

Graph neural networks and graph isomorphism

Soledad Villar

Center for Data Science
Courant Institute of Mathematical Sciences



NEW YORK UNIVERSITY

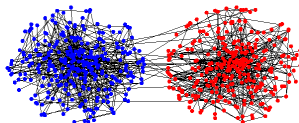
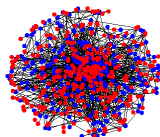
Geometry of Deep Learning
Microsoft Research, August 27 2019

Motivating example 1

Clustering the stochastic block model

$A \sim SBM(p, q, n, 2)$ (two equal-sized communities):

$$\mathbb{P}(A_{ij} = 1) = \begin{cases} p & \text{if } i, j \text{ in the same community} \\ q & \text{if } i, j \text{ in different communities} \end{cases}$$

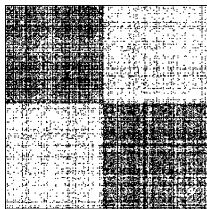
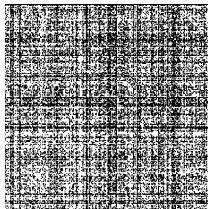
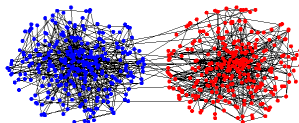
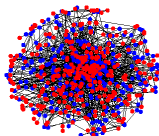


Motivating example 1

Clustering the stochastic block model

$A \sim SBM(p, q, n, 2)$ (two equal-sized communities):

$$\mathbb{P}(A_{ij} = 1) = \begin{cases} p & \text{if } i, j \text{ in the same community} \\ q & \text{if } i, j \text{ in different communities} \end{cases}$$



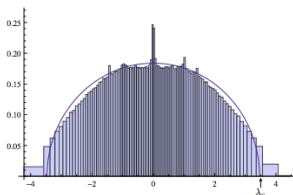
Clustering the stochastic block model

$A \sim SBM(a/n, b/n, n, 2)$ sparse.

Statistical threshold for detection: $(a - b)^2 > 2(a + b)$.

Spectrum doesn't concentrate (high degree vertices dominate it)

Laplacian is not useful for clustering



Other methods succeed. Example: semidefinite programming.

Krzakala, Moore, Mossel, Neeman, Sly, Zdeborová, Zhang, 2013

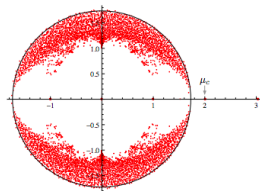
Deshpande, Abbe, Montanari, 2014

Abbe, Bandeira, Hall, 2014

Spectral redemption

Consider the non-backtracking operator (from linearized BP)

$$B_{(i \rightarrow j)(i' \rightarrow j')} = \begin{cases} 1 & \text{if } j = i' \text{ and } j' \neq i \\ 0 & \text{otherwise} \end{cases}$$



Second eigenvector of B reveals clustering structure

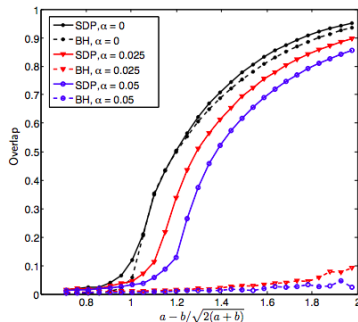
Bethe Hessian

$$BH(r) = (r^2 - 1)I - rA + D$$

Fixed points of BP \longleftrightarrow Stationary points of Bethe free energy

Second eigenvector reveals clustering structure

Pitfall: highly dependent in the model.



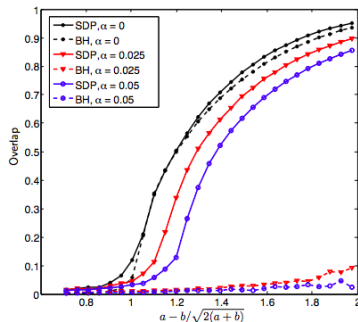
Bethe Hessian

$$BH(r) = (r^2 - 1)I - rA + D$$

Fixed points of BP \longleftrightarrow Stationary points of Bethe free energy

Second eigenvector reveals clustering structure

Pitfall: highly dependent in the model.



Goal: Combine graph operators I, D, A, \dots to generate robust “data-driven spectral methods” for problems in graphs

Graph neural networks

sGNN(\mathcal{M})

Power method: $v^{t+1} = Mv^t \quad t = 1, \dots, T.$

Graph neural networks

sGNN(\mathcal{M})

Power method: $v^{t+1} = Mv^t \quad t = 1, \dots, T.$

Graph with adjacency matrix A . Set $\mathcal{M} = \{I_n, D, A, A^2, \dots, A^{2^J}\},$

$$v^{t+1} = \left(\sum_{M \in \mathcal{M}} Mv^t \theta_M \right),$$

with $v^t \in \mathbb{R}^{n \times d_t},$

$\Theta = \{\theta_1^t, \dots, \theta_{|\mathcal{M}|}^t\}_t, \theta_M^t \in \mathbb{R}^{d_t \times d_{t+1}}$ trainable parameters.

Graph neural networks

sGNN(\mathcal{M})

Power method: $v^{t+1} = Mv^t \quad t = 1, \dots, T.$

Graph with adjacency matrix A . Set $\mathcal{M} = \{I_n, D, A, A^2, \dots, A^{2^J}\},$

$$v_l^{t+1} = \left(\sum_{M \in \mathcal{M}} Mv^t \theta_{M,l} \right), \quad l = 1, \dots, d_{t+1}$$

with $v^t \in \mathbb{R}^{n \times d_t},$

$\Theta = \{\theta_1^t, \dots, \theta_{|\mathcal{M}|}^t\}_t, \theta_M^t \in \mathbb{R}^{d_t \times d_{t+1}}$ trainable parameters.

Graph neural networks

sGNN(\mathcal{M})

Power method: $v^{t+1} = Mv^t \quad t = 1, \dots, T.$

Graph with adjacency matrix A . Set $\mathcal{M} = \{I_n, D, A, A^2, \dots, A^{2^J}\},$

$$v_l^{t+1} = \left(\sum_{M \in \mathcal{M}} M v^t \theta_{M, l}^t \right), \quad l = 1, \dots, d_{t+1}$$

with $v^t \in \mathbb{R}^{n \times d_t},$

$\Theta = \{\theta_1^t, \dots, \theta_{|\mathcal{M}|}^t\}_t, \theta_M^t \in \mathbb{R}^{d_t \times d_{t+1}}$ trainable parameters.

Graph neural networks

sGNN(\mathcal{M})

Power method: $v^{t+1} = Mv^t \quad t = 1, \dots, T.$

Graph with adjacency matrix A . Set $\mathcal{M} = \{I_n, D, A, A^2, \dots, A^{2^J}\},$

$$v_l^{t+1} = \rho \left(\sum_{M \in \mathcal{M}} M v^t \theta_{M, l}^t \right), \quad l = 1, \dots, d_{t+1}$$

with $v^t \in \mathbb{R}^{n \times d_t},$

$\Theta = \{\theta_1^t, \dots, \theta_{|\mathcal{M}|}^t\}_t, \theta_M^t \in \mathbb{R}^{d_t \times d_{t+1}}$ trainable parameters.

Graph neural networks

sGNN(\mathcal{M})

Power method: $v^{t+1} = Mv^t \quad t = 1, \dots, T.$

Graph with adjacency matrix A . Set $\mathcal{M} = \{I_n, D, A, A^2, \dots, A^{2^J}\},$

$$v_l^{t+1} = \rho \left(\sum_{M \in \mathcal{M}} M v^t \theta_{M, l}^t \right), \quad l = 1, \dots, d_{t+1}$$

with $v^t \in \mathbb{R}^{n \times d_t},$

$\Theta = \{\theta_1^t, \dots, \theta_{|\mathcal{M}|}^t\}_t, \theta_M^t \in \mathbb{R}^{d_t \times d_{t+1}}$ trainable parameters.

- ▶ Extension to line graph (GNN with non-backtracking).
- ▶ Extension to power graph $\min(1, A^t)$

Graph neural networks

sGNN(\mathcal{M})

Power method: $v^{t+1} = Mv^t \quad t = 1, \dots, T.$

Graph with adjacency matrix A . Set $\mathcal{M} = \{I_n, D, A, A^2, \dots, A^{2^J}\},$

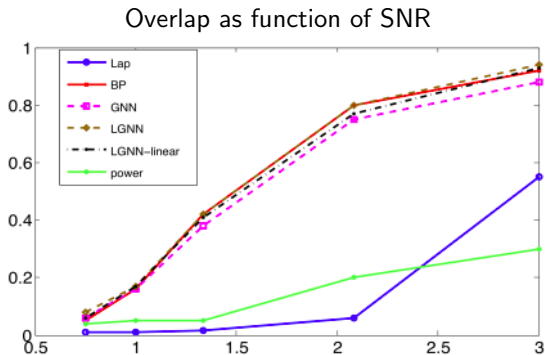
$$v_l^{t+1} = \rho \left(\sum_{M \in \mathcal{M}} M v^t \theta_{M, l}^t \right), \quad l = 1, \dots, d_{t+1}$$

with $v^t \in \mathbb{R}^{n \times d_t},$

$\Theta = \{\theta_1^t, \dots, \theta_{|\mathcal{M}|}^t\}_t, \theta_M^t \in \mathbb{R}^{d_t \times d_{t+1}}$ trainable parameters.

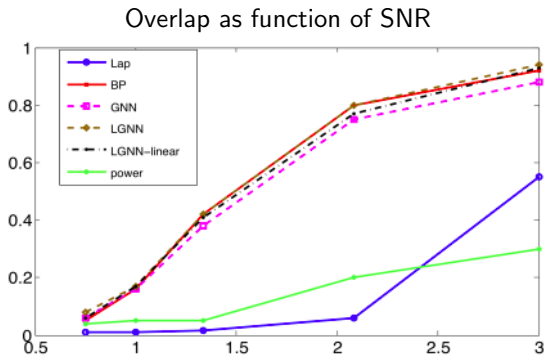
- ▶ Extension to line graph (GNN with non-backtracking).
- ▶ Extension to power graph $\min(1, A^t)$
- ▶ Equivariant wrt permutations $G \mapsto \phi(G)$ then $G_{\Pi} \mapsto \Pi \phi(G).$

Numerical performance. SBM $k = 2$



Theoretical result: Under simplifications one can show that all local minima have small loss.

Numerical performance. SBM $k = 2$



Theoretical result: Under simplifications one can show that all local minima have small loss.

Extension to unsupervised setting: Max-cut on random regular graphs.

Motivating example 2

Quadratic assignment problem

A, B $n \times n$ matrices. Π : set of $n \times n$ permutation matrices.

Quadratic assignment : $\max_{X \in \Pi} \text{Trace}(AXBX^T)$

Motivating example 2

Quadratic assignment problem

A, B $n \times n$ matrices. Π : set of $n \times n$ permutation matrices.

$$\text{Quadratic assignment : } \max_{X \in \Pi} \text{Trace}(AXBX^T)$$

It includes many relevant problem as particular cases:

- ▶ Graph matching: $\min_{X \in \Pi} \|AX - XB\|_F^2$

Motivating example 2

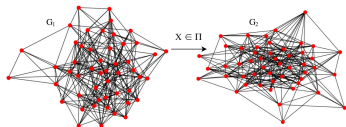
Quadratic assignment problem

A, B $n \times n$ matrices. Π : set of $n \times n$ permutation matrices.

$$\text{Quadratic assignment : } \max_{X \in \Pi} \text{Trace}(AXBX^T)$$

It includes many relevant problem as particular cases:

- Graph matching: $\min_{X \in \Pi} \|AX - XB\|_F^2 = \|AX\|_F^2 + \|XB\|_F^2 - 2\langle AX, XB \rangle$



Motivating example 2

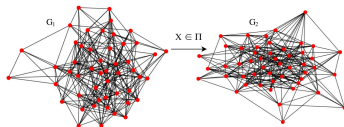
Quadratic assignment problem

A, B $n \times n$ matrices. Π : set of $n \times n$ permutation matrices.

$$\text{Quadratic assignment : } \max_{X \in \Pi} \text{Trace}(AXBX^T)$$

It includes many relevant problem as particular cases:

- ▶ Graph matching: $\min_{X \in \Pi} \|AX - XB\|_F^2 = \|AX\|_F^2 + \|XB\|_F^2 - 2\langle AX, XB \rangle$



- ▶ Graph isomorphism

Motivating example 2

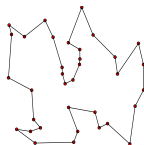
Quadratic assignment problem

A, B $n \times n$ matrices. Π : set of $n \times n$ permutation matrices.

$$\text{Quadratic assignment : } \max_{X \in \Pi} \text{Trace}(AXBX^T)$$

It includes many relevant problem as particular cases:

- ▶ Graph matching: $\min_{X \in \Pi} \|AX - XB\|_F^2$.
- ▶ Graph isomorphism.
- ▶ Traveling salesman problem.



Motivating example 2

Quadratic assignment problem

A, B $n \times n$ matrices. Π : set of $n \times n$ permutation matrices.

$$\text{Quadratic assignment : } \max_{X \in \Pi} \text{Trace}(AXBX^T)$$

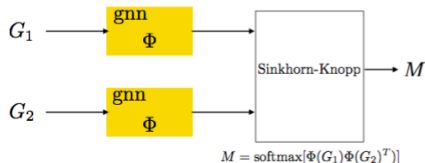
It includes many relevant problem as particular cases:

- ▶ Graph matching: $\min_{X \in \Pi} \|AX - XB\|_F^2$.
- ▶ Graph isomorphism.
- ▶ Traveling salesman problem.
- ▶ Gromov-Hausdorff distance of finite metric spaces.

It is NP-hard, even to approximate it.

GNN approach to quadratic assignment

Siamese neural network:

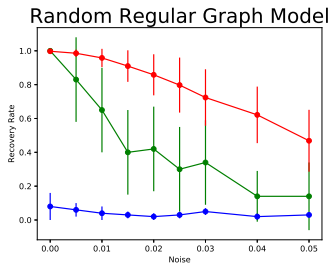
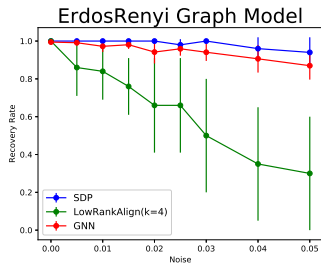


$$G_2 = \pi \star G_1 \oplus N \quad N \sim \text{i.i.d. bit flip}$$

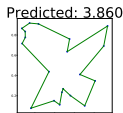
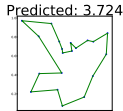
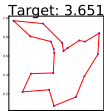
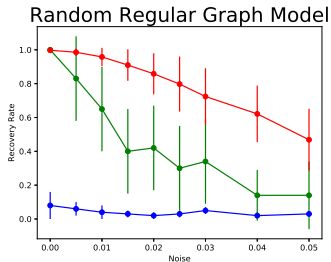
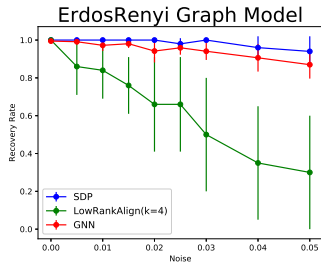
$$G_1 \sim \text{Erdos-Renyi}$$

$$G_1 \sim \text{Random regular}$$

Numerical experiments



Numerical experiments



Other GNN formulation

Message passing neural network

$$a_v^{(k)} = \text{AGGREGATE}^{(k)} \left(\{h_u^{(k-1)} : u \in \mathcal{N}(v)\} \right)$$

$$h_v^{(k)} = \text{COMBINE}^{(k)} \left(h_v^{(k-1)}, a_v^{(k)} \right)$$

Other GNN formulation

Message passing neural network

$$a_v^{(k)} = \text{AGGREGATE}^{(k)} \left(\{h_u^{(k-1)} : u \in \mathcal{N}(v)\} \right)$$

$$h_v^{(k)} = \text{COMBINE}^{(k)} \left(h_v^{(k-1)}, a_v^{(k)} \right)$$

- ▶ There exist many formulations of GNN
- ▶ All satisfy one essential property:
 - ▶ invariance or equivariance with respect to permutations
 - ▶ node labels are not intrinsic

How powerful are graph neural networks?

Q: How good are they at distinguishing non-isomorphic graphs?

How powerful are graph neural networks?

Q: How good are they at distinguishing non-isomorphic graphs?

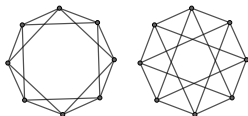
A: MPNN can be as powerful as the Weisfeler-Lehman test (1968).
W-L test is as powerful as the LP relaxation (Ullman et al 1994).

How powerful are graph neural networks?

Q: How good are they at distinguishing non-isomorphic graphs?

A: MPNN can be as powerful as the Weisfeler-Lehman test (1968).
W-L test is as powerful as the LP relaxation (Ullman et al 1994).

In particular MPNN cannot distinguish between non-isomorphic regular graphs with the same degree.



Invariant functions on graphs

► Linear case:

- If $L : \mathbb{R}^{n^k} \rightarrow \mathbb{R}$ invariant, then $\text{vec}(L) = \pi^{\otimes k} \text{vec}(L)$.
- If $L : \mathbb{R}^{n^k} \rightarrow \mathbb{R}^{n^k}$ equivariant, then $\text{vec}(L) = \pi^{\otimes 2k} \text{vec}(L)$
- The space of invariant [equivariant] linear functions on k -tensors has dimension $b(k)$ [$b(2k)$].
($b(k)$ denotes Bell Number: number of partitions of a size k set).

Invariant functions on graphs

- ▶ Linear case:
 - ▶ If $L : \mathbb{R}^{n^k} \rightarrow \mathbb{R}$ invariant, then $\text{vec}(L) = \pi^{\otimes k} \text{vec}(L)$.
 - ▶ If $L : \mathbb{R}^{n^k} \rightarrow \mathbb{R}^{n^k}$ equivariant, then $\text{vec}(L) = \pi^{\otimes 2k} \text{vec}(L)$
 - ▶ The space of invariant [equivariant] linear functions on k -tensors has dimension $b(k)$ [$b(2k)$].
($b(k)$ denotes Bell Number: number of partitions of a size k set).
- ▶ Universal approximation:
 - ▶ Invariant networks constructed by composition of linear invariant layers $L_t : \mathbb{R}^{n^k \times a} \rightarrow \mathbb{R}^b$ with ReLU or sigmoid activation functions universally approximate the space of invariant functions.
 - ▶ Extension to equivariant functions.

Invariant functions on graphs

- ▶ Linear case:

- ▶ If $L : \mathbb{R}^{n^k} \rightarrow \mathbb{R}$ invariant, then $\text{vec}(L) = \pi^{\otimes k} \text{vec}(L)$.
- ▶ If $L : \mathbb{R}^{n^k} \rightarrow \mathbb{R}^{n^k}$ equivariant, then $\text{vec}(L) = \pi^{\otimes 2k} \text{vec}(L)$
- ▶ The space of invariant [equivariant] linear functions on k -tensors has dimension $b(k)$ [$b(2k)$].
($b(k)$ denotes Bell Number: number of partitions of a size k set).

- ▶ Universal approximation:

- ▶ Invariant networks constructed by composition of linear invariant layers $L_t : \mathbb{R}^{n^k \times a} \rightarrow \mathbb{R}^b$ with ReLU or sigmoid activation functions universally approximate the space of invariant functions.
- ▶ Extension to equivariant functions.

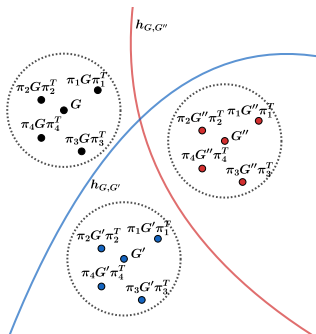
Arbitrary high order tensors are needed.

Rates of convergence are not known.

Graph isomorphism equivalence to universal approximation

Is-discriminating class of functions

A class \mathcal{C} of permutation-invariant functions from $\mathcal{X}^{n \times n}$ to \mathbb{R} so that for all pairs $G_1 \not\cong G_2 \in \mathcal{X}^{n \times n}$, there exists $h \in \mathcal{C}$ such that $h(G_1) \neq h(G_2)$.



Graph isomorphism equivalence to universal approximation

Universally approximating

A class \mathcal{C} of permutation-invariant functions from $\mathcal{X}^{n \times n}$ to \mathbb{R} so that for all permutation-invariant function f from $\mathcal{X}^{n \times n}$ to \mathbb{R} , and for all $\epsilon > 0$, there exists $h_{f,\epsilon} \in \mathcal{C}$ such that

$$\|f - h_{f,\epsilon}\|_{\infty} := \sup_{G \in \mathcal{X}^{n \times n}} |f(G) - h_{f,\epsilon}(G)| < \epsilon$$

Remark

Universally approximating classes of functions are also GIs-discriminating.

Graph isomorphism equivalence to universal approximation

\mathcal{C}^{+L}

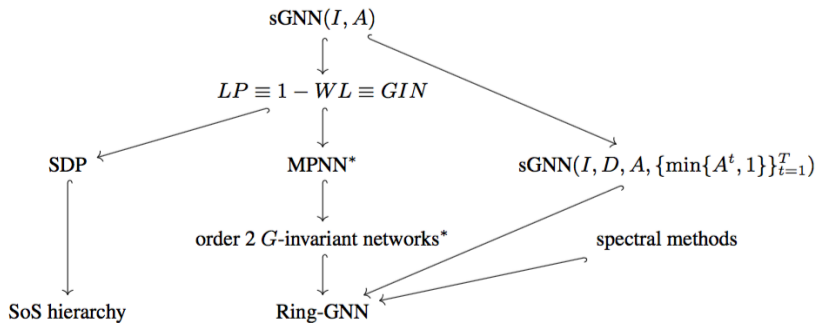
If \mathcal{C} is a collection of functions from $\mathcal{X}^{n \times n}$ to \mathbb{R} , consider the set of functions from graphs G to $\mathcal{NN}([h_1(G), \dots, h_d(G)])$ for some finite d and $h_1, \dots, h_d \in \mathcal{C}$, where \mathcal{NN} is a feed-forward neural network with ReLU and L layers.

Theorem

If \mathcal{C} is Glso-discriminating \mathcal{C}^{+2} is universally approximating.

Comparison of classes of functions through GSo

$\mathcal{C} \subseteq \mathcal{C}'$ if for all pairs of non-isomorphic graphs G_1, G_2 , if there exists $h \in \mathcal{C}$ so that $h(G_1) \neq h(G_2)$ then there exists $h' \in \mathcal{C}'$ so that $h'(G_1) \neq h'(G_2)$.



Ring GNN

Input: Graph with n nodes and d features: $A \in \mathbb{R}^{n \times n \times d}$.

Equivariant linear layer from $\mathbb{R}^{n \times n \times d}$ to $\mathbb{R}^{n \times n \times d'}$. For $\theta \in \mathbb{R}^{d \times d' \times 17}$:

$$L_{\theta}(A)_{\cdot,\cdot,k'} = \sum_{k=1}^d \sum_{i=1}^{15} \theta_{k,k',i} L_i(A_{\cdot,\cdot,i}) + \sum_{i=16}^{17} \theta_{k,k',i} \bar{L}_i.$$

Set $A^{(0)} = A$.

$$\begin{aligned} B_1^{(t)} &= \rho(L_{\alpha^{(t)}}(A^{(t)})) \\ B_2^{(t)} &= \rho(L_{\beta^{(t)}}(A^{(t)}) \cdot L_{\gamma^{(t)}}(A^{(t)})) \\ A^{(t+1)} &= k_1^{(t)} B_1^{(t)} + k_2^{(t)} B_2^{(t)} \end{aligned}$$

where $k_1^{(t)}, k_2^{(t)} \in \mathbb{R}$, $\alpha^{(t)}, \beta^{(t)}, \gamma^{(t)} \in \mathbb{R}^{d^{(t)} \times d'^{(t)} \times 17}$ are learnable parameters.

Scalar output: $\theta_S \sum_{i,j} A_{ij}^{(T)} + \theta_D \sum_{i,i} A_{ii}^{(T)} + \sum_i \theta_i \lambda_i(A^{(T)})$, where $\theta_S, \theta_D, \theta_1, \dots, \theta_n \in \mathbb{R}$ are trainable parameters, and $\lambda_i(A^{(T)})$ is the i -th eigenvalue of $A^{(T)}$.

Extensions - Future work

- ▶ Explicit rates:
Connect GNN depth/architecture with classes of graphs they separate.
- ▶ Optimization landscape of GNNs:
Current analysis of optimization landscape relies in simplified models to show that all local minima are confined in low-energy configurations.
- ▶ Connection with SoS:
For some classes of “detecting hidden structures problems” existence of degree- d SoS refutations implies success of certain (typically non-explicit) spectral methods.
 - ▶ Can we express such class of spectral methods with GNNs.
 - ▶ Can we learn them?

References

Supervised Community Detection with Hierarchical Graph Neural Networks

Z. Chen, L. Li, J. Bruna, ICLR 2019

arXiv:1705.08415

A note on learning algorithms for quadratic assignment with Graph Neural Networks

A. Nowak, S. Villar, A. S. Bandeira, J. Bruna, ICML Workshop (PADL) 2017

arXiv:1706.07450

On the equivalence between graph isomorphism testing and function approximation with GNNs

Z. Chen, S. Villar, L. Chen, J. Bruna, NeurIPS 2019

arXiv:1905.12560



SIMONS
FOUNDATION