Graph Neural Networks

Fernando Gama

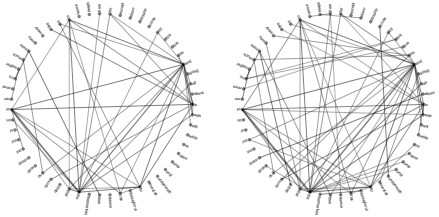
Department of Electrical Engineering and Computer Sciences, University of California, Berkeley

CosmoStat seminar on Machine Learning for Astrophysics
March 5th, 2021

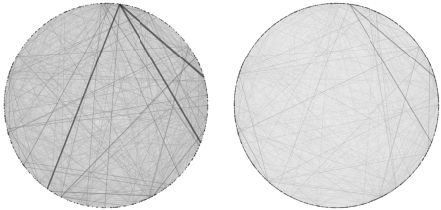
Thanks: D. Owerko, E. Tolstaya, L. Ruiz, Q. Li, T.-K. Hu, A. Prorok, A. G. Marques, Z. Wang, E. Isufi, G. Leus, J. Bruna, S. Sojoudi and A. Ribeiro

- Graphs are generic models of signal structure that can help to learn in several practical problems

Authorship Attribution



Recommendation Systems



Gama, Ribeiro, Bruna, "Diffusion Scattering Transforms on Graphs", ICLR 2019

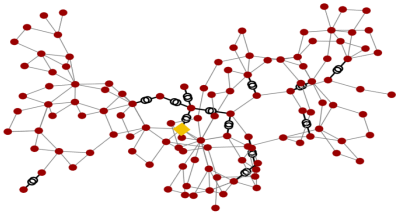
Gama, Bruna, Ribeiro, "Stability of Graph Neural Networks to Relative Perturbations", IEEE ICASSP 2020

Machine Learning on Graphs

- Graphs are generic models of signal structure that can help to learn in several practical problems

Decentralized Control of Autonomous Systems

Smart Grids



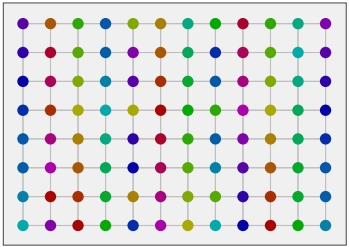
Tolstaya, **Gama**, Paulos, Pappas, Kumar, Ribeiro, "Learning Decentralized Controllers for Robot Swarms with Graph Neural Networks", CoRL 2019

Owerko, **Gama**, Ribeiro, "Optimal Power Flow Using Graph Neural Networks", IEEE ICASSP 2020

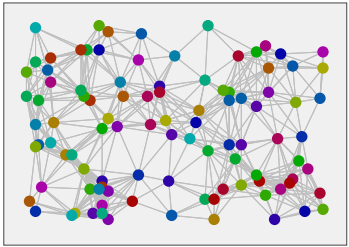
Machine Learning on Graphs

- Successful machine learning leverages structure $\Rightarrow$ Convolutional neural networks (CNNs)

We are good at learning over this



Challenge is we want to learn on this



- Scales, exploits data structure, and has an efficient implementation (distributed)
- Graph Convolutions $\Rightarrow$ Graph Signal Processing $\Rightarrow$ Graph Filtering

Machine Learning on Graphs

Graph Neural Networks

- Graph signal processing $\Rightarrow$ Mathematical framework
- Graph convolutions $\Rightarrow$ Local, distributed
  $\Rightarrow$ Generalize time convolutions

Equivariance and Stability

- Permutation equivariance $\Rightarrow$ Exploit structure
- Stability to changes in the underlying network
- Transferability and scalability

Distributed Collaborative Intelligent Systems

- Team of agents $\Rightarrow$ Collaborate to accomplish global task
- Autonomy of agents $\Rightarrow$ Decentralized actions
- Global objective vs. Local actions

Outline

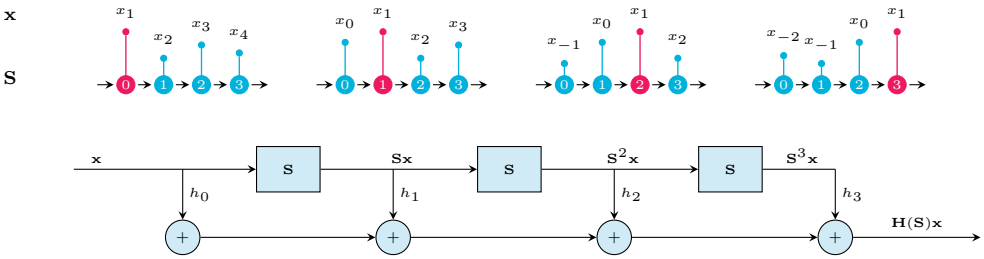
Graph Convolutions

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

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

- Notion of shift $\mathbf{S}$ $\Rightarrow$ Matrix description of graph $\Rightarrow$ $\mathbf{Sx}$ 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, Isufi, Leus, Ribeiro, "Graphs, Convolutions and Neural Networks: From Graph Filters to Graph Neural Networks", IEEE SPM, 2020

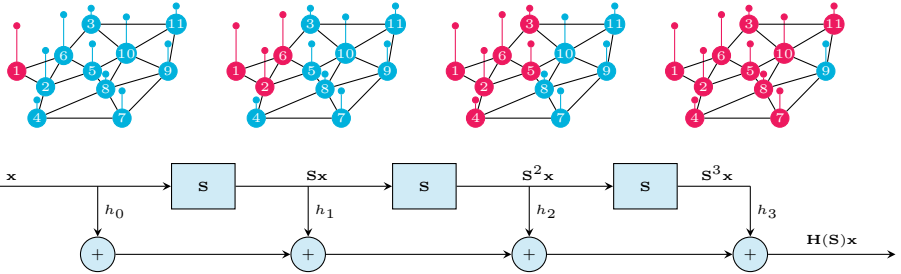
Graph Convolutions

- 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

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



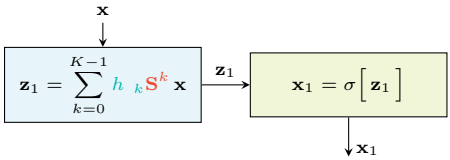
Gama, Isufi, Leus, Ribeiro, "Graphs, Convolutions and Neural Networks: From Graph Filters to Graph Neural Networks", IEEE SPM, 2020

Nonlinear Graph Signal Processing

- Traditional signal processing
  $\Rightarrow$ Best linear filter that exploits structure

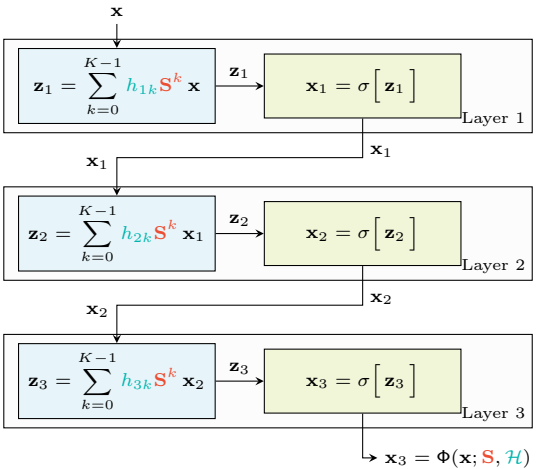
$$\min_{\{h_k\}} \mathsf{J}(\mathbf{z}_1) = \min_{\{h_k\}} \mathsf{J}(\mathbf{H}(\mathbf{S})\mathbf{x})$$

- Linear models $\Rightarrow$ Limited representation
  $\Rightarrow$ Nonlinear graph signal processing
- Graph perceptron $\Rightarrow$ Nonlinear processing
  $\Rightarrow$ Graph filter $\Rightarrow$ Pointwise nonlinearity
  $\Rightarrow$ *Learn* graph filter $\{h_k\}$ $\Rightarrow$ $\min_{\{h_k\}} J(\mathbf{x}_1)$
- Basic nonlinear description of models
  $\Rightarrow$ Increase representation power $\Rightarrow$ Repeat



Gama, Isufi, Leus, Ribeiro, "Graphs, Convolutions and Neural Networks: From Graph Filters to Graph Neural Networks", IEEE SPM, 2020

- 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

Graph Neural Networks: Why?

- Time convolutions are intuitive. Graph convolutions not so much.
  $\Rightarrow$ Local information, distributed implementation
- 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

Gama, Ribeiro, Bruna, "Diffusion Scattering Transforms on Graphs", ICLR 2019

Gama, Bruna, Ribeiro, "Stability of Graph Scattering Transforms", NeurIPS 2019

Permutation Equivariance

- 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 (Gama, Bruna, Ribeiro)

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

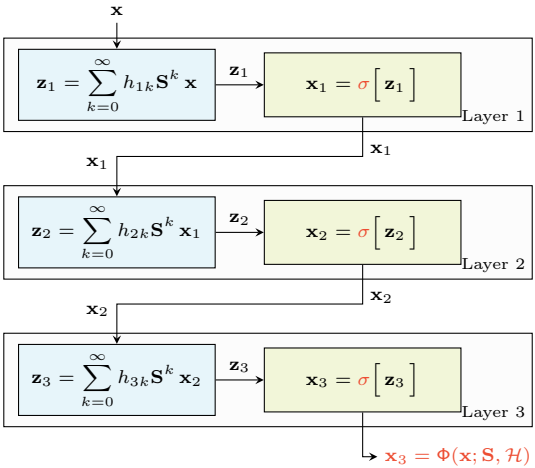
$$\mathbf{H}(\hat{\mathbf{S}})\hat{\mathbf{x}} = \mathbf{P}^{\mathsf{T}}\mathbf{H}(\mathbf{S})\mathbf{x}$$

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

Gama, Bruna, Ribeiro, "Stability of Graph Scattering Transforms", NeurIPS 2019

GNNs Inherit Permutation Equivariance from Graph Filters

- GNN is a cascade of layers
  $\Rightarrow$ Graph filters and pointwise nonlinearities
- Pointwise operation $\Rightarrow$ No mixing of node values
  $\Rightarrow$ Independent of the graph
- GNN retains permutation equivariance



Permutation Equivariance of Graph Neural Networks

Theorem (Gama, Ribeiro, Bruna)

GNNs 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

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

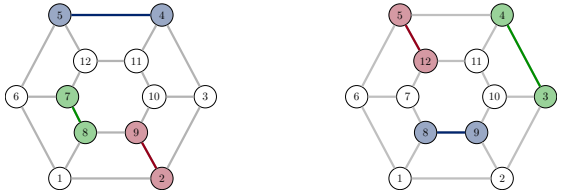
where $\mathbf{\Phi}(\hat{\mathbf{x}}; \hat{\mathbf{S}}, \mathcal{H})$ is the output of processing $\hat{\mathbf{x}}$ on $\hat{\mathbf{S}}$ with GNN $\mathcal{H}$ and $\mathbf{\Phi}(\mathbf{x}; \mathbf{S}, \mathcal{H})$ is the output of processing $\mathbf{x}$ on $\mathbf{S}$ with the same GNN $\mathcal{H}$.

- Signal Processing with Graph Neural Networks is independent of labeling

Gama, Bruna, Ribeiro, "Stability Properties of Graph Neural Networks", TSP, 2020

Equivariance to Permutations is More Valuable than Apparent

- 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 process (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?
- 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 Convolutions in the Frequency Domain

- 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

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

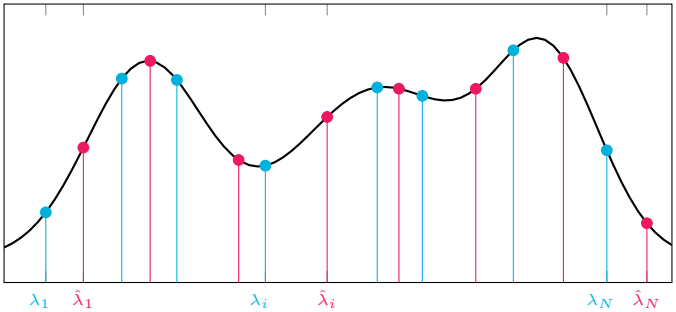
  where we have used the graph Fourier transform (GFT) definitions $\tilde{\mathbf{x}} = \mathbf{V}^{\mathsf{H}}\mathbf{x}$ and $\tilde{\mathbf{y}} = \mathbf{V}^{\mathsf{H}}\mathbf{y}$

- Graph convolution is a pointwise operation in the spectral domain $\Rightarrow \tilde{y}_i = \tilde{\mathsf{h}}(\lambda_i)\tilde{x}_i$

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

Graph Frequency Response

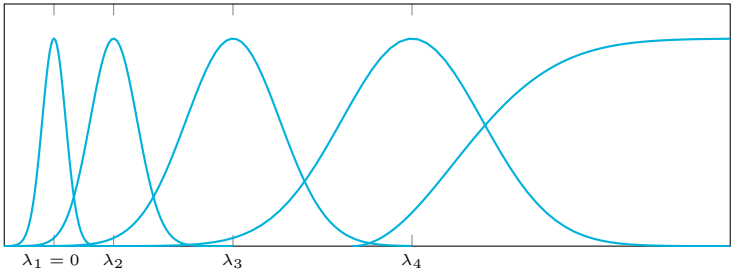
- We can reinterpret the frequency response as a polynomial on continuous λ $\Rightarrow \tilde{\mathsf{h}}(\lambda) = \sum_{k=0}^{\infty} h_k \lambda^k$



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

Restricting the Class of Allowable Filters

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



- Integral Lipschitz filters have to be wide for large λ; but they can be thin for low λ

Stability of Graph Neural Networks with Integral Lipschitz Filters

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

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

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 $d(\mathbf{S}, \hat{\mathbf{S}}) \leq \varepsilon/2$. The matrix $\mathbf{E}$ that achieves minimum distance satisfies $\|\mathbf{E}/\|\mathbf{E}\| - \mathbf{I}\| \leq \varepsilon$. It holds that for all signals $\mathbf{x}$

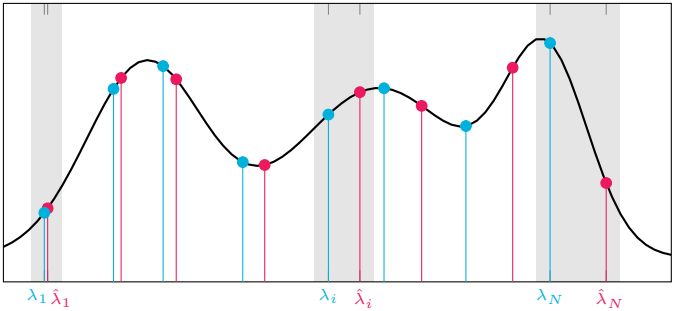
$$\min_{\mathbf{P} \in \mathcal{P}} \|\mathbf{\Phi}(\mathbf{x}; \hat{\mathbf{S}}, \mathcal{H}) - \mathbf{P}^{\mathsf{T}} \mathbf{\Phi}(\mathbf{x}; \mathbf{S}, \mathcal{H})\| \leq CL\, \varepsilon + \mathcal{O}(\varepsilon^2)$$

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

Gama, Bruna, Ribeiro, "Stability Properties of Graph Neural Networks", TSP, 2020

Insights: Edge Dilations

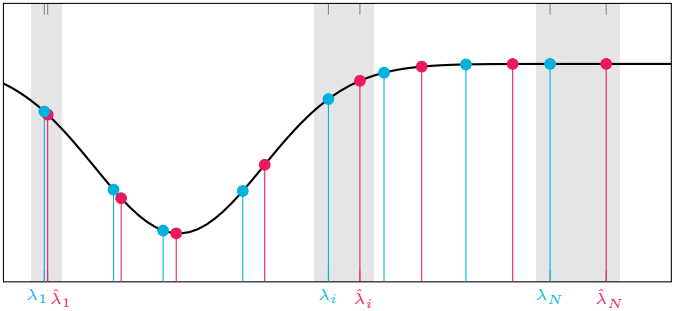
- Obtain valuable inisights $\Rightarrow$ Consider particular case of edge dilation $\Rightarrow \hat{\mathbf{S}} = (1+\varepsilon)\mathbf{S}$
- An edge dilation just produces a spectrum dilation $\Rightarrow \hat{\lambda}_i = (1+\varepsilon)\lambda_i$, $\mathbf{E} = (\varepsilon/2)\mathbf{I}$



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

Insights: Edge Dilations

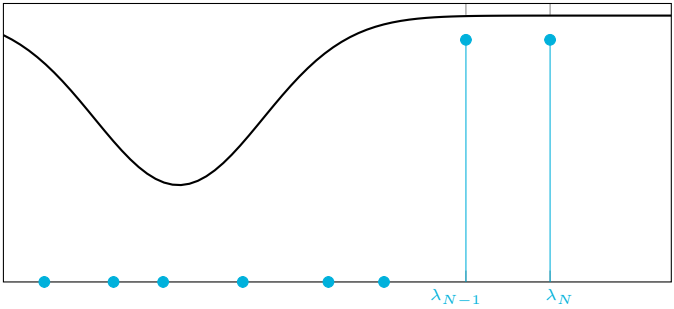
- Obtain valuable inisights $\Rightarrow$ Consider particular case of edge dilation $\Rightarrow \hat{\mathbf{S}} = (1+\varepsilon)\mathbf{S}$
- An edge dilation just produces a spectrum dilation $\Rightarrow \hat{\lambda}_i = (1+\varepsilon)\lambda_i$, $\mathbf{E} = (\varepsilon/2)\mathbf{I}$



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

Discriminative Graph Filter Banks are Unstable

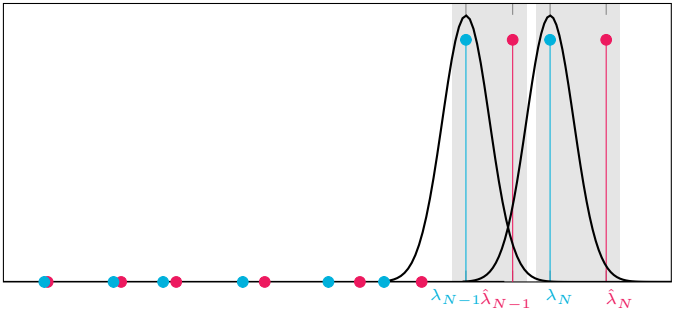
- Q2: What is wrong with linear graph convolutions?
- Cannot be simultaneously stable to deformations and discriminate features at large eigenvalues



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

Discriminative Graph Filter Banks are Unstable

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

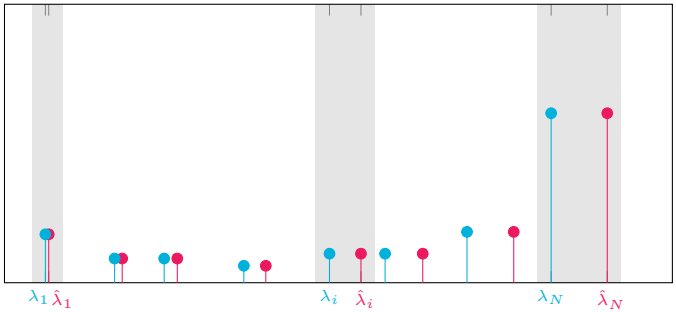


- Limits their value in machine learning problems where features at large eigenvalues are important

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

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

Distributed Coordination of a Robot Swarm

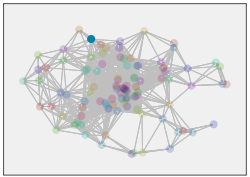
- We want the team to coordinate on their individual velocities without colliding with each other

- This is a very easy problem to solve if we allow for centralized coordination $\Rightarrow \mathbf{u}_i = \sum_{i=1}^{N} \mathbf{v}_i$
- But it is very difficult to solve if we do do not allow for centralized coordination $\Rightarrow \mathbf{u}_i = \ldots$

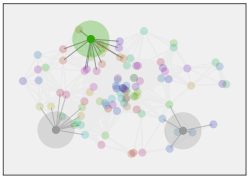
Tolstaya, Gama, Paulos, Pappas, Kumar, Ribeiro, "Learning Decentralized Controllers for Robot Swarms with Graph Neural Networks", CoRL 2019

Information Structure on Distributed Systems

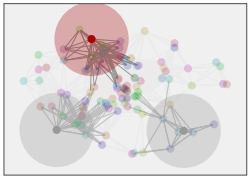
- The challenge in designing behaviors for distributed systems is the partial information structure



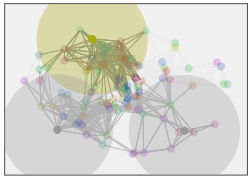
$\mathbf{x}_i(t)$



$\mathbf{x}_j(t-1)$ for $j \in \mathcal{N}_i^1$



$\mathbf{x}_j(t-2)$ for $j \in \mathcal{N}_i^2$



$\mathbf{x}_j(t-3)$ for $j \in \mathcal{N}_i^3$

- Node i has access to its own local information at time t $\Rightarrow \mathbf{x}_i(t)$
- And the information of its 1-hop neighbors at time $t-1$ $\Rightarrow \mathbf{x}_j(t-1)$ for all $j \in \mathcal{N}_i^1$
- And the information of its 2-hop neighbors at time $t-2$ $\Rightarrow \mathbf{x}_j(t-2)$ for all $j \in \mathcal{N}_i^2$
- And the information of its 3-hop neighbors at time $t-3$ $\Rightarrow \mathbf{x}_j(t-3)$ for all $j \in \mathcal{N}_i^3$

Tolstaya, Gama, Paulos, Pappas, Kumar, Ribeiro, "Learning Decentralized Controllers for Robot Swarms with Graph Neural Networks", CoRL 2019

Learning Distributed Actions

- Optimal centralized actions act directly on all states $\pi^\star(\mathbf{x}(t))$ and can be readily computed
- Distributed actions can only depend on information history $\Rightarrow \mathcal{X}_i(t) = \bigcup_{k=0}^{K-1} \left\{ \mathbf{x}_j(t-k) : j \in \mathcal{N}_i^k \right\}$
  - $\Rightarrow$ Optimal distributed actions $\mathbf{u}(t) = \boldsymbol{\Phi}^\star(\mathcal{X}_i(t); \mathcal{G})$ are famously difficult to find [Witsenhausen '68]
  - $\Rightarrow$ When optimal solutions are out of reach we resort to heuristics $\Rightarrow$ data driven heuristics

Gama, Li, Tolstaya, Prorok, Ribeiro, "Decentralized Control with Graph Neural Networks", arxiv.org/abs/2012.14906

Imitation Learning of Distributed Actions

- Parametrize $\Phi(\mathcal{X}_i(t); \mathcal{G})$ with a graph neural network $\Phi(\mathbf{x}(t); \mathbf{S}(t), \mathcal{H})$
  $\Rightarrow$ Adopt learning architecture that naturally processes distributed partial information $\mathcal{X}_i(t)$
- Train $\Phi(\mathbf{x}(t); \mathbf{S}(t), \mathcal{H})$ by using **imitation learning**
- Optimal centralized policy $\pi^\star(\mathbf{x}(t))$ can be computed during training time
- Find the parameters $\mathcal{H}$ that make $\Phi(\mathbf{x}(t); \mathbf{S}(t), \mathcal{H})$ closer to $\pi^\star(\mathbf{x}(t))$

$$\mathcal{H}^\star = \arg\min_{\mathcal{H}} \mathbb{E}_{\pi^\star} \left[\mathsf{J}\Big(\Phi(\mathbf{x}(t); \mathbf{S}(t), \mathcal{H}), \pi^\star(\mathbf{x}(t)) \Big) \right]$$

- Centralized policy required at train time but **not at test time**

Gama, Li, Tolstaya, Prorok, Ribeiro, "Decentralized Control with Graph Neural Networks", arxiv.org/abs/2012.14906

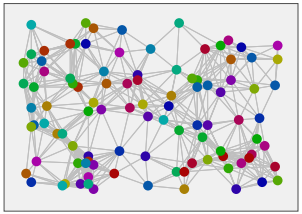
GNN Properties Play a Key Role

- GNN controllers $\mathbf{u}(t) = \boldsymbol{\Phi}(\mathbf{x}(t); \mathbf{S}(t), \mathcal{H})$ respect the partial information structure $\mathcal{X}_i(t)$
  $\Rightarrow$ Graph filters $\mathbf{H}(\mathcal{X}_i(t)) = \sum_{k=0}^{K-1} h_k \, \mathbf{S}(t) \cdots \mathbf{S}(t-(k-1))\mathbf{x}(t-k)$
- Permutation equivariance
  $\Rightarrow$ If two agents observe the same input
  $\Rightarrow$ Their k-hop neighbors observe the same inputs
  $\Rightarrow$ And the local neighborhood structures of the graph are the same
- Then the output of the control policy is the same at both nodes
- Stability $\Rightarrow$ If graphs are similar, GNN outputs will be similar
- These properties are a necessity for offline training

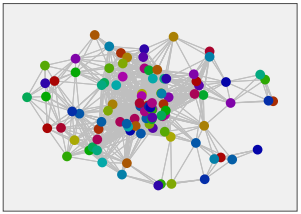
Gama, Li, Tolstaya, Prorok, Ribeiro, "Decentralized Control with Graph Neural Networks", arxiv.org/abs/2012.14906

Offline Training vs Online Execution

- If we want to train offline and execute online we can't assume the graph is the same
- Train offline on a graph like this



- And execute online on a graph like this



- Graph convolutions $\Rightarrow$ Learn parameters $\mathcal{H}$ $\Rightarrow$ Run on any $\mathbf{S}$: $\mathbf{H}(\mathbf{S}) = \sum_{k=0}^{K-1} h_k \, \mathbf{S}^k$
- Permutation equivariance $\Rightarrow$ Exploit symmetries, reuse data
- Stability $\Rightarrow$ Similar graphs yield similar outputs

Gama, Li, Tolstaya, Prorok, Ribeiro, "Decentralized Control with Graph Neural Networks", arxiv.org/abs/2012.14906

Team Dynamics

- Team of N agents with positions $\mathbf{r}_i(t)$, velocities $\mathbf{v}_i(t)$ and acceleration $\mathbf{u}_i(t)$
- System dynamics $\Rightarrow$ Constant acceleration during each interval T_s

$$\begin{cases} \mathbf{r}_i(t+1) = \mathbf{u}_i(t) T_s^2/2 + \mathbf{v}_i(t) T_s + \mathbf{r}_i(t) \\ \mathbf{v}_i(t+1) = \mathbf{u}_i(t) T_s + \mathbf{v}_i(t) \end{cases}$$

- Control acceleration $\mathbf{u}_i(t)$ $\Rightarrow$ Design sequence of acceleration values $\{\mathbf{u}_i(t)\}$
- Proof of concept $\Rightarrow$ More realistic scenarios are available [Tolstaya et al '19, Li et al '20, Hu et al '21]

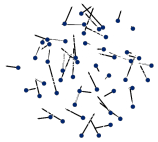
Tolstaya, Gama, Paulos, Pappas, Kumar, Ribeiro, "Learning Decentralized Controllers for Robot Swarms with GNNs", CoRL 2019.
Li, Gama, Ribeiro, Prorok, "Graph Neural Networks for Decentralized Multi-Robot Path Planning", IRoS 2020.
Hu, Gama, Wang, Ribeiro, Sadler "VGAI: A Vision-Based Decentralized Controller Learning Framework for Robot Swarms", ICASSP 2021
Gama, Li, Tolstaya, Prorok, Ribeiro, "Decentralized Control with Graph Neural Networks", arxiv.org/abs/2012.14906

Problem Statement

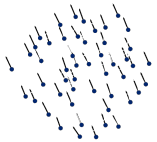
- Coordinate the velocities $\mathbf{v}_i(t)$ of all agents to be the same while avoiding collisions
- Measure the cost by computing the velocity variation across the team for the entire trajectory

$$\mathbf{J}\big[\{\mathbf{v}_i(t)\}\big] = \frac{1}{N}\sum_{t}\sum_{i=1}^{N}\|\mathbf{v}_i(t) - \bar{\mathbf{v}}_j(t)\|^2 \;,\; \bar{\mathbf{v}}_j = \frac{1}{N}\sum_{j=1}^{N}\mathbf{v}_j(t)$$

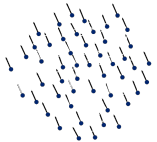
- Control accelerations $\mathbf{u}_i(t)$ such that $\min_{\mathbf{u}_i(t)\;,\;t\geq 0} \mathbf{J}\big[\{\mathbf{v}_i(t)\}\big] \;\Rightarrow$ Velocity $\mathbf{v}_i(t+1) = \mathbf{u}_i(t)T_s + \mathbf{v}_i(t)$



$t = 0\text{s}$



$t = 1\text{s}$



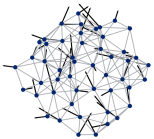
$t = 2\text{s}$

Gama, Li, Tolstaya, Prorok, Ribeiro, "Decentralized Control with Graph Neural Networks", arxiv.org/abs/2012.14906

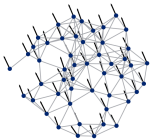
Communication Network

- Decentralized setting $\Rightarrow$ Agents can only communicate if $\|\mathbf{r}_i(t) - \mathbf{r}_j(t)\| \leq R$
- Communication graph $\mathbf{S}(t)$ $\Rightarrow$ Edge $[\mathbf{S}(t)]_{ij}$ if agents can communicate
- Use GNNs that respect the communicaton graph $\Rightarrow$ Local and distributed operations

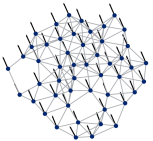
$$\mathbf{U}(\mathcal{X}_i(t), \mathcal{H}) = \Phi(\mathbf{X}(t); \mathbf{S}(t), \mathcal{H}) \ , \ \mathbf{H}(\mathcal{X}_i(t)) = \sum_{k=0}^{K-1} h_k \ \mathbf{S}(t) \cdots \mathbf{S}(t-(k-1))\mathbf{x}(t-k)$$



$t = 0$s



$t = 1$s



$t = 2$s

Gama, Li, Tolstaya, Prorok, Ribeiro, "Decentralized Control with Graph Neural Networks", arxiv.org/abs/2012.14906

Imitation Learning

- Train GNN $\boldsymbol{\Phi}(\mathbf{X}(t); \mathbf{S}(t), \mathcal{H})$ by using imitation learning $\Rightarrow$ Find $\mathcal{H}$ that make $\boldsymbol{\Phi}$ closer to $\mathbf{U}^{\star}(t)$

$$\mathcal{H}^{\star} = \arg\min_{\mathcal{H}} \mathbb{E}_{\mathbf{U}^{\star}} \left[\| \boldsymbol{\Phi}(\mathbf{X}(t); \mathbf{S}(t), \mathcal{H}) - \mathbf{U}^{\star}(t) \| \right]$$

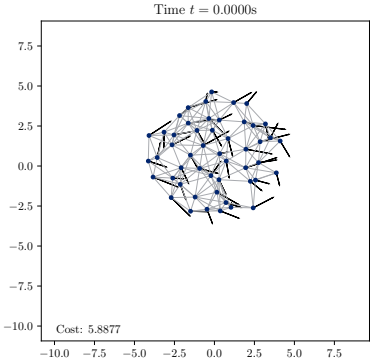
- The optimal centralized solution $\mathbf{U}^{\star}(t)$ that avoids collisions is given by

$$\mathbf{u}_i^{\star}(t) = - \sum_{j=1}^{N} \left(\mathbf{v}_i(t) - \mathbf{v}_j(t) \right) \underbrace{- \sum_{j=1}^{N} \nabla_{\mathbf{r}_i(t)} P(\mathbf{r}_i(t), \mathbf{r}_j(t))}_{\text{collision avoidance}}$$

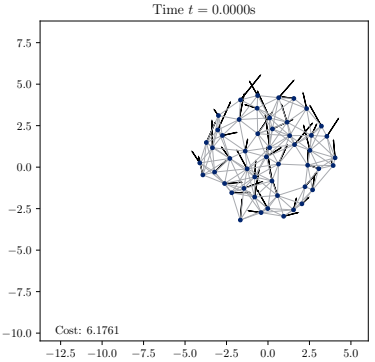
- The state $\mathbf{x}_i(t)$ for each agent is set to $\nabla_{\mathbf{r}_i(t)} P(\mathbf{r}_i(t), \mathbf{r}_j(t)) \Rightarrow$ Local to each agent
- Centralized controller $\mathbf{U}^{\star}$ is required at train time **but not** at testing time

Gama, Li, Tolstaya, Prorok, Ribeiro, "Decentralized Control with Graph Neural Networks", arxiv.org/abs/2012.14906

Flocking: Transferring



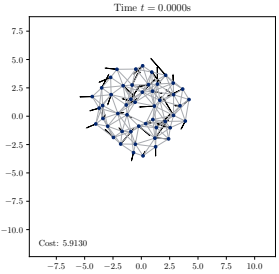
Offline optimal trajectory



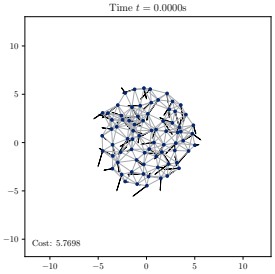
Online learned trajectory

Gama, Li, Tolstaya, Prorok, Ribeiro, "Decentralized Control with Graph Neural Networks", arxiv.org/abs/2012.14906

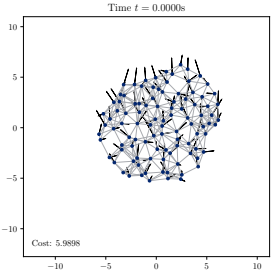
Flocking: Scalability



50 agents.



75 agents.



100 agents.

Gama, Li, Tolstaya, Prorok, Ribeiro, "Decentralized Control with Graph Neural Networks", arxiv.org/abs/2012.14906

Flocking: Transferring at Scale

- Train on 50 agents. Test on 100 agents.

Gama, Li, Tolstaya, Prorok, Ribeiro, "Decentralized Control with Graph Neural Networks", arxiv.org/abs/2012.14906

Flocking: Takeaways

- GNNs learn controllers to drive a team to fly at the same velocity while avoiding collisions
  $\Rightarrow$ Control actions taken by each agent based only on outdated neighboring information

- Transfer $\Rightarrow$ The communication network changes from time to time $\Rightarrow$ GNNs work
- Scale $\Rightarrow$ Learn to control teams of increasing number of agents $\Rightarrow$ GNNs work
- Transfer at scale $\Rightarrow$ Train on a small team, test on a big team $\Rightarrow$ GNNs work

Tolstaya, Gama, Paulos, Pappas, Kumar, Ribeiro, "Learning Decentralized Controllers for Robot Swarms with Graph Neural Networks", CoRL 2019

Gama, Li, Tolstaya, Prorok, Ribeiro, "Decentralized Control with Graph Neural Networks", arxiv.org/abs/2012.14906

- 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
- Learning decentralized controllers $\Rightarrow$ Distributed systems $\Rightarrow$ Partial information structure $\Rightarrow$ Parametrize controller with GNN $\Rightarrow$ Naturally adapts to partial information structure
- Flocking $\Rightarrow$ Coordinate a team of robots to fly at the same velocities $\Rightarrow$ Transfers at scale

Thank you!