = we require the properties of previous slide

- **Insights from Graph Signal Processing**
  - ✦ Interplay between Structure (graph) & Attributes (signals / features)
  - ✦ Multiscale view is required

Method	Continuous Attributes	Complexity (Training)	Complexity (Inference)	End-to-End Differentiable	Supervised
Kernel methods, e.g., **WL**-OA [27], **WWL** [4]	✓	$\mathcal{O}(N^2)$*	$\mathcal{O}(N)$*	✗	✗
Graph2Vec [2]	✗	$\mathcal{O}(N)$	✗	✗	✗
GIN [32], DiffPool [17], MinCutPool [18]	✓	$\mathcal{O}(N)$	$\mathcal{O}(1)$	✓	✓
HG2V (Section 4), Infograph [13]	✓	$\mathcal{O}(N)$	$\mathcal{O}(1)$	✓	✗

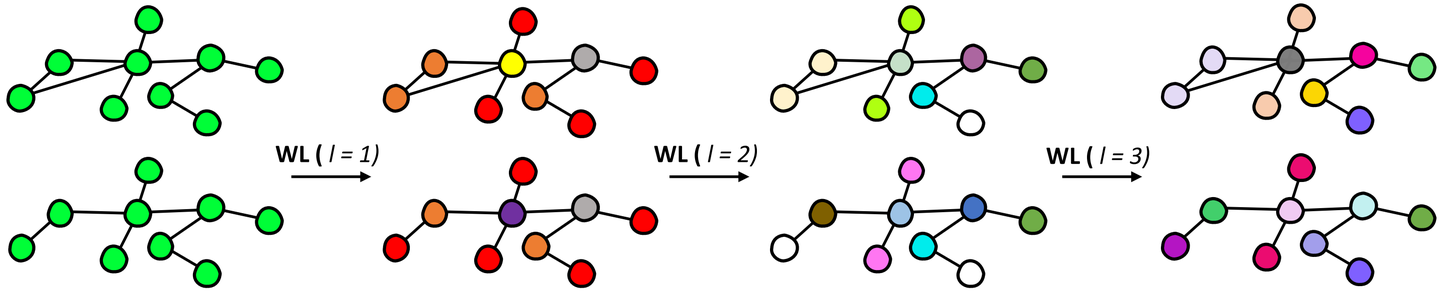
Characterize a Graph from its Nodes

- The **Weisfeiler-Lehman Test** – Characterize a Graph by its Sub-Trees



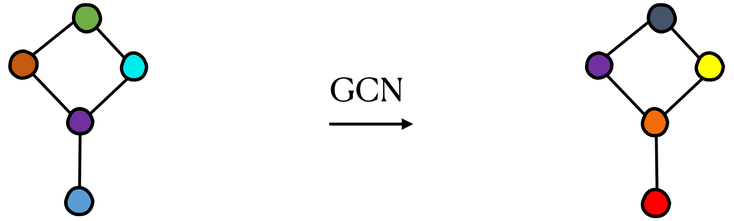
- Does not work for continuous attributes –> Use Graph Neural Networks

- Does not consider Global Structures

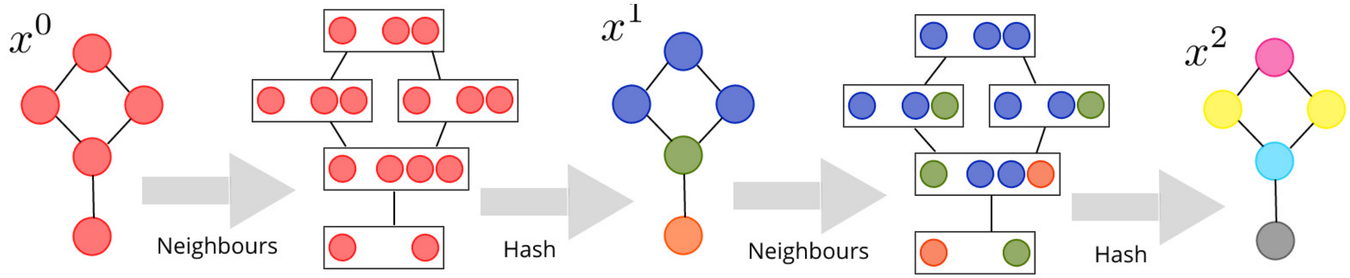


Characterize a Graph from its Nodes

- From Weisfeiler-Lehman to GNN (here Conv. one)



- Can use continuous attributes (or discrete ones)
- Continuous with respect to topology of the graph (not explained here)
- Discriminative power can be as good as WL (see GIN in "How Powerful are Graph Neural Networks ?" ICLR 2019)
- GNN can be aptly stacked if one chooses a correct one (e.g., GCN of [Kipf&Welling, ICLR 2017] ; Truncated Krylov [Luan et al. NeurIPS 2019]

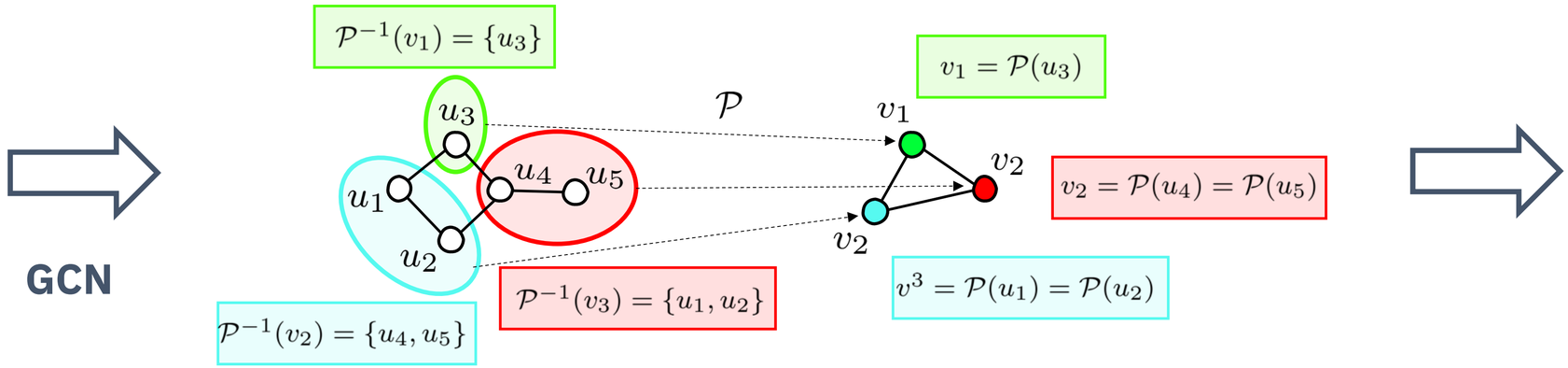


Characterize a Graph from its Nodes: the issue of Pooling and of the Global View

- Problem with GNN: impossible to characterise several global properties,

cf. A. Loukas et al. “What graph neural networks cannot learn: deepth vs. width“ ICLR 2020

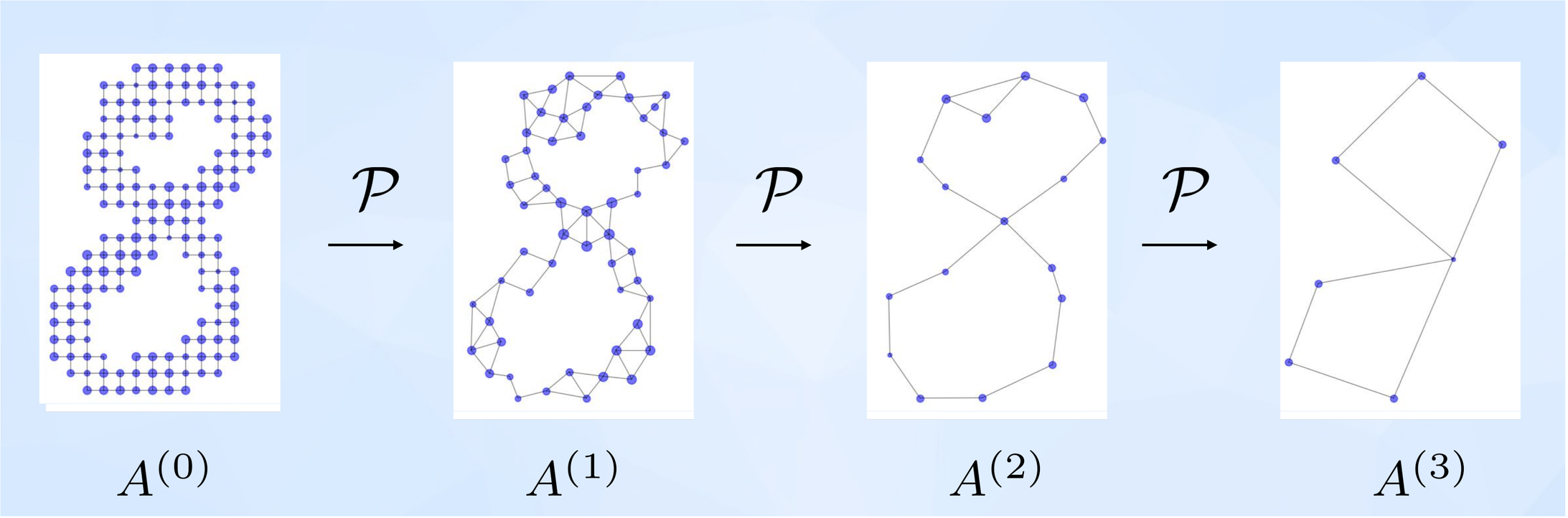
- Proposed architecture: use a Pooling method between GCN layers -> **Hierarchical method**



- Allows to change the scale from one level to another
- Pooling can have various definition – Key point = how to cope with irregularities
- [Loukas, 2019] Spectral pooling that keeps the structure, thanks to Laplacian Spectal properties of the associated pooling

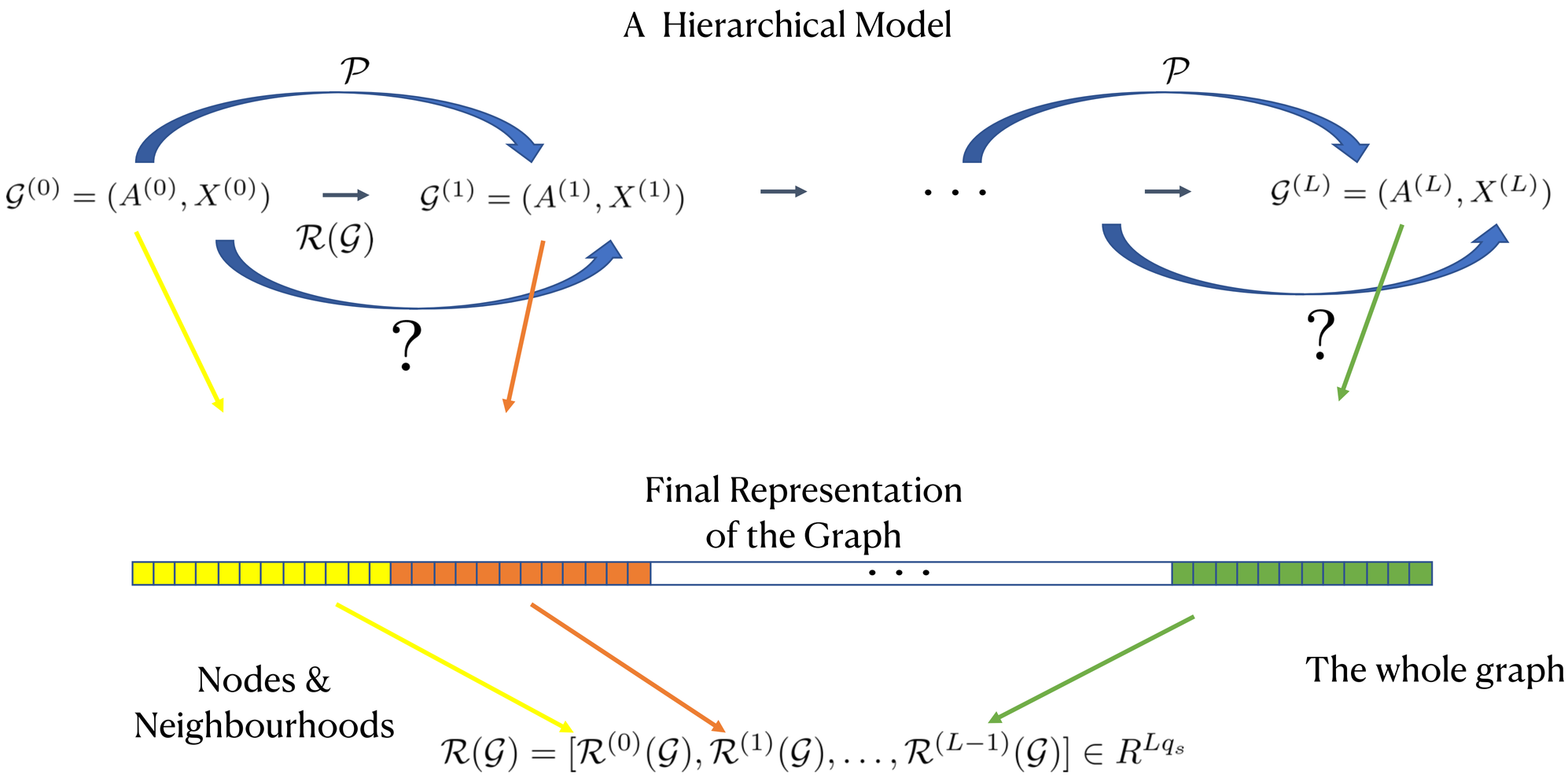
Characterize a Graph from its Nodes: the issue of Pooling

- [Loukas, 2019] Spectral pooling that keeps the structure, thanks to Laplacian Spectal properties of the associated pooling



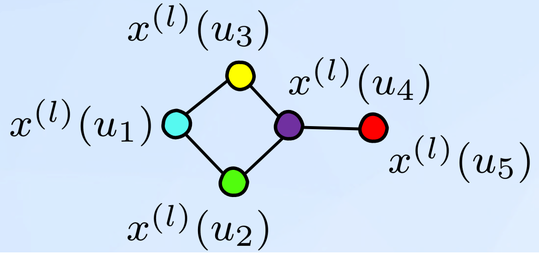
- Each step preserves the global (low-pass) structures of the graph

HG2V: Overview of the proposed method



HG2V: More details of the attributes of Nodes and Neighbourhoods

- Each node has a representation per scale, that will be used in the global representation

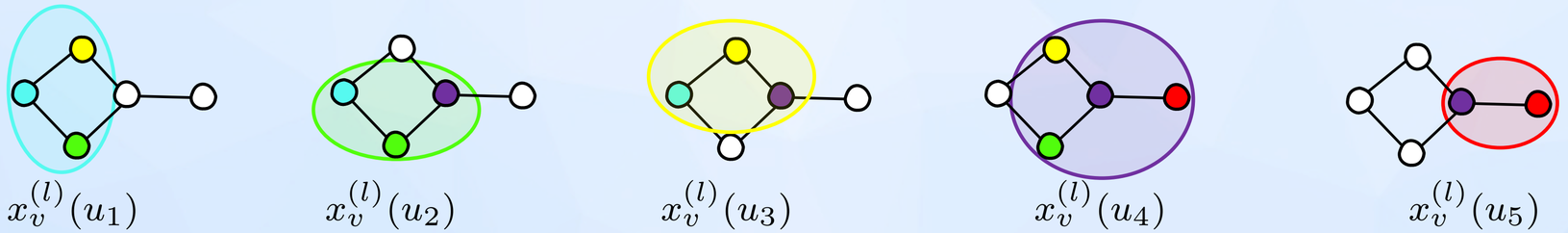


$$x^{(0)}(u_i) = M_\theta(x(u_i))$$

$$x^{(l)}(u_i) = F_\theta^{(l)}(\bigcup_{u \in \mathcal{P}^{-1}(u_i)} \{x^{(l-1)}(v) | v \in \mathcal{V}_p(u)\})$$

$$\mathcal{R}^{(l)}(\mathcal{G}) = \left(\sum_{u \in V} x^{(l)}(u)\right)^T \in \mathbb{R}^{1 \times q_s}$$

- Each neighbourhood is also represented (here: neighbourhoods of level 1)

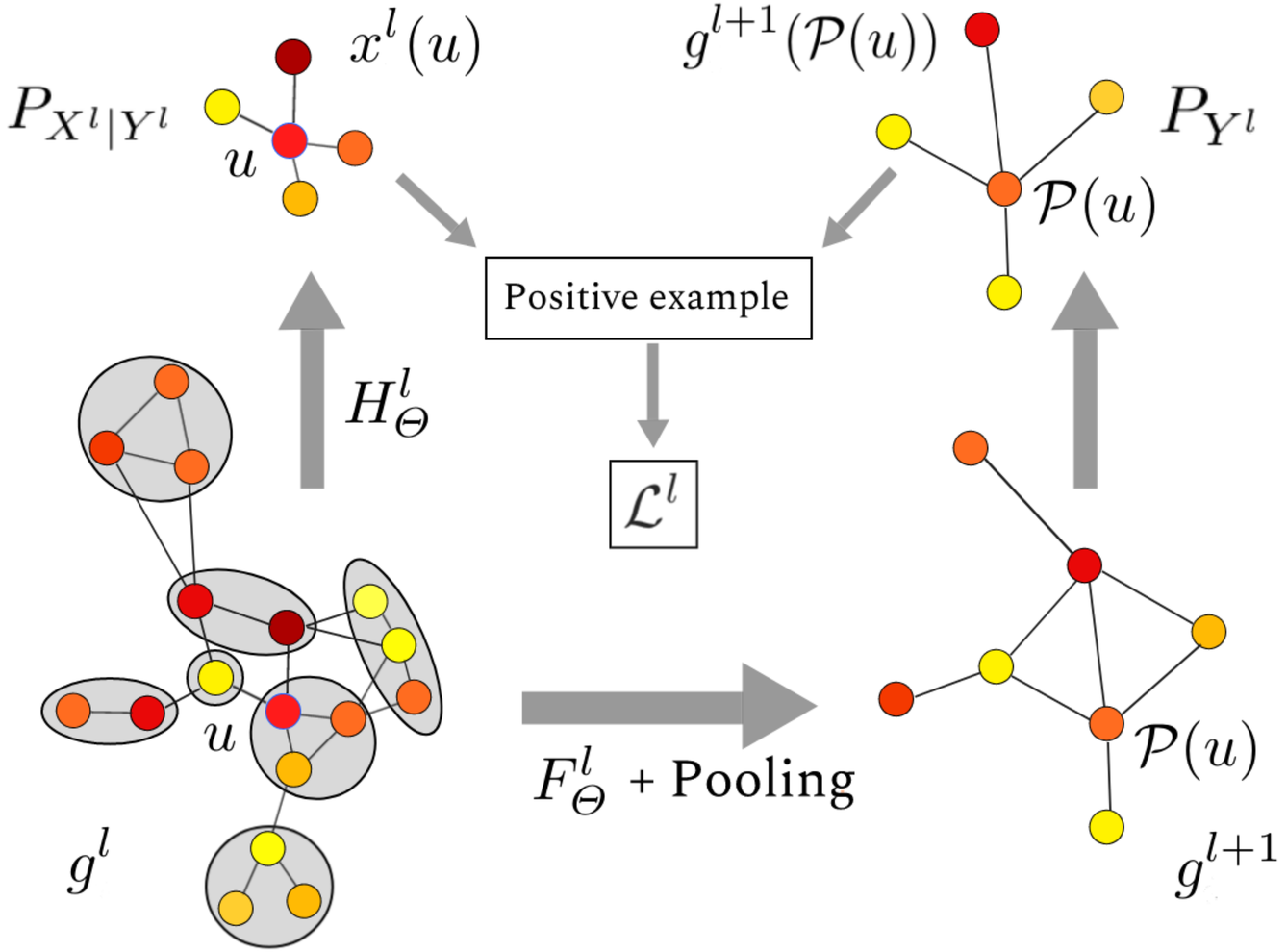


$$x_v^{(l)}(u_i) = H_\theta^{(l)}(\{x^{(l)}(u_j) | u_j \in \mathcal{V}_p(u_i)\})$$

HG2V: More details about the Learning procedure = Negative sampling

e..g. Hjelm et al. "Learning deep representations by mutual information estimation and maximization. ICLR 2019

- Sketch one level of the hierarchy: **sampling positive examples and negative examples** (sampled from independent prob.), we ensure that nodes and neighbors **minimize the cross-entropy**



HG2V: More details about the Learning procedure = Negative sampling

e..g. Hjelm et al. "Learning deep representations by mutual information estimation and maximization. ICLR 2019

- Mutual Information to be minimised

$$\mathcal{L}^{(l)} = \mathbb{E}_{(x_v^{(l)}(u), x^{l+1}(w)) \sim P(\mathbb{X}_v^{(l)}, \mathbb{X}^{(l+1)})} \log \tau(x_v^{(l)}(u) \cdot x^{l+1}(w)) + \mathbb{E}_{(x_v^{(l)}(u), x^{(l+1)}(w)) \sim P(\mathbb{X}_v^{(l)}) \otimes P(\mathbb{X}^{(l+1)})} \log \tau(-x_v^{(l)}(u) \cdot x^{(l+1)}(w))$$

- Aggregated Score as loss function / Training with usual algorithms (SGD with Adam)

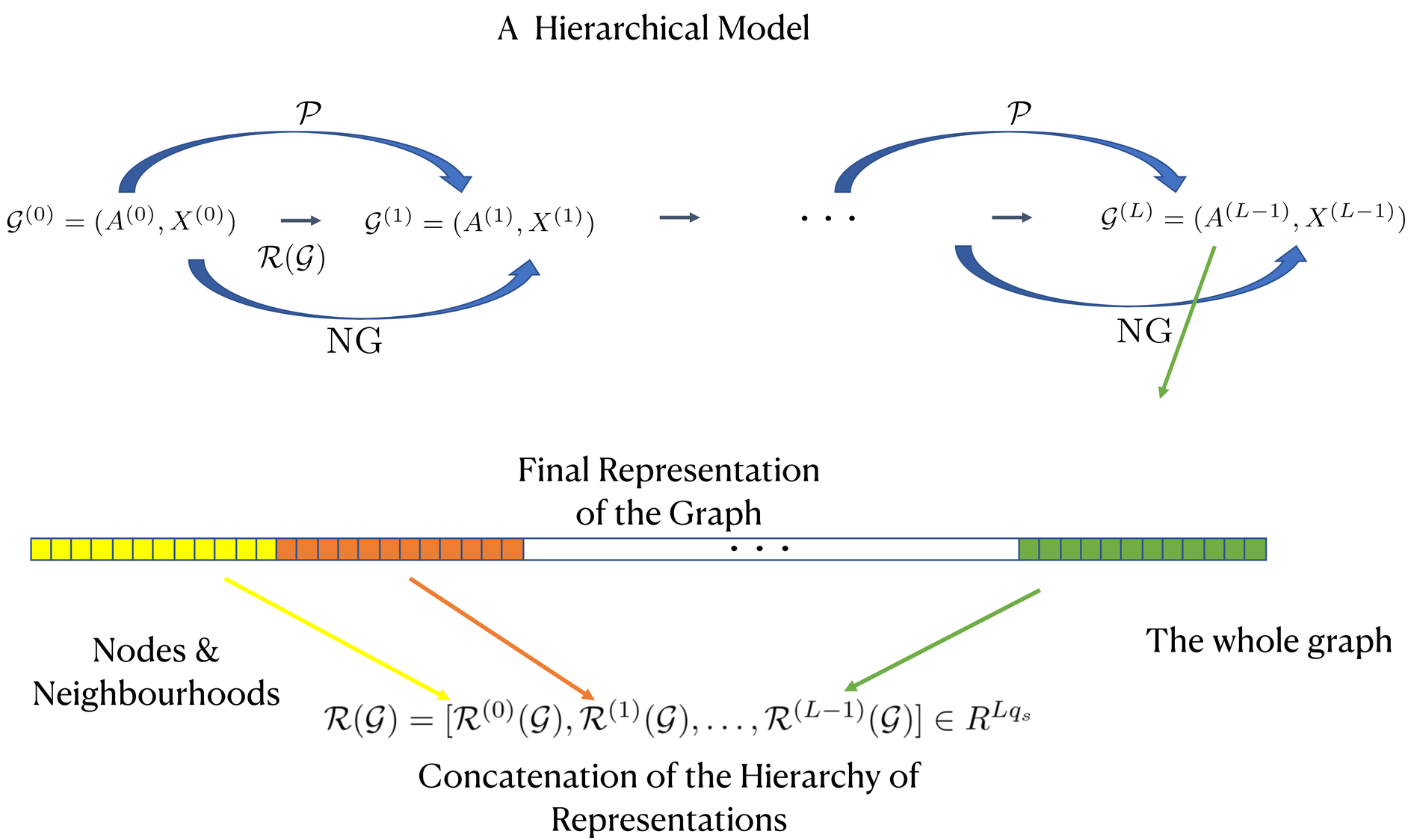
$$\min_\theta \sum_{\mathcal{G} \in \mathbb{G}_x} \sum_{l \in \{0,\ldots,L-1\}} \mathcal{L}_{\mathcal{G}}^{(l)}$$

- Final Representation = concatenation of nodes' repr. at all scales

$$\mathcal{R}^{(l)}(\mathcal{G}) = \left(\sum_{u \in V} x^{(l)}(u)\right)^T \in \mathbb{R}^{1 \times q_s}$$

$$\mathcal{R}(\mathcal{G}) = [\mathcal{R}^{(0)}(\mathcal{G}), \mathcal{R}^{(1)}(\mathcal{G}), \ldots, \mathcal{R}^{(L-1)}(\mathcal{G})] \in R^{Lq_s}$$

HG2V: Overview of the proposed method



HG2V: Numerical experiments – Dataset

	Méthodes	#graphe	#classe	#noeud moy.	#arête moy.
Biologie/Chimie	PROTEIN	1113	2	39.06	72.82
	ENZYMES	600	6	32.63	64.14
	DD	1178	2	284.32	715.66
	NCI1	4110	2	29.87	32.30
	NCI109	4127	2	29.68	32.13
	MUTAG	188	2	17.93	19.79
	PTC_FR	351	2	14.56	15
Réseaux	IMDB BINARY	1000	2	19.77	96.53
	IMDB MULTI	1500	3	13.00	65.94
	REDDIT BINARY	2000	2	429.63	497.75
	REDDIT 5K	4999	5	508.82	594.87
Synthétiques	FRANKENSTEIN	4337	2	16.90	17.88
	DLA	1000	2	500	499
Images	MNIST	10000	10	144	237.76
	USPS	9298	10	89.04	141.35
	FASHION-MNIST	10000	10	366.13	672.36

HG2V: Numerical experiments – Supervised Classification with SVM

Jeu de données	HG2V	Graph2Vec	Infograph	DiffPool (supervisé)	GIN (supervisé)	MinCutPool (supervisé)	WL-OA (noyau)	WWL (noyau)
IMDB-m	47.9 ± 1.0	$\mathbf{50.4 \pm 0.9}$	49.6 ± 0.5	45.6 ± 3.4	48.5 ± 3.3	✗	✗	✗
PTC_FR	$\mathbf{67.5 \pm 0.5}$	60.2 ± 6.9	✗	✗	✗	✗	$\mathbf{63.6 \pm 1.5}$	✗
FRANK.	$\mathbf{65.3 \pm 0.7}$	60.4 ± 1.3	✗	✗	✗	✗	✗	✗
MUTAG	81.8 ± 1.8	83.1 ± 9.2	$\mathbf{89.0 \pm 1.1}$	✗	✗	✗	84.5 ± 1.7	$\mathbf{87.3 \pm 1.5}$
IMDB-b	71.3 ± 0.8	63.1 ± 0.1	$\mathbf{73.0 \pm 0.9}$	68.4 ± 3.3	71.2 ± 3.9	✗	✗	$\mathbf{74.4 \pm 0.8}$
NCI1	76.3 ± 0.8	73.2 ± 1.8	✗	76.9 ± 1.9	$\mathbf{80.0 \pm 1.4}$	✗	$\mathbf{86.1 \pm 0.2}$	85.8 ± 0.2
NCI109	$\mathbf{75.6 \pm 0.7}$	74.3 ± 1.5	✗	✗	✗	✗	$\mathbf{86.3 \pm 0.2}$	✗
ENZYMES	$\mathbf{66.0 \pm 2.5}$	51.8 ± 1.8	✗	59.5 ± 5.6	59.6 ± 4.5	✗	59.9 ± 1.1	$\mathbf{73.3 \pm 0.9}$
PROTEINS	75.7 ± 0.7	73.3 ± 2.0	✗	73.7 ± 3.5	73.3 ± 4.0	$\mathbf{76.5 \pm 2.6}$	76.4 ± 0.4	$\mathbf{77.9 \pm 0.8}$
MNIST	$\mathbf{96.1 \pm 0.2}$	56.3 ± 0.7	✗	✗	✗	✗	✗	✗
D&D	79.2 ± 0.8	58.6 ± 0.1	✗	75.0 ± 3.5	75.3 ± 2.9	$\mathbf{80.8 \pm 2.3}$	79.2 ± 0.4	$\mathbf{79.7 \pm 0.5}$
REDDIT-b	91.2 ± 0.6	75.7 ± 1.0	82.5 ± 1.4	87.8 ± 2.5	89.9 ± 1.9	$\mathbf{91.4 \pm 1.5}$	89.3	✗
DLA	$\mathbf{99.9 \pm 0.1}$	77.2 ± 2.5	✗	✗	✗	✗	✗	✗
REDDIT-5K	55.5 ± 0.7	47.9 ± 0.3	53.5 ± 1.0	53.8 ± 1.4	$\mathbf{56.1 \pm 1.7}$	✗	✗	✗

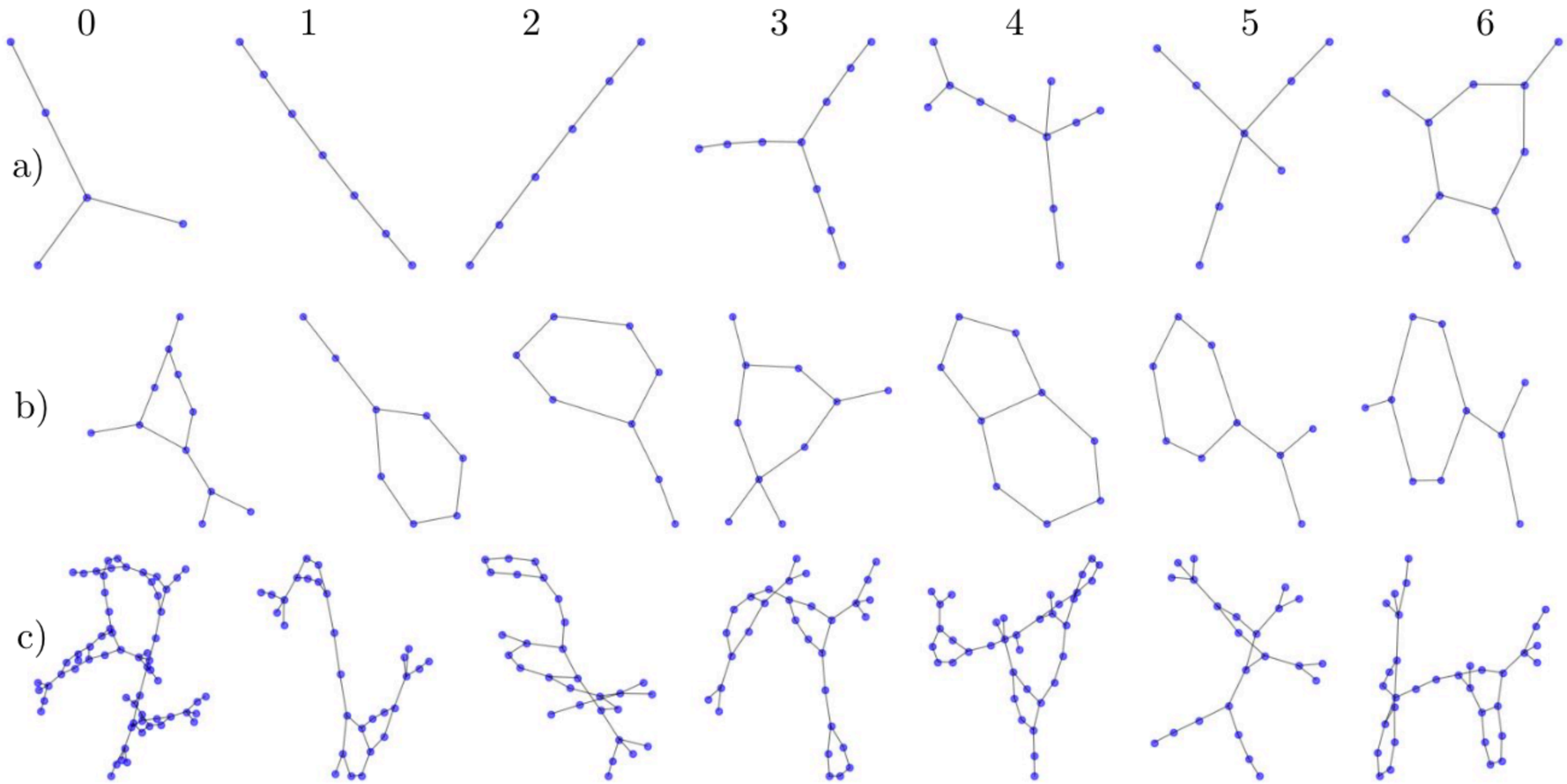
Reference Method

Similar Performance

High variability
Performance worse

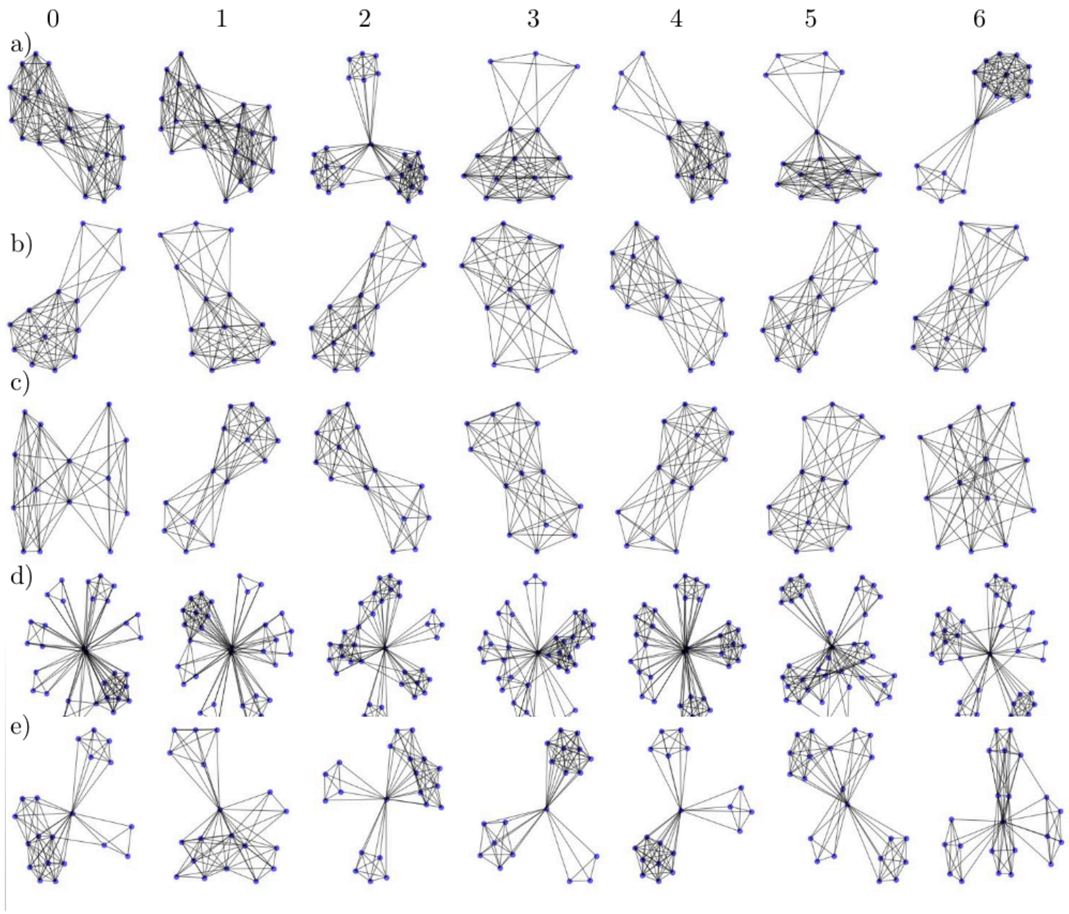
Best Performance
Quadratic in cost

HG2V: Numerical experiments – Visualisation (I)

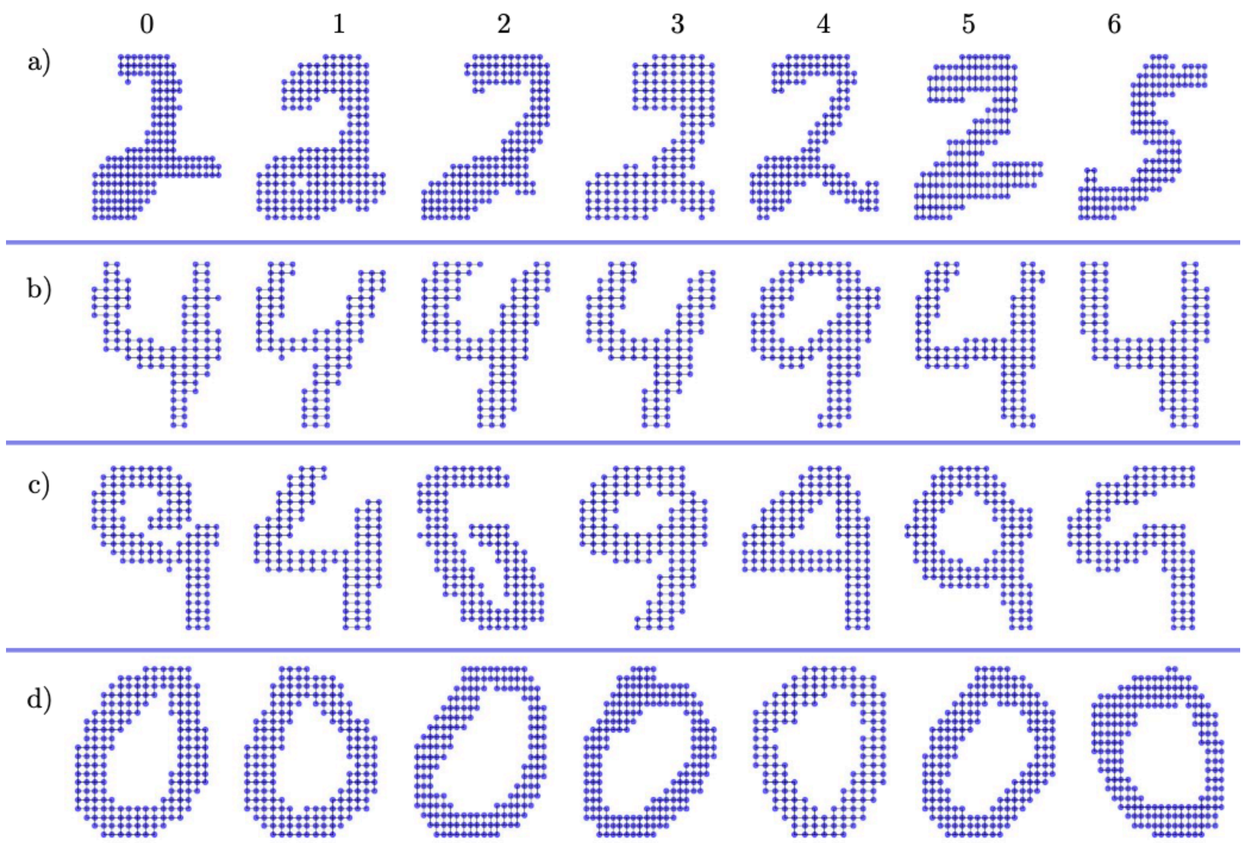


PTC dataset

HG2V: Numerical experiments – Visualisation (II)



IMDB-b



MNIST

HG2V: Conclusion on this part

- **Additional experiments & features**
  - ✦ Ablative study -> Loukas' pooling needed on large graphs ; GNN needed on continuous features
  - ✦ Experiments in inductive learning & transfer learning
  - ✦ End-to-End differentiability

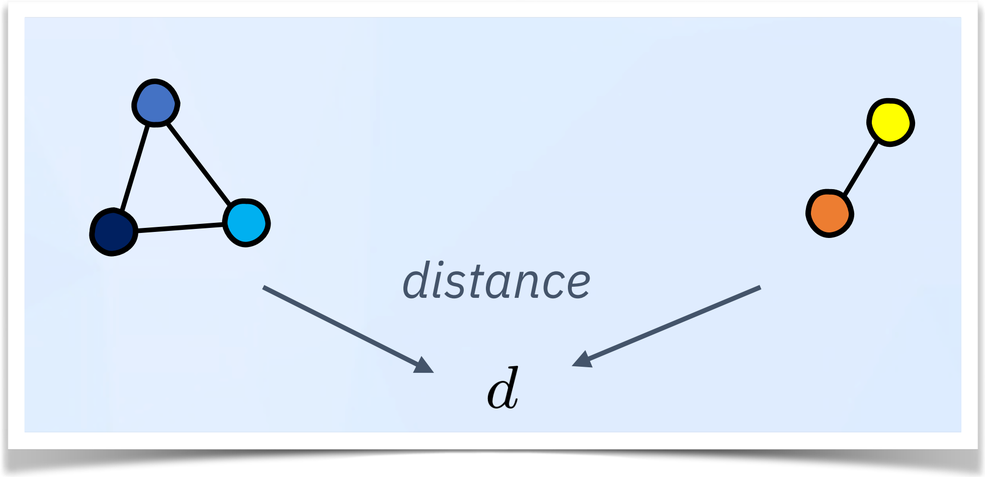
- **To wrap-up**
  - ✦ A good, state-of-the-art, representation method for attributed graphs
  - ✦ Linear in the number of input graphs -> scalable
  - ✦ Training is unsupervised and inductive
  - ✦ Can be incorporated in larger models to solve whatever task

➡ **HG2vec** = $\mathbb{G}_x \xrightarrow{\mathcal{R}} \mathbb{R}^d$

A Simple Model to Learn Metrics Between Attributed Graphs

From Yacouba Kaloga's thesis ; 12/2021

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



An even lower-level task: compute distances

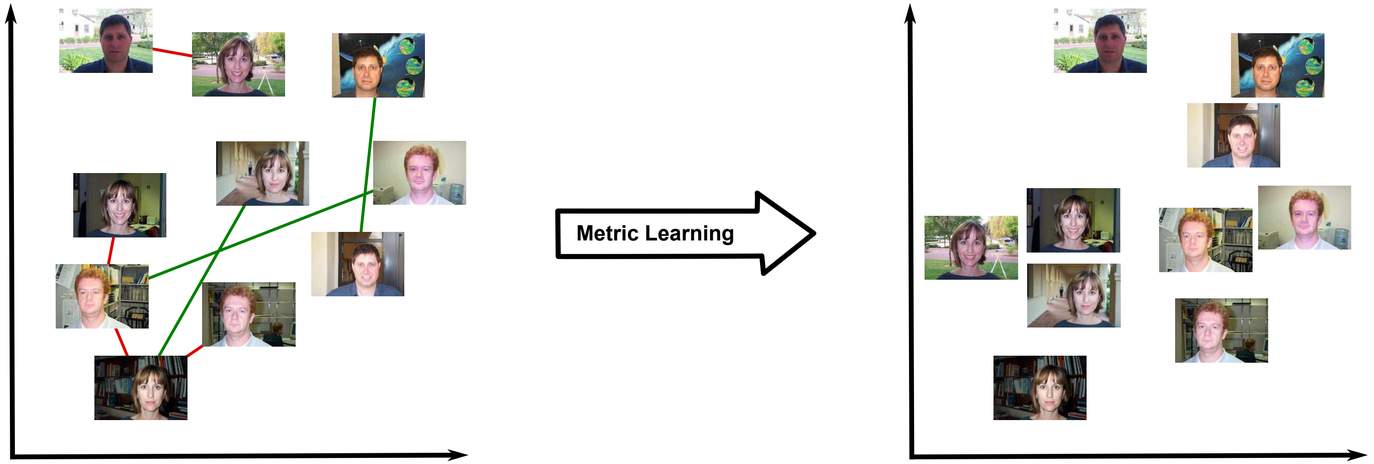
- Why ? At the input of many (many!) methods

  “Real“ distances between graphs are often hard to compute (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 Existing Works

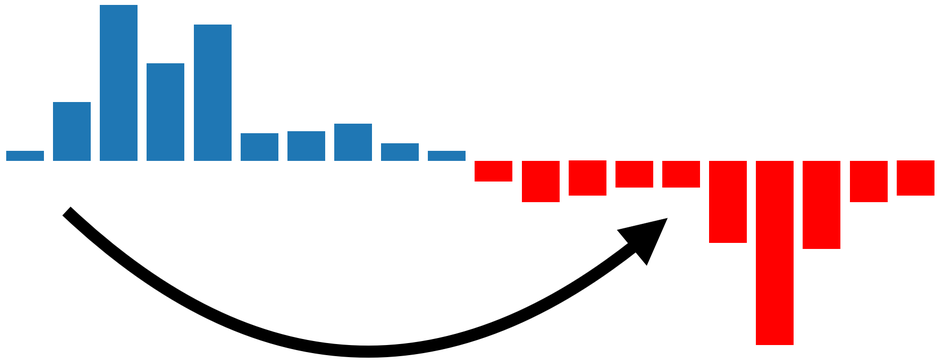
- The main objective is to **jointly code for topologies & attributes**
- Existing Solutions :
  - ❖ In ML: low scalability when methods rely of GED (Graph Edit Distance)
  - ❖ In GSP: some works where topology G is set and distances between attributes on G

    [Graph Optimal Transport, Maretic et al. NeuRIPS 2019]
  - ❖ With kernels: usually nonparametric (exception multiple kernel learning)
  - ❖ Change the point-of-view: **Optimal Transport** between distributions

Optimal Transport ~~for Graphs or Attributed Graphs~~

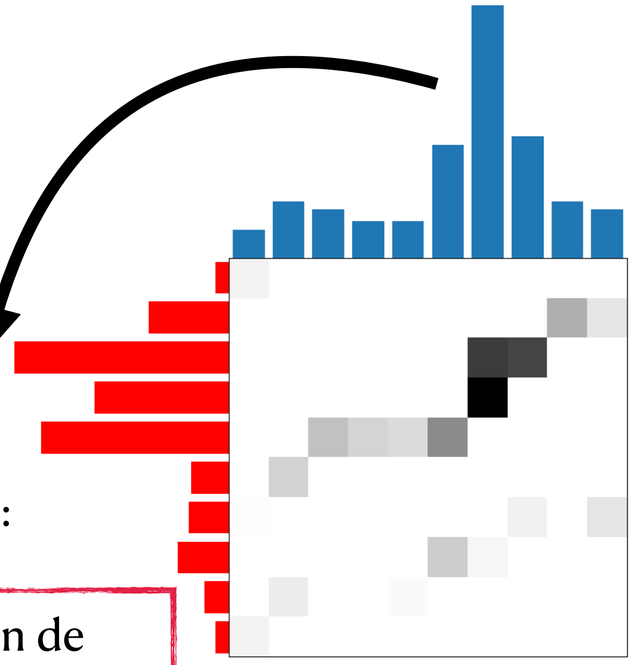
- Optimal Transport: compute a distance between 2 distributions, while finding the optimal coupling (or transport plan) between these 2
- cf. "Computational Optimal Transport" (G. Peyré & M. Cuturi), 2019

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



Problème de Monge : « Mémoire sur la théorie des déblais et des remblais », 1776

Une solution :



Avec relaxation de Kantorovich

Optimal Transport ~~for Graphs or Attributed Graphs~~

- **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 build the **2-Wasserstein distance** $\mathscr{W}_2$ as:

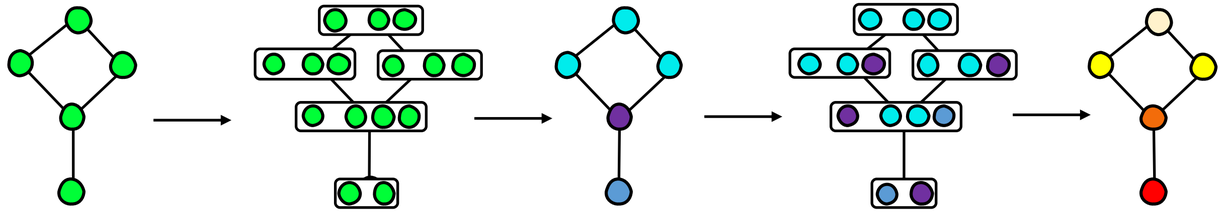
$$\mathscr{W}_2(\mu, \nu) = \inf_{\pi_{i,j} \in \Pi_{a,b}} \left(\sum_{i,j=1}^{n,n'} \pi_{i,j} c(\mathbf{x}_i, \mathbf{x}'_j)^2 \right)^{\frac{1}{2}}$$

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

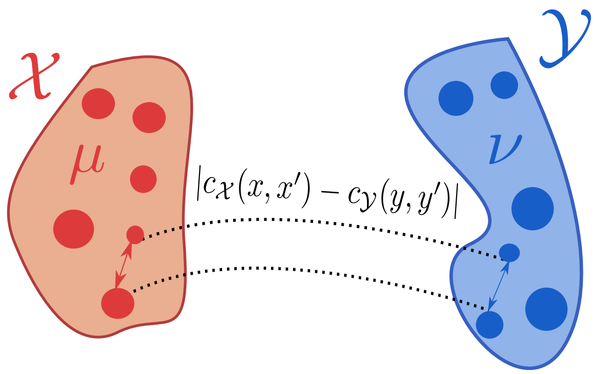
whose marginals are the distributions $\mu = \sum_{\mathbf{x}'_i \in \mathbb{X}'} \pi(\cdot, \mathbf{x}'_i)$ and $\nu = \sum_{\mathbf{x}_i \in \mathbb{X}} \pi(\mathbf{x}_i, \cdot)$

Optimal Transport for Graphs ~~or Attributed Graphs~~

- For Graphs: one has to **Associate a distribution to a graph**
  - We already have seen one: the Weisfeiler-Lehman method
  - cf. Togninalli et al., "Wasserstein Weisfeiler-Lehman graph kernels" NeurIPS 2019



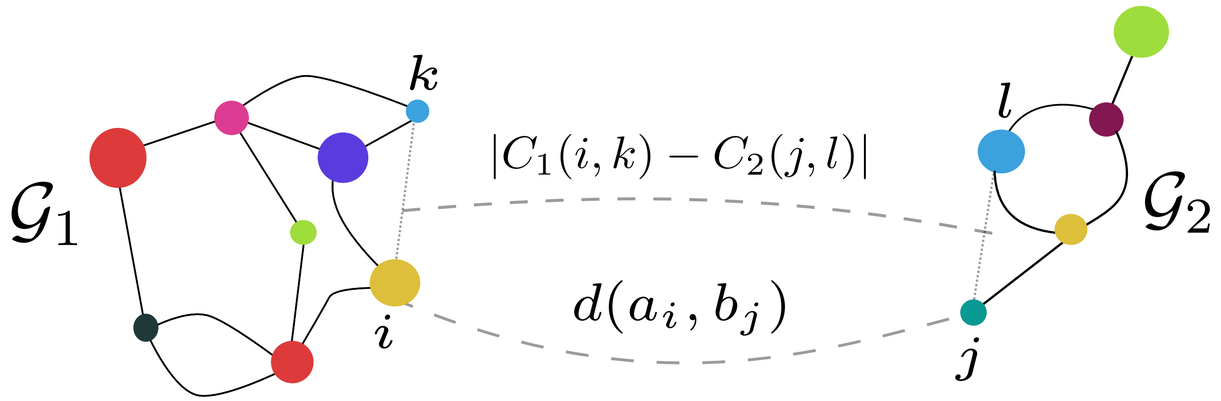
- A second solution: **The Gromov Wasserstein distance**
  - [Mémoli, Found. Comp. Math. 2011; Peyré, Cuturi, Solomon, ICML 2016]
  - structures are compared through their pairwise distances
  - cf. also N. Courty, R. Flamary, T. Vayer [PhD 2020]



Optimal Transport for ~~Graphs or~~ Attributed Graphs

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

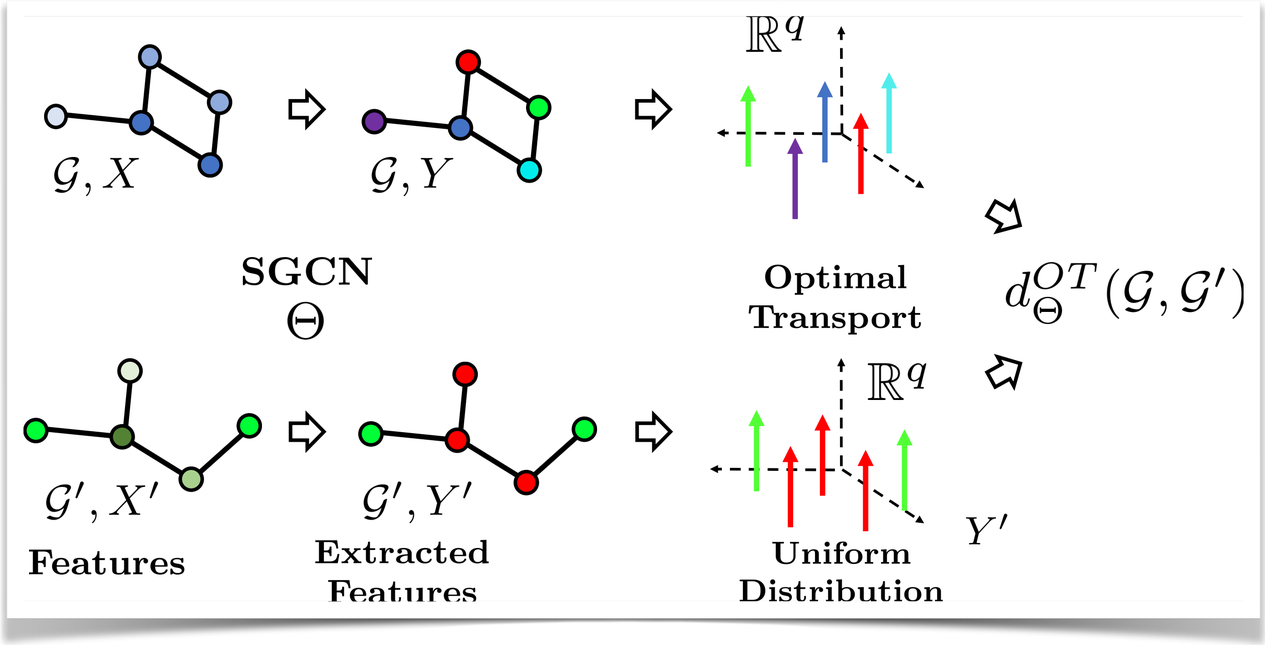
  - “Fused Gromov-Wasserstein distance“ [Vayer et al., ICML 2019]



- Shameful advertisement: see **The Diffusion Wasserstein distance,** [Barbe et al. 2020-201]

Optimal Transport for Attributed Graphs, with Metric Learning

- The idea is to **parametrize (graphs+attributes) through a GCN**
  - Then **compute distance** between them **by optimal transport**



- **Trainable parameters:** the parameters of the GCN
  - Use an understandable GCN with few parameters (hence: not a deep one); **Simple GCN**
  - **[Wu et al. "Simplifying graph convolutional networks". PLMR 2019]**

Optimal Transport for Attributed Graphs, with Metric Learning and Reduced Computational Load

- For Optimal Transport: Use the **Sliced methods**

  - **[N. Bonneel et al., "Sliced and Radon Wasserstein barycenters of measures", JMIV 2015]**

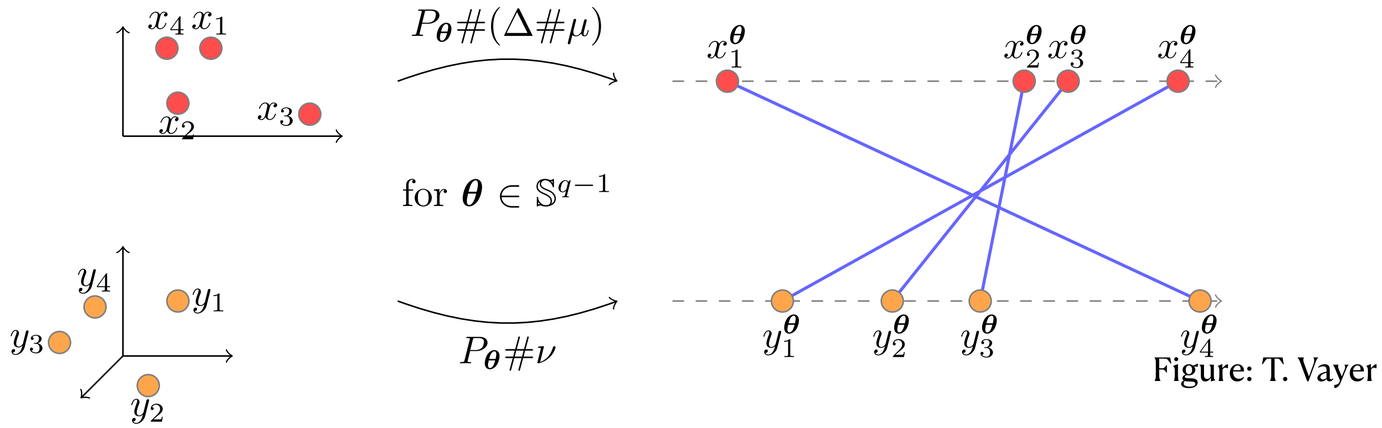


Figure: T. Vayer

- To Extract Features for Attributed Graphs: **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 ; New $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_n$ Features $\mathbf{Y}$ as

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

Metric Learning for Attributed Graphs, with Optimal Transport and Reduced Computational Load

- Objective function ?

- Go back to slide on Bellet et al.

- Here: a variant of NCA

- **Nearest Class Cloud Metric Learning**

- Designed to boost 1-NN classif.

$\mathcal{G}_i \in \mathbb{G}_x$ *Attributed graph* $\quad e \in \mathbb{L}$ *label*

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

Probability for the graphs G to have label e

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

Maximize the probability for each graph to have is own label

Metric Learning for Attributed Graphs,

Numerical Experiments

- Graph Datsets

Datasets	**BZR**	**COX2**	**ENZYMES**	**MUTAG**	**NCI1**	**PTC-MR**
#Graphs	405	467	600	188	4110	344
#Nodes	35.75	41.22	32.63	17.93	29.97	14.29
Node attributes	cont.	cont.	lab.	deg.	lab.	lab.
q	3	3	18	4	38	18

- Task of Supervised Classification
  - k-Nearest Neighbors classifier
  - SVM with induced kernel

Dataset	**Continuous attributes**		**Discrete attributes**			
Method	**BZR**	**COX2**	**MUTAG**	**NCI1**	**PTC-MR**	**ENZYMES**
SGML - $\mathcal{RPW}_2$ (ML-kNN)	**85.61 ± 2.98**	**79.79 ± 2.18**	**90.00 ± 7.60**	72.12 ± 1.65	58.86 ± 5.88	49.00 ± 8.17
Net-LSD-heat (1-NN)	✗	✗	84.90	65.89	55.30	31.99
FGSD (1-NN)	✗	✗	86.47	75.77	60.28	41.58
NetSimile (1-NN)	✗	✗	84.09	66.56	61.26	33.23
SGML - $\mathcal{RPW}_2$ (ML-OT-SVM)	84.39 ± 3.81	**78.51 ± 0.01**	**88.95 ± 7.61**	74.84 ± 1.81	58.29 ± 6.29	54.00 ± 7.07
WWL (OT-SVM)	**84.42 ± 2.03**	78.29 ± 0.47	87.27 ± 1.50	85.75 ± 0.25	**66.31 ± 1.21**	**59.13 ± 0.80**
$\mathcal{FSW}$ (OT-SVM)	85.12 ± 4.15	77.23 ± 4.86	83.26 ± 10.30	72.82 ± 1.46	55.71 ± 6.74	✗
$\mathcal{FSW}$-WL [p = 4] (OT-SVM)	✗	✗	88.42 ± 5.67	**86.42 ± 1.63**	65.31 ± 7.90	✗
HGK-SP (SVM)	76.42 ± 0.72	72.57 ± 1.18	✗	✗	✗	✗
WL-OA (SVM)	-	-	87.15 ± 1.82	86.08 ± 0.27	60.58 ± 1.35	58.97 ± 0.82
PSCN [K = 10] (GCN)	80.00 ± 4.47	71.70 ± 3.57	83.47 ± 10.26	70.65 ± 2.58	58.34 ± 7.71	✗

Metric Learning for Attributed Graphs,
Visualisation of Numerical Experiments

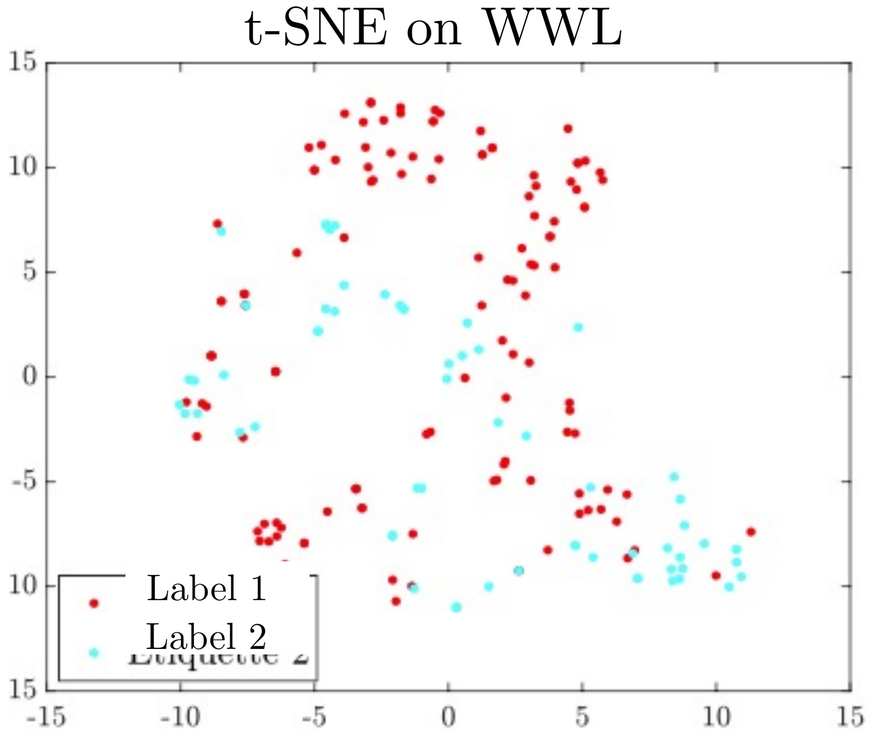
- For MUTAG Dataset

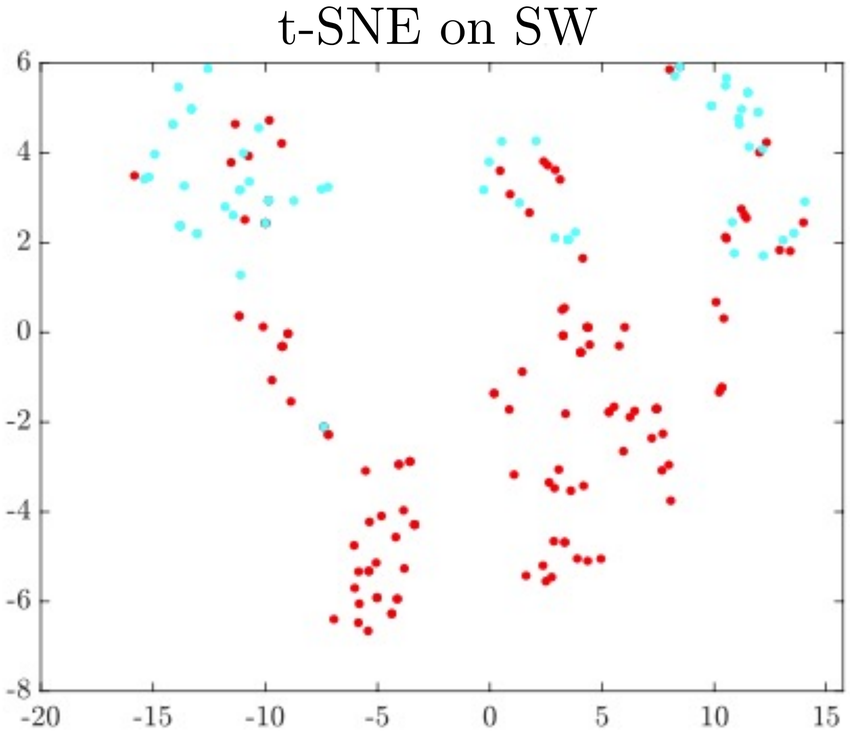
Kernel classification

Méthode	$\mathcal{SW}$	$\mathcal{PSW}$	Fused Wass.	WWL
MUTAG	**0.900** (0.873)	**0.900** (**0.900**)	0.884	0.873

K-nearest neighbors

Méthode	$\mathcal{SW}$	$\mathcal{PSW}$	Net-LSD
MUTAG	0.858 (0.858)	**0.890** (**0.895**)	0.865





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

- **A conclusion: it works** as initially wished; time complexity in $O(|\mathbb{G}|EB^2pq\tilde{n}^2)$.

Now is the time to conclude

- Two examples of models for **Representation Learning of (Attributed) Graphs**
- Non-Linear methods easily obtained thanks to GNN / GCN
- Saturation of many ML (GNN) methods on graphs (dataset, performance,...)
- => Favor the **simpler methods**, with a specific objectives and reduced costs

- **A way forward:** Introduce some explainability in these graph-based methods
- Last (still shameful) Advertisement: we hire a post-doc

Lab PHYS ENS de LYON

cnrs

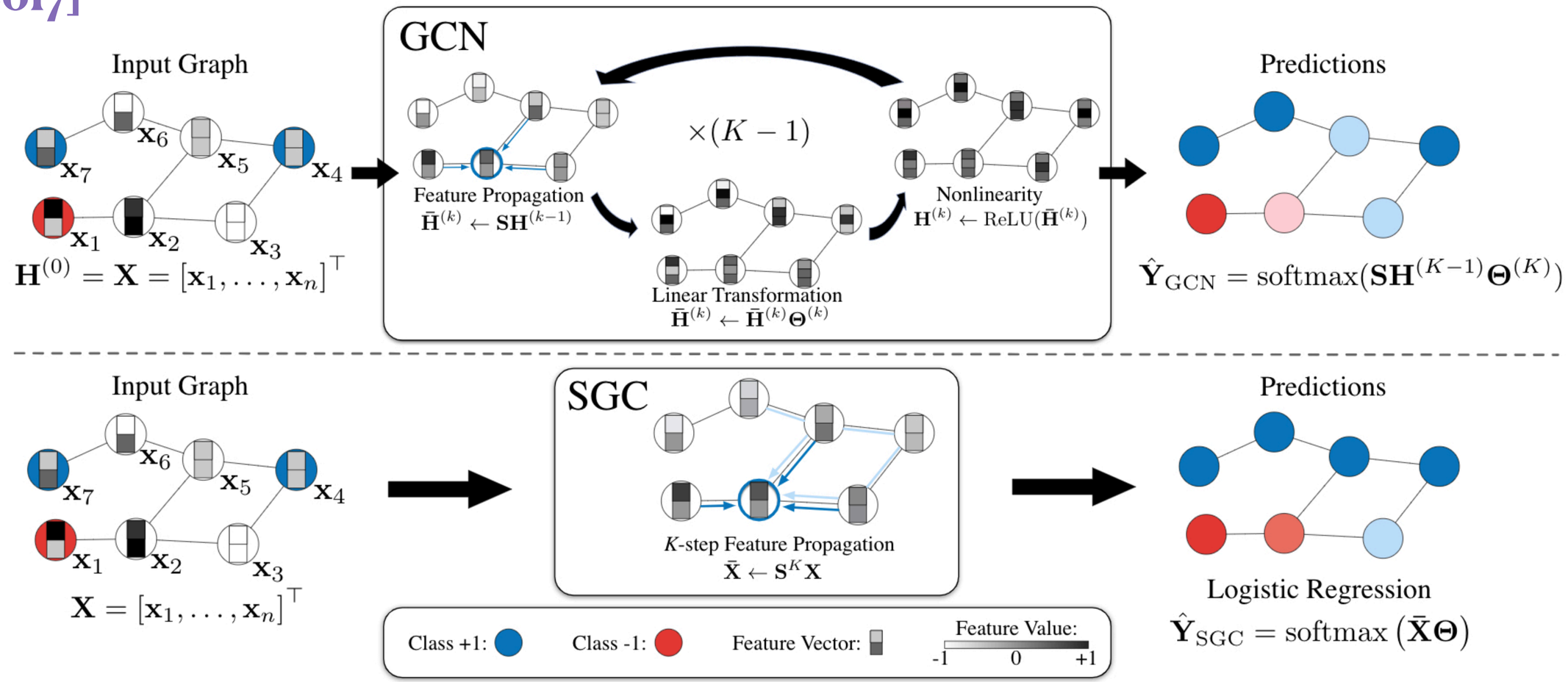
ENS DE LYON

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

Supplements

More (visual) details on GCN vs. Simple GCN

[Kipf & Welling "Semi-supervised classification with graph convolutional networks". ICLR 2017]



[Wu et al. "Simplifying graph convolutional networks". PLMR 2019]