LanczosNet: Multi-Scale Deep Graph Convolutional Networks

Credit: Renjie Liao, Zhizhen Zhao, Raquel Urtasun, Richard S. Zemely

https://qdata.github.io/deep2Read

Presenter: Ryan McCampbell
https://qdata.github.io/deep2Read
Outline

1. Introduction
2. Background
3. LanczosNet
4. AdaLanczosNet
5. Experiments
6. Conclusions
Outline

1. Introduction
2. Background
3. LanczosNet
4. AdaLanczosNet
5. Experiments
6. Conclusions
Introduction

Two main issues with current GCN approaches

1. How to efficiently leverage multi-scale information
   - Graph coarsening - fixed process
   - Powers of graph Laplacian - expensive

2. Spectral filters are mostly fixed
   - Learning filters can produce more useful representations
Introduction

Idea:

- Use low-rank approximation of graph Laplacian
  - Enables efficient computation of matrix powers for multi-scale information

- Design learnable spectral filters
Graph Laplacian $L = D - A$, $L = I - D^{-1}A$, or $L = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$

Affinity matrix $S = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$

Spectral decomposition $S = U\Lambda U^T$

Graph Fourier Transform $Y = U^TX$ and $\hat{X} = UY$

We can filter in the spectral domain
Localized Polynomial Filter

- $\tau$-localized polynomial filter:

$$g_w(\Lambda) = \sum_{t=0}^{\tau-1} w_t \Lambda^t$$

- Leverages information from nodes $\leq \tau$ hops away

- More general form:

$$Y = \sum_{t=0}^{\tau-1} g_t(S, ..., S^t, X) W_t$$

- Krylov subspace $\mathcal{K}_t(S, x) = \text{span}\{x, Sx, ..., S^{t-1}x\}$
Lanczos Algorithm

Given affinity matrix $S$ and node features $x$, the N-step Lanczos algorithm computes orthogonal $Q$ and symmetric tridiagonal $T$ with $Q^T S Q = T$.

$$
T = \begin{bmatrix}
\gamma_1 & \beta_1 \\
\beta_1 & \ddots & \ddots \\
& \ddots & \ddots & \beta_{N-1} \\
& & \beta_{N-1} & \gamma_N \\
\end{bmatrix}
$$

- $Q$ forms orthogonal basis of $\mathcal{K}_N(S, x)$
- First $K$ cols of $Q$ form orthogonal basis of $\mathcal{K}_K(S, x)$
Algorithm 1: Lanczos Algorithm

1: **Input:** $S, x, K, \epsilon$
2: **Initialization:** $\beta_0 = 0$, $q_0 = 0$, and $q_1 = \frac{x}{\|x\|}$
3: **For** $j = 1, 2, \ldots, K$:
   4: $z = Sq_j$
   5: $\gamma_j = q_j^T z$
   6: $z = z - \gamma_j q_j - \beta_{j-1} q_{j-1}$
   7: $\beta_j = \|z\|_2$
   8: **If** $\beta_j < \epsilon$, **quit**
   9: $q_{j+1} = \frac{z}{\beta_j}$
10:
11: $Q = [q_1, q_2, \cdots, q_K]$
12: Construct $T$ following Eq. (2)
13: Eigen decomposition $T = B R B^T$
14: Return $V = QB$ and $R$.  

Credit: Renjie Liao, Zhizhen Zhao, Raquel Urtasun, Richard S. Zemely (University of Virginia)
Localized polynomial filter

- Run Lanczos for $K$ steps starting with $X_i$ to compute orthonormal basis $Q$ of $\mathcal{K}_K(S, X_i)$

$$Y_j = Qw_{i,j}$$

- $Q$ depends on $X_i$: separate run of Lanczos is needed for each graph convolution layer

- Ideally, we want to only compute Lanczos once during inference on a graph
Alternate view:

- Choose random starting vector $x$
- Treat $K$ step Lanczos output as low-rank approximation $S \approx QTQ^T$
- Decompose tridiagonal matrix into Ritz values $T = BRB^T$
  - $R$ is diagonal: approximation of eigenvalues
- $S \approx VRV^T$ where $V = QB$
- Rewrite graph convolution as

$$Y_j = [X_i, SX_i, \ldots, S^{K-1}X_i]w_{i,j} \approx [X_i, VRV^TX_i, \ldots, VR^{K-1}V^TX_i]w_{i,j}$$
Learning the Spectral Filter

- Use $K$ different spectral filters with $k$th output
  $\hat{R}(k) = f_k([R^1, \ldots, R^{K-1}])$, where $f_k$ is MLP
  
  $$Y_j = [X_i, V\hat{R}(1)V^TX_i, \ldots, V\hat{R}(K-1)V^TX_i]w_{i,j}$$
Multi-Scale Graph Convolution

\[ Y = [L^{S_1}X, \ldots, L^{S_M}X, V\hat{R}(I_1)V^TX, \ldots, V\hat{R}(I_N)V^TX]W \]

- **S**: Short-scale parameters, e.g. 0, ..., 5
- **I**: Long-scale parameters, e.g. 10, 20, ..., 50
Algorithm 2 : LanczosNet

1: **Input:** Signal $X$, Lanczos output $V$ and $R$, scale index sets $S$ and $\mathcal{I}$,
2: **Initialization:** $Y_0 = X$
3: **For** $\ell = 1, 2, \ldots, \ell_c$:
4: \hspace{1cm} $Z = Y_{\ell-1}$, $\mathcal{Z} = \{\emptyset\}$
5: \hspace{1cm} **For** $j = 1, 2, \ldots, \max(S)$:
6: \hspace{2cm} $Z = S \mathcal{Z}$
7: \hspace{1cm} **If** $j \in S$:
8: \hspace{2cm} $\mathcal{Z} = \mathcal{Z} \cup Z$
9: \hspace{1cm} **For** $i \in \mathcal{I}$:
10: \hspace{2cm} $\mathcal{Z} = \mathcal{Z} \cup V \hat{R}(\mathcal{I}_i) V^T Y_{\ell-1}$
11: \hspace{1cm} $Y_{\ell} = \text{concat}(\mathcal{Z}) W_{\ell}$
12: **If** $\ell < L$
13: \hspace{1cm} $Y_{\ell} = \text{Dropout}(\sigma(Y_{\ell}))$
14: **Return** $Y_{\ell_c}$.  

Credit: Renjie Liao, Zhizhen Zhao, Raquel Urtasun, Richard S. Zemely (University of Virginia)

LanczosNet: Multi-Scale Deep Graph Convolutional Networks

Presenter: Ryan McCampbell

https://qdata.github.io/deep2Read
AdaLanczosNet

- Back-propagate through Lanczos algorithm
- Facilitates learning graph kernel and/or node embeddings
Learnable anisotropic graph kernel

\[ k(x_i, x_j) = \exp \left( -\frac{\| f_\theta(x_i) - f_\theta(x_j) \|^2}{\epsilon} \right) \]

- \( f_\theta \) is an MLP
- We can construct adjacency matrix \( A_{i,j} = k(x_i, x_j) \) if \((i, j) \in E\) and 0 otherwise
- Use to define affinity matrix \( S = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \)
- We can discard \( f \) to learn node embeddings on \( X \)
Tridiagonal Decomposition

- Backpropagation through the eigendecomposition of tridiagonal matrix is unstable
- Instead directly use approximation $S = QTQ^T$

$$Y = [L^{S_1}X, ..., L^{S_M}X, Qf_1(T^{I_1})Q^T X, ..., Qf_N(T^{I_N})Q^T X]W$$

- $f_i$ is learnable filter
<table>
<thead>
<tr>
<th></th>
<th>GCN-FP</th>
<th>GGNN</th>
<th>DCNN</th>
<th>ChebyNet</th>
<th>GCN</th>
<th>MPNN</th>
<th>GraphSAGE</th>
<th>GAT</th>
<th>LNet</th>
<th>AdaLNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public</td>
<td>74.6 ± 0.7</td>
<td>77.6 ± 1.7</td>
<td>79.7 ± 0.8</td>
<td>78.0 ± 1.2</td>
<td>80.5 ± 0.8</td>
<td>78.0 ± 1.1</td>
<td>74.5 ± 0.8</td>
<td><strong>82.6 ± 0.7</strong></td>
<td>79.5 ± 1.8</td>
<td>80.4 ± 1.1</td>
</tr>
<tr>
<td>3%</td>
<td>71.7 ± 2.4</td>
<td>73.1 ± 2.3</td>
<td>76.7 ± 2.5</td>
<td>62.1 ± 6.7</td>
<td>74.0 ± 2.8</td>
<td>72.0 ± 4.6</td>
<td>64.2 ± 4.0</td>
<td>56.8 ± 7.9</td>
<td>76.3 ± 2.3</td>
<td><strong>77.7 ± 2.4</strong></td>
</tr>
<tr>
<td>1%</td>
<td>59.6 ± 6.5</td>
<td>60.5 ± 7.1</td>
<td>66.4 ± 8.2</td>
<td>44.2 ± 5.6</td>
<td>61.0 ± 7.2</td>
<td>56.7 ± 5.9</td>
<td>49.0 ± 5.8</td>
<td>48.6 ± 8.0</td>
<td>66.1 ± 8.2</td>
<td><strong>67.5 ± 8.7</strong></td>
</tr>
<tr>
<td>0.5%</td>
<td>50.5 ± 6.0</td>
<td>48.2 ± 5.7</td>
<td>59.0 ± 10.7</td>
<td>33.9 ± 5.0</td>
<td>52.9 ± 7.4</td>
<td>46.5 ± 7.5</td>
<td>37.5 ± 5.4</td>
<td>41.4 ± 6.9</td>
<td>58.1 ± 8.2</td>
<td><strong>60.8 ± 9.0</strong></td>
</tr>
<tr>
<td>Citeseer</td>
<td>GCN-FP</td>
<td>GGNN</td>
<td>DCNN</td>
<td>ChebyNet</td>
<td>GCN</td>
<td>MPNN</td>
<td>GraphSAGE</td>
<td>GAT</td>
<td>LNet</td>
<td>AdaLNet</td>
</tr>
<tr>
<td>Public</td>
<td>61.5 ± 0.9</td>
<td>64.6 ± 1.3</td>
<td>69.4 ± 1.3</td>
<td>70.1 ± 0.8</td>
<td>68.1 ± 1.3</td>
<td>64.0 ± 1.9</td>
<td>67.2 ± 1.0</td>
<td><strong>72.2 ± 0.9</strong></td>
<td>66.2 ± 1.9</td>
<td>68.7 ± 1.0</td>
</tr>
<tr>
<td>1%</td>
<td>54.3 ± 4.4</td>
<td>56.0 ± 3.4</td>
<td>62.2 ± 2.5</td>
<td>59.4 ± 5.4</td>
<td>58.3 ± 4.0</td>
<td>54.3 ± 3.5</td>
<td>51.0 ± 5.7</td>
<td>46.5 ± 9.3</td>
<td>61.3 ± 3.9</td>
<td><strong>63.3 ± 1.8</strong></td>
</tr>
<tr>
<td>0.5%</td>
<td>43.9 ± 4.2</td>
<td>44.3 ± 3.8</td>
<td>53.1 ± 4.4</td>
<td>45.3 ± 6.6</td>
<td>47.7 ± 4.4</td>
<td>41.8 ± 5.0</td>
<td>33.8 ± 7.0</td>
<td>38.2 ± 7.1</td>
<td>53.2 ± 4.0</td>
<td><strong>53.8 ± 4.7</strong></td>
</tr>
<tr>
<td>0.3%</td>
<td>38.4 ± 5.8</td>
<td>36.5 ± 5.1</td>
<td>44.3 ± 5.1</td>
<td>39.3 ± 4.9</td>
<td>39.2 ± 6.3</td>
<td>36.0 ± 6.1</td>
<td>25.7 ± 6.1</td>
<td>30.9 ± 6.9</td>
<td>44.4 ± 4.5</td>
<td><strong>46.7 ± 5.6</strong></td>
</tr>
<tr>
<td>Pubmed</td>
<td>GCN-FP</td>
<td>GGNN</td>
<td>DCNN</td>
<td>ChebyNet</td>
<td>GCN</td>
<td>MPNN</td>
<td>GraphSAGE</td>
<td>GAT</td>
<td>LNet</td>
<td>AdaLNet</td>
</tr>
<tr>
<td>Public</td>
<td>76.0 ± 0.7</td>
<td>75.8 ± 0.9</td>
<td>76.8 ± 0.8</td>
<td>69.8 ± 1.1</td>
<td>77.8 ± 0.7</td>
<td>75.6 ± 1.0</td>
<td>76.8 ± 0.6</td>
<td>76.7 ± 0.5</td>
<td><strong>78.3 ± 0.3</strong></td>
<td>78.1 ± 0.4</td>
</tr>
<tr>
<td>0.1%</td>
<td>70.3 ± 4.7</td>
<td>70.4 ± 4.5</td>
<td>73.1 ± 4.7</td>
<td>55.2 ± 6.8</td>
<td>73.0 ± 5.5</td>
<td>67.3 ± 4.7</td>
<td>65.4 ± 6.2</td>
<td>59.6 ± 9.5</td>
<td><strong>73.4 ± 5.1</strong></td>
<td>72.8 ± 4.6</td>
</tr>
<tr>
<td>0.05%</td>
<td>63.2 ± 4.7</td>
<td>63.3 ± 4.0</td>
<td>66.7 ± 5.3</td>
<td>48.2 ± 7.4</td>
<td>64.6 ± 7.5</td>
<td>59.6 ± 4.0</td>
<td>53.0 ± 8.0</td>
<td>50.4 ± 9.7</td>
<td><strong>68.8 ± 5.6</strong></td>
<td>66.0 ± 4.5</td>
</tr>
<tr>
<td>0.03%</td>
<td>56.2 ± 7.7</td>
<td>55.8 ± 7.7</td>
<td>60.9 ± 8.2</td>
<td>45.3 ± 4.5</td>
<td>57.9 ± 8.1</td>
<td>53.9 ± 6.9</td>
<td>45.4 ± 5.5</td>
<td>50.9 ± 8.8</td>
<td>60.4 ± 8.6</td>
<td><strong>61.0 ± 8.7</strong></td>
</tr>
</tbody>
</table>

Table 1: Test accuracy with 10 runs on citation networks. The public splits in Cora, Citeseer and Pubmed contain 5.2%, 3.6% and 0.3% training examples respectively.
<table>
<thead>
<tr>
<th>Methods</th>
<th>Validation MAE ($\times 1.0e^{-3}$)</th>
<th>Test MAE ($\times 1.0e^{-3}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN-FP [29]</td>
<td>15.06 ± 0.04</td>
<td>14.80 ± 0.09</td>
</tr>
<tr>
<td>GGNN [37]</td>
<td>12.94 ± 0.05</td>
<td>12.67 ± 0.22</td>
</tr>
<tr>
<td>DCNN [8]</td>
<td>10.14 ± 0.05</td>
<td>9.97 ± 0.09</td>
</tr>
<tr>
<td>ChebyNet [7]</td>
<td>10.24 ± 0.06</td>
<td>10.07 ± 0.09</td>
</tr>
<tr>
<td>GCN [11]</td>
<td>11.68 ± 0.09</td>
<td>11.41 ± 0.10</td>
</tr>
<tr>
<td>MPNN [62]</td>
<td>11.16 ± 0.13</td>
<td>11.08 ± 0.11</td>
</tr>
<tr>
<td>GraphSAGE [39]</td>
<td>13.19 ± 0.04</td>
<td>12.95 ± 0.11</td>
</tr>
<tr>
<td>GPNN [40]</td>
<td>12.81 ± 0.80</td>
<td>12.39 ± 0.77</td>
</tr>
<tr>
<td>GAT [33]</td>
<td>11.39 ± 0.09</td>
<td>11.02 ± 0.06</td>
</tr>
<tr>
<td>LanczosNet</td>
<td>9.65 ± 0.19</td>
<td>9.58 ± 0.14</td>
</tr>
<tr>
<td>AdaLanczosNet</td>
<td>10.10 ± 0.22</td>
<td>9.97 ± 0.20</td>
</tr>
</tbody>
</table>

Table 2: Mean absolute error on QM8 dataset.
Outline

1. Introduction
2. Background
3. LanczosNet
4. AdaLanczosNet
5. Experiments
6. Conclusions
Conclusions

- This method enables more powerful learning on graphs, incorporating multi-scale information as well as learned spectral filters and graph kernels.
- It outperforms many other graph networks on difficult problems.