LAGRANGIAN PROPAGATION GRAPH NEURAL NETWORKS

Matteo Tiezzi

December 4, 2020

SAILab, University of Siena

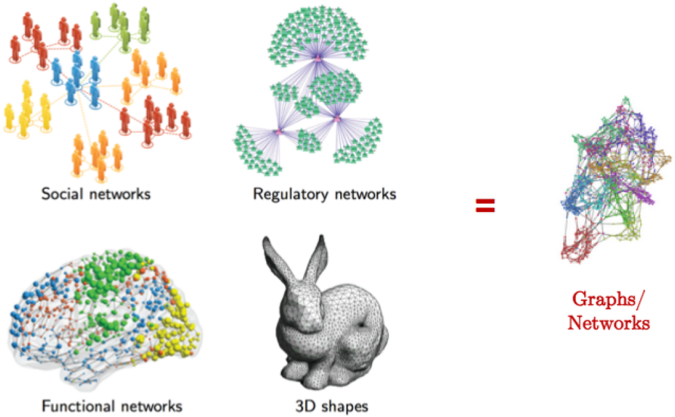
https://sailab.diism.unisi.it

https://mtiezzi.github.io/

@TiezziMatteo

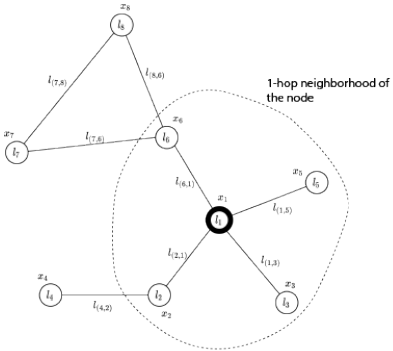
SAILab
Siena Artificial Intelligence Lab

LEARNING IN STRUCTURED DOMAINS



- Non-Euclidean (graph or manifold-structured) data such as social networks, molecular graphs and 3D point clouds in computer vision

GRAPH STRUCTURED DATA

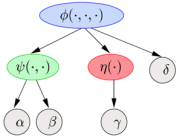


- Graph $G = (V, E)$, where V is a finite set of *nodes* and $E \subseteq V \times V$ collects the *arcs*
- l_i node i features, $l_{(i,j)}$ arc (i, j) features (both optional)
- Structures allowing to represent relationships
- GOAL Learn a mapping $f : V \to \mathcal{Y}$ predicting some graph property (at node/graph level)

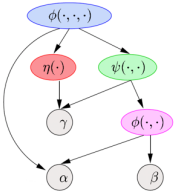
$$\tau(G)$$

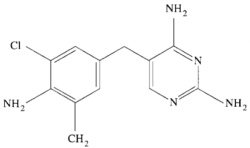
On the truth of logic statements

$\phi(\psi(\alpha,\beta),\eta(\gamma),\delta)$



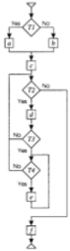
$\phi(\alpha,\eta(\gamma),\psi(\gamma,\phi(\alpha,\beta)))$





Program behavior

```
program  name (list);
var
 . . .
begin
if  T1 then
        a
else
        b;
c;
while T2 do
        begin
        d;
        if  T3 then
        while T4 do
                e
        end;
f
end.
```



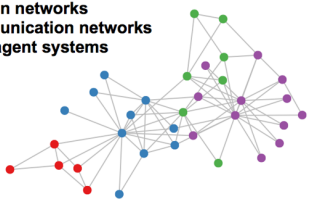
Physicochemical behavior

$$\tau(G, n)$$

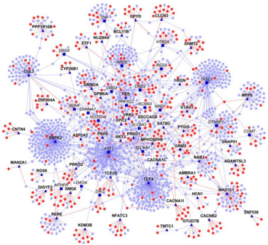
Social nets

here we need to make prediction at node level!

Social networks
Citation networks
Communication networks
Multi-agent systems



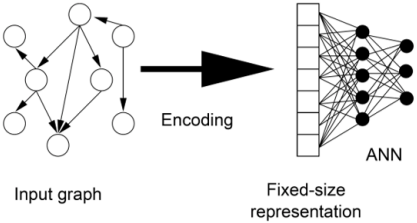
Karate club network



Protein Interaction Network

GRAPH REPRESENTATION

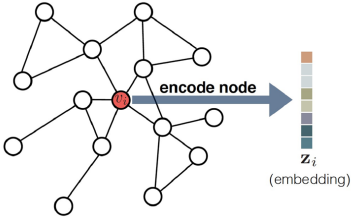
- Traditional machine learning approaches assume to deal with flat data
- Flat representations relying on summary graph statistics, kernel functions, graph traversals procedures etc.
- Pre-processing step, using hand-engineered statistics to extract structural information into simpler encodings

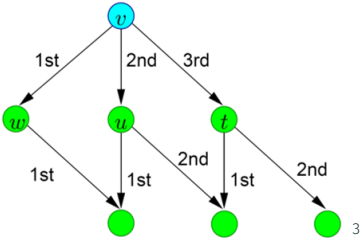


- Limited approaches – loosing useful information, not able to adapt during learning

EMBEDDING A GRAPH

- Map a graph to a real valued vector[1,2] – concatenate node features, following an order derived from the connection topology





[3]

- Not well defined for any category of graph – it holds for Directed Ordered Acyclic Graphs (DOAGs), does not hold for generic cyclic graphs

[1] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. "Deepwalk: Online learning of social representations". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 701–710.

[2] Aditya Grover and Jure Leskovec. "node2vec: Scalable feature learning for networks". In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016, pp. 855–864.

[3] Figure credit to William L. Hamilton

The Graph Neural Network Model

THE GRAPH NEURAL NETWORK MODEL (GNN)

- Introduced by Scarselli et al.[4] in 2005
- No need of a preprocessing embedding step
- No limitation on the graph type (more general w.r.t Recursive nets[5])
- Neural networks exploited to learn how to encode nodes of a graph for a given task
- Take into account information local to each node and the whole graph topology
- The learning process requires, for each epoch, an iterative diffusion mechanism up to convergence to a stable fixed point

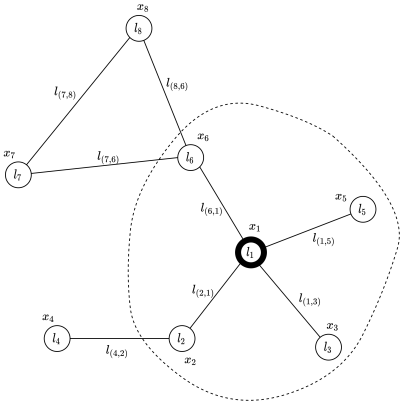
[4] Franco Scarselli et al. "Graph neural networks for ranking web pages". In: *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*. IEEE. 2005, pp. 666–672.

[5] Paolo Frasconi, Marco Gori, and Alessandro Sperduti. "A general framework for adaptive processing of data structures". In: *IEEE transactions on Neural Networks* 9.5 (1998), pp. 768–786.

GNNS- A TWO PHASES COMPUTATION

- GNNs apply a two-phase computation on each graph $G = (V, E)$
- Encoding (aggregate) phase compute a state vector x_v for each node in V by (iteratively) combining the states of neighboring nodes (i.e. nodes $u, v \in V$ that are connected by an arc $(u, v) \in E$) – exploiting the state transition function f_w
- Output (readout) phase the final latent representations encoded by the states stored in each node are exploited to compute the model output – exploiting the output function g_w

TRANSITION AND OUTPUT FUNCTIONS



Neighbors state

Neighbors features

$$x_v^{(t+1)} = f_a(x_{ne[v]}^{(t)} l_{ne[v]}, l_v, l_{(ne[v],v)} | \theta_{f_a})$$

$$y_v = f_r(x_v^{(T)} | \theta_{f_r})$$

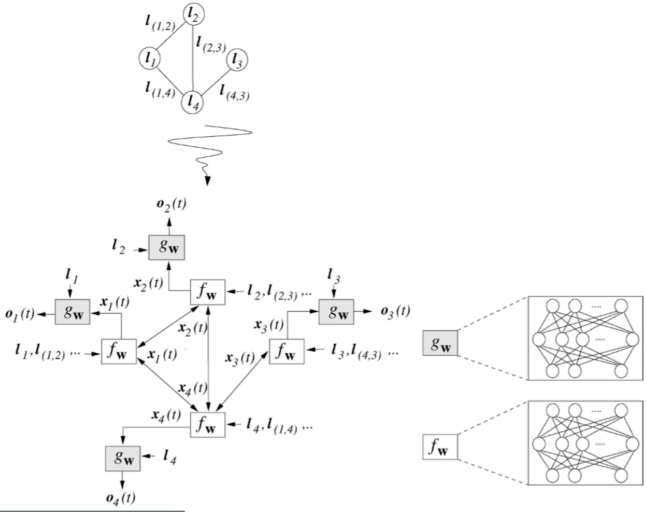
Arc feature

Node feature

$$f_{a,v}^{(\mathrm{SUM})} = \sum_{u \in ne[v]} h(x_u, l_u, l_v, l_{(u,v)} \mid \theta_h)$$

$$f_{a,v}^{(\mathrm{AVG})} = \frac{1}{|ne[v]|} \sum_{u \in ne[v]} h(x_u, l_u, l_v, l_{(u,v)} \mid \theta_h)$$

GRAPH ENCODING



[6]

[6] Figure credit to Scarselli et al.

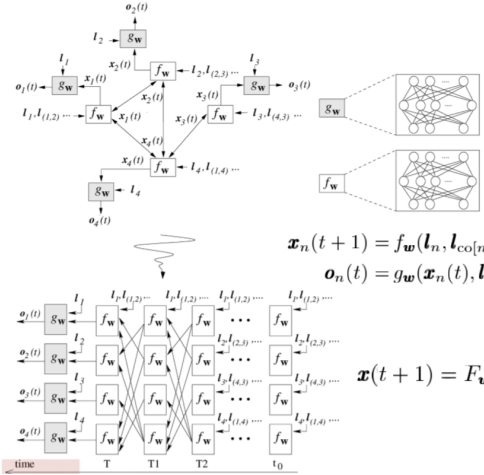
CONVERGENCE PROCEDURE

- The recursive application of the state transition function $f_w()$ on the graph nodes yields a diffusion mechanism, whose range depends on T
- In the original GNN model[7] the convergence procedure is executed until convergence of the state representation, i.e. until $x_n^{(t)} \simeq x_n^{(t-1)}, v \in V$.
- Corresponds to the computation of the *fixed point* of $f_w()$ on the input graph.
- To guarantee the convergence of this phase, the transition function is required to be a contraction map - **Banach Fixed Point Theorem**

[7] Franco Scarselli et al. "The graph neural network model". In: *IEEE Transactions on Neural Networks* 20.1 (2008), pp. 61–80.

How we get the equilibrium points?

Relaxation to an equilibrium



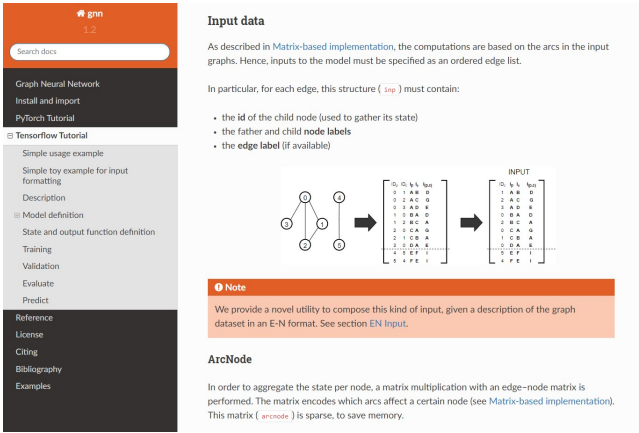
$$\boldsymbol{x}_n(t+1) = f_{\boldsymbol{w}}(\boldsymbol{l}_n, \boldsymbol{l}_{\mathrm{co}[n]}, \boldsymbol{x}_{\mathrm{ne}[n]}(t), \boldsymbol{l}_{\mathrm{ne}[n]})$$

$$\boldsymbol{o}_n(t) = g_{\boldsymbol{w}}(\boldsymbol{x}_n(t), \boldsymbol{l}_n), \qquad n \in \boldsymbol{N}.$$

$$\boldsymbol{x}(t+1) = F_{\boldsymbol{w}}(\boldsymbol{x}(t), \boldsymbol{l})$$

GNN FRAMEWORK[8]

http://sailab.diism.unisi.it/gnn/

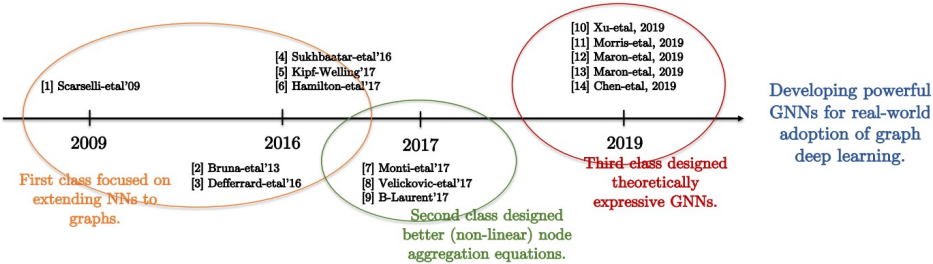


TensorFlow

PYTORCH

[8] Alberto Rossi et al. "Inductive–transductive learning with graph neural networks". In: *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*. Springer. 2018, pp. 201–212.

TIMELINE OF GNN MODELS



[1] Scarselli, Gori, Tsoi, Hagenbuchner, Monfardini, The Graph Neural Network Model, 2009
[2] Bruna, Zaremba, Szlam, LeCun, Spectral networks and locally connected networks on graphs, 2013
[3] Defferrard, Bresson, Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, 2016
[4] Sukhbaatar, Szlam, Fergus, Learning multiagent communication with backpropagation, 2016
[5] Kipf, Welling, Semi-supervised classification with graph convolutional networks, 2017
[6] Hamilton, Ying, Leskovec, Inductive representation learning on large graphs, 2017
[7] Monti, Boscaini, Masci, Rodola, Svoboda, Bronstein, Geometric deep learning on graphs using mixture model cnns, 2017
[8] Velickovic, Cucurull, Casanova, Romero, Lio, Bengio, Graph attention networks, 2017
[9] Bresson, Laurent, Residual gated graph convnets, 2017
[10] Xu, Hu, Leskovec, Jegelka, How powerful are graph neural networks?, 2019
[11] Morris, Ritzert, Fey, Hamilton, Lenssen, Rattan, Grohe, Weisfeiler and leman go neural: Higher-order graph networks, 2019
[12] Maron, Ben-Hamu, Shamir, Lipman, Invariant and equivariant graph networks, 2019
[13] Maron, Ben-Hamu, Serviansky, Lipman, Provably powerful graph networks, 2019
[14] Chen, Villar, Chen, Bruna, On the equivalence graph isomorphism testing and function approximation with gnns, 2019

Apologize for not citing more works (4000+ GNN papers). [9]

[9] Figure credit to Xavier Bresson

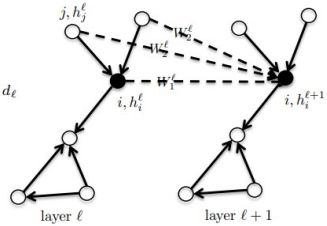
GRAPH CONVOLUTIONAL NETWORK

$$h_i^{\ell+1} = \sigma\Big(W_1^\ell\, h_i^\ell + \sum_{j\in\mathcal{N}_i} W_2^\ell h_j^\ell\Big),$$

ReLU

Sum/mean /max

$$h^{\ell+1} \in \mathbb{R}^{n\times d_{\ell+1}},\ h^\ell \in \mathbb{R}^{n\times d_\ell},\ W_{1,2}^\ell \in \mathbb{R}^{d_{\ell+1}\times d_\ell}$$



- First Order Model Message passing scheme written in matrix form

$$H^{\ell+1} = \sigma(H^\ell W_0^\ell + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} H^\ell W_1^\ell) \tag{1}$$

- $D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ is the normalized adjacency matrix
- H^0 is the matrix of input node features
- H^ℓ is the matrix of states at the ℓ-layer, computed aggregating the node from layer below
- Layered structure - depth foster aggregation

Describe the models under a common framework - MPNN[10]

- Minimal requirements to design a message passing node aggregation function:
  - Permutation invariance
  - Independent on neighborhood size (1-hop)
  - Exploit same aggregation function among the nodes (gain generalization)
  - Linear complexity on the Edges

MPNN

$$m_{i\leftarrow j}^{(\ell)} = \mathrm{MSG}_\ell\left(x_i^{(\ell-1)}, x_j^{(\ell-1)}, l_i,\ l_j, a_{i\leftarrow j}\right)$$

$$x_i^{(\ell)} = \mathrm{UP}_\ell\left(\sum_{v_j\in\mathcal{N}_i^*} m_{i\leftarrow j}^{(\ell)}\right)$$

[10] Justin Gilmer et al. "Neural message passing for quantum chemistry". In: *arXiv preprint arXiv:1704.01212* (2017).

ISOTROPIC GCN

- In the aggregation phase, every neighbor contributes equally
- Models
  - GCN[11]
  - GraphSAGE[12]
- Every direction is treated in the same way

[11] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016).
[12] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs". In: *Advances in neural information processing systems*. 2017, pp. 1024–1034.

ANISOTROPIC GCN

- Gaining a directional structure
  - Adding Edge Features if available[13]
  - Learn anisotropy to treat neighbors differently
    - Graph Attention Networks[14]
    - MoNet[15]

$$h_i^{\ell+1} = \sigma\Big(W_1^\ell\, h_i^\ell + \sum_{j\in\mathcal{N}_i} \eta_{ij}^\ell W_2^\ell h_j^\ell\Big),\ h^{\ell+1}\in\mathbb{R}^{n\times d_{\ell+1}},\ h^\ell\in\mathbb{R}^{n\times d_\ell},\ W_{1,2}^\ell\in\mathbb{R}^{d_{\ell+1}\times d_\ell},$$

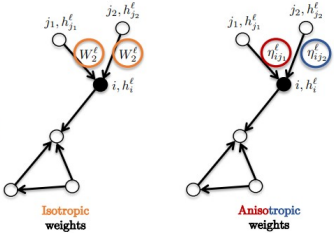
$$\eta_{ij}^\ell = g_{W_{QK}}(h_i^\ell, h_j^\ell)$$

$$= \mathrm{Softmax}_{\mathcal{N}_i}(e_{ij}^\ell) = \frac{\exp(e_{ij}^\ell)}{\sum_{j'\in\mathcal{N}_i}\exp(e_{ij'}^\ell)}$$

Attention mechanism in 1-hop neighborhood

$$\text{with } e_{ij}^\ell = (W_Q^\ell h_i^\ell)^T (W_K^\ell h_j^\ell),\ W_Q, W_K \in \mathbb{R}^{d_{\ell+1}\times d_\ell}$$

Query Key



[13] Scarselli et al., "The graph neural network model"; Gilmer et al., "Neural message passing for quantum chemistry".

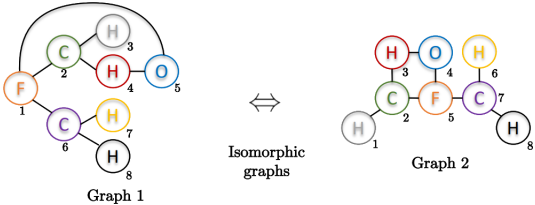
[14] Petar Veličković et al. "Graph attention networks". In: *arXiv preprint arXiv:1710.10903* (2017).

[15] Federico Monti et al. "Geometric deep learning on graphs and manifolds using mixture model cnns". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5115–5124.

Theoretically Expressive GNNs

WEISFELER-LEHMAN GNNS

- Study expressivity power of GNNs through isomorphism
- Two graph are isomorphic if topologically equivalent
  - There exist a node index permutation preserving adjacencies
- The Weisfeler-Lehman test[16] guarantee that two graphs are not isomorphic
  - Not sufficient to guarantee isomorphism



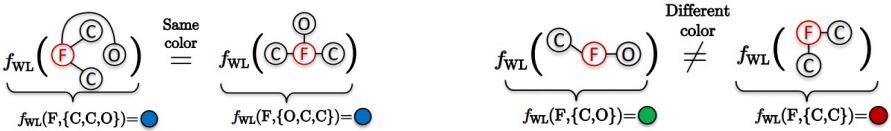
[16] Boris Weisfeiler and Andrei A Lehman. "A reduction of a graph to a canonical form and an algebra arising during this reduction". In: *Nauchno-Technicheskaya Informatsia* 2.9 (1968), pp. 12–16.

WL TEST

- Multiset: set that allows multiple instances for its elements
- Defining a coloring function f_{WL} that given a node and its neighborhood, hashes a color:

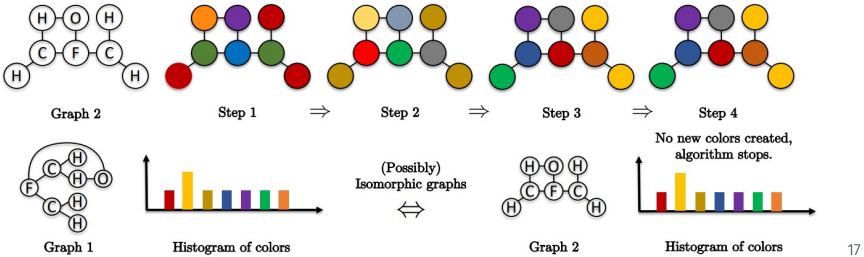
$$c_i^{(t+1)} = f_{WL}(c_i^t, \{c_j^t\}_{j \in \mathcal{N}_i}) \tag{2}$$

- f_{WL} must be defined on multiset and map different inputs to different ouputs - injective



WL TEST

- Iteratively apply the coloring function f_{WL} until color convergence
- Graph represented by an histogram
  - if histogram is different → non-isomorphic
  - if same histogram → not sufficient condition for isomorphism



[17]

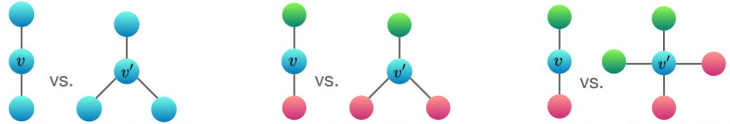
[17] Figure credit to Xavier Bresson

GRAPH ISOMORPHISM NETWORK

- Design a GNN able to distinguish non-isomorphic graphs - GIN[18]
  - same representational power of WL test

$$c_i^{(t+1)} = f_{WL}(c_i^t, \{c_j^t\}_{j \in \mathcal{N}_i}) \tag{3}$$

- Use an injective aggregation function $\rightarrow$ Sum



(a) Mean and Max both fail (b) Max fails (c) Mean and Max both fail [19]

- Mean captures the proportion/distribution of elements of a given type
- Max ignores multiplicities (reduces the multiset to a simple set).

[18] Keyulu Xu et al. "How powerful are graph neural networks?" In: *arXiv preprint arXiv:1810.00826* (2018).
[19] Figure credit to Keyulu Xu et al.

GRAPH ISOMORPHISM NETWORK

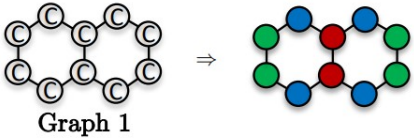
- Design a GNN able to distinguish non-isomorphic graphs - GIN[20]
  - same representational power of WL test

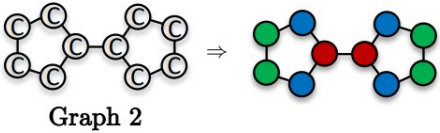
$$c_i^{(t+1)} = f_{WL}(c_i^t, \{c_j^t\}_{j \in \mathcal{N}_i}) \tag{4}$$

- Use an injective aggregation function

$$h_i^{\ell+1} = f_{\text{GIN}}\left(h_i^\ell, \{h_j^\ell\}_{j \in \mathcal{N}_i}\right) = \text{MLP}^\ell\left((1+\varepsilon)h_i^\ell + \sum_{j \in \mathcal{N}_i} h_j^\ell\right)$$

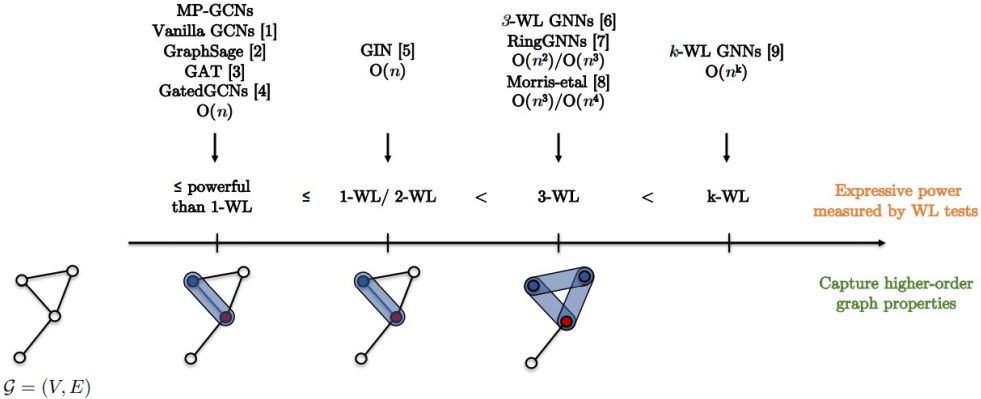
- Cons – WL test is not a sufficient condition and can fail to distinguish non-isomorphic graphs





[20] Xu et al., "How powerful are graph neural networks?"

MORE EXPRESSIVE MODELS – K-WL AND BEYOND

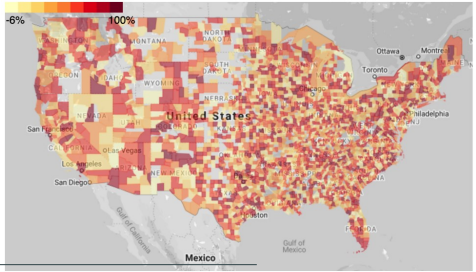


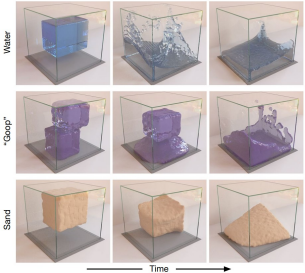
[1] Kipf, Welling, Semi-supervised classification with graph convolutional networks, 2017
[2] Hamilton, Ying, Leskovec, Inductive representation learning on large graphs, 2017
[3] Velickovic, Cucurull, Casanova, Romero, Lio, Bengio, Graph attention networks, 2017
[4] Bresson, Laurent, Residual gated graph convnets, 2017
[5] Xu, Hu, Leskovec, Jegelka, How powerful are graph neural networks?, 2019
[6] Maron, Ben-Hamu, Serviansky, Lipman, Provably powerful graph networks, 2019
[7] Chen, Villar, Chen, Bruna, On the equivalence between graph isomorphism testing and function approximation with gnns, 2019
[8] Morris, Ritzert, Fey, Hamilton, Lenssen, Rattan, Grohe, Weisfeiler and leman go neural: Higher-order graph neural networks, 2019
[9] Maron, Ben-Hamu, Shamir, Lipman, Invariant and equivariant graph networks, 2019

OTHER TRENDS AND APPLICATIONS

DGL — Zheng Zhang, GRL+ Talk
PyTorch geometric — Matthias Fey, GRL+ Talk
Spektral

- Benchmarking GNNs[21] - evaluate GNNs under same experimental setting[22]
- Scale to bigger graphs[23] and real-world tasks (physics, mobility prediction, COVID-19 forecasting[24])





[21] Vijay Prakash Dwivedi et al. "Benchmarking graph neural networks". In: *arXiv preprint arXiv:2003.00982* (2020).
[22] Federico Errica et al. "A fair comparison of graph neural networks for graph classification". In: *arXiv preprint arXiv:1912.09893* (2019).
[23] Aleksandar Bojchevski et al. "Scaling graph neural networks with approximate pagerank". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 2464–2473.
[24] Amol Kapoor et al. "Examining covid-19 forecasting using spatio-temporal graph neural networks". In: *arXiv preprint arXiv:2007.03113* (2020).

A Lagrangian Approach to Information diffusion in GNNs

LP-GNNs

Matteo Tiezzi, Giuseppe Marra, Stefano Melacci, Marco Maggini, Marco Gori

MESSAGE PASSING GRAPH NEURAL NETWORKS

Recent debate on capabilities and expressive power of message-passing graph neural networks (MPNN).

MPNN

$$m_{i\leftarrow j}^{(\ell)} = \textsc{Msg}_\ell\left(x_i^{(\ell-1)}, x_j^{(\ell-1)}, l_i,\ l_j, a_{i\leftarrow j}\right)$$

$$x_i^{(\ell)} = \textsc{Up}_\ell\Big(\sum_{v_j\in\mathcal{N}_i^*} m_{i\leftarrow j}^{(\ell)}\Big)$$

In these models, layerwise feature extraction fosters message propagation:

- Pros[25] – Turing universal if their capacity (width * depth) is large enough
- Cons[26] – Many layers will wash away node features information

[25] Andreas Loukas. "What graph neural networks cannot learn: depth vs width". In: *International Conference on Learning Representations*. 2019.
[26] Qimai Li, Zhichao Han, and Xiao-Ming Wu. "Deeper insights into graph convolutional networks for semi-supervised learning". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

By Scarselli et al. – a more general message passing process on graphs:

GNN

$$x_v^{(t+1)} = f_a(x_{ne[v]}^{(t)}, l_{ne[v]}, l_v, a_{v \leftarrow ne[v]})$$
$$y_v = f_r(x_v^{(T)})$$

Node update is repeated until convergence of the state representation, i.e. until $x_v^{(T)} \simeq x_v^{(T-1)}, v \in V$. Hence, f_a reaches its fixed point, satisfying the constraint:

$$\forall v \in V, x_v = f_{a,v} \ .$$

- Pros – Diffusion mechanism involving all the graph, not only a k-hop neighborhood (k layers).
- Cons – Epoch wise ad-hoc iterative convergence and BackProp.

- Avoid the explicit iterative computation of the fixed point.
- Cast the learning problem as constrained optimization.
- Add free variables x_v (to be optimized) corresponding to the node states.

Problem

$$\min_{\Theta_{f_a}, \theta_{f_r}, X} \quad \sum_{v \in S} L(f_r(x_v), y_v)$$

$$\text{subject to} \quad \mathcal{G}(x_v - f_{a,v}) = 0, \quad \forall\, v \in V$$

With $\mathcal{G}(0) = 0$. Enforce the constraint satisfaction – express relationship between each node and its neighborhood.

Model evolution:

- Introduce a set of K states for each node $v \in V$, organized into K layers, $\{x_{v,k}, k = 0, \ldots, K-1\}$.
- Node states as additional input to the upper layer state transition function f_a^k.

$$\mathcal{G}(x_{v,k} - f_{a,v}^k) = 0, \quad \forall\, v \in V,\ \forall\, k \in [0, K-1]$$

CONSTRAINTS TOLERANCE - $\mathcal{G}$ FUNCTION

$$\mathcal{G}(a) = \textit{lin-}\epsilon(a) := \max\{a, \varepsilon\} - \max\{-a, \varepsilon\} =$$

,

$$\mathcal{G}(a) = \textit{abs-}\epsilon(a) := \max\{|a| - \varepsilon, 0\} =$$

.

Table 1: The considered variants of the $\mathcal{G}$ function. By introducing ϵ-insensitive constraint satisfaction, we can inject into our hard-optimization scheme a controlled amount (i.e. ϵ) of unsatisfaction tolerance.

	lin	*lin-*ϵ	*abs*	*abs-*ϵ	*squared*
$\mathcal{G}(a)$	a	$\max(a, \epsilon) - \max(-a, \epsilon)$	$\lvert a \rvert$	$\max(\lvert a \rvert - \epsilon, 0)$	a^2
Unilateral	×	×	✓	✓	✓
ϵ-insensitive	×	✓	×	✓	×

- Stabilize the learning process
- Improved generalization and tolerance to noise

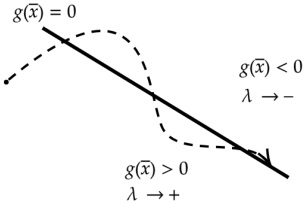
A LAGRANGIAN APPROACH

We tackle the constrained problem introducing the Lagrangian

$$\mathcal{L}(\theta_{f_a}, \theta_{f_r}, X, \Lambda) = \sum_{v \in S} \left[L(f_r(x_v), y_v) + \sum_{k=0}^{K-1} \lambda_v^k \mathcal{G}\left(x_v - f_{a,v}\right) \right]$$

and then looking for saddle points of this function, in a gradient ascent-descent scheme[27].

$$\min_{\Theta_{f_a}, \theta_{f_r}, X} \max_{\Lambda} \mathcal{L}(\Theta_{f_a}, \theta_{f_r}, X, \Lambda)$$



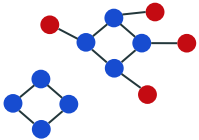
[27] John C Platt and Alan H Barr. "Constrained differential optimization". In: *Neural Information Processing Systems*. 1988, pp. 612–621.

MOTIVATIONS AND STRENGTHS

- Jointly optimize the model weights and the state representations without the need of separate ad-hoc optimization stages.
- Diffuse information layerwise by gradually enforcing the convergence of the state transition function to a fixed point (by virtue of the constraints).
- LP-GNNs[28] strictly split deep feature extraction from the diffusion mechanism.
- Our scheme can be plugged into all SOTA models, leveraging powerfull aggregation functions empowered by diffusion over the graph.

[28] Matteo Tiezzi et al. "A Lagrangian Approach to Information Propagation in Graph Neural Networks". In: vol. 325. Giacomo, Giuseppe De. IOS Press, 2020, pp. 1539–1546. URL: *https://doi.org/10.3233/FAIA200262*.

EXPERIMENTS – SUBGRAPH MATCHING AND CLIQUE DETECTION



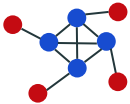


Table 2: Accuracies on the artificial datasets, for the proposed model (Lagrangian Propagation GNN - LP-GNN) and the standard GNN model for different settings.

Model			Subgraph		Clique	
	$\mathcal{G}$	ϵ	*Acc(avg)*	*Acc(std)*	*Acc(avg)*	*Acc(std)*
LP-GNN	*abs*	0.00	96.25	0.96	88.80	4.82
		0.01	**96.30**	0.87	88.75	5.03
		0.10	95.80	0.85	85.88	4.13
	lin	0.00	95.94	0.91	84.61	2.49
		0.01	95.94	0.91	85.21	0.54
		0.10	95.80	0.85	85.14	2.17
	squared	-	96.17	1.01	**93.07**	2.18
GNN	-	-	95.86	0.64	91.86	1.12

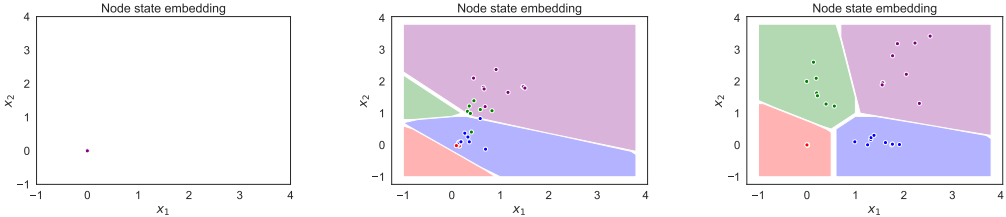
EXPERIMENTS – GRAPH CLASSIFICATION

Table 3: Average and standard deviation of the classification accuracy on the graph classification benchmarks, evaluated on the test set, for different GNN models.

Datasets	IMDB-B	IMDB-M	MUTAG	PROT.	PTC	NCI1
# graphs	1000	1500	188	1113	344	4110
# classes	2	3	2	2	2	2
Avg # nodes	19.8	13.0	17.9	39.1	25.5	29.8
DCNN	49.1	33.5	67.0	61.3	56.6	62.6
PATCHYSAN	71.0 ± 2.2	45.2 ± 2.8	**92.6** ± 4.2	75.9 ± 2.8	60.0 ± 4.8	78.6 ± 1.9
DGCNN	70.0	47.8	85.8	75.5	58.6	74.4
AWE	74.5 ± 5.9	51.5 ± 3.6	87.9 ± 9.8	–	–	–
GRAPHSAGE	72.3 ± 5.3	50.9 ± 2.2	85.1 ± 7.6	75.9 ± 3.2	63.9 ± 7.7	77.7 ± 1.5
GIN	75.1 ± 5.1	**52.3** ± 2.8	89.4 ± 5.6	76.2 ± 2.8	64.6 ± 7.0	**82.7** ± 1.7
GNN	60.9 ± 5.7	41.1 ± 3.8	88.8 ± 11.5	76.4 ± 4.4	61.2 ± 8.5	51.5 ± 2.6
LP-GNN-SINGLE	71.2 ± 4.7	46.6 ± 3.7	90.5 ± 7.0	77.1 ± 4.3	64.4 ± 5.9	68.4 ± 2.1
LP-GNN-MULTI	**76.2** ± 3.2	51.1 ± 2.1	**92.2** ± 5.6	**77.5** ± 5.2	**67.9** ± 7.2	74.9 ± 2.4

ADDITIONAL EXPERIMENTS – STATE EVOLUTION IN FEATURELESS DATA

- Completely removed node-attached features from the Karate Club dataset, in order to exploit only topological properties.
- No dependence on node features (l_v^0), the states are continuous representations of topological features of the nodes in the graph.



ADDITIONAL EXPERIMENTS – DEPTH VS DIFFUSION

- GCN-like models need to stack multiple layers to achieve information diffusion.
- Some tasks suffice a shallow representation of the nodes, but still need a diffusion process to take place.
- LP-GNN naturally model this diffusion, without the need of deep architectures: the diffusion process is independent of the depth of the network.

Table 4: Average test accuracy on the IMDB-B dataset for LP-GNN and GIN model with state layers $K \in [1, 5]$.

Model	Number of State Layers			
	1	2	3	5
GIN[29]	52	72.6	72.7	75.1
LP-GNN	**71.2**	**73.7**	**73.9**	**76.2**

[29] Keyulu Xu et al. "How Powerful are Graph Neural Networks?" In: *International Conference on Learning Representations*. 2018.

Thank you for listening!

A Lagrangian Approach to Information Propagation in Graph Neural Networks

LPGNN-Single (ECAI 2020): https://arxiv.org/abs/2002.07684
Technical report – Deep LPGNN: https://arxiv.org/abs/2005.02392
GNN framework (TF) : https://github.com/sailab-code/gnn

GNN framework (PyTorch) : https://github.com/mtiezzi/torch_gnn

GNN documentation: http://sailab.diism.unisi.it/gnn

LP-GNNs repository: https://github.com/mtiezzi/lpgnn

Matteo Tiezzi
https://mtiezzi.github.io/
@TiezziMatteo

References

- Bojchevski, Aleksandar et al. “Scaling graph neural networks with approximate pagerank”. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 2464–2473.
- Dwivedi, Vijay Prakash et al. “Benchmarking graph neural networks”. In: *arXiv preprint arXiv:2003.00982* (2020).
- Errica, Federico et al. “A fair comparison of graph neural networks for graph classification”. In: *arXiv preprint arXiv:1912.09893* (2019).
- Frasconi, Paolo, Marco Gori, and Alessandro Sperduti. “A general framework for adaptive processing of data structures”. In: *IEEE transactions on Neural Networks* 9.5 (1998), pp. 768–786.
- Gilmer, Justin et al. “Neural message passing for quantum chemistry”. In: *arXiv preprint arXiv:1704.01212* (2017).
- Grover, Aditya and Jure Leskovec. “node2vec: Scalable feature learning for networks”. In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016, pp. 855–864.
- Hamilton, Will, Zhitao Ying, and Jure Leskovec. “Inductive representation learning on large graphs”. In: *Advances in neural information processing systems*. 2017, pp. 1024–1034.
- Kapoor, Amol et al. “Examining covid-19 forecasting using spatio-temporal graph neural networks”. In: *arXiv preprint arXiv:2007.03113* (2020).

Kipf, Thomas N and Max Welling. “Semi-supervised classification with graph convolutional networks”. In: *arXiv preprint arXiv:1609.02907* (2016).

Li, Qimai, Zhichao Han, and Xiao-Ming Wu. “Deeper insights into graph convolutional networks for semi-supervised learning”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

Loukas, Andreas. “What graph neural networks cannot learn: depth vs width”. In: *International Conference on Learning Representations*. 2019.

Monti, Federico et al. “Geometric deep learning on graphs and manifolds using mixture model cnns”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5115–5124.

Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena. “Deepwalk: Online learning of social representations”. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 701–710.

Platt, John C and Alan H Barr. “Constrained differential optimization”. In: *Neural Information Processing Systems*. 1988, pp. 612–621.

Rossi, Alberto et al. “Inductive–transductive learning with graph neural networks”. In: *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*. Springer. 2018, pp. 201–212.

Scarselli, Franco et al. “Graph neural networks for ranking web pages”. In: *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI’05)*. IEEE. 2005, pp. 666–672.

Scarselli, Franco et al. “The graph neural network model”. In: *IEEE Transactions on Neural Networks* 20.1 (2008), pp. 61–80.

Tiezzi, Matteo et al. “A Lagrangian Approach to Information Propagation in Graph Neural Networks”. In: vol. 325. Giacomo, Giuseppe De. IOS Press, 2020, pp. 1539–1546. URL: *https://doi.org/10.3233/FAIA200262*.

Veličković, Petar et al. “Graph attention networks”. In: *arXiv preprint arXiv:1710.10903* (2017).

Weisfeiler, Boris and Andrei A Lehman. “A reduction of a graph to a canonical form and an algebra arising during this reduction”. In: *Nauchno-Technicheskaya Informatsia* 2.9 (1968), pp. 12–16.

Xu, Keyulu et al. “How powerful are graph neural networks?” In: *arXiv preprint arXiv:1810.00826* (2018).

– .“How Powerful are Graph Neural Networks?” In: *International Conference on Learning Representations*. 2018.