Aggregation Graph Neural Networks

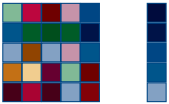
Fernando Gama, Antonio G. Marques,
Geert Leus & Alejandro Ribeiro

Dept. of Electrical and Systems Engineering
University of Pennsylvania

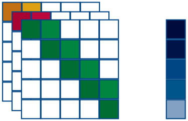
IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)
Brighton, UK — May 16, 2019

Graph Neural Networks

- Neural Networks $\Rightarrow$ Information processing architectures (models)
  - $\Rightarrow$ Linear transform followed by activation function
- Design linear transform to *fit* a training set $\Rightarrow$ Generalization
  - $\Rightarrow$ Minimize a cost function over the training set $\Rightarrow$ Learn
- Linear transforms depend on the size of data $\Rightarrow$ **Do not scale**
- Convolutional Neural Networks $\Rightarrow$ Regularize linear operation
  - $\Rightarrow$ Linear transform is now a bank of filters $\Rightarrow$ **Convolution**



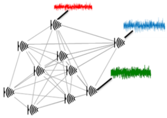
linear transform



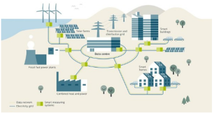
bank of filters

Graph Neural Networks

- Network data $\Rightarrow$ Data elements related by pairwise relationships
  $\Rightarrow$ Irregular structure $\Rightarrow$ Convolution does not work

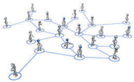


Wireless sensor networks



Power grids

Transportation network



Team of autonomous agents

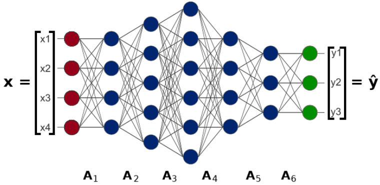
- **Aggregation graph neural networks**
  $\Rightarrow$ Exploit underlying graph topology
  $\Rightarrow$ Regularize linear transform $\Rightarrow$ Local architecture
  $\Rightarrow$ Tools from Graph Signal Processing (GSP) framework

Neural Networks (NNs)

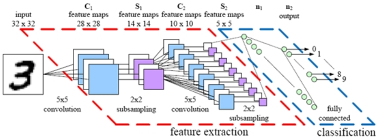
- Training set $\mathcal{T} = \{(\mathbf{x}, \mathbf{y})\}$ with input-output pairs $(\mathbf{x}, \mathbf{y})$
- Learning = Estimate output $\hat{\mathbf{y}}$ associated with input $\mathbf{x} \notin \mathcal{T}$
  - $\Rightarrow$ Adopt a neural network architecture to map between $\mathbf{x}$ and $\hat{\mathbf{y}}$
- Layer ℓ $\Rightarrow$ Linear transform followed by pointwise nonlinearity
  - $\Rightarrow$ Cascade L layers (input $\mathbf{x}_0 = \mathbf{x}$ and output $\hat{\mathbf{y}} = \mathbf{x}_L$)

$$\mathbf{x}_1 = \sigma_1\Big(\mathbf{A}_1\mathbf{x}\Big), \ \ldots, \ \mathbf{x}_\ell = \sigma_\ell\Big(\mathbf{A}_\ell\mathbf{x}_{\ell-1}\Big), \ \ldots, \ \mathbf{x}_L = \sigma_L\Big(\mathbf{A}_L\mathbf{x}_{L-1}\Big)$$

- Use $\mathcal{T}$ to find $\{\mathbf{A}_\ell\}$ that optimize loss function $\sum_{\mathcal{T}} \mathcal{L}(\mathbf{y}, \mathbf{x}_L)$



- Linear transform $\mathbf{A}_\ell$ $\Rightarrow$ Contains parameters to learn
  - $\Rightarrow$ Depends on the size of the input data (feature extraction)
  - $\Rightarrow$ Curse of dimensionality, large datasets, computationally costly, ...
- CNNs $\Rightarrow$ Regularize linear transform $\Rightarrow$ Small-support filters
  - $\Rightarrow$ Number of learnable parameters independent of size of data
  - $\Rightarrow$ Filtering $\Rightarrow$ Output computed by convolution (efficiently)
  - $\Rightarrow$ Exploit underlying regular structure of data
  - $\Rightarrow$ Pooling $\Rightarrow$ Local summaries $\Rightarrow$ Multi-resolution
- Structural information of data $\Rightarrow$ Constrain space of models



Network Data

- Relationship between data elements given by a network

  $\Rightarrow$ Modeled by a graph $\mathcal{G}$ with N nodes and edge set $\mathcal{E}$
- $[\mathbf{x}]_i$ =Data value stored at node i $\Rightarrow$ Graph signal $\mathbf{x} \in \mathbb{R}^N$
- Graph topology encoded in graph shift operator (GSO) $\mathbf{S} \in \mathbb{R}^{N \times N}$

$$[\mathbf{S}]_{ij} \neq 0 \quad \Longleftrightarrow \quad i = j \text{ or } (j, i) \in \mathcal{E}$$

- Linear operation $\mathbf{Sx}$ locally relates data with underlying network

$$[\mathbf{Sx}]_i = \sum_{j \in \mathcal{N}_i} [\mathbf{S}]_{ij} [\mathbf{x}]_j \qquad ([\mathbf{S}]_{ij} = 0 \text{ if } (j, i) \notin \mathcal{E})$$

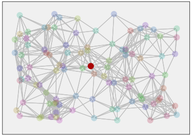
  $\Rightarrow$ Linear combination of signal values in the one-hop neighborhood

- Extend descriptive power of GSP $\Rightarrow$ Assign a vector to each node

  $\Rightarrow$ $\mathbf{x} : \mathcal{V} \to \mathbb{R}^F$ $\Rightarrow$ $\mathbf{x} = \{\mathbf{x}^f\}_{f=1}^F$, $\mathbf{x}^f$: graph signal for feature f

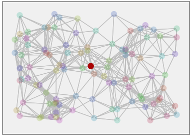
Aggregation Graph Neural Networks

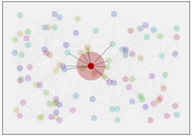
- Input signal defined over graph with N nodes $\Rightarrow$ Select a node
- Gather values from repeated exchanges with neighbors
- Resultant signal collected at the node has a regular structure
  $\Rightarrow$ Consecutive values encode nearby information in the graph
- Regular convolution linearly relates neighboring values
- Regular pooling constructs adequate neighborhood summaries
  $\Rightarrow$ Effective aggregation of information from local to global



Input 

$$\mathbf{z}_p = \Big[[\mathbf{x}_0^g]_p, [\mathbf{S}\mathbf{x}_0^g]_p, [\mathbf{S}^2\mathbf{x}_0^g]_p, [\mathbf{S}^3\mathbf{x}_0^g]_p, \ldots, [\mathbf{S}^{N-1}\mathbf{x}_0^g]_p \Big]$$

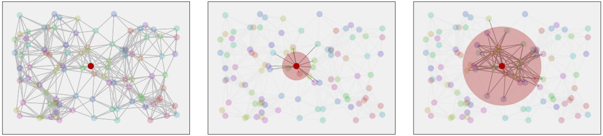




Input

$$\mathbf{z}_p = \Big[[\mathbf{x}_0^g]_p, [\mathbf{S}\mathbf{x}_0^g]_p, [\mathbf{S}^2\mathbf{x}_0^g]_p, [\mathbf{S}^3\mathbf{x}_0^g]_p, \ldots, [\mathbf{S}^{N-1}\mathbf{x}_0^g]_p \Big]$$

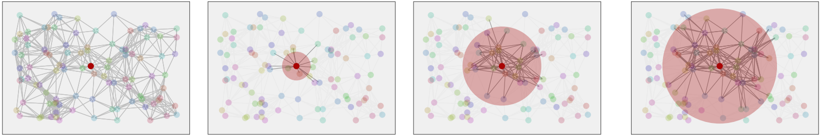
Aggregation GNN



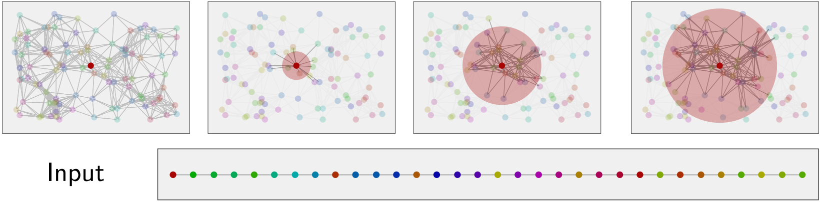
Input

$$\mathbf{z}_p = \Big[[\mathbf{x}_0^g]_p, [\mathbf{S}\mathbf{x}_0^g]_p, [\mathbf{S}^2\mathbf{x}_0^g]_p, [\mathbf{S}^3\mathbf{x}_0^g]_p, \ldots, [\mathbf{S}^{N-1}\mathbf{x}_0^g]_p \Big]$$

Aggregation GNN

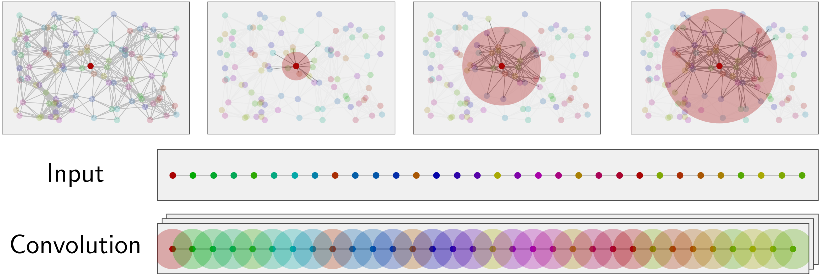


$$\mathbf{z}_p = \Big[[\mathbf{x}_0^g]_p, [\mathbf{S}\mathbf{x}_0^g]_p, [\mathbf{S}^2\mathbf{x}_0^g]_p, [\mathbf{S}^3\mathbf{x}_0^g]_p, \ldots, [\mathbf{S}^{N-1}\mathbf{x}_0^g]_p \Big]$$



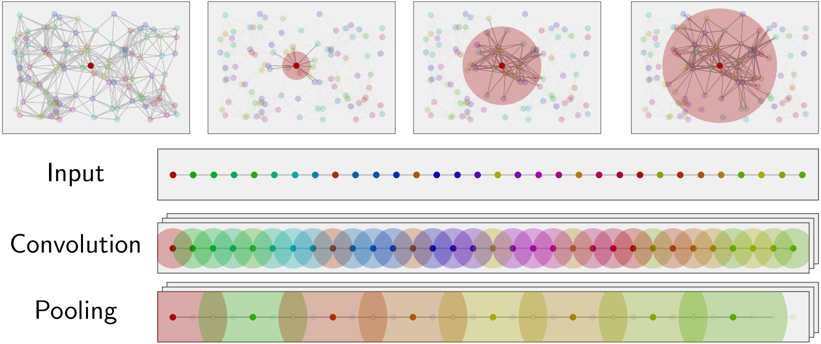
$$\mathbf{z}_p = \Big[[\mathbf{x}_0^g]_p, [\mathbf{S}\mathbf{x}_0^g]_p, [\mathbf{S}^2\mathbf{x}_0^g]_p, [\mathbf{S}^3\mathbf{x}_0^g]_p, \ldots, [\mathbf{S}^{N-1}\mathbf{x}_0^g]_p \Big]$$

Aggregation GNN



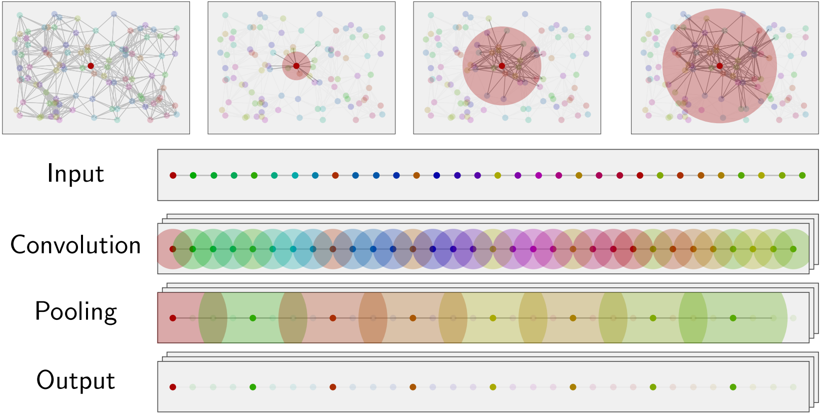
$$\left[\mathbf{u}_1^{fg}\right]_n = \left[\mathbf{h}_1^{fg} * \mathbf{z}_p\right]_n = \sum_{k=0}^{K_1-1} \left[\mathbf{h}_1^{fg}\right]_k \left[\mathbf{z}_p\right]_{n-k} = \sum_{k=0}^{K_1-1} \left[\mathbf{h}_1^{fg}\right]_k \left[\mathbf{S}^{n-k}\mathbf{x}_0^{g}\right]_p$$

Aggregation GNN



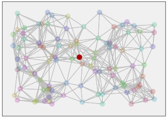
$$\left[\mathbf{v}_1^f\right]_n = \rho_1\left(\left[\mathbf{u}_1^f\right]_{\mathbf{n}_1}\right) = \varrho_1\left(\left[\mathbf{z}_p\right]_{n\in\mathbf{n}_1}\right) = \varrho_1\left(\left[\mathbf{S}^n\mathbf{x}_0^g\right]_p\right)_{n\in\mathbf{n}_1}$$

Aggregation GNN



$$\mathbf{z}_1^f = \sigma_1(\mathbf{C}_1 \mathbf{v}_1^f)$$

Regular Convolution



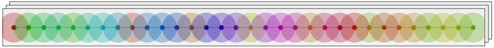
- Input $\mathbf{x}_0^g$ is a signal over known N-node graph
- Select node $p \in \mathcal{V}$ $\Rightarrow$ Perform local exchanges
- Consecutive elements encode nearby neighbors

$$\mathbf{z}_p = \Big[[\mathbf{x}_0^g]_p, [\mathbf{S}\mathbf{x}_0^g]_p, [\mathbf{S}^2\mathbf{x}_0^g]_p, \ldots, [\mathbf{S}^{N-1}\mathbf{x}_0^g]_p \Big]^\mathsf{T}$$

- Feature $\mathbf{u}_1^{fg}$ is obtained from regular convolution

$$\Big[\mathbf{u}_1^{fg}\Big]_n = \Big[\mathbf{h}_1^{fg} * \mathbf{z}_p\Big]_n = \sum_{k=0}^{K_1-1} \Big[\mathbf{h}_1^{fg}\Big]_k \Big[\mathbf{z}_p\Big]_{n-k} = \sum_{k=0}^{K_1-1} \Big[\mathbf{h}_1^{fg}\Big]_k \Big[\mathbf{S}^{n-k}\mathbf{x}_0^g\Big]_p$$

$\Rightarrow$ Effectively relates neighboring information encoded by the graph



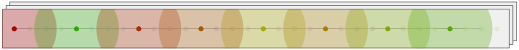
Regular Pooling

- Regular pooling $\Rightarrow \mathbf{n}_1 := \{\alpha_1$ consecutive elements of $\mathbf{u}_1^f\}$

$$\left[\mathbf{v}_1^f\right]_n = \rho_1\left(\left[\mathbf{u}_1^f\right]_{\mathbf{n}_1}\right) = \varrho_1\left(\left[\mathbf{z}_p\right]_{n\in\mathbf{n}_1}\right) = \varrho_1\left(\left[\mathbf{S}^n\mathbf{x}_0^g\right]_p\right)_{n\in\mathbf{n}_1}$$

$$= \varrho_1\left(\left[\mathbf{S}^{n+\alpha_1}\mathbf{x}_0^g\right]_p, \ldots, \left[\mathbf{S}^{n-K_1}\mathbf{x}_0^g\right]_p\right)$$

$\Rightarrow$ Summary for the $\alpha_1 + K_1$ neighborhood (of the original graph)



- Regular downsampling $\Rightarrow$ One every N_1 elements $\Rightarrow \mathbf{z}_1^f = \sigma_1(\mathbf{C}_1\mathbf{v}_1^f)$

$\Rightarrow [\mathbf{z}_1^f]_n \Rightarrow$ Summary from $[(n-1)N_1 + \alpha_1 + K_1]$ to $[nN_1 + \alpha_1 + K_1]$

Next Hidden Layers

- Input $\mathbf{z}_{\ell-1}^{g}$ to layer ℓ exhibits a regular structure
  - $\Rightarrow$ Element $[\mathbf{z}_{\ell-1}^{g}]_n$ represents a neighborhood summary
  - $\Rightarrow$ Consecutive elements contain nearby summaries
- Apply regular convolution $\Rightarrow$ Linearly relate nearby summaries

$$\left[\mathbf{u}_{\ell}^{fg}\right]_n = \left[\mathbf{h}_{\ell}^{fg} * \mathbf{z}_{\ell-1}^{g}\right]_n = \sum_{k=0}^{K_1-1} \left[\mathbf{h}_{1}^{fg}\right]_k \left[\mathbf{z}_{\ell-1}^{g}\right]_{n-k}$$

- Regular pooling $\Rightarrow$ $\mathbf{n}_\ell = \{\alpha_\ell$ consecutive elements of $\mathbf{u}_\ell^f\}$

$$\left[\mathbf{v}_{\ell}^{f}\right]_n = \rho_\ell\left(\left[\mathbf{u}_{\ell}^{f}\right]_{\mathbf{n}_\ell}\right) = \varrho_\ell\left(\left[\mathbf{z}_{\ell-1}^{g}\right]_{n\in\mathbf{n}_\ell}\right)$$

  - $\Rightarrow$ Summary of a larger neighborhood $\Rightarrow$ Change in resolution
- Regular downsampling $\Rightarrow$ Select one every N_ℓ consecutive elements

$$\mathbf{z}_{\ell}^{f} = \sigma_\ell\left(\mathbf{C}_\ell \mathbf{v}_\ell^f\right)$$

  - $\Rightarrow$ Reduce dimensionality $\Rightarrow$ Keep larger neighborhood summaries

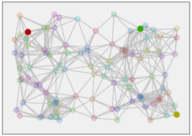
Aggregation GNN: Observations

- Entirely local architecture $\Rightarrow$ Only one node selected
  - $\Rightarrow$ Node gather all relevant information by local exchanges
  - $\Rightarrow$ The desired output is obtained at a single node
- Collected data has regular structure $\Rightarrow$ Traditional CNN
  - $\Rightarrow$ Existing results on CNNs can be used in the design
- Large networks might demand too many local exchanges
  - $\Rightarrow$ Long time to collect all relevant information

Multi-Node Aggregation GNN

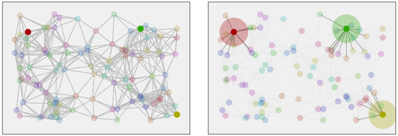
- Determine an initial subset of nodes (as opposed to only one)
  $\Rightarrow$ Aggregate local information (at those nodes) $\Rightarrow$ Few exchanges
- Regular structure $\Rightarrow$ Aggregation GNN stage (regular CNN)
  $\Rightarrow$ Obtain descriptive features of the aggregated neighborhood
- Features collected at a subset of nodes of original graph
  $\Rightarrow$ Disseminate information $\Rightarrow$ Zero-pad to fit the graph
- Select a smaller subset of nodes $\Rightarrow$ Aggregate local information
- Aggregation GNN stage $\Rightarrow$ Construct descriptive features
- Zero-pad, exchange, and so on...

Multi-Node Aggregation GNN



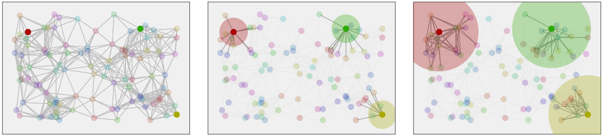
Input

Multi-Node Aggregation GNN



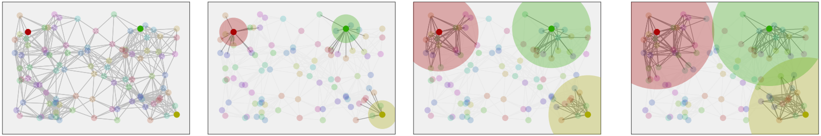
Input

Multi-Node Aggregation GNN



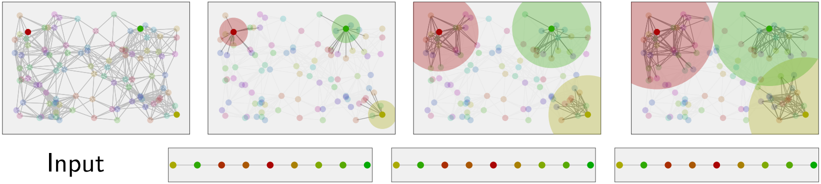
Input

Multi-Node Aggregation GNN

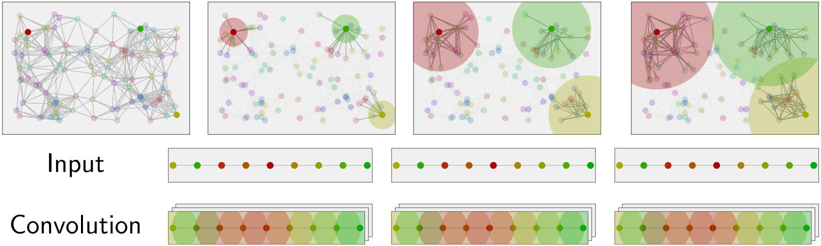


Input

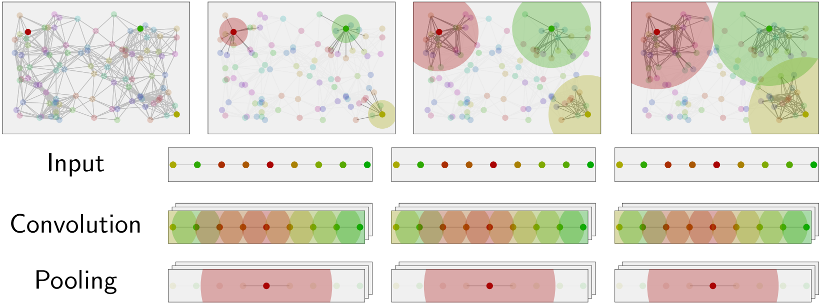
Multi-Node Aggregation GNN



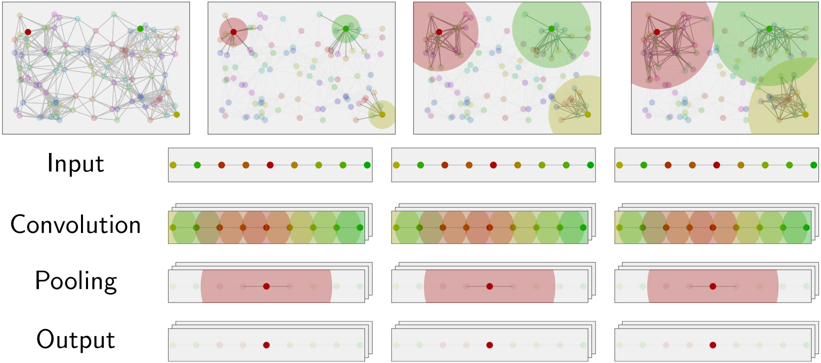
Multi-Node Aggregation GNN



Multi-Node Aggregation GNN

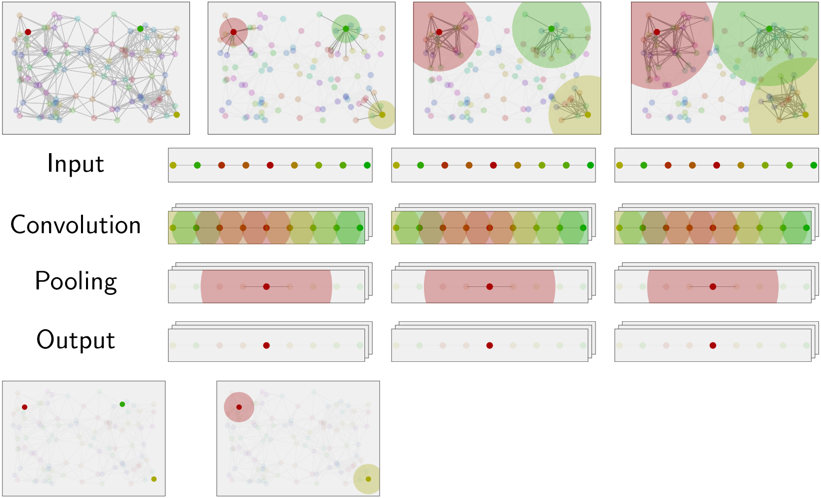


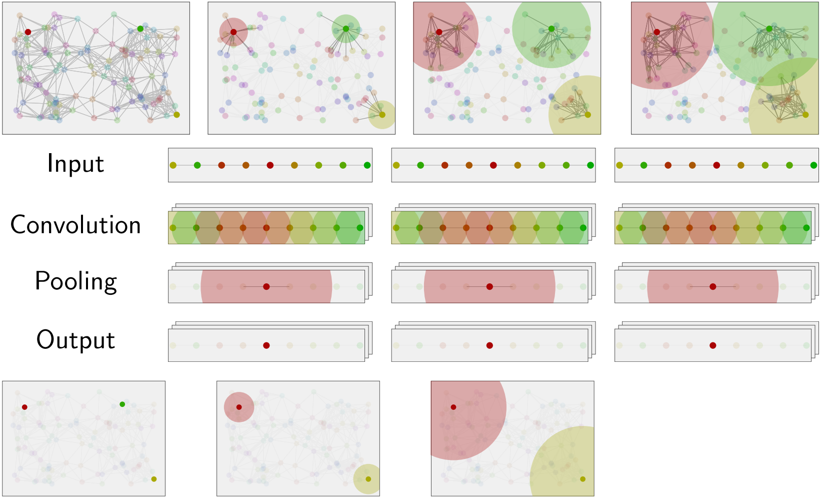
Multi-Node Aggregation GNN



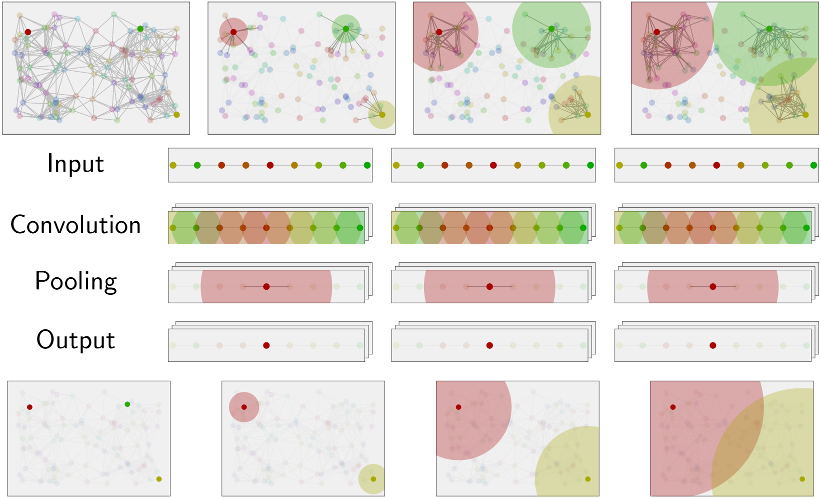
Multi-Node Aggregation GNN

Multi-Node Aggregation GNN




Input
Convolution
Pooling
Output

Multi-Node Aggregation GNN



Multi-Node Aggregation GNN: First Layer

- Consider data matrix $\mathbf{X}_0^g \in \mathbb{R}^{N \times N}$ obtained from input $\mathbf{x}_0^g$

$$\mathbf{X}_0^g = \left[\mathbf{S}^0\mathbf{x}_0^g, \mathbf{S}^1\mathbf{x}_0^g, \ldots, \mathbf{S}^{N-1}\mathbf{x}_0^g\right] = \begin{bmatrix} [\mathbf{S}^0\mathbf{x}_0^g]_1 & [\mathbf{S}^1\mathbf{x}_0^g]_1 & \cdots & [\mathbf{S}^{N-1}\mathbf{x}_0^g]_1 \\ [\mathbf{S}^0\mathbf{x}_0^g]_2 & [\mathbf{S}^1\mathbf{x}_0^g]_2 & \cdots & [\mathbf{S}^{N-1}\mathbf{x}_0^g]_2 \\ \vdots & \vdots & \ddots & \vdots \\ [\mathbf{S}^0\mathbf{x}_0^g]_N & [\mathbf{S}^1\mathbf{x}_0^g]_N & \cdots & [\mathbf{S}^{N-1}\mathbf{x}_0^g]_N \end{bmatrix}$$

- Select a subset $\mathcal{P}_1$ of nodes of the original graph ($\mathcal{P}_1$ row selection)
- Perform Q_1 exchanges of information (Q_1 column selection)

$$\mathbf{z}_1^g(0,p) = \left[[\mathbf{S}^0\mathbf{x}_0^g]_p, [\mathbf{S}^1\mathbf{x}_0^g]_p, \cdots, [\mathbf{S}^{Q_1-1}\mathbf{x}_0^g]_p\right], \ p \in \mathcal{P}_1$$

$\Rightarrow$ Each node gathers information up to the Q_1-hop neighborhood

- Data gathered at each node has regular structure

$\Rightarrow$ Aggregation GNN with L_1 layers at each node $\Rightarrow F_1$ features

Next layer: Dimension mismatch

- The output $\mathbf{z}_1(L_1, p) \in \mathbb{R}^{F_1}$ is obtained from Aggregation GNN
  - $\Rightarrow$ Defined only over the set $\mathcal{P}_1$ of nodes $\Rightarrow$ Not a graph signal
  - $\Rightarrow$ No GSO to keep exchanging information with neighbors
- Define the collection of feature f at each node

$$\mathbf{x}_1^f = \Big[[\mathbf{z}_1(L_1, p_1)]_f, \ldots, [\mathbf{z}_1(L_1, p_{|\mathcal{P}_1|})]_f\Big] \ , \ p_k \in \mathcal{P}_1$$

  - $\Rightarrow$ Zero-pad to obtain $\tilde{\mathbf{x}}_1^f = \mathcal{D}_1^{\mathsf{T}} \mathbf{x}_1^f$ that fits the original graph
- For outer layer r $\Rightarrow$ Select a subset $\mathcal{P}_r \subset \mathcal{P}_{r-1}$ to further collect data
- Perform Q_r exchanges with neighbors $\Rightarrow$ Regular structure data

$$\mathbf{z}_r^g(0, p) = \Big[[\tilde{\mathbf{x}}_{r-1}^g]_p, [\mathbf{S}\tilde{\mathbf{x}}_{r-1}^g]_p, \cdots, [\mathbf{S}^{Q_r-1}\tilde{\mathbf{x}}_{r-1}^g]_p\Big] \ , \ p \in \mathcal{P}_r$$

  - $\Rightarrow$ Q_r-hop nodes have information from their Q_{r-1} neighborhood
- Aggregation GNN to create F_r features $\Rightarrow$ $\mathbf{z}_r(L_r, p) \in \mathbb{R}^{F_r}$

Numerical Experiments: Source Localization

- Consider a stochastic block model (SBM) with $N = 100$ nodes
  $\Rightarrow$ $C = 5$ communities, 20 nodes each, $p_{c_i c_i} = 0.8$, $p_{c_i c_j} = 0.2$
- Assume node c started a diffusion at time $t = 0$
  $\Rightarrow$ Graph signal $\mathbf{e}_c$ has 1 in node c and zeros elsewhere
- Consider observations $\mathbf{x} = \mathbf{A}^t \mathbf{e}_c$ for some unknown $t > 0$
- Localize the community c that originated the diffusion

- Dataset: $8,000$ training, $2,000$ validation, 200 test
- 10 graph realizations, 10 dataset realizations for each graph
- ADAM optimizer: learning rate 0.001; 40 epochs, 100 batch size

- Degree, experimentally designed sampling (EDS) and spectral proxies (SP)

Source Localization: Results

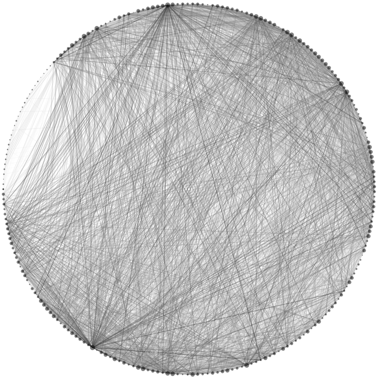
- (A): $L = 2$, $K^{(1)} = 4$, $K^{(2)} = 8$, $F^{(1)} = 16$, $F^{(2)} = 32$, half-pooling
- (MN): $K^{(1)} = K^{(2)} = 3$, $F^{(1)} = 16$, $F^{(2)} = 32$, $P^{(1)} = 10$, $P^{(2)} = 5$, $Q^{(1)} = 7$, $Q^{(2)} = 5$, half-pooling
- Clustering (C): $L = 2$, $F^{(1)} = F^{(2)} = 32$, $K^{(1)} = K^{(2)} = 5$

Architecture	Accuracy
Aggregation (A) Degree	94.2(±4.7)%
Aggregation (A) EDS	96.5(±3.1)%
Aggregation (A) SP	95.2(±4.4)%
Multinode (MN) Degree	96.1(±3.4)%
Multinode (MN) EDS	96.0(±3.5)%
Multinode (MN) SP	**97.3(±2.7)%**
Graph Coarsening (C) Clustering	87.4(±3.2)%

Numerical Experiments: Facebook network

- Same source localization problem $\Rightarrow$ Identify community
  - $\Rightarrow$ 234 Facebook network subgraph with 2 communities (McAuley '12)

- Dataset: 8,000 training, 2,000 validation, 200 test
- 10 random dataset realizations
- ADAM optimizer: learning rate 0.001; 80 epochs, 100 batch size
- Degree, experimentally designed sampling (EDS) and spectral proxies (SP)



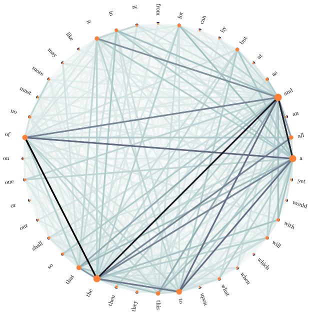
Facebook Network: Results

- (A): $L = 2$, $K^{(1)} = K^{(2)} = 4$, $F^{(1)} = 32$, $F^{(2)} = 64$, half-pooling
- (MN): $K^{(1)} = K^{(2)} = 3$, $F^{(1)} = 16$, $F^{(2)} = 32$, $P^{(1)} = 30$, $P^{(2)} = 10$, $Q^{(1)} = Q^{(2)} = 5$, half-pooling
- Clustering (C): $L = 2$, $F^{(1)} = F^{(2)} = 32$, $K^{(1)} = K^{(2)} = 5$

Architecture	Accuracy
Aggregation (A) Degree	95.8(±1.6)%
Aggregation (A) EDS	96.9(±1.2)%
Aggregation (A) SP	95.8(±1.4)%
Multinode (MN) Degree	97.6(±1.3)%
Multinode (MN) EDS	96.8(±1.2)%
Multinode (MN) SP	**99.0**(±**0.8**)%
Graph Coarsening (C) Clustering	95.2(±1.2)%

Numerical Experiments: Authorship Attribution

- Identify author of text excerpt
- Build word adjacency network
  - ⇒ From training excerpts
- Word frequency as graph signal
- 19th century authors
  - ⇒ Emily Brontë



- Dataset: 546 texts by Brontë to build WAN, 1000 words (nodes)
  - ⇒ 1,092 training texts excerpts, 272 testing text excerpts
- ADAM optimizer: learning rate 0.001; 40 epochs, 100 batch size

Authorship Attribution: Results

- (A): $L = 3$, $K^{(1)} = 6$, $K^{(2)} = K^{(3)} = 4$, $F^{(1)} = 32$, $F^{(2)} = 64$, $F^{(3)} = 128$, half-pooling
- (MN): $K^{(1)} = K^{(2)} = 3$, $F^{(1)} = 16$, $F^{(2)} = 32$, $P^{(1)} = 30$, $P^{(2)} = 10$, $Q^{(1)} = Q^{(2)} = 5$, half-pooling
- Clustering (C): $L = 2$, $F^{(1)} = F^{(2)} = 32$, $K^{(1)} = K^{(2)} = 5$

Architecture	Accuracy
Aggregation (A) Degree	69.5(±2.0)%
Aggregation (A) EDS	71.0(±2.8)%
Aggregation (A) SP	69.2(±4.0)%
Multinode (MN) Degree	80.4(±2.0)%
Multinode (MN) EDS	**80.5**(±**2.6**)%
Multinode (MN) SP	79.9(±2.8)%
Graph Coarsening (C) Clustering	65.2(±5.0)%

Conclusions & Other Extensions

- Regularize neural networks to exploit underlying graph topology
  - ⇒ Local architecture ⇒ Exchanges with neighboring nodes
- **Aggregation GNN:** collects data at one node ⇒ Regular structure
  - ⇒ Process regular data by using traditional CNNs
  - ⇒ **Multi-node GNN:** avoids the need of a large number of exchanges
- Tested on source localization and authorship attribution
- Journal: IEEE Trans. Signal Process., 67(10), 1034-1049, Feb. 2019.
- Other extensions in graph neural networks:
  - ⇒ Extend nonlinearities to include neighborhoods: `arXiv:1903.12575`, **today 6pm**, syndicate 1.
  - ⇒ Stability of GNNs under topology perturbations: `arXiv:1905.04497`
  - ⇒ Gated graph recurrent neural networks: `arXiv:1903.01888`
  - ⇒ Generalization through edge-varying recursions: `arXiv:1903.01298`
  - ⇒ Application to learning decentralized controllers: `arXiv:1903.10527`