On the Ability of Graph Neural Networks to Model Interactions Between Vertices

Noam Razin

Joint work with Tom Verbin & Nadav Cohen

Tel Aviv University

Learning on Graphs and Geometry Reading Group

16 January 2023

Outline

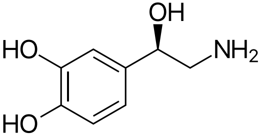
Graph Neural Networks (GNNs)

Neural networks purposed for modeling interactions over graph data

Graph Neural Networks (GNNs)

Neural networks purposed for modeling interactions over graph data

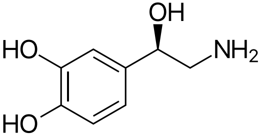
- Molecular data — graph prediction



Graph Neural Networks (GNNs)

Neural networks purposed for modeling interactions over graph data

- Molecular data — graph prediction

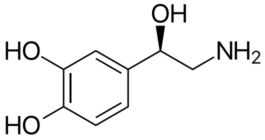


- Social networks — vertex prediction

Graph Neural Networks (GNNs)

Neural networks purposed for modeling interactions over graph data

- Molecular data — graph prediction



- Social networks — vertex prediction

- Many more applications: recommender systems, ETA prediction,...

Mathematical Theory of GNNs

Mathematical Theory of GNNs

Challenge

Develop mathematical theory for GNNs

Mathematical Theory of GNNs

Challenge

Develop mathematical theory for GNNs

Fundamental Question

Expressivity: which functions can GNNs realize?

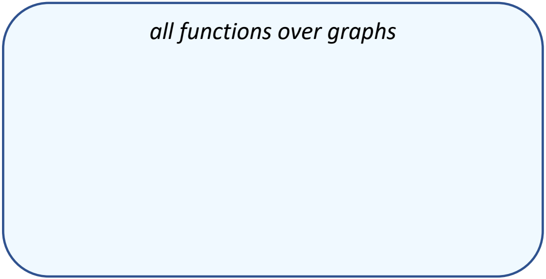
Mathematical Theory of GNNs

Challenge

Develop mathematical theory for GNNs

Fundamental Question

Expressivity: which functions can GNNs realize?



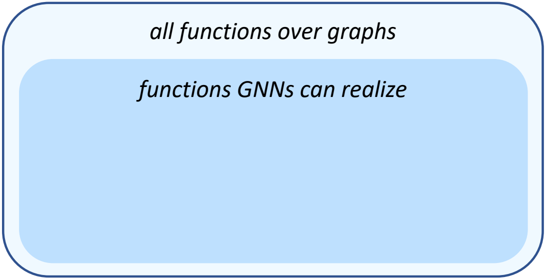
Mathematical Theory of GNNs

Challenge

Develop mathematical theory for GNNs

Fundamental Question

Expressivity: which functions can GNNs realize?



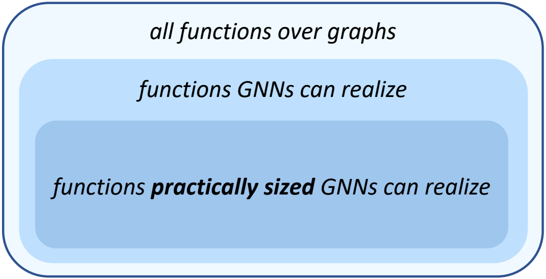
Mathematical Theory of GNNs

Challenge

Develop mathematical theory for GNNs

Fundamental Question

Expressivity: which functions can GNNs realize?

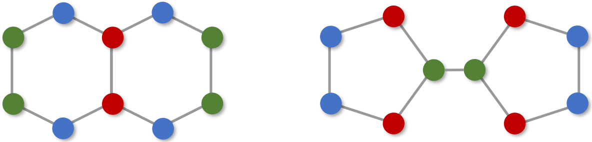


Existing Analyses of Expressivity

Existing Analyses of Expressivity

(1) Ability to distinguish non-isomorphic graphs

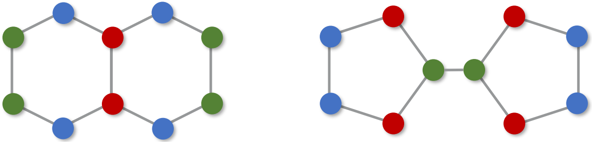
(*e.g.* Xu et al. 2019, Morris et al. 2019, Maron et al. 2019b, Geerts & Reutter 2022)



Existing Analyses of Expressivity

(1) Ability to distinguish non-isomorphic graphs

(*e.g.* Xu et al. 2019, Morris et al. 2019, Maron et al. 2019b, Geerts & Reutter 2022)



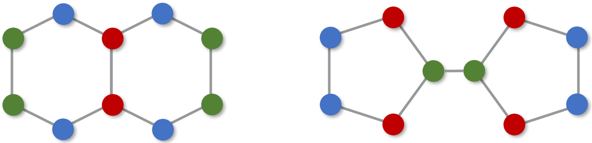
(2) Universality

(*e.g.* Maron et al. 2019a, Keriven & Peyré 2019, Chen et al. 2019, Azizian & Lelarge 2021)

Existing Analyses of Expressivity

(1) Ability to distinguish non-isomorphic graphs

(*e.g.* Xu et al. 2019, Morris et al. 2019, Maron et al. 2019b, Geerts & Reutter 2022)



(2) Universality

(*e.g.* Maron et al. 2019a, Keriven & Peyré 2019, Chen et al. 2019, Azizian & Lelarge 2021)

(3) Computability of graph properties: shortest paths, diameter,...

(e.g. Dehmamy et al. 2019, Garg et al. 2020, Loukas 2020, Chen et al. 2020)

Limitations of Existing Analyses

Despite progress in understanding expressivity of GNNs:

Limitations of Existing Analyses

Despite progress in understanding expressivity of GNNs:

(1) Analyses often treat asymptotic regimes of unbounded width or depth

Limitations of Existing Analyses

Despite progress in understanding expressivity of GNNs:

(1) Analyses often treat asymptotic regimes of unbounded width or depth

(2) Lack formalization for ability to model interactions between vertices

Limitations of Existing Analyses

Despite progress in understanding expressivity of GNNs:

(1) Analyses often treat asymptotic regimes of unbounded width or depth

(2) Lack formalization for ability to model interactions between vertices

Question

How do graph structure and GNN architecture affect interactions?

Promo: Our Contributions

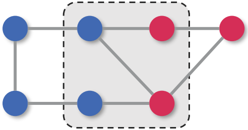
Promo: Our Contributions

Theory

Promo: Our Contributions

Theory

Characterize ability of certain GNNs to model interactions between vertices



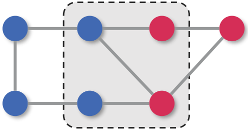
Promo: Our Contributions

Theory

Characterize ability of certain GNNs to model interactions between vertices

↑

formalized via *separation rank*



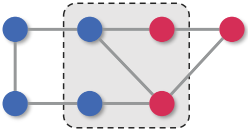
Promo: Our Contributions

Theory

Characterize ability of certain GNNs to model interactions between vertices

↑

formalized via *separation rank*

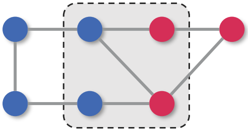


Practical Application

Promo: Our Contributions

Theory

Characterize ability of certain GNNs to model interactions between vertices

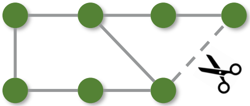


↑

formalized via *separation rank*

Practical Application

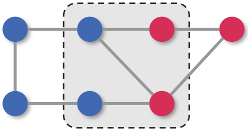
Use theory to derive an *edge sparsification* method preserving interactions



Promo: Our Contributions

Theory

Characterize ability of certain GNNs to model interactions between vertices

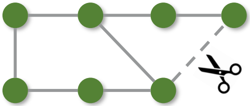


↑

formalized via *separation rank*

Practical Application

Use theory to derive an *edge sparsification* method preserving interactions



It is simple, efficient, and outperforms alternative methods

Outline

Outline

Separation Rank

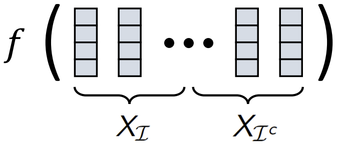
Separation Rank

Known measure for interaction modeled across partition of input variables

Separation Rank

Known measure for interaction modeled across partition of input variables

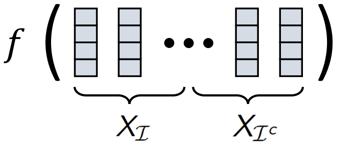
Let $f : (\mathbb{R}^D)^N \to \mathbb{R}$ and subset of variables $\mathcal{I} \subseteq [N]$



Separation Rank

Known measure for interaction modeled across partition of input variables

Let $f : (\mathbb{R}^D)^N \to \mathbb{R}$ and subset of variables $\mathcal{I} \subseteq [N]$

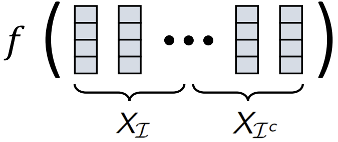


$$\mathrm{sep}(f; \mathcal{I}) := \min R \text{ s.t. } f(X) = \sum_{r=1}^{R} g_r(X_{\mathcal{I}}) \cdot \bar{g}_r(X_{\mathcal{I}^c})$$

Separation Rank

Known measure for interaction modeled across partition of input variables

Let $f : (\mathbb{R}^D)^N \to \mathbb{R}$ and subset of variables $\mathcal{I} \subseteq [N]$



$$\mathrm{sep}(f;\mathcal{I}) := \min R \text{ s.t. } f(X) = \textstyle\sum_{r=1}^{R} g_r(X_{\mathcal{I}}) \cdot \bar{g}_r(X_{\mathcal{I}^c})$$

Higher $\mathrm{sep}(f;\mathcal{I}) \implies$ stronger interaction between $X_{\mathcal{I}}$ and $X_{\mathcal{I}^c}$

Usages of Separation Rank

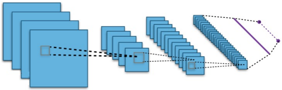
Usages of Separation Rank

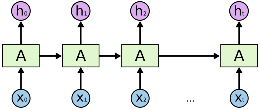
- Measure of entanglement in quantum mechanics

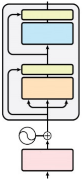
Usages of Separation Rank

- Measure of entanglement in quantum mechanics

- Analyses of convolutional, recurrent, and self-attention NNs
  (*e.g.* Cohen & Shashua 2017, Levine et al. 2018;2020, **R** et al. 2022)



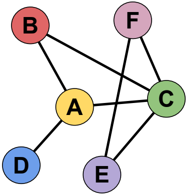




Outline

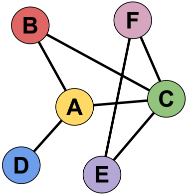
Message-Passing GNNs

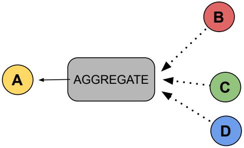
Message-Passing GNNs



Inputs: graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, vertex features $X = (x^{(1)}, \ldots, x^{(|\mathcal{V}|)})$

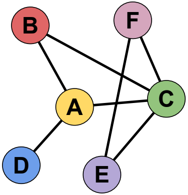
Message-Passing GNNs

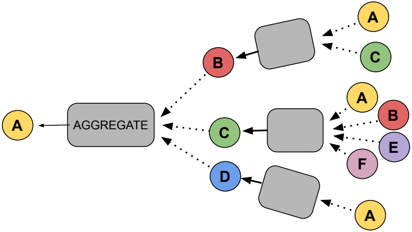




Inputs: graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, vertex features $X = (x^{(1)}, \ldots, x^{(|\mathcal{V}|)})$

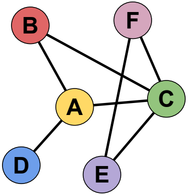
Message-Passing GNNs

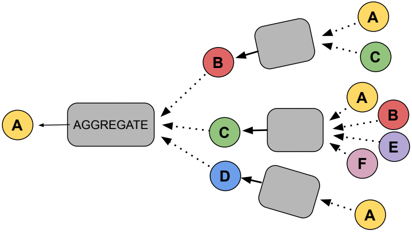




Inputs: graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, vertex features $X = (x^{(1)}, \ldots, x^{(|\mathcal{V}|)})$

Message-Passing GNNs

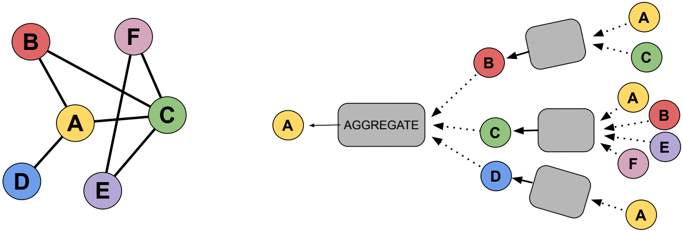




Inputs: graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, vertex features $X = (x^{(1)}, \ldots, x^{(|\mathcal{V}|)})$

Initialize: $h^{(0,i)} := x^{(i)}$ for $i \in \mathcal{V}$

Message-Passing GNNs

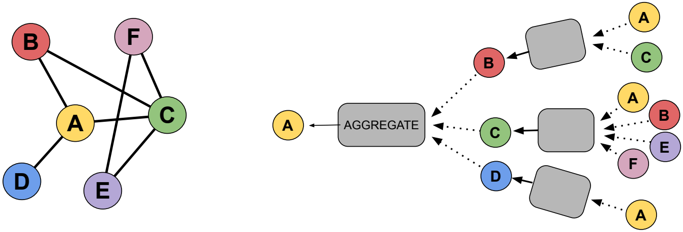


Inputs: graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, vertex features $X = (x^{(1)}, \ldots, x^{(|\mathcal{V}|)})$

Initialize: $h^{(0,i)} := x^{(i)}$ for $i \in \mathcal{V}$

Common update rule: at layer $l = 1, 2, \ldots, L$ for $i \in \mathcal{V}$

Message-Passing GNNs



Inputs: graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, vertex features $X = (x^{(1)}, \ldots, x^{(|\mathcal{V}|)})$

Initialize: $h^{(0,i)} := x^{(i)}$ for $i \in \mathcal{V}$

Common update rule: at layer $l = 1, 2, \ldots, L$ for $i \in \mathcal{V}$

$$h^{(l,i)} = \textsc{agg}\Big(\Big\{ W^{(l)} h^{(l-1,j)} : j \in \mathrm{neighbors}(i) \Big\}\Big)$$

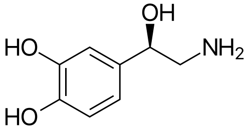
GNNs for Vertex vs Graph Prediction

After L layers the GNN produces $h^{(L,1)}, \ldots, h^{(L,|\mathcal{V}|)}$

GNNs for Vertex vs Graph Prediction

After L layers the GNN produces $h^{(L,1)}, \ldots, h^{(L,|\mathcal{V}|)}$

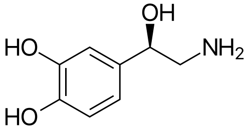
Graph prediction: single output for the whole graph



GNNs for Vertex vs Graph Prediction

After L layers the GNN produces $h^{(L,1)}, \ldots, h^{(L,|\mathcal{V}|)}$

Graph prediction: single output for the whole graph

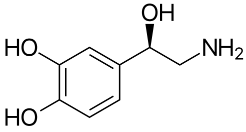


$$GNN(X) = W^{(o)} \mathrm{AGG}\big(h^{(L,1)}, \ldots, h^{(L,|\mathcal{V}|)}\big)$$

GNNs for Vertex vs Graph Prediction

After L layers the GNN produces $h^{(L,1)}, \ldots, h^{(L,|\mathcal{V}|)}$

Graph prediction: single output for the whole graph



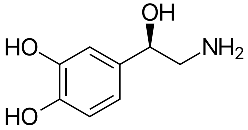
$$GNN(X) = W^{(o)} \mathrm{AGG}\left(h^{(L,1)}, \ldots, h^{(L,|\mathcal{V}|)}\right)$$

Vertex prediction: output for every $t \in \mathcal{V}$

GNNs for Vertex vs Graph Prediction

After L layers the GNN produces $h^{(L,1)}, \ldots, h^{(L,|\mathcal{V}|)}$

Graph prediction: single output for the whole graph



$$GNN(X) = W^{(o)} \mathrm{AGG}\left(h^{(L,1)}, \ldots, h^{(L,|\mathcal{V}|)}\right)$$

Vertex prediction: output for every $t \in \mathcal{V}$

$$GNN^{(t)}(X) = W^{(o)} h^{(L,t)}$$

Studying Modeled Interactions via Tensor Networks

Our aim: investigate ability of GNNs to model interactions

Studying Modeled Interactions via Tensor Networks

Our aim: investigate ability of GNNs to model interactions

Prior work: study interactions for other NNs w/ polynomial non-linearity

(*e.g.* Cohen et al. 2016, Khrulkov et al. 2018, Levine et al. 2020, **R** et al. 2021;2022)

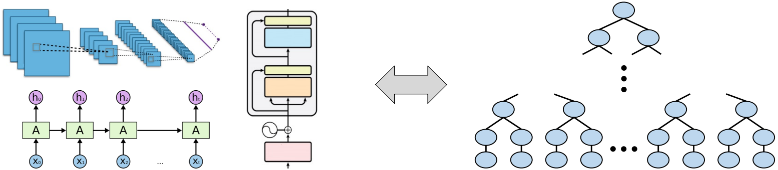
Studying Modeled Interactions via Tensor Networks

Our aim: investigate ability of GNNs to model interactions

Prior work: study interactions for other NNs w/ polynomial non-linearity

(*e.g.* Cohen et al. 2016, Khrulkov et al. 2018, Levine et al. 2020, **R** et al. 2021;2022)

NNs w/ polynomial non-linearity ***Tensor networks***

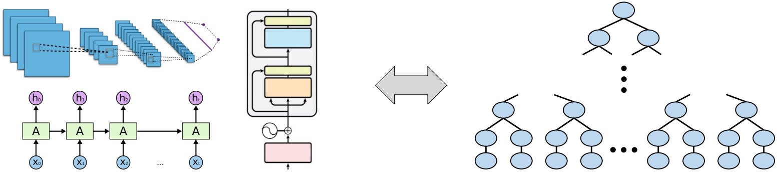


Studying Modeled Interactions via Tensor Networks

Our aim: investigate ability of GNNs to model interactions

Prior work: study interactions for other NNs w/ polynomial non-linearity
(*e.g.* Cohen et al. 2016, Khrulkov et al. 2018, Levine et al. 2020, **R** et al. 2021;2022)

NNs w/ polynomial non-linearity ***Tensor networks***



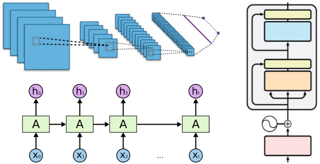
Why analyze NNs w/ polynomial non-linearity?

Studying Modeled Interactions via Tensor Networks

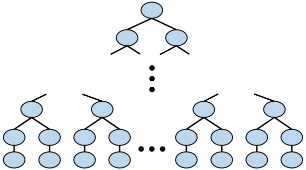
Our aim: investigate ability of GNNs to model interactions

Prior work: study interactions for other NNs w/ polynomial non-linearity
(*e.g.* Cohen et al. 2016, Khrulkov et al. 2018, Levine et al. 2020, **R** et al. 2021;2022)

NNs w/ polynomial non-linearity



Tensor networks



Why analyze NNs w/ polynomial non-linearity?

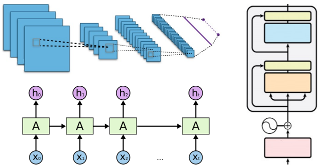
- Competitive empirical performance (*e.g.* Chrysos et al. 2020, Hua et al. 2022)

Studying Modeled Interactions via Tensor Networks

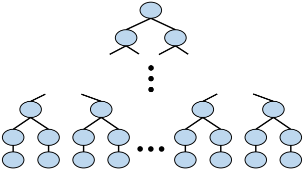
Our aim: investigate ability of GNNs to model interactions

Prior work: study interactions for other NNs w/ polynomial non-linearity
(*e.g.* Cohen et al. 2016, Khrulkov et al. 2018, Levine et al. 2020, **R** et al. 2021;2022)

NNs w/ polynomial non-linearity



Tensor networks



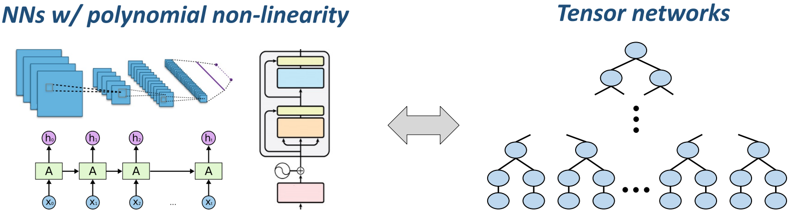
Why analyze NNs w/ polynomial non-linearity?

- Competitive empirical performance (*e.g.* Chrysos et al. 2020, Hua et al. 2022)
- Compatible with quantum computing (*e.g.* Grant et al. 2018, Bhatia et al. 2019)

Studying Modeled Interactions via Tensor Networks

Our aim: investigate ability of GNNs to model interactions

Prior work: study interactions for other NNs w/ polynomial non-linearity
(*e.g.* Cohen et al. 2016, Khrulkov et al. 2018, Levine et al. 2020, **R** et al. 2021;2022)



Why analyze NNs w/ polynomial non-linearity?

- Competitive empirical performance (*e.g.* Chrysos et al. 2020, Hua et al. 2022)
- Compatible with quantum computing (*e.g.* Grant et al. 2018, Bhatia et al. 2019)
- Insights and practical tools for more common models

GNNs With Product Aggregation

We theoretically study GNNs with product aggregation (polynomial in X)

GNNs With Product Aggregation

We theoretically study GNNs with product aggregation (polynomial in X)

$$h^{(l,i)} = \text{AGG}\Big(\Big\{W^{(l)} h^{(l-1,j)} : j \in \text{neighbors}(i)\Big\}\Big)$$

GNNs With Product Aggregation

We theoretically study GNNs with product aggregation (polynomial in X)

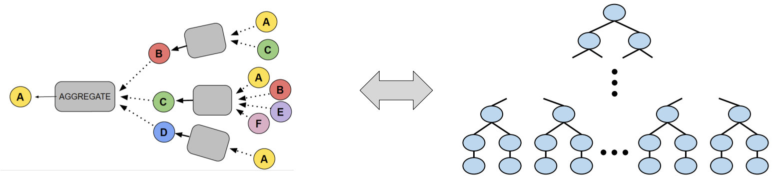
$$h^{(l,i)} = \textsc{prod}\left(\left\{ W^{(l)} h^{(l-1,j)} : j \in \text{neighbors}(i) \right\}\right)$$

GNNs With Product Aggregation

We theoretically study GNNs with product aggregation (polynomial in X)

$$h^{(l,i)} = \text{PROD}\left(\left\{W^{(l)} h^{(l-1,j)} : j \in \text{neighbors}(i)\right\}\right)$$

GNNs w/ product aggregation ***Tensor networks***

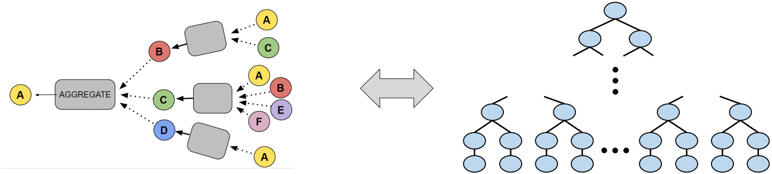


GNNs With Product Aggregation

We theoretically study GNNs with product aggregation (polynomial in X)

$$h^{(l,i)} = \text{PROD}\left(\left\{W^{(l)} h^{(l-1,j)} : j \in \text{neighbors}(i)\right\}\right)$$

GNNs w/ product aggregation ***Tensor networks***

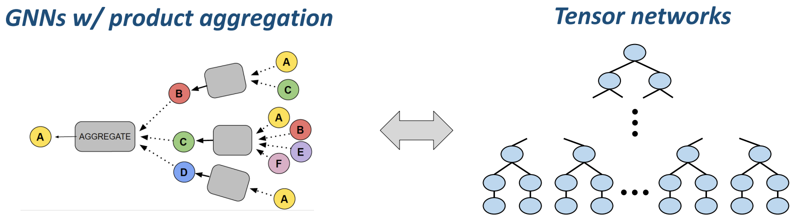


- Variant of the competitive Tensorized GNN (Hua et al. 2022)

GNNs With Product Aggregation

We theoretically study GNNs with product aggregation (polynomial in X)

$$h^{(l,i)} = \text{PROD}\left(\left\{W^{(l)} h^{(l-1,j)} : j \in \text{neighbors}(i)\right\}\right)$$



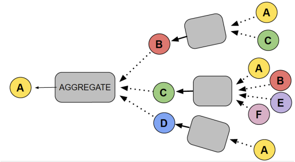
- Variant of the competitive Tensorized GNN (Hua et al. 2022)
- Demonstrate findings empirically on GNNs with ReLU non-linearity

GNNs With Product Aggregation

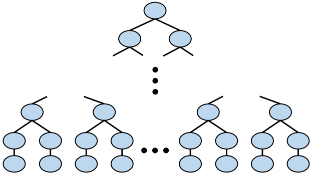
We theoretically study GNNs with product aggregation (polynomial in X)

$$h^{(l,i)} = \textsc{prod}\left(\left\{W^{(l)} h^{(l-1,j)} : j \in \text{neighbors}(i)\right\}\right)$$

GNNs w/ product aggregation



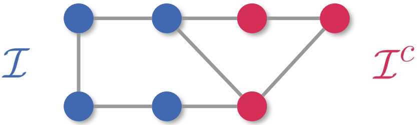
Tensor networks



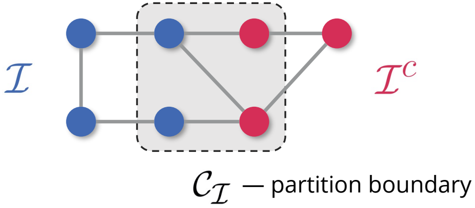
- Variant of the competitive Tensorized GNN (Hua et al. 2022)
- Demonstrate findings empirically on GNNs with ReLU non-linearity
- Based on theory: derive an edge sparsification algorithm

Outline

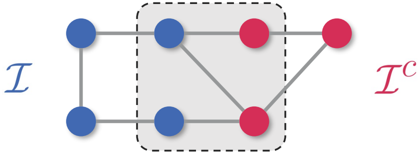
Walk Index (WI) of a Partition of Vertices



Walk Index (WI) of a Partition of Vertices



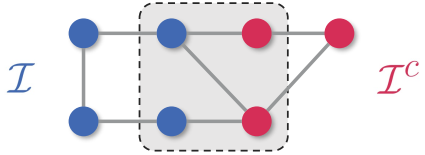
Walk Index (WI) of a Partition of Vertices



L — GNN depth

$\mathcal{C}_{\mathcal{I}}$ — partition boundary

Walk Index (WI) of a Partition of Vertices



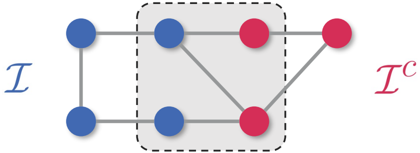
L — GNN depth

$\mathcal{C}_{\mathcal{I}}$ — partition boundary

Graph prediction:

$$\mathrm{WI}_{L-1}(\mathcal{I}) := \#\text{ length } L-1 \text{ walks from } \mathcal{C}_{\mathcal{I}}$$

Walk Index (WI) of a Partition of Vertices



L — GNN depth

$\mathcal{C}_{\mathcal{I}}$ — partition boundary

Graph prediction:

$$\mathrm{WI}_{L-1}(\mathcal{I}) := \#\text{ length } L-1 \text{ walks from } \mathcal{C}_{\mathcal{I}}$$

Vertex prediction:

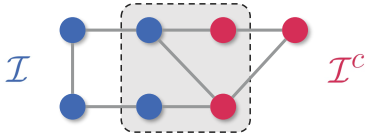
$$\mathrm{WI}_{L-1,t}(\mathcal{I}) := \#\text{ length } L-1 \text{ walks from } \mathcal{C}_{\mathcal{I}} \text{ to } t \in \mathcal{V}$$

Main Result: Strength of Interaction $\propto$ Walk Index

Main Result: Strength of Interaction $\propto$ Walk Index

Theorem

For a depth L GNN with width D_h and $\mathcal{I} \subseteq \mathcal{V}$:

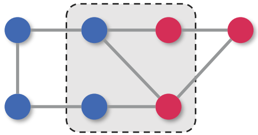


Main Result: Strength of Interaction $\propto$ Walk Index

Theorem

For a depth L GNN with width D_h and $\mathcal{I} \subseteq \mathcal{V}$:

(graph prediction) $\quad \mathrm{sep}(GNN; \mathcal{I}) = D_h^{\mathcal{O}\left(\mathbf{WI}_{L-1}(\mathcal{I})\right)}$

$\mathcal{I}$  $\mathcal{I}^c$

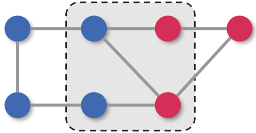
Main Result: Strength of Interaction $\propto$ Walk Index

Theorem

For a depth L GNN with width D_h and $\mathcal{I} \subseteq \mathcal{V}$:

$$\textit{(graph prediction)} \quad \mathrm{sep}(GNN; \mathcal{I}) = D_h^{\mathcal{O}\left(\mathbf{WI}_{L-1}(\mathcal{I})\right)}$$

$$\textit{(vertex prediction)} \quad \mathrm{sep}(GNN^{(t)}; \mathcal{I}) = D_h^{\mathcal{O}\left(\mathbf{WI}_{L-1,t}(\mathcal{I})\right)}$$

$\mathcal{I}$  $\mathcal{I}^c$

Main Result: Strength of Interaction $\propto$ Walk Index

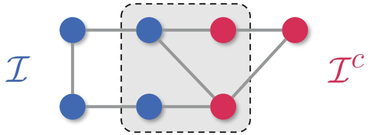
Theorem

For a depth L GNN with width D_h and $\mathcal{I} \subseteq \mathcal{V}$:

(graph prediction) $\quad \mathrm{sep}(GNN; \mathcal{I}) = D_h^{\mathcal{O}\left(\mathbf{WI}_{L-1}(\mathcal{I})\right)}$

(vertex prediction) $\quad \mathrm{sep}(GNN^{(t)}; \mathcal{I}) = D_h^{\mathcal{O}\left(\mathbf{WI}_{L-1,t}(\mathcal{I})\right)}$

* *Nearly matching lower bounds*



Main Result: Strength of Interaction $\propto$ Walk Index

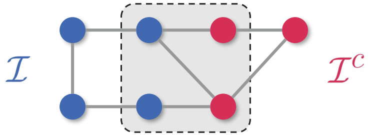
Theorem

For a depth L GNN with width D_h and $\mathcal{I} \subseteq \mathcal{V}$:

$$\textit{(graph prediction)} \quad \mathrm{sep}(GNN; \mathcal{I}) = D_h^{\mathcal{O}\left(\mathbf{WI}_{L-1}(\mathcal{I})\right)}$$

$$\textit{(vertex prediction)} \quad \mathrm{sep}(GNN^{(t)}; \mathcal{I}) = D_h^{\mathcal{O}\left(\mathbf{WI}_{L-1,t}(\mathcal{I})\right)}$$

** Nearly matching lower bounds*



Strength of interaction modeled across partition of vertices is determined by its walk index

Main Result — Proof Sketch

Theorem

(graph prediction) $\mathrm{sep}(GNN; \mathcal{I}) = D_h^{\mathcal{O}\left(\mathbf{WI}_{L-1}(\mathcal{I})\right)}$

Main Result — Proof Sketch

Theorem

$$\textit{(graph prediction)} \quad \mathrm{sep}(GNN; \mathcal{I}) = D_h^{\mathcal{O}\left(\mathbf{WI}_{L-1}(\mathcal{I})\right)}$$

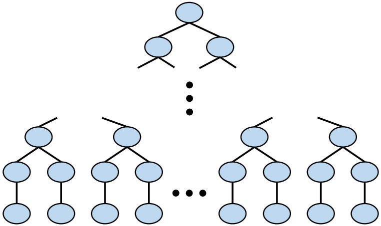
GNN w/ product aggregation can be represented as tensor network

Main Result — Proof Sketch

Theorem

(graph prediction) $\mathrm{sep}(GNN;\mathcal{I}) = D_h^{\mathcal{O}\left(\mathbf{WI}_{L-1}(\mathcal{I})\right)}$

GNN w/ product aggregation can be represented as tensor network

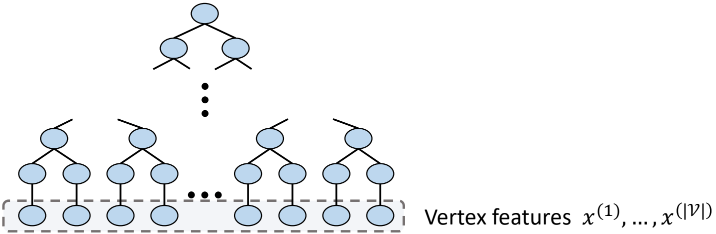


Main Result — Proof Sketch

Theorem

(graph prediction) $\mathrm{sep}(GNN; \mathcal{I}) = D_h^{\mathcal{O}\left(\mathbf{WI}_{L-1}(\mathcal{I})\right)}$

GNN w/ product aggregation can be represented as tensor network

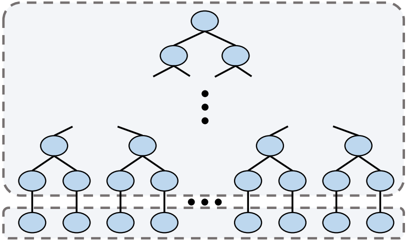


Main Result — Proof Sketch

Theorem

(graph prediction) $\mathrm{sep}(GNN; \mathcal{I}) = D_h^{\mathcal{O}(\mathbf{WI}_{L-1}(\mathcal{I}))}$

GNN w/ product aggregation can be represented as tensor network

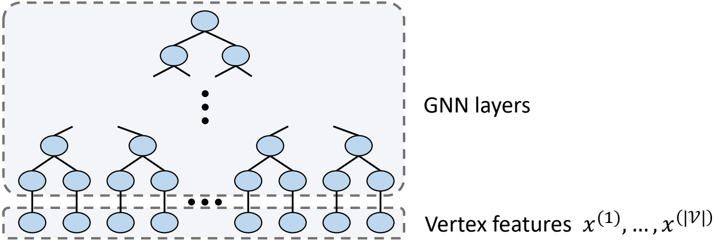


Main Result — Proof Sketch

Theorem

(graph prediction) $\mathrm{sep}(GNN; \mathcal{I}) = D_h^{\mathcal{O}\left(\mathbf{WI}_{L-1}(\mathcal{I})\right)}$

GNN w/ product aggregation can be represented as tensor network



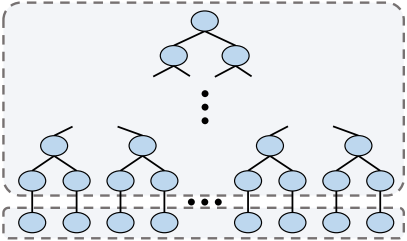
$\mathrm{sep}(GNN; \mathcal{I})$ upper bounded by min cut in tensor network

Main Result — Proof Sketch

Theorem

(graph prediction) $\mathrm{sep}(GNN; \mathcal{I}) = D_h^{\mathcal{O}\left(\mathbf{WI}_{L-1}(\mathcal{I})\right)}$

GNN w/ product aggregation can be represented as tensor network



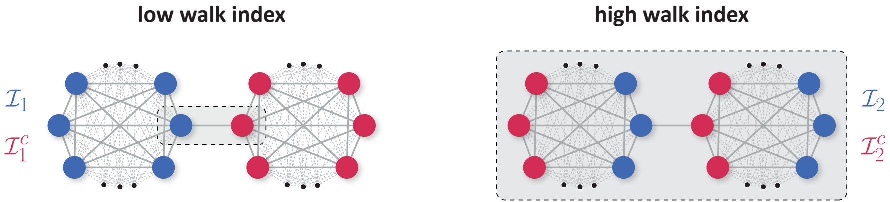
$\mathrm{sep}(GNN; \mathcal{I})$ upper bounded by min cut in tensor network

$\uparrow$

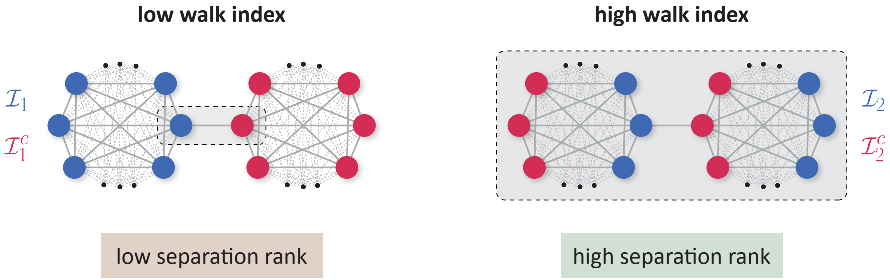
separating leaves in $\mathcal{I}$ from leaves in $\mathcal{I}^c$

Implication of Main Result

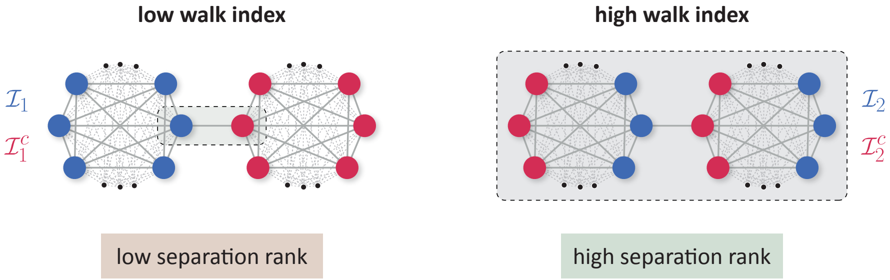
Implication of Main Result



Implication of Main Result

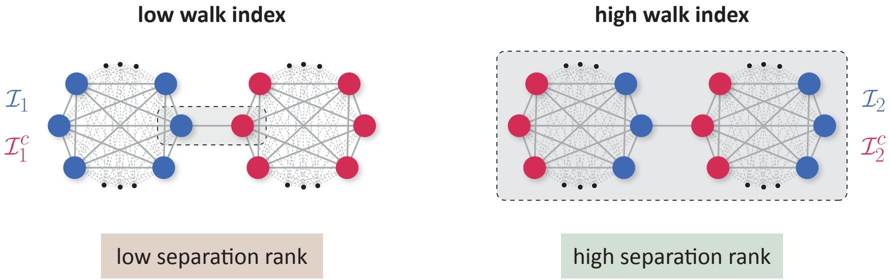


Implication of Main Result



GNNs can model stronger interactions across partitions with higher walk index

Implication of Main Result



GNNs can model stronger interactions across partitions with higher walk index

Formalizes intuition: more interconnected $\Longrightarrow$ stronger interaction

Empirical Demonstration on GNNs with ReLU

Theory Suggests

Empirical Demonstration on GNNs with ReLU

Theory Suggests

GNNs perform better on datasets requiring strong interactions across higher walk index partitions

Empirical Demonstration on GNNs with ReLU

Theory Suggests

GNNs perform better on datasets requiring strong interactions across higher walk index partitions

Experiment

Empirical Demonstration on GNNs with ReLU

Theory Suggests

GNNs perform better on datasets requiring strong interactions across higher walk index partitions

Experiment

GNNs w/ ReLU non-linearity on low vs high walk index datasets

Empirical Demonstration on GNNs with ReLU

Theory Suggests

GNNs perform better on datasets requiring strong interactions across higher walk index partitions

Experiment

GNNs w/ ReLU non-linearity on low vs high walk index datasets

Task (graph prediction): predict if two FMNIST images have same class

Empirical Demonstration on GNNs with ReLU

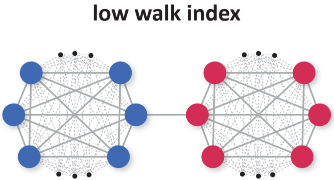
Theory Suggests

GNNs perform better on datasets requiring strong interactions across higher walk index partitions

Experiment

GNNs w/ ReLU non-linearity on low vs high walk index datasets

Task (graph prediction): predict if two FMNIST images have same class



Empirical Demonstration on GNNs with ReLU

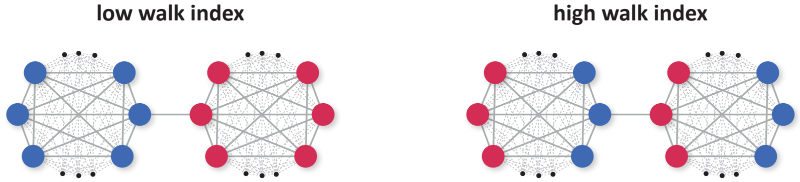
Theory Suggests

GNNs perform better on datasets requiring strong interactions across higher walk index partitions

Experiment

GNNs w/ ReLU non-linearity on low vs high walk index datasets

Task (graph prediction): predict if two FMNIST images have same class



blue vertices: patches of 1st image | red vertices: patches of 2nd image

Empirical Demonstration on GNNs with ReLU

Experiment Results

Empirical Demonstration on GNNs with ReLU

Experiment Results

		Partition Walk Index	
		Low	High
GCN	Train	70.4 ± 1.7	$\mathbf{81.4} \pm 2.0$
	Test	52.7 ± 1.9	$\mathbf{66.2} \pm 1.1$
GAT	Train	82.8 ± 2.6	$\mathbf{88.5} \pm 1.1$
	Test	69.6 ± 0.6	$\mathbf{72.1} \pm 1.2$
GIN	Train	83.2 ± 0.8	$\mathbf{94.2} \pm 0.8$
	Test	53.7 ± 1.8	$\mathbf{64.8} \pm 1.4$

Empirical Demonstration on GNNs with ReLU

Experiment Results

		Partition Walk Index	
		Low	High
GCN	Train	70.4 ± 1.7	$\mathbf{81.4} \pm 2.0$
	Test	52.7 ± 1.9	$\mathbf{66.2} \pm 1.1$
GAT	Train	82.8 ± 2.6	$\mathbf{88.5} \pm 1.1$
	Test	69.6 ± 0.6	$\mathbf{72.1} \pm 1.2$
GIN	Train	83.2 ± 0.8	$\mathbf{94.2} \pm 0.8$
	Test	53.7 ± 1.8	$\mathbf{64.8} \pm 1.4$

In accordance with our theory:

GNNs perform better on tasks entailing strong interactions across partitions with higher walk index

Outline

1. Expressivity in Graph Neural Networks (GNNs)
2. Theory: Quantifying Ability of GNNs to Model Interactions
   - Formalizing Interaction via Separation Rank
   - Analyzed GNN Architecture
   - Characterizing Strength of Modeled Interaction
3. Application: Expressivity Preserving Edge Sparsification
4. Conclusion

Edge Sparsification

Edge Sparsification

Computations over large-scale graphs are expensive

Edge Sparsification

Computations over large-scale graphs are expensive

Edge sparsification: removing edges while maintaining graph properties

(*e.g.* Baswana & Sen 2007, Spielman & Srivastava 2011, Hamann et al. 2016)

Edge Sparsification

Computations over large-scale graphs are expensive

Edge sparsification: removing edges while maintaining graph properties

(*e.g.* Baswana & Sen 2007, Spielman & Srivastava 2011, Hamann et al. 2016)

GNNs perspective: maintain accuracy when removing edges

(*e.g.* Li et al. 2020, Chen et al. 2021)

Edge Sparsification

Computations over large-scale graphs are expensive

Edge sparsification: removing edges while maintaining graph properties

(*e.g.* Baswana & Sen 2007, Spielman & Srivastava 2011, Hamann et al. 2016)

GNNs perspective: maintain accuracy when removing edges

(*e.g.* Li et al. 2020, Chen et al. 2021)

Our theory $\implies$ simple & effective recipe for pruning edges

General Walk Index Sparsification Scheme

Theory: walk index of $\mathcal{I} \subseteq \mathcal{V}$ key for modeling interaction across $\mathcal{I}, \mathcal{I}^c$

General Walk Index Sparsification Scheme

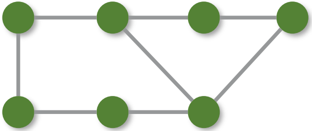
Theory: walk index of $\mathcal{I} \subseteq \mathcal{V}$ key for modeling interaction across $\mathcal{I}, \mathcal{I}^c$

Idea: greedily prune edge whose removal harms interactions the least

General Walk Index Sparsification Scheme

Theory: walk index of $\mathcal{I} \subseteq \mathcal{V}$ key for modeling interaction across $\mathcal{I}, \mathcal{I}^c$

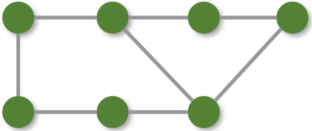
Idea: greedily prune edge whose removal harms interactions the least



General Walk Index Sparsification Scheme

Theory: walk index of $\mathcal{I} \subseteq \mathcal{V}$ key for modeling interaction across $\mathcal{I}, \mathcal{I}^c$

Idea: greedily prune edge whose removal harms interactions the least

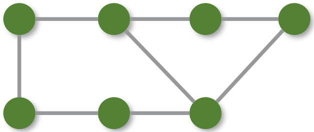


Algorithm: until desired # edges are removed:

General Walk Index Sparsification Scheme

Theory: walk index of $\mathcal{I} \subseteq \mathcal{V}$ key for modeling interaction across $\mathcal{I}, \mathcal{I}^c$

Idea: greedily prune edge whose removal harms interactions the least



Algorithm: until desired # edges are removed:

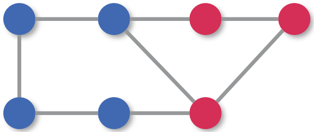
(1) Choose partitions $\mathcal{I}_1, \ldots, \mathcal{I}_M$ to preserve modeled interactions for

General Walk Index Sparsification Scheme

Theory: walk index of $\mathcal{I} \subseteq \mathcal{V}$ key for modeling interaction across $\mathcal{I}, \mathcal{I}^c$

Idea: greedily prune edge whose removal harms interactions the least

$\mathcal{I}_1$



Algorithm: until desired # edges are removed:

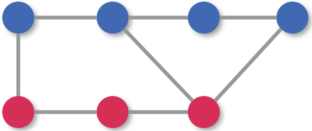
(1) Choose partitions $\mathcal{I}_1, \ldots, \mathcal{I}_M$ to preserve modeled interactions for

General Walk Index Sparsification Scheme

Theory: walk index of $\mathcal{I} \subseteq \mathcal{V}$ key for modeling interaction across $\mathcal{I}, \mathcal{I}^c$

Idea: greedily prune edge whose removal harms interactions the least

$\mathcal{I}_2$



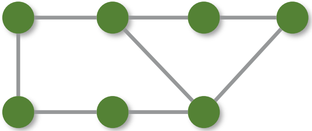
Algorithm: until desired # edges are removed:

(1) Choose partitions $\mathcal{I}_1, \ldots, \mathcal{I}_M$ to preserve modeled interactions for

General Walk Index Sparsification Scheme

Theory: walk index of $\mathcal{I} \subseteq \mathcal{V}$ key for modeling interaction across $\mathcal{I}, \mathcal{I}^c$

Idea: greedily prune edge whose removal harms interactions the least



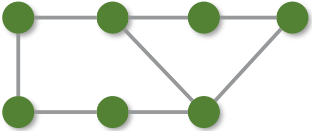
Algorithm: until desired # edges are removed:

(1) Choose partitions $\mathcal{I}_1, \ldots, \mathcal{I}_M$ to preserve modeled interactions for

General Walk Index Sparsification Scheme

Theory: walk index of $\mathcal{I} \subseteq \mathcal{V}$ key for modeling interaction across $\mathcal{I}, \mathcal{I}^c$

Idea: greedily prune edge whose removal harms interactions the least



Algorithm: until desired # edges are removed:

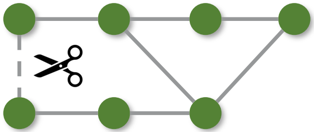
(1) Choose partitions $\mathcal{I}_1, \ldots, \mathcal{I}_M$ to preserve modeled interactions for

(2) Per edge, compute $(L-1)$-walk indices of $\mathcal{I}_1, \ldots, \mathcal{I}_M$ after its removal

General Walk Index Sparsification Scheme

Theory: walk index of $\mathcal{I} \subseteq \mathcal{V}$ key for modeling interaction across $\mathcal{I}, \mathcal{I}^c$

Idea: greedily prune edge whose removal harms interactions the least



$s_1 =$

Algorithm: until desired # edges are removed:

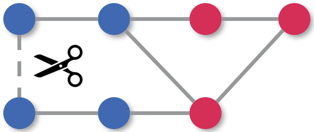
(1) Choose partitions $\mathcal{I}_1, \ldots, \mathcal{I}_M$ to preserve modeled interactions for

(2) Per edge, compute $(L-1)$-walk indices of $\mathcal{I}_1, \ldots, \mathcal{I}_M$ after its removal

General Walk Index Sparsification Scheme

Theory: walk index of $\mathcal{I} \subseteq \mathcal{V}$ key for modeling interaction across $\mathcal{I}, \mathcal{I}^c$

Idea: greedily prune edge whose removal harms interactions the least



$$s_1 = WI_{L-1}(\mathcal{I}_1)$$

Algorithm: until desired # edges are removed:

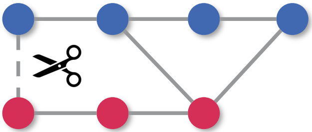
(1) Choose partitions $\mathcal{I}_1, \ldots, \mathcal{I}_M$ to preserve modeled interactions for

(2) Per edge, compute $(L-1)$-walk indices of $\mathcal{I}_1, \ldots, \mathcal{I}_M$ after its removal

General Walk Index Sparsification Scheme

Theory: walk index of $\mathcal{I} \subseteq \mathcal{V}$ key for modeling interaction across $\mathcal{I}, \mathcal{I}^c$

Idea: greedily prune edge whose removal harms interactions the least



$$s_1 = WI_{L-1}(\mathcal{I}_1)\,, WI_{L-1}(\mathcal{I}_2)$$

Algorithm: until desired # edges are removed:

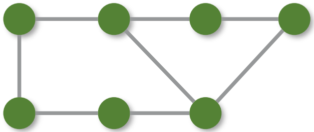
(1) Choose partitions $\mathcal{I}_1, \ldots, \mathcal{I}_M$ to preserve modeled interactions for

(2) Per edge, compute $(L-1)$-walk indices of $\mathcal{I}_1, \ldots, \mathcal{I}_M$ after its removal

General Walk Index Sparsification Scheme

Theory: walk index of $\mathcal{I} \subseteq \mathcal{V}$ key for modeling interaction across $\mathcal{I}, \mathcal{I}^c$

Idea: greedily prune edge whose removal harms interactions the least



Algorithm: until desired # edges are removed:

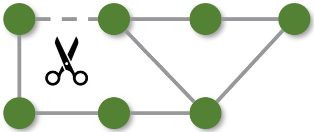
(1) Choose partitions $\mathcal{I}_1, \ldots, \mathcal{I}_M$ to preserve modeled interactions for

(2) Per edge, compute $(L-1)$-walk indices of $\mathcal{I}_1, \ldots, \mathcal{I}_M$ after its removal

General Walk Index Sparsification Scheme

Theory: walk index of $\mathcal{I} \subseteq \mathcal{V}$ key for modeling interaction across $\mathcal{I}, \mathcal{I}^c$

Idea: greedily prune edge whose removal harms interactions the least



$s_2 =$

Algorithm: until desired # edges are removed:

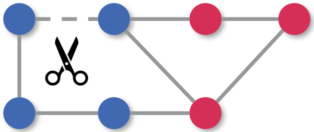
(1) Choose partitions $\mathcal{I}_1, \ldots, \mathcal{I}_M$ to preserve modeled interactions for

(2) Per edge, compute $(L-1)$-walk indices of $\mathcal{I}_1, \ldots, \mathcal{I}_M$ after its removal

General Walk Index Sparsification Scheme

Theory: walk index of $\mathcal{I} \subseteq \mathcal{V}$ key for modeling interaction across $\mathcal{I}, \mathcal{I}^c$

Idea: greedily prune edge whose removal harms interactions the least



$s_2 = WI_{L-1}(\mathcal{I}_1)$

Algorithm: until desired # edges are removed:

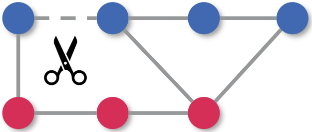
(1) Choose partitions $\mathcal{I}_1, \ldots, \mathcal{I}_M$ to preserve modeled interactions for

(2) Per edge, compute $(L-1)$-walk indices of $\mathcal{I}_1, \ldots, \mathcal{I}_M$ after its removal

General Walk Index Sparsification Scheme

Theory: walk index of $\mathcal{I} \subseteq \mathcal{V}$ key for modeling interaction across $\mathcal{I}, \mathcal{I}^c$

Idea: greedily prune edge whose removal harms interactions the least



$s_2 = WI_{L-1}(\mathcal{I}_1) \, , WI_{L-1}(\mathcal{I}_2)$

Algorithm: until desired # edges are removed:

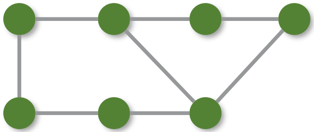
(1) Choose partitions $\mathcal{I}_1, \ldots, \mathcal{I}_M$ to preserve modeled interactions for

(2) Per edge, compute $(L-1)$-walk indices of $\mathcal{I}_1, \ldots, \mathcal{I}_M$ after its removal

General Walk Index Sparsification Scheme

Theory: walk index of $\mathcal{I} \subseteq \mathcal{V}$ key for modeling interaction across $\mathcal{I}, \mathcal{I}^c$

Idea: greedily prune edge whose removal harms interactions the least



$s_1, s_2, \ldots, s_8$

Algorithm: until desired # edges are removed:

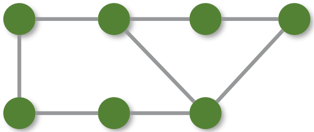
(1) Choose partitions $\mathcal{I}_1, \ldots, \mathcal{I}_M$ to preserve modeled interactions for

(2) Per edge, compute $(L-1)$-walk indices of $\mathcal{I}_1, \ldots, \mathcal{I}_M$ after its removal

General Walk Index Sparsification Scheme

Theory: walk index of $\mathcal{I} \subseteq \mathcal{V}$ key for modeling interaction across $\mathcal{I}, \mathcal{I}^c$

Idea: greedily prune edge whose removal harms interactions the least



$s_1, s_2, \ldots, s_8$

Algorithm: until desired # edges are removed:

(1) Choose partitions $\mathcal{I}_1, \ldots, \mathcal{I}_M$ to preserve modeled interactions for

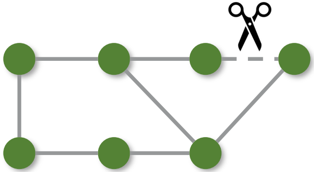
(2) Per edge, compute $(L-1)$-walk indices of $\mathcal{I}_1, \ldots, \mathcal{I}_M$ after its removal

(3) Remove edge with maximal walk index tuple

General Walk Index Sparsification Scheme

Theory: walk index of $\mathcal{I} \subseteq \mathcal{V}$ key for modeling interaction across $\mathcal{I}, \mathcal{I}^c$

Idea: greedily prune edge whose removal harms interactions the least



$s_1, s_2, \ldots, s_8$

Algorithm: until desired # edges are removed:

(1) Choose partitions $\mathcal{I}_1, \ldots, \mathcal{I}_M$ to preserve modeled interactions for

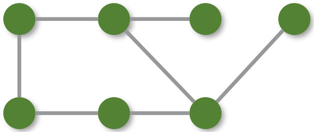
(2) Per edge, compute $(L-1)$-walk indices of $\mathcal{I}_1, \ldots, \mathcal{I}_M$ after its removal

(3) Remove edge with maximal walk index tuple

General Walk Index Sparsification Scheme

Theory: walk index of $\mathcal{I} \subseteq \mathcal{V}$ key for modeling interaction across $\mathcal{I}, \mathcal{I}^c$

Idea: greedily prune edge whose removal harms interactions the least



Algorithm: until desired # edges are removed:

(1) Choose partitions $\mathcal{I}_1, \ldots, \mathcal{I}_M$ to preserve modeled interactions for

(2) Per edge, compute $(L-1)$-walk indices of $\mathcal{I}_1, \ldots, \mathcal{I}_M$ after its removal

(3) Remove edge with maximal walk index tuple

Walk Index Sparsification (WIS)

We focus on vertex prediction (most relevant in large graphs)

Walk Index Sparsification (WIS)

We focus on vertex prediction (most relevant in large graphs)

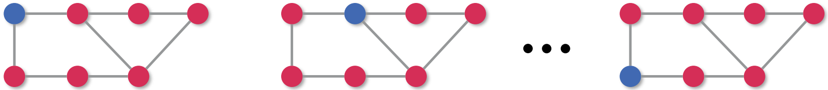
$(L-1)$-Walk Index Sparsification (WIS)

Walk Index Sparsification (WIS)

We focus on vertex prediction (most relevant in large graphs)

$(L-1)$-Walk Index Sparsification (WIS)

- Choose partitions separating a vertex from all others

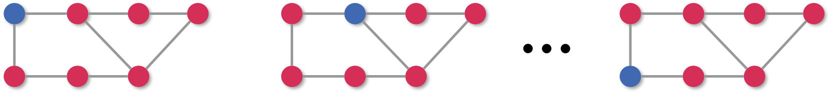


Walk Index Sparsification (WIS)

We focus on vertex prediction (most relevant in large graphs)

($L-1$)-Walk Index Sparsification (WIS)

- Choose partitions separating a vertex from all others



- Order tuples by minimal entry, breaking ties using second smallest,...

1-Walk Index Sparsification (1-WIS)

1-Walk Index Sparsification (1-WIS)

Particularly simple & efficient implementation

1-Walk Index Sparsification (1-WIS)

Particularly simple & efficient implementation

Algorithm: until desired # edges are removed:

1-Walk Index Sparsification (1-WIS)

Particularly simple & efficient implementation

Algorithm: until desired # edges are removed:

(1) Compute vertex degrees

1-Walk Index Sparsification (1-WIS)

Particularly simple & efficient implementation

Algorithm: until desired # edges are removed:

(1) Compute vertex degrees

(2) Remove edge $\{i, j\}$ with maximal $\min\{\deg(i), \deg(j)\}$

1-Walk Index Sparsification (1-WIS)

Particularly simple & efficient implementation

Algorithm: until desired # edges are removed:

(1) Compute vertex degrees

(2) Remove edge $\{i, j\}$ with maximal $\min\{\deg(i), \deg(j)\}$

(break ties via $\max\{\deg(i), \deg(j)\}$)

Comparison of Edge Sparsification Methods

Experiment

Compare edge sparsification methods over standard benchmarks

Comparison of Edge Sparsification Methods

Experiment

Compare edge sparsification methods over standard benchmarks

Baselines: random

Comparison of Edge Sparsification Methods

Experiment

Compare edge sparsification methods over standard benchmarks

Baselines: random, spectral (Spielman & Srivastava 2011),

Comparison of Edge Sparsification Methods

Experiment

Compare edge sparsification methods over standard benchmarks

Baselines: random, spectral (Spielman & Srivastava 2011), UGS (Chen et al. 2021)

Comparison of Edge Sparsification Methods

Experiment

Compare edge sparsification methods over standard benchmarks

Baselines: random, spectral (Spielman & Srivastava 2011), UGS (Chen et al. 2021)

Model: depth $L = 3$ GCN (similar results using GIN)

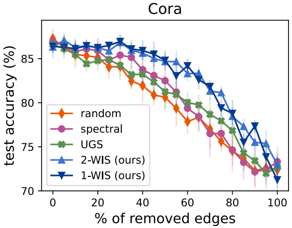
Comparison of Edge Sparsification Methods

Experiment

Compare edge sparsification methods over standard benchmarks

Baselines: random, spectral (Spielman & Srivastava 2011), UGS (Chen et al. 2021)

Model: depth $L = 3$ GCN (similar results using GIN)



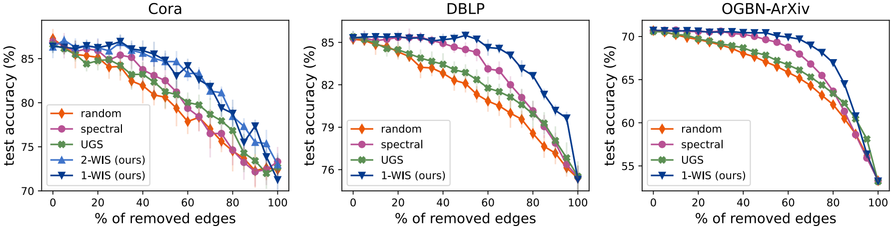
Comparison of Edge Sparsification Methods

Experiment

Compare edge sparsification methods over standard benchmarks

Baselines: random, spectral (Spielman & Srivastava 2011), UGS (Chen et al. 2021)

Model: depth $L = 3$ GCN (similar results using GIN)



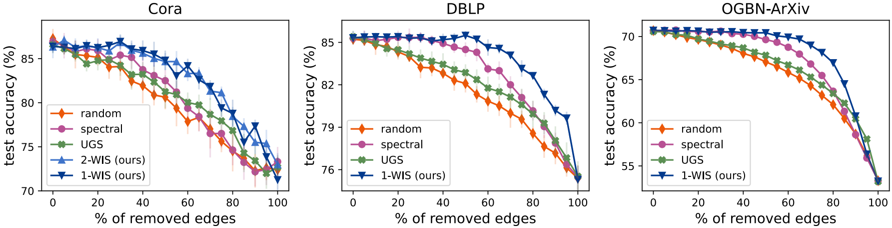
Comparison of Edge Sparsification Methods

Experiment

Compare edge sparsification methods over standard benchmarks

Baselines: random, spectral (Spielman & Srivastava 2011), UGS (Chen et al. 2021)

Model: depth $L = 3$ GCN (similar results using GIN)



WIS outperforms existing methods while being simple & efficient

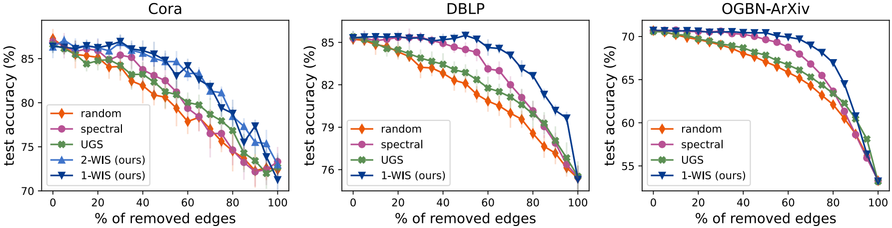
Comparison of Edge Sparsification Methods

Experiment

Compare edge sparsification methods over standard benchmarks

Baselines: random, spectral (Spielman & Srivastava 2011), UGS (Chen et al. 2021)

Model: depth $L = 3$ GCN (similar results using GIN)



WIS outperforms existing methods while being simple & efficient

Code: `https://github.com/noamrazin/gnn_interactions`

Outline

Conclusion

Conclusion

Theoretical Analysis

Conclusion

Theoretical Analysis

- Characterized ability of certain GNNs to model interactions

Conclusion

Theoretical Analysis

- Characterized ability of certain GNNs to model interactions
- Walk index of a partition controls strength of interaction

Conclusion

Theoretical Analysis

- Characterized ability of certain GNNs to model interactions
- Walk index of a partition controls strength of interaction

Practical Application

Conclusion

Theoretical Analysis

- Characterized ability of certain GNNs to model interactions
- Walk index of a partition controls strength of interaction

Practical Application

- Derived WIS: an edge sparsification algorithm preserving interactions

Conclusion

Theoretical Analysis

- Characterized ability of certain GNNs to model interactions
- Walk index of a partition controls strength of interaction

Practical Application

- Derived WIS: an edge sparsification algorithm preserving interactions
- WIS is simple, efficient, and outperforms alternative methods

Conclusion

Theoretical Analysis

- Characterized ability of certain GNNs to model interactions
- Walk index of a partition controls strength of interaction

Practical Application

- Derived WIS: an edge sparsification algorithm preserving interactions
- WIS is simple, efficient, and outperforms alternative methods

Going Forward

Conclusion

Theoretical Analysis

- Characterized ability of certain GNNs to model interactions
- Walk index of a partition controls strength of interaction

Practical Application

- Derived WIS: an edge sparsification algorithm preserving interactions
- WIS is simple, efficient, and outperforms alternative methods

Going Forward

Studying modeled interactions may be key for:

Conclusion

Theoretical Analysis

- Characterized ability of certain GNNs to model interactions
- Walk index of a partition controls strength of interaction

Practical Application

- Derived WIS: an edge sparsification algorithm preserving interactions
- WIS is simple, efficient, and outperforms alternative methods

Going Forward

Studying modeled interactions may be key for:

- Understanding aspects beyond expressivity: *e.g.* generalization

Conclusion

Theoretical Analysis

- Characterized ability of certain GNNs to model interactions
- Walk index of a partition controls strength of interaction

Practical Application

- Derived WIS: an edge sparsification algorithm preserving interactions
- WIS is simple, efficient, and outperforms alternative methods

Going Forward

Studying modeled interactions may be key for:

- Understanding aspects beyond expressivity: *e.g.* generalization
- Improving performance of GNNs

Thank You!

Work supported by:
Apple Scholars in AI/ML PhD fellowship, Google Research Scholar Award, Google Research Gift, the Yandex Initiative in Machine Learning, the Israel Science Foundation (grant 1780/21), Len Blavatnik and the Blavatnik Family Foundation, Tel Aviv University Center for AI and Data Science, and Amnon and Anat Shashua.