Stability of Graph Neural Networks to Relative Perturbations

Fernando Gama, Joan Bruna, and Alejandro Ribeiro
Dept. of Electrical and Systems Engineering
University of Pennsylvania

Supported by NSF CCF 1717120, ARO W911NF1710438,
ARL DCIST CRA W911NF-17-2-0181, ISTC-WAS and Intel DevCloud

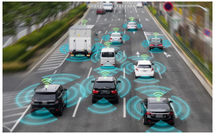
May 8, 2020
45th Int. Conf. Acoustics, Speech and Signal Processing (ICASSP 2020)

- Graphs are models of signal structure ⇒ Network data ⇒ Leverage in learning from network data

Robot coordination

Smart grids

Remote sensing



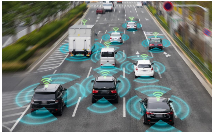
Traffic coordination

- Graphs are models of signal structure $\Rightarrow$ Network data $\Rightarrow$ Leverage in learning from network data

Robot coordination

Smart grids

Remote sensing



Traffic coordination

- Scalability $\Rightarrow$ Process data from arbitrarily large networks
- Exploit data structure, local information $\Rightarrow$ Fast training and moderate dataset size
- Distributed computations $\Rightarrow$ Efficient implementation

- **Graph Neural Networks**
  - $\Rightarrow$ Graph Signal Processing $\Rightarrow$ Mathematical framework
  - $\Rightarrow$ Graph convolutions $\Rightarrow$ Local, distributed $\Rightarrow$ Generalize time convolutions

- **Graph Neural Networks**
  - $\Rightarrow$ Graph Signal Processing $\Rightarrow$ Mathematical framework
  - $\Rightarrow$ Graph convolutions $\Rightarrow$ Local, distributed $\Rightarrow$ Generalize time convolutions
- **Equivariance and stability** $\Rightarrow$ Transferability and scalability
  - $\Rightarrow$ Permutation equivariance $\Rightarrow$ Exploit structure
  - $\Rightarrow$ Stability to changes in the underlying network

- **Graph Neural Networks**
  - $\Rightarrow$ Graph Signal Processing $\Rightarrow$ Mathematical framework
  - $\Rightarrow$ Graph convolutions $\Rightarrow$ Local, distributed $\Rightarrow$ Generalize time convolutions
- **Equivariance and stability** $\Rightarrow$ Transferability and scalability
  - $\Rightarrow$ Permutation equivariance $\Rightarrow$ Exploit structure
  - $\Rightarrow$ Stability to changes in the underlying network

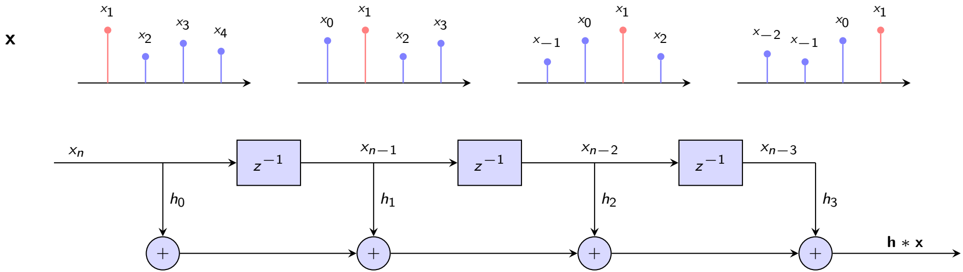
Stability to Perturbations

A small change in the graph support causes a small change in the output of the GNN

Outline

- Graph convolution $\Rightarrow$ Linear combination of shifted versions of the signal $\mathbf{x}$

$$\mathbf{x} * \mathbf{h} = \sum_{k=0}^{K-1} h_k x_{n-k}$$



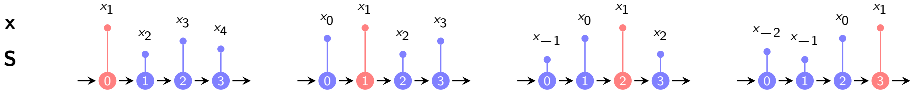
Gama, Marques, Leus, Ribeiro, "Convolutional Graph Neural Networks", Asilomar, 2019.

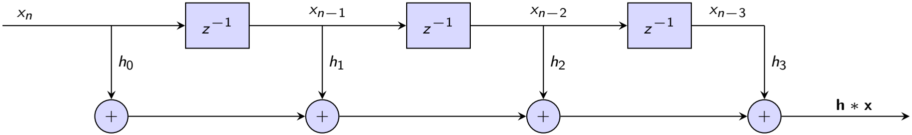
- Graph convolution $\Rightarrow$ Linear combination of shifted versions of the signal $\mathbf{x}$

$$\mathbf{x} * \mathbf{h} = \sum_{k=0}^{K-1} h_k x_{n-k}$$

- Notion of shift $\mathbf{S}$ $\Rightarrow$ Matrix description of graph (adjacency, Laplacian)

$$\begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & 0 & 0 & 0 & \cdots \\ \cdots & 1 & 0 & 0 & \cdots \\ \cdots & 0 & 1 & 0 & \cdots \\ \cdots & 0 & 0 & 1 & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix}$$





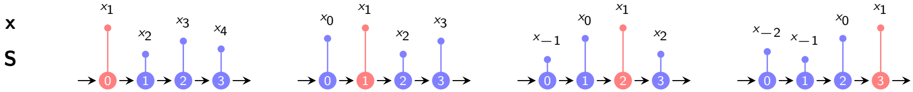
Gama, Marques, Leus, Ribeiro, "Convolutional Graph Neural Networks", Asilomar, 2019.

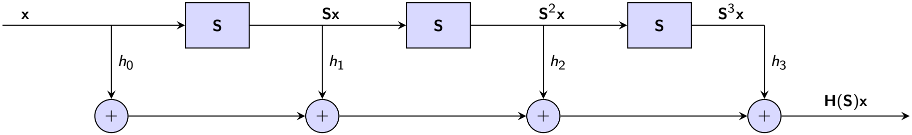
- Graph convolution $\Rightarrow$ Linear combination of shifted versions of the signal $\mathbf{x}$

$$\mathbf{x} *_{\mathbf{S}} \mathbf{h} = \sum_{k=0}^{K-1} h_k \, \mathbf{S}^k \mathbf{x}$$

- Notion of shift $\mathbf{S}$ $\Rightarrow$ Matrix description of graph $\Rightarrow$ $\mathbf{S}\mathbf{x}$ shifts the signal $\mathbf{x}$

$$\begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & 0 & 0 & 0 & \cdots \\ \cdots & 1 & 0 & 0 & \cdots \\ \cdots & 0 & 1 & 0 & \cdots \\ \cdots & 0 & 0 & 1 & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix} \begin{bmatrix} \vdots \\ x_1 \\ x_2 \\ x_3 \\ \vdots \end{bmatrix}$$





Gama, Marques, Leus, Ribeiro, "Convolutional Graph Neural Networks", Asilomar, 2019.

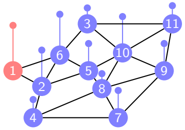
- Graph convolution $\Rightarrow$ Linear combination of shifted versions of the signal $\mathbf{x}$

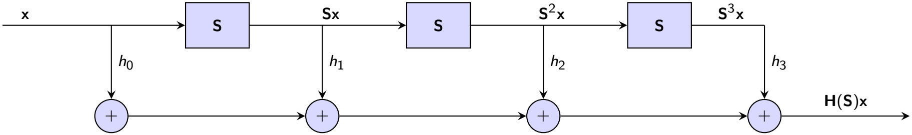
$$\mathbf{x} *_{\mathbf{S}} \mathbf{h} = \sum_{k=0}^{K-1} h_k \, \mathbf{S}^k \mathbf{x}$$

- Notion of shift $\mathbf{S}$ $\Rightarrow$ Matrix description of graph $\Rightarrow$ $\mathbf{Sx}$ shifts the signal $\mathbf{x}$

$$\begin{bmatrix} 0&1&0&0&0&1&0&0&0&0&0 \\ 1&0&0&1&1&1&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&1&1 \\ 0&1&0&0&0&0&1&1&0&0&0 \\ 0&1&0&0&1&0&0&1&0&1&0 \\ 1&1&1&0&1&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&1&1&0&0 \\ 0&0&0&1&1&0&1&0&1&1&0 \\ 0&0&0&0&0&0&1&1&0&1&1 \\ 0&0&0&0&1&0&0&1&1&0&1 \\ 0&0&1&0&0&0&0&0&1&1&0 \end{bmatrix}$$

$(\mathbf{x}; \mathbf{S})$





Gama, Marques, Leus, Ribeiro, "Convolutional Graph Neural Networks", Asilomar, 2019.

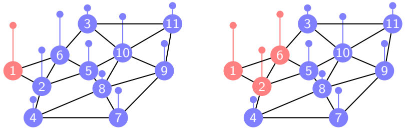
- Graph convolution $\Rightarrow$ Linear combination of shifted versions of the signal $\mathbf{x}$

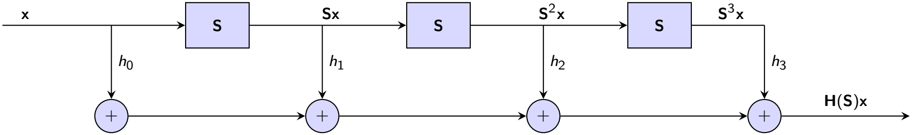
$$\mathbf{x} *_{\mathbf{S}} \mathbf{h} = \sum_{k=0}^{K-1} h_k \, \mathbf{S}^k \mathbf{x}$$

- Notion of shift $\mathbf{S}$ $\Rightarrow$ Matrix description of graph $\Rightarrow$ $\mathbf{Sx}$ shifts the signal $\mathbf{x}$

$$\begin{bmatrix} 0&1&0&0&0&1&0&0&0&0&0 \\ 1&0&0&1&1&1&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&1&1 \\ 0&1&0&0&0&0&1&1&0&0&0 \\ 0&1&0&0&1&0&0&1&0&1&0 \\ 1&1&1&0&1&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&1&1&0&0 \\ 0&0&0&1&1&0&1&0&1&1&0 \\ 0&0&0&0&0&0&1&1&0&1&1 \\ 0&0&0&0&1&0&0&1&1&0&1 \\ 0&0&1&0&0&0&0&0&1&1&0 \end{bmatrix}$$

$(\mathbf{x}; \mathbf{S})$





Gama, Marques, Leus, Ribeiro, "Convolutional Graph Neural Networks", Asilomar, 2019.

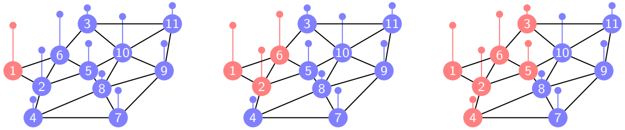
- Graph convolution $\Rightarrow$ Linear combination of shifted versions of the signal $\mathbf{x}$

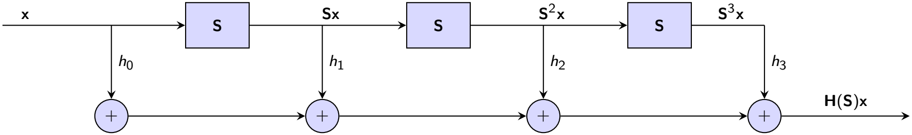
$$\mathbf{x} *_{\mathbf{S}} \mathbf{h} = \sum_{k=0}^{K-1} h_k \, \mathbf{S}^k \mathbf{x}$$

- Notion of shift $\mathbf{S}$ $\Rightarrow$ Matrix description of graph $\Rightarrow$ $\mathbf{Sx}$ shifts the signal $\mathbf{x}$

$$\begin{bmatrix} 0&1&0&0&0&1&0&0&0&0&0 \\ 1&0&0&1&1&1&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&1&1 \\ 0&1&0&0&0&0&1&1&0&0&0 \\ 0&1&0&0&1&0&0&1&0&1&0 \\ 1&1&1&0&1&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&1&1&0&0 \\ 0&0&0&1&1&0&1&0&1&1&0 \\ 0&0&0&0&0&0&1&1&0&1&1 \\ 0&0&0&0&1&0&0&1&1&0&1 \\ 0&0&1&0&0&0&0&0&1&1&0 \end{bmatrix}$$

$(\mathbf{x}; \mathbf{S})$





Gama, Marques, Leus, Ribeiro, “Convolutional Graph Neural Networks”, Asilomar, 2019.

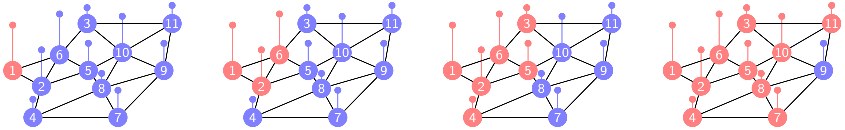
- Graph **convolution** $\Rightarrow$ Linear combination of shifted versions of the signal $\mathbf{x}$

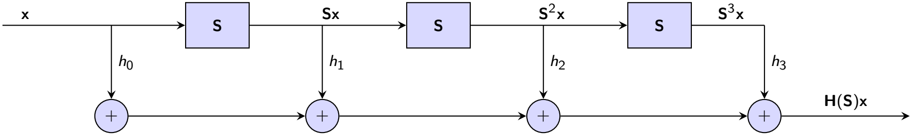
$$\mathbf{x} *_{\mathbf{S}} \mathbf{h} = \sum_{k=0}^{K-1} h_k \, \mathbf{S}^k \mathbf{x}$$

$$\begin{bmatrix}
0&1&0&0&0&1&0&0&0&0&0\\
1&0&0&1&1&1&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&1&1\\
0&1&0&0&0&0&1&1&0&0&0\\
0&1&0&0&1&0&0&1&0&1&0\\
1&1&1&0&1&0&0&0&0&0&0\\
0&0&0&1&0&0&0&1&1&0&0\\
0&0&0&1&1&0&1&0&1&1&0\\
0&0&0&0&0&0&1&1&0&1&1\\
0&0&0&0&1&0&0&1&1&0&1\\
0&0&1&0&0&0&0&0&1&1&0
\end{bmatrix}$$

- Notion of shift $\mathbf{S}$ $\Rightarrow$ Matrix description of graph $\Rightarrow$ $\mathbf{S}\mathbf{x}$ **shifts** the signal $\mathbf{x}$

$(\mathbf{x}; \mathbf{S})$





Gama, Marques, Leus, Ribeiro, "Convolutional Graph Neural Networks", Asilomar, 2019.

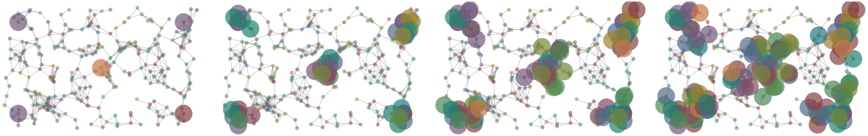
Graph Convolutions

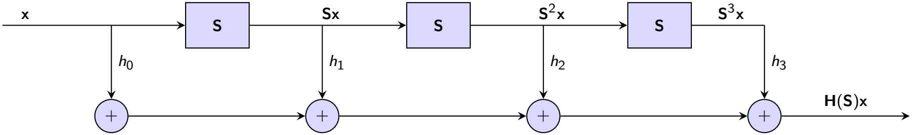
- Graph convolution $\Rightarrow$ Linear combination of shifted versions of the signal

$$\mathbf{x} *_{\mathbf{S}} \mathbf{h} = \sum_{k=0}^{K-1} h_k \, \mathbf{S}^k \mathbf{x} = \mathbf{H}(\mathbf{S})\mathbf{x}$$

- Notion of shift $\mathbf{S}$ $\Rightarrow$ Matrix description of graph (adjacency, Laplacian)
- Linear combination of neighboring signal $\Rightarrow$ Local operation

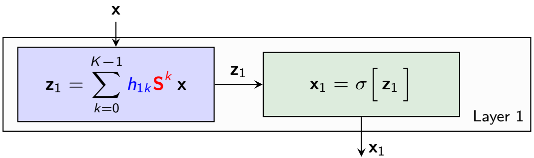
$(\mathbf{x}; \mathbf{S})$





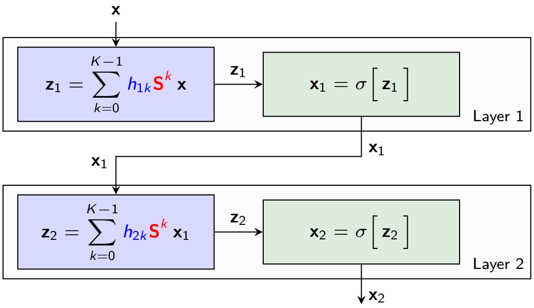
Gama, Marques, Leus, Ribeiro, "Convolutional Graph Neural Networks", Asilomar, 2019.

- Cascade of L layers
  - $\Rightarrow$ Graph convolutions with filters $\mathcal{H} = \{\mathbf{h}_\ell\}$
  - $\Rightarrow$ Pointwise nonlinearity (activation functions)



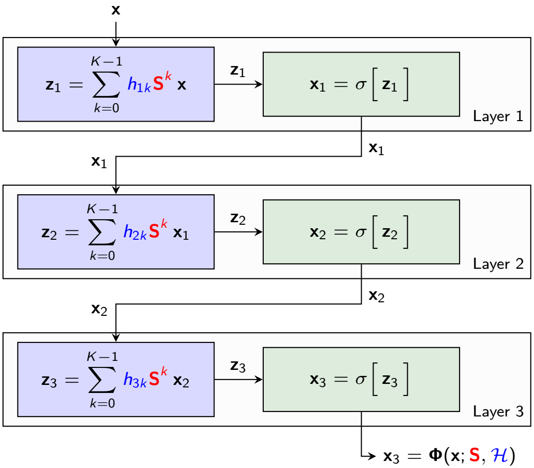
Gama, Marques, Leus, Ribeiro, "Convolutional Neural Network Architectures for Signals Supported on Graphs", IEEE TSP, 2019

- Cascade of L layers
  - $\Rightarrow$ Graph convolutions with filters $\mathcal{H} = \{\mathbf{h}_\ell\}$
  - $\Rightarrow$ Pointwise nonlinearity (activation functions)



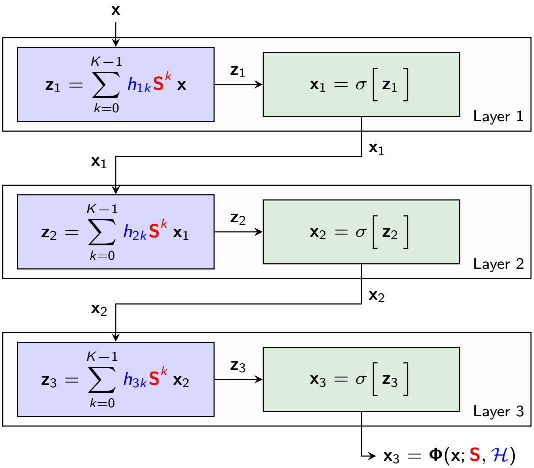
Gama, Marques, Leus, Ribeiro, "Convolutional Neural Network Architectures for Signals Supported on Graphs", IEEE TSP, 2019

- Cascade of L layers
  - $\Rightarrow$ Graph convolutions with filters $\mathcal{H} = \{\mathbf{h}_\ell\}$
  - $\Rightarrow$ Pointwise nonlinearity (activation functions)
- The GNN $\mathbf{\Phi}(\mathbf{x}; \mathbf{S}, \mathcal{H})$ depends on the filters $\mathcal{H}$
  - $\Rightarrow$ Learn filter taps $\mathcal{H}$ from training data
  - $\Rightarrow$ Also depends on the graph $\mathbf{S}$



Gama, Marques, Leus, Ribeiro, "Convolutional Neural Network Architectures for Signals Supported on Graphs", IEEE TSP, 2019

- Cascade of L layers
  - $\Rightarrow$ Graph convolutions with filters $\mathcal{H} = \{\mathbf{h}_\ell\}$
  - $\Rightarrow$ Pointwise nonlinearity (activation functions)
- The GNN $\mathbf{\Phi}(\mathbf{x}; \mathbf{S}, \mathcal{H})$ depends on the filters $\mathcal{H}$
  - $\Rightarrow$ Learn filter taps $\mathcal{H}$ from training data
  - $\Rightarrow$ Also depends on the graph $\mathbf{S}$
- Nonlinear mapping $\mathbf{\Phi}(\mathbf{x}; \mathbf{S}, \mathcal{H})$
  - $\Rightarrow$ Exploit underlying graph structure $\mathbf{S}$
  - $\Rightarrow$ Local information
  - $\Rightarrow$ Distributed implementation



Gama, Marques, Leus, Ribeiro, "Convolutional Neural Network Architectures for Signals Supported on Graphs", IEEE TSP, 2019

- Time convolutions are intuitive. Graph convolutions not so much.
  $\Rightarrow$ Local information, efficient implementation (distributed)
- CNNs are good at machine learning $\Rightarrow$ Translation equivariant, stable [Mallat '12]
- Permutation equivariance $\Rightarrow$ Exploit internal symmetries of the graph
- Stability to graph perturbations $\Rightarrow$ Similar graphs yield similar outputs
- Permutation Equivariance + Stablity $\Rightarrow$ Scalability and transferability

- Consider the graph convolution operator $\mathbf{H}(\mathbf{S})\mathbf{x} = \sum_{k=0}^{\infty} h_k \mathbf{S}^k \mathbf{x}$
- Depends on filter parameters $\mathbf{h} = \{h_k\}_{k=0}^{\infty}$ and shift operator $\mathbf{S}$; applied to the input signal $\mathbf{x}$

- Consider the graph convolution operator $\mathbf{H}(\mathbf{S})\mathbf{x} = \sum_{k=0}^{\infty} h_k \, \mathbf{S}^k \, \mathbf{x}$
- Depends on filter parameters $\mathbf{h} = \{h_k\}_{k=0}^{\infty}$ and shift operator $\mathbf{S}$; applied to the input signal $\mathbf{x}$

Theorem

Graph convolutions are equivariant to permutations. For graphs with permuted shift operators $\hat{\mathbf{S}} = \mathbf{P}^{\mathsf{T}}\mathbf{S}\mathbf{P}$ and permuted graph signals $\hat{\mathbf{x}} = \mathbf{P}^{\mathsf{T}}\mathbf{x}$ it holds

$$\mathbf{H}(\hat{\mathbf{S}})\hat{\mathbf{x}} = \mathbf{P}^{\mathsf{T}}\mathbf{H}(\mathbf{S})\mathbf{x}$$

Proof $\Rightarrow$ $\mathbf{H}(\hat{\mathbf{S}})\hat{\mathbf{x}} = \sum_{k=0}^{\infty} h_k \, \hat{\mathbf{S}}^k \hat{\mathbf{x}} = \sum_{k=0}^{\infty} h_k \, (\mathbf{P}^{\mathsf{T}}\mathbf{S}\mathbf{P})^k \mathbf{P}^{\mathsf{T}}\mathbf{x} = \mathbf{P}^{\mathsf{T}} \left(\sum_{k=0}^{\infty} h_k \, \mathbf{S}^k \mathbf{x} \right) = \mathbf{P}^{\mathsf{T}}\mathbf{H}(\mathbf{S})\mathbf{x}$

- Consider the graph convolution operator $\mathbf{H}(\mathbf{S})\mathbf{x} = \sum_{k=0}^{\infty} h_k \, \mathbf{S}^k \, \mathbf{x}$
- Depends on filter parameters $\mathbf{h} = \{h_k\}_{k=0}^{\infty}$ and shift operator $\mathbf{S}$; applied to the input signal $\mathbf{x}$

Theorem

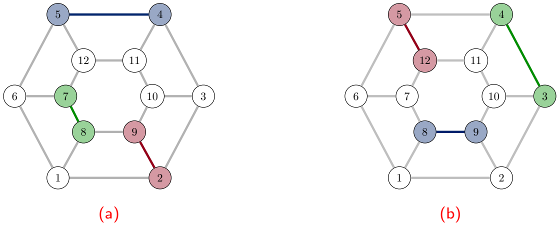
Graph convolutions are equivariant to permutations. For graphs with permuted shift operators $\hat{\mathbf{S}} = \mathbf{P}^\mathsf{T}\mathbf{S}\mathbf{P}$ and permuted graph signals $\hat{\mathbf{x}} = \mathbf{P}^\mathsf{T}\mathbf{x}$ it holds

$$\mathbf{H}(\hat{\mathbf{S}})\hat{\mathbf{x}} = \mathbf{P}^\mathsf{T}\mathbf{H}(\mathbf{S})\mathbf{x}$$

Proof $\Rightarrow$ $\mathbf{H}(\hat{\mathbf{S}})\hat{\mathbf{x}} = \sum_{k=0}^{\infty} h_k \, \hat{\mathbf{S}}^k \hat{\mathbf{x}} = \sum_{k=0}^{\infty} h_k \, (\mathbf{P}^\mathsf{T}\mathbf{S}\mathbf{P})^k \mathbf{P}^\mathsf{T}\mathbf{x} = \mathbf{P}^\mathsf{T}\left(\sum_{k=0}^{\infty} h_k \, \mathbf{S}^k \mathbf{x}\right) = \mathbf{P}^\mathsf{T}\mathbf{H}(\mathbf{S})\mathbf{x}$

- GNN $\Rightarrow$ Graph convolution + Pointwise nonlinearity $\Rightarrow$ Pointwise does not mix node values
  $\Rightarrow$ GNN retains permutation equivariance $\Rightarrow$ $\mathbf{\Phi}(\hat{\mathbf{x}}; \hat{\mathbf{S}}, \mathcal{H}) = \mathbf{P}^\mathsf{T}\mathbf{\Phi}(\mathbf{x}; \mathbf{S}, \mathcal{H})$
- Signal processing with graph neural networks is independent of labeling

- Invariance to node relabelings allows GNNs to exploit internal symmetries of graph signals
- Although different, signals on (a) and (b) are permutations of one other
  $\Rightarrow$ Permutation equivariance means that the GNN can learn to classify (b) from seeing (a)



- Permutation Equivariance is not a good idea in all problems $\Rightarrow$ Edge-Variant GNNs

Isufi, **Gama**, Ribeiro, "EdgeNets: Edge Varying Graph Neural Networks", arXiv:2001.07620, 2020

Equivariance to Permutations is a Property of Graph Filters

- Permutation equivariance is a property of graph convolutions inherited to GNNs
  - $\Rightarrow$ Exploits data structure (internal symmetries of the graph)
- Why choose GNNs over graph convolutions?
  - $\Rightarrow$ Q1: What is good about pointwise nonlinearities?
  - $\Rightarrow$ Q2: What is wrong with linear graph convolutions?

- Permutation equivariance is a property of graph convolutions inherited to GNNs
  - $\Rightarrow$ Exploits data structure (internal symmetries of the graph)
- Why choose GNNs over graph convolutions?
  - $\Rightarrow$ Q1: What is good about pointwise nonlinearities?
  - $\Rightarrow$ Q2: What is wrong with linear graph convolutions?
- A2: They can be unstable to perturbations of the graph if we push their discriminative power
- A1: They make GNNs stable to perturbations while retaining discriminability

Equivariance to Permutations is a Property of Graph Filters

- Permutation equivariance is a property of graph convolutions inherited to GNNs
  - ⇒ Exploits data structure (internal symmetries of the graph)
- Why choose GNNs over graph convolutions?
  - ⇒ Q1: What is good about pointwise nonlinearities?
  - ⇒ Q2: What is wrong with linear graph convolutions?
- A2: They can be unstable to perturbations of the graph if we push their discriminative power
- A1: They make GNNs stable to perturbations while retaining discriminability
- These questions can be answered with an analysis in the **spectral domain**

- Graph convolution is a polynomial on the shift operator $\Rightarrow \mathbf{y} = \sum_{k=0}^{\infty} h_k \mathbf{S}^k \mathbf{x}$

- Graph convolution is a polynomial on the shift operator $\Rightarrow \mathbf{y} = \sum_{k=0}^{\infty} h_k \mathbf{S}^k \mathbf{x}$
- Decompose operator as $\mathbf{S} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^{\mathsf{H}}$ to write the spectral representation of the graph convolution

$$\mathbf{V}^{\mathsf{H}}\mathbf{y} = \mathbf{V}^{\mathsf{H}} \sum_{k=0}^{\infty} h_k (\mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^{\mathsf{H}})^k \mathbf{x} \quad \Rightarrow \quad \tilde{\mathbf{y}} = \sum_{k=0}^{\infty} h_k \boldsymbol{\Lambda}^k \tilde{\mathbf{x}}$$

- where we have used the graph Fourier transform (GFT) definitions $\tilde{\mathbf{x}} = \mathbf{V}^{\mathsf{H}}\mathbf{x}$ and $\tilde{\mathbf{y}} = \mathbf{V}^{\mathsf{H}}\mathbf{y}$

- Graph convolution is a polynomial on the shift operator $\Rightarrow \mathbf{y} = \sum_{k=0}^{\infty} h_k \mathbf{S}^k \mathbf{x}$
- Decompose operator as $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{\mathsf{H}}$ to write the spectral representation of the graph convolution

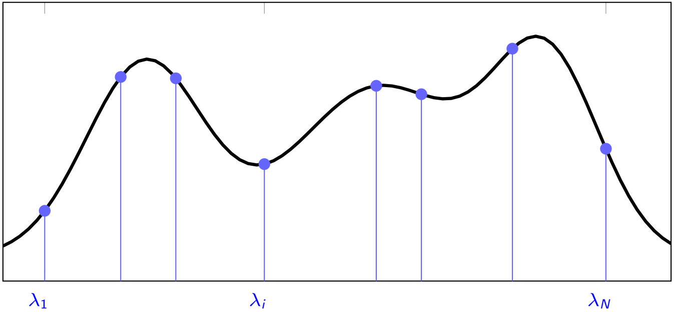
$$\mathbf{V}^{\mathsf{H}}\mathbf{y} \;=\; \mathbf{V}^{\mathsf{H}} \sum_{k=0}^{\infty} h_k (\mathbf{V}\mathbf{\Lambda}\mathbf{V}^{\mathsf{H}})^k \;\mathbf{x} \qquad \Rightarrow \qquad \tilde{\mathbf{y}} \;=\; \sum_{k=0}^{\infty} h_k \mathbf{\Lambda}^k \;\tilde{\mathbf{x}}$$

- where we have used the graph Fourier transform (GFT) definitions $\tilde{\mathbf{x}} = \mathbf{V}^{\mathsf{H}}\mathbf{x}$ and $\tilde{\mathbf{y}} = \mathbf{V}^{\mathsf{H}}\mathbf{y}$
- Graph convolution is a pointwise operation in the spectral domain

$$\tilde{y}_i \;=\; \tilde{h}(\lambda_i) \cdot \tilde{x}_i$$

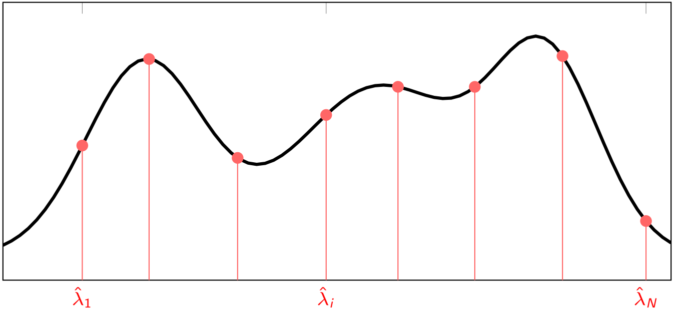
$\Rightarrow$ Determined by the (graph) frequency response $\Rightarrow \sum_{k=0}^{\infty} h_k \lambda_i^k = \tilde{h}(\lambda_i)$

- We can reinterpret the frequency response as a polynomial on continuous λ $\Rightarrow \tilde{h}(\lambda) = \sum_{k=0}^{\infty} h_k \lambda^k$



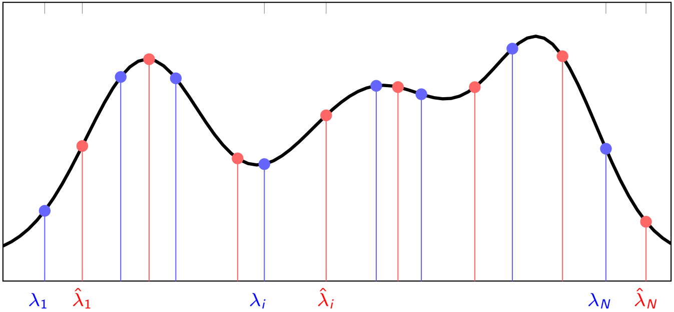
- Frequency response is the same no matter the graph $\Rightarrow$ It's instantiated on its particular spectrum

- We can reinterpret the frequency response as a polynomial on continuous λ $\Rightarrow \tilde{h}(\lambda) = \sum_{k=0}^{\infty} h_k \lambda^k$



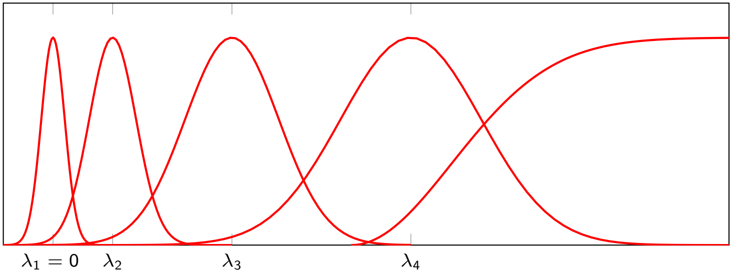
- Frequency response is the same no matter the graph $\Rightarrow$ It's instantiated on its particular spectrum

- We can reinterpret the frequency response as a polynomial on continuous λ $\Rightarrow \tilde{h}(\lambda) = \sum_{k=0}^{\infty} h_k \lambda^k$



- Frequency response is the same no matter the graph $\Rightarrow$ It's instantiated on its particular spectrum

- Let $h(\lambda)$ be the frequency response of filter $\mathbf{H}$. We say $\mathbf{H}$ is integral Lipschitz if $|\lambda h'(\lambda)| \leq C$



- Integral Lipschitz filters have to be wide for large λ $\Rightarrow$ They cannot discriminate
- But they can be thin for low λ $\Rightarrow$ They can discriminate. Arbitrarily discriminate

- Relative distance between $\mathbf{S}$ and $\hat{\mathbf{S}}$ $\Rightarrow$ Smallest matrix $\mathbf{E}$ that maps $\mathbf{S}$ into a permutation of $\hat{\mathbf{S}}$

$$\mathcal{E} = \left\{ \mathbf{E} : \mathbf{P}^{\mathsf{T}} \hat{\mathbf{S}} \mathbf{P} = \mathbf{S} + \mathbf{E}^{\mathsf{T}} \mathbf{S} + \mathbf{S} \mathbf{E} \right\} \quad \Rightarrow \quad d(\mathbf{S}, \hat{\mathbf{S}}) = \min_{\mathbf{E} \in \mathcal{E}} \| \mathbf{E} \| \leq \frac{\| \hat{\mathbf{S}} - \mathbf{S} \|}{\| \mathbf{S} \|}$$

- Relative distance between $\mathbf{S}$ and $\hat{\mathbf{S}}$ $\Rightarrow$ Smallest matrix $\mathbf{E}$ that maps $\mathbf{S}$ into a permutation of $\hat{\mathbf{S}}$

$$\mathcal{E} = \left\{ \mathbf{E} : \mathbf{P}^{\mathsf{T}} \hat{\mathbf{S}} \mathbf{P} = \mathbf{S} + \mathbf{E}^{\mathsf{T}} \mathbf{S} + \mathbf{S} \mathbf{E} \right\} \quad \Rightarrow \quad d(\mathbf{S}, \hat{\mathbf{S}}) = \min_{\mathbf{E} \in \mathcal{E}} \|\mathbf{E}\| \leq \frac{\|\hat{\mathbf{S}} - \mathbf{S}\|}{\|\mathbf{S}\|}$$

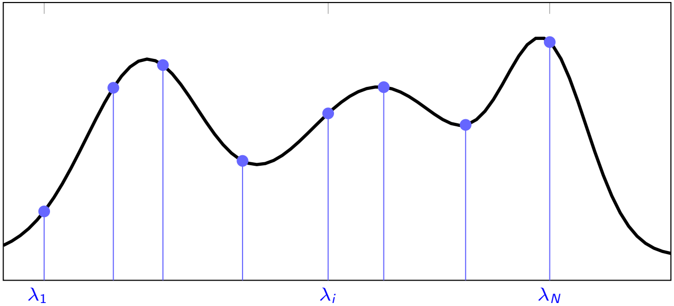
Theorem

Consider a GNN with L layers having integral Lipschitz filter $\mathbf{H}_\ell$ with constant C. Graphs $\mathbf{S}$ and $\hat{\mathbf{S}}$ satisfy $d(\mathbf{S}, \hat{\mathbf{S}}) \leq \epsilon/2$. The matrix $\mathbf{E}$ that achieves minimum distance satisfies $\|\mathbf{E}/\|\mathbf{E}\| - \mathbf{I}\| \leq \varepsilon$. It holds that for all signals $\mathbf{x}$

$$\min_{\mathbf{P} \in \mathcal{P}} \|\Phi(\mathbf{x}; \hat{\mathbf{S}}, \mathcal{H}) - \mathbf{P}^{\mathsf{T}} \Phi(\mathbf{x}; \mathbf{S}, \mathcal{H})\| \leq CL\, \varepsilon + \mathcal{O}(\varepsilon^2)$$

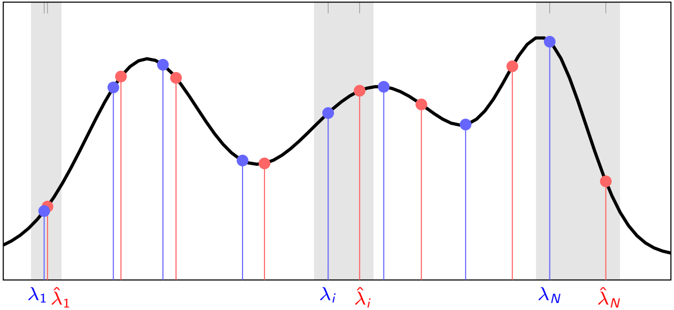
- GNNs can be made stable to graph perturbations if filters are integral Lipschitz

- The GNN stability theorem is elementary to prove for an edge dilation $\Rightarrow \hat{\mathbf{S}} = (1+\varepsilon)\mathbf{S}$
- An edge dilation just produces a spectrum dilation $\Rightarrow \hat{\lambda}_i = (1+\varepsilon)\lambda_i$, $\mathbf{E} = (\varepsilon/2)\mathbf{I}$



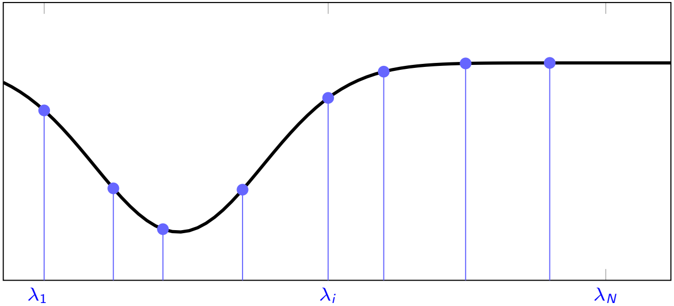
- Small deformations may result in large filter variations for large λ if filter is not integral Lipschitz

- The GNN stability theorem is elementary to prove for an edge dilation $\Rightarrow \hat{\mathbf{S}} = (1+\varepsilon)\mathbf{S}$
- An edge dilation just produces a spectrum dilation $\Rightarrow \hat{\lambda}_i = (1+\varepsilon)\lambda_i$, $\mathbf{E} = (\varepsilon/2)\mathbf{I}$



- Small deformations may result in large filter variations for large λ if filter is not integral Lipschitz

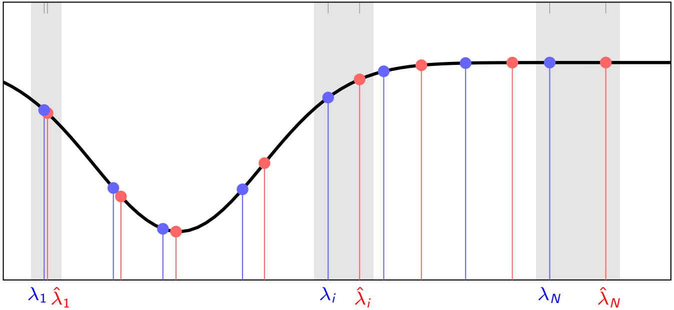
- The GNN stability theorem is elementary to prove for an edge dilation $\Rightarrow \hat{\mathbf{S}} = (1+\varepsilon)\mathbf{S}$
- An edge dilation just produces a spectrum dilation $\Rightarrow \hat{\lambda}_i = (1+\varepsilon)\lambda_i$, $\mathbf{E} = (\varepsilon/2)\mathbf{I}$



- Integral Lipschitz is always stable $\Rightarrow$ Eigenvalue does not move or filter does not move

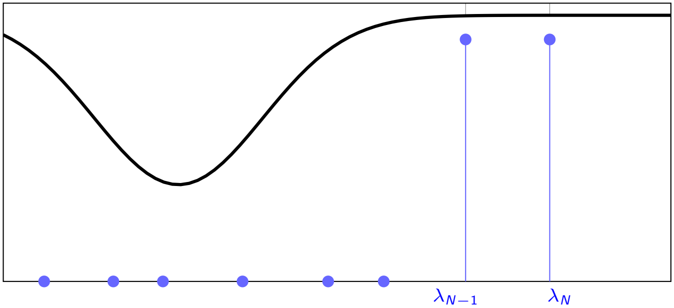
Proof of Stability Theorem

- The GNN stability theorem is elementary to prove for an edge dilation $\Rightarrow \hat{\mathbf{S}} = (1+\varepsilon)\mathbf{S}$
- An edge dilation just produces a spectrum dilation $\Rightarrow \hat{\lambda}_i = (1+\varepsilon)\lambda_i$, $\mathbf{E} = (\varepsilon/2)\mathbf{I}$



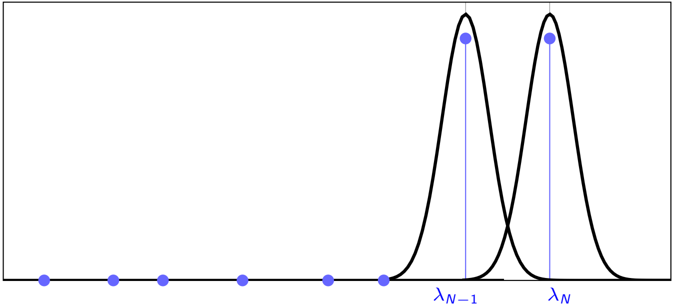
- Integral Lipschitz is always stable $\Rightarrow$ Eigenvalue does not move or filter does not move

- Q2: What is wrong with linear graph convolutions?
- Cannot be simultaneously stable to deformations and discriminate features at large eigenvalues



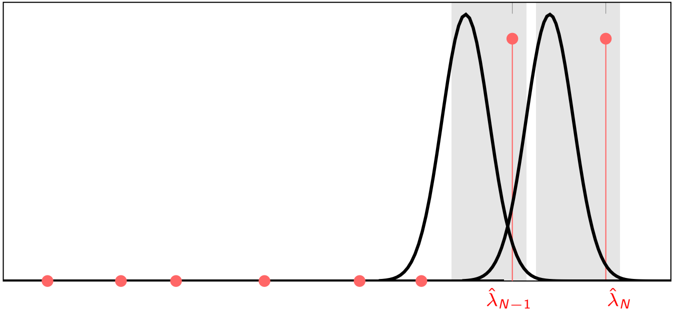
- Limits their value in machine learning problems where features at large eigenvalues are important

- Q2: What is wrong with linear graph convolutions?
- Cannot be simultaneously stable to deformations and discriminate features at large eigenvalues



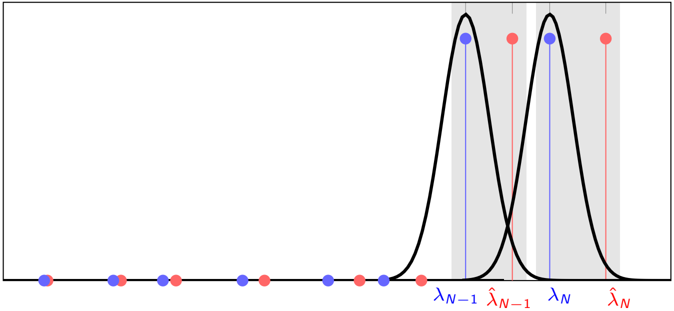
- Limits their value in machine learning problems where features at large eigenvalues are important

- Q2: What is wrong with linear graph convolutions?
- Cannot be simultaneously stable to deformations and discriminate features at large eigenvalues



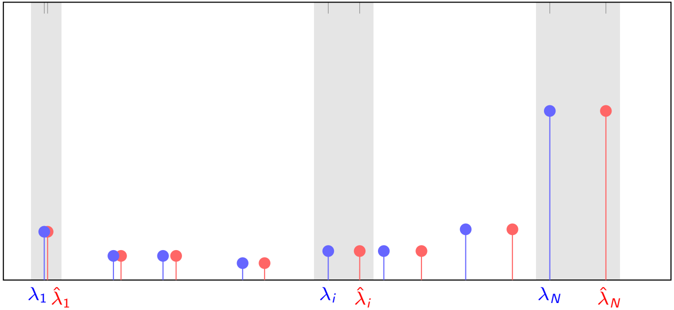
- Limits their value in machine learning problems where features at large eigenvalues are important

- Q2: What is wrong with linear graph convolutions?
- Cannot be simultaneously stable to deformations and discriminate features at large eigenvalues



- Limits their value in machine learning problems where features at large eigenvalues are important

- Q1: What is good about pointwise nonlinearities?
- Preserve permutation equivariance while generating low graph frequency components
  $\Rightarrow$ Which we can discriminate with stable filters

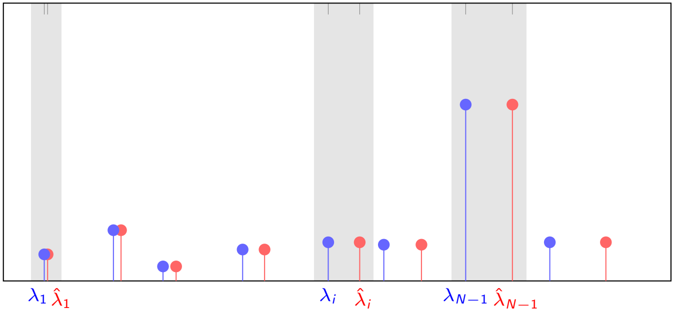


Spectrum of rectified graph signal

$$\mathbf{x}_{\text{relu}} = \max(\mathbf{x}, 0)$$

- The nonlinearity demodulates. It creates low frequency content that is stable

- Q1: What is good about pointwise nonlinearities?
- Preserve permutation equivariance while generating low graph frequency components
  $\Rightarrow$ Which we can discriminate with stable filters



Spectrum of rectified graph signal

$$\mathbf{x}_{\text{relu}} = \max(\mathbf{x}, 0)$$

- The nonlinearity demodulates. It creates low frequency content that is stable

Nonlinearities Create Low Frequency Components

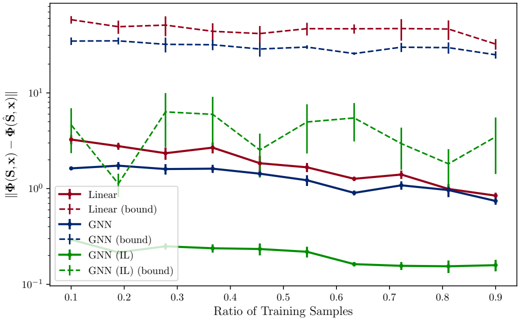
- Q1: What is good about pointwise nonlinearities?
- Preserve permutation equivariance while generating low graph frequency components
  $\Rightarrow$ Which we can discriminate with stable filters

GNNs are **stable** and **selective** information processing architectures

- The nonlinearity demodulates. It creates low frequency content that is stable

Example: Movie Recommendation Systems

- Movie recommendation problem $\Rightarrow$ Each node is a movie, each edge is the rating similarity
- Rating similarities estimated from training set $\Rightarrow$ Changing training set changes graph



- GNN trained with integral Lipschitz filters is more stable to graph estimation errors

Gama, Isufi, Leus, Ribeiro, "Graphs, Convolutions, and Neural Networks", arXiv:2003.03777, 2020

Gama, Tolstaya, Ribeiro, "Graph Neural Networks for Decentralized Controllers", arXiv:2003.10280, 2020

- Successful learning on graphs $\Rightarrow$ Scalability, exploit data structure, distributed implementation
- Graph neural networks (GNNs) $\Rightarrow$ Graph convolutions followed by pointwise nonlinearities
- GNNs are permutation equivariant and stable to changes in the graph $\Rightarrow$ Scale, transfer
- Graph convolutions are either stable or selective, but cannot be both
- Nonlinearities $\Rightarrow$ GNNs are both **stable** and **selective** information processing architectures
- Movie recommendation $\Rightarrow$ Stable to estimation errors in the rating similarity

Journal version:

Gama, Bruna, Ribeiro, "Stability Properties of Graph Neural Networks", arXiv:1905.04497, 2020.

Thank You!