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 ⇒ Information processing architectures (models)
  ⇒ Linear transform followed by activation function
- Design linear transform to *fit a training set* ⇒ Generalization
  ⇒ Minimize a cost function over the training set ⇒ Learn
- Linear transforms depend on the size of data ⇒ *Do not scale*

- Convolutional Neural Networks ⇒ Regularize linear operation
  ⇒ Linear transform is now a bank of filters ⇒ *Convolution*

linear transform

bank of filters
Graph Neural Networks

- **Network data** ⇒ Data elements related by **pairwise relationships**
  ⇒ Irregular structure ⇒ Convolution does not work

![Examples of graph data](wireless_sensor_networks.png, power grids.png, transportation_network.png, team_of_autonomous_agents.png)

- **Aggregation graph neural networks**
  ⇒ Exploit underlying graph topology
  ⇒ Regularize linear transform ⇒ Local architecture
  ⇒ Tools from Graph Signal Processing (GSP) framework
Neural Networks (NNs)

- Training set $\mathcal{T} = \{(x, y)\}$ with input-output pairs $(x, y)$
- Learning = Estimate output $\hat{y}$ associated with input $x \notin \mathcal{T}$
  $\Rightarrow$ Adopt a neural network architecture to map between $x$ and $\hat{y}$

- Layer $\ell$ $\Rightarrow$ Linear transform followed by pointwise nonlinearity
  $\Rightarrow$ Cascade $L$ layers (input $x_0 = x$ and output $\hat{y} = x_L$)
  
  $$x_1 = \sigma_1(A_1 x), \ldots, x_\ell = \sigma_\ell(A_\ell x_{\ell-1}), \ldots, x_L = \sigma_L(A_L x_{L-1})$$

- Use $\mathcal{T}$ to find $\{A_\ell\}$ that optimize loss function $\sum_{\mathcal{T}} \mathcal{L}(y, x_L)$

![Diagram of neural network]
Convolutional Neural Networks (CNNs)

- Linear transform $A_\ell$ ⇒ Contains parameters to learn
  ⇒ Depends on the size of the input data (feature extraction)
  ⇒ Curse of dimensionality, large datasets, computationally costly, ...

- CNNs ⇒ Regularize linear transform ⇒ Small-support filters
  ⇒ Number of learnable parameters independent of size of data
  ⇒ Filtering ⇒ Output computed by convolution (efficiently)
  ⇒ Exploit underlying regular structure of data
  ⇒ Pooling ⇒ Local summaries ⇒ Multi-resolution

- Structural information of data ⇒ Constrain space of models
Network Data

- Relationship between data elements given by a network
  ⇒ Modeled by a graph \( G \) with \( N \) nodes and edge set \( \mathcal{E} \)
- \([x]_i\) = Data value stored at node \( i \) ⇒ Graph signal \( x \in \mathbb{R}^N \)
- Graph topology encoded in graph shift operator (GSO) \( S \in \mathbb{R}^{N \times N} \)
  \[ [S]_{ij} \neq 0 \iff i = j \text{ or } (j, i) \in \mathcal{E} \]
- Linear operation \( Sx \) locally relates data with underlying network
  \[ [Sx]_i = \sum_{j \in \mathcal{N}_i} [S]_{ij} [x]_j \quad (\text{if } [S]_{ij} = 0 \text{ if } (j, i) \notin \mathcal{E}) \]
  ⇒ Linear combination of signal values in the one-hop neighborhood
- Extend descriptive power of GSP ⇒ Assign a vector to each node
  ⇒ \( x : \mathcal{V} \rightarrow \mathbb{R}^F \) ⇒ \( x = \{x^f\}_{f=1}^F \), \( x^f \): graph signal for feature \( f \)
Input signal defined over graph with $N$ nodes ⇒ Select a node

Gather values from repeated exchanges with neighbors

Resultant signal collected at the node has a regular structure
⇒ Consecutive values encode nearby information in the graph

Regular convolution linearly relates neighboring values

Regular pooling constructs adequate neighborhood summaries
⇒ Effective aggregation of information from local to global
Input

\[ z_p = \left[ x_0^g \right]_p, [Sx_0^g]_p, [S^2x_0^g]_p, [S^3x_0^g]_p, \ldots, [S^{N-1}x_0^g]_p \]
\[ z_p = \left[ [x_0^g]_p, [Sx_0^g]_p, [S^2x_0^g]_p, [S^3x_0^g]_p, \ldots, [S^{N-1}x_0^g]_p \right] \]
\[ z_p = \left[ x_0^g \right]_p, [Sx_0^g]_p, [S^2x_0^g]_p, [S^3x_0^g]_p, \ldots, [S^{N-1}x_0^g]_p \]
\[ z_p = \left[ x_0^g \right]_p, \left[ S x_0^g \right]_p, \left[ S^2 x_0^g \right]_p, \left[ S^3 x_0^g \right]_p, \ldots, \left[ S^{N-1} x_0^g \right]_p \]
$z_p = \left[ x_0^g p, Sx_0^g p, S^2 x_0^g p, S^3 x_0^g p, \ldots, S^{N-1} x_0^g p \right]$
\[
\begin{align*}
\mathbf{u}_1^{fg} &= \mathbf{h}_1^{fg} \ast \mathbf{z}_p \\
&= \sum_{k=0}^{K_1-1} \mathbf{h}_1^{fg} \mathbf{z}_p \\
&= \sum_{k=0}^{K_1-1} \mathbf{h}_1^{fg} \mathbf{S}^{n-k} \mathbf{x}_0
\end{align*}
\]
Aggregation GNN

\[
\begin{align*}
\mathbf{v}_1^n &= \rho_1 \left( \mathbf{u}_1^n \right) = \rho_1 \left( \mathbf{z}_p \right) = \rho_1 \left( \mathbf{S}_n \mathbf{x}_0^g \right) \\
\end{align*}
\]
Aggregation GNN

Input

Convolution

Pooling

Output

\[ z_1^f = \sigma_1(C_1v_1^f) \]
Input $x^g_0$ is a signal over known $N$-node graph

Select node $p \in \mathcal{V} \Rightarrow$ Perform local exchanges

Consecutive elements encode nearby neighbors

$$z_p = \left[ [x^g_0]_p, [Sx^g_0]_p, [S^2x^g_0]_p, \ldots, [S^{N-1}x^g_0]_p \right]^T$$

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

$$\left[ u^{fg}_1 \right]_n = \left[ h^{fg}_1 \ast z_p \right]_n = \sum_{k=0}^{K_1-1} \left[ h^{fg}_1 \right]_k \left[ z_p \right]_{n-k} = \sum_{k=0}^{K_1-1} \left[ h^{fg}_1 \right]_k \left[ S^{n-k}x^g_0 \right]_p$$

⇒ Effectively relates neighboring information encoded by the graph
**Regular Pooling**

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

\[
\left[ \mathbf{v}_1^f \right]_n = \rho_1 \left( \left[ \mathbf{u}_1^f \right]_{\mathbf{n}_1} \right) = \rho_1 \left( \left[ \mathbf{z}_p \right]_{n \in \mathbf{n}_1} \right) = \rho_1 \left( \left[ \mathbf{S}^n \mathbf{x}_0^g \right]_p \right)_{n \in \mathbf{n}_1} \\
= \rho_1 \left( \left[ \mathbf{S}^{n+\alpha_1} \mathbf{x}_0^g \right]_p, \ldots, \left[ \mathbf{S}^{n-K_1} \mathbf{x}_0^g \right]_p \right)
\]

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

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

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

- Input $\mathbf{z}^g_{\ell-1}$ to layer $\ell$ exhibits a regular structure
  - Element $[\mathbf{z}^g_{\ell-1}]_n$ represents a neighborhood summary
  - Consecutive elements contain nearby summaries

- Apply regular convolution  $\Rightarrow$ Linearly relate nearby summaries

$$
\begin{align*}
\mathbf{u}^{fg}_\ell &= \mathbf{h}^{fg}_\ell \ast \mathbf{z}^g_{\ell-1} = \sum_{k=0}^{K_1-1} \left[ h^{fg}_1 \right]_k \left[ \mathbf{z}^g_{\ell-1} \right]_{n-k}
\end{align*}
$$

- Regular pooling  $\Rightarrow$ $n_\ell = \{ \alpha_\ell \text{ consecutive elements of } \mathbf{u}^f_\ell \}$

$$
\begin{align*}
\mathbf{v}^f_\ell &= \rho_\ell \left( \left[ \mathbf{u}^f_\ell \right]_{n_\ell} \right) = \rho_\ell \left( \left[ \mathbf{z}^g_{\ell-1} \right]_{n \in n_\ell} \right)
\end{align*}
$$

  $\Rightarrow$ Summary of a larger neighborhood  $\Rightarrow$ Change in resolution

- Regular downsampling  $\Rightarrow$ Select one every $N_\ell$ consecutive elements

$$
\begin{align*}
\mathbf{z}^f_\ell &= \sigma_\ell \left( \mathbf{C}_\ell \mathbf{v}^f_\ell \right)
\end{align*}
$$

  $\Rightarrow$ Reduce dimensionality  $\Rightarrow$ Keep larger neighborhood summaries
Aggregation GNN: Observations

- **Entirely local architecture** ⇒ Only one node selected
  ⇒ Node gather all relevant information by local exchanges
  ⇒ The desired output is obtained at a single node

- **Collected data has regular structure** ⇒ Traditional CNN
  ⇒ Existing results on CNNs can be used in the design

- **Large networks might demand too many local exchanges**
  ⇒ Long time to collect all relevant information
Multi-Node Aggregation GNN

- Determine an initial **subset of nodes** (as opposed to only one)
  - Aggregate local information (at those nodes) ⇒ Few exchanges

- Regular structure ⇒ **Aggregation GNN stage** (regular CNN)
  - Obtain descriptive features of the aggregated neighborhood

- Features collected at a subset of nodes of original graph
  - Disseminate information ⇒ **Zero-pad** to fit the graph

- Select a smaller subset of nodes ⇒ Aggregate local information

- Aggregation GNN stage ⇒ Construct descriptive features

- Zero-pad, exchange, and so on...
Multi-Node Aggregation GNN

Input
Multi-Node Aggregation GNN

Input

![Graph 1](image1)

![Graph 2](image2)
Multi-Node Aggregation GNN

Input
Multi-Node Aggregation GNN

Input

Convolution

Pooling
Multi-Node Aggregation GNN

Input
Convolution
Pooling
Output
Multi-Node Aggregation GNN

Input

Convolution

Pooling

Output
Multi-Node Aggregation GNN

Input

Convolution

Pooling

Output
Multi-Node Aggregation GNN

Input
Convolution
Pooling
Output
Multi-Node Aggregation GNN: First Layer

- Consider data matrix $X^g_0 \in \mathbb{R}^{N \times N}$ obtained from input $x^g_0$

  $X^g_0 = [S^0 x^g_0, S^1 x^g_0, \ldots, S^{N-1} x^g_0] = \begin{bmatrix} [S^0 x^g_0]^1_1 & [S^1 x^g_0]^1_1 & \cdots & [S^{N-1} x^g_0]^1_1 \\ [S^0 x^g_0]^2_2 & [S^1 x^g_0]^2_2 & \cdots & [S^{N-1} x^g_0]^2_2 \\ \vdots & \vdots & \ddots & \vdots \\ [S^0 x^g_0]^N_N & [S^1 x^g_0]^N_N & \cdots & [S^{N-1} x^g_0]^N_N \end{bmatrix}$

- 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)

  $z^g_1(0, p) = \begin{bmatrix} [S^0 x^g_0]^p_1 \\ [S^1 x^g_0]^p_1 \\ \vdots \\ [S^{Q_1-1} x^g_0]^p_1 \end{bmatrix}, \ p \in \mathcal{P}_1$

  ⇒ Each node gathers information up to the $Q_1$-hop neighborhood
  ⇒ Data gathered at each node has regular structure
  ⇒ Aggregation GNN with $L_1$ layers at each node ⇒ $F_1$ features
The output $z_1(L_1, p) \in \mathbb{R}^{F_1}$ is obtained from Aggregation GNN

- Defined only over the set $\mathcal{P}_1$ of nodes
- Not a graph signal
- No GSO to keep exchanging information with neighbors

Define the collection of feature $f$ at each node

$$x_1^f = \left[ [z_1(L_1, p_1)]_f, \ldots, [z_1(L_1, p_{|\mathcal{P}_1|})]_f \right], \ p_k \in \mathcal{P}_1$$

- Zero-pad to obtain $\tilde{x}_1^f = D_1^T x_1^f$ that fits the original graph

For outer layer $r$

- Select a subset $\mathcal{P}_r \subset \mathcal{P}_{r-1}$ to further collect data
- Perform $Q_r$ exchanges with neighbors

Perform $Q_r$ exchanges with neighbors

$$z_r^g(0, p) = \left[ [\tilde{x}_{r-1}^g]_p, [S\tilde{x}_{r-1}^g]_p, \ldots, [S^{Q_r-1}\tilde{x}_{r-1}^g]_p \right], \ p \in \mathcal{P}_r$$

- $Q_r$-hop nodes have information from their $Q_{r-1}$ neighborhood

Aggregation GNN to create $F_r$ features

$$z_r(L_r, p) \in \mathbb{R}^{F_r}$$
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 $e_c$ has 1 in node $c$ and zeros elsewhere
- Consider observations $x = A^t e_c$ 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

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

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregation (A) Degree</td>
<td>94.2(±4.7)%</td>
</tr>
<tr>
<td>Aggregation (A) EDS</td>
<td>96.5(±3.1)%</td>
</tr>
<tr>
<td>Aggregation (A) SP</td>
<td>95.2(±4.4)%</td>
</tr>
<tr>
<td>Multinode (MN) Degree</td>
<td>96.1(±3.4)%</td>
</tr>
<tr>
<td>Multinode (MN) EDS</td>
<td>96.0(±3.5)%</td>
</tr>
<tr>
<td><strong>Multinode (MN) SP</strong></td>
<td><strong>97.3(±2.7)%</strong></td>
</tr>
<tr>
<td>Graph Coarsening (C) Clustering</td>
<td>87.4(±3.2)%</td>
</tr>
</tbody>
</table>
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): $L = 2$, $F^{(1)} = F^{(2)} = 32$, $K^{(1)} = K^{(2)} = 5$

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregation (A) Degree</td>
<td>95.8(±1.6)%</td>
</tr>
<tr>
<td>Aggregation (A) EDS</td>
<td>96.9(±1.2)%</td>
</tr>
<tr>
<td>Aggregation (A) SP</td>
<td>95.8(±1.4)%</td>
</tr>
<tr>
<td>Multinode (MN) Degree</td>
<td>97.6(±1.3)%</td>
</tr>
<tr>
<td>Multinode (MN) EDS</td>
<td>96.8(±1.2)%</td>
</tr>
<tr>
<td><strong>Multinode (MN) SP</strong></td>
<td><strong>99.0(±0.8)%</strong></td>
</tr>
<tr>
<td>Graph Coarsening (C) Clustering</td>
<td>95.2(±1.2)%</td>
</tr>
</tbody>
</table>
Numerical Experiments: Authorship Attribution

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

- Dataset: 546 texts by Brontë to build WAN, 1000 words (nodes)
  \(\Rightarrow\) 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): $L = 2$, $F^{(1)} = F^{(2)} = 32$, $K^{(1)} = K^{(2)} = 5$

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregation (A) Degree</td>
<td>69.5(±2.0)%</td>
</tr>
<tr>
<td>Aggregation (A) EDS</td>
<td>71.0(±2.8)%</td>
</tr>
<tr>
<td>Aggregation (A) SP</td>
<td>69.2(±4.0)%</td>
</tr>
<tr>
<td>Multinode (MN) Degree</td>
<td>80.4(±2.0)%</td>
</tr>
<tr>
<td><strong>Multinode (MN) EDS</strong></td>
<td><strong>80.5(±2.6)%</strong></td>
</tr>
<tr>
<td>Multinode (MN) SP</td>
<td>79.9(±2.8)%</td>
</tr>
<tr>
<td>Graph Coarsening (C) Clustering</td>
<td>65.2(±5.0)%</td>
</tr>
</tbody>
</table>
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


- Other extensions in graph neural networks:
  - Extend nonlinearities to include neighborhoods: arXiv:1903.12575, **today 6pm**, syndicate 1.
  - Generalization through edge-varying recursions: arXiv:1903.01298
  - Application to learning decentralized controllers: arXiv:1903.10527