Graph Signal Processing using Deep Neural Networks

Jayaraman J. Thiagarajan
Joint Work With

Rushil Anirudh
LLNL

Bhavya Kailkhura
LLNL

Prasanna Sattigeri
IBM Research

Karthi Ramamurthy
IBM Research
Graphs are Everywhere

Graphics  Social Networks  Medicine

Graphs can capture complex relational characteristics for analysis and inference
Undirected Graphs – Basic Notations

Graph $G = (\mathcal{V}, E, w)$. 

Adjacency matrix $A$

Degree matrix $D = \text{diag}\{d_i\}$

Laplacian matrix $L = D - A$.

Normalized Laplacian matrix $\mathcal{L} = D^{-1/2}LD^{-1/2}$
Signals on a Graph can be as Interesting as the Graph Itself

Graph signal: a function $f : \mathcal{V} \rightarrow \mathbb{R}$ that assigns real values to each vertex of the graph

- Examples:
  - Brain networks and fMRI signals
  - Gene regulatory networks and gene expression levels
  - Social networks and information cascades

Analyze characteristics of the signal by exploiting the graph structure
Connection to Fourier Analysis – Eigenvectors as Frequencies

The “classical” graph:

\[
L_{cl} = \begin{bmatrix}
2 & -1 & 0 & \cdots & 0 & -1 \\
-1 & 2 & -1 & \cdots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & 2 & -1 \\
-1 & 0 & 0 & \cdots & -1 & 2
\end{bmatrix}
\]

Any graph:

By analogy, any graph’s Fourier modes are the eigenvectors of its Laplacian matrix \( L \).

All classical Fourier modes are the eigenvectors of \( L_{cl} \)
Graph Fourier Transform

- Using the eigenvectors of the Laplacian, we can define

\[
\hat{f}(\lambda_\ell) := \langle f, u_\ell \rangle = \sum_{i=1}^{N} f(i) u_\ell^*(i)
\]

\[
f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\lambda_\ell) u_\ell(i)
\]

Number of zero crossings
Graph Filtering is the Key to Designing Graph Convolutional Neural Networks

Graphs vs Euclidean grids
- Irregular sampling.
- Weighted edges.

A graph filter $H : \mathbb{R}^N \rightarrow \mathbb{R}^N$ is a map between graph signals.

Focus on linear filters
⇒ map represented by an $N \times N$ matrix

\[
\hat{f}_{out}(\lambda_\ell) = \hat{f}_{in}(\lambda_\ell) \hat{h}(\lambda_\ell)
\]
Graph Convolutional Networks Learn a Sequence of Graph Filters

Chebyshev Approximation

\[ g_\theta \ast x = \hat{\theta}_0 x - \hat{\theta}_1 D^{-1/2} W D^{-1/2} x \]
Problem: Using Graph CNN for Autism Spectrum Disorder Classification

Neuropathology studies map variations in brain functionality to clinical measures

<table>
<thead>
<tr>
<th>Population Graph Design</th>
<th>Signal Construction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-imaging features (e.g. gender/site) or combination of imaging and non-imaging features</td>
<td>Statistics from imaging features (e.g. fMRI)</td>
</tr>
</tbody>
</table>
Fine-Tuning Graph CNNs is Highly Challenging due to their Sensitivity to Graph Design

We adopt a bootstrapping approach that performs dropout in the input layer of the network.
Results: Autism Brain Imaging Data Exchange Initiative

- 872 patients from 20 different sites
- Signal Construction: Upper-Triangular part of covariance matrix + PCA

<table>
<thead>
<tr>
<th>Baselines</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear SVM</td>
<td>64.71</td>
</tr>
<tr>
<td>Kernel SVM</td>
<td>65.72</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ensemble G-CNN</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(Gender, Site)</td>
<td>67.7</td>
</tr>
<tr>
<td>Linear Kernel +</td>
<td>67.7</td>
</tr>
<tr>
<td>(Gender, Site)</td>
<td></td>
</tr>
<tr>
<td>Graph Kernel +</td>
<td>68.28</td>
</tr>
<tr>
<td>(Gender, Site)</td>
<td></td>
</tr>
</tbody>
</table>

Bootstrapped Graph CNNs improve the prediction accuracy on the challenging ABIDE dataset
Problem: Neighborhood Graph Construction is Sensitive to Sample Density and Noise

- Same signal defined on different graphs can lead to completely different results.
Idea 1: Graph Auto-Encoders can Recover the Optimal Embeddings for Data

- Auto-encoders aim to minimize the reconstruction loss

\[ \text{Loss}(\theta_1, \theta_2) = \sum_{i=1}^{N} l\left(x_i, g\left(f(x_i; \theta_1); \theta_2\right)\right) \]

- The “reconstruction nature” of spectral clustering can be viewed as an auto-encoder

\[ \arg \min_{\tilde{W}} \| \mathbf{W} - \tilde{\mathbf{W}} \|_F = \mathbf{U}\tilde{\Sigma}\mathbf{V}^T, \text{ s.t. } \text{rank}(\tilde{\mathbf{W}}) = k \]

- Using L2 loss – Hidden features will converge to the smallest eigenvectors of the Laplacian matrix
Idea 2: Spectral Embedding with Quantile Loss Enables Robust Sparsification of Graphs

Piecewise Linear Quadratic Loss typically used to explore heterogeneous datasets

\[ \rho_\tau(r) = \begin{cases} 
\frac{\tau |r|}{2} - \frac{\kappa \tau^2}{2} & \text{if } r < -\tau \kappa, \\
\frac{1}{2\kappa} r^2 & \text{if } r \in [-\kappa \tau, (1 - \tau) \kappa], \\
(1 - \tau) |r| - \frac{\kappa (1 - \tau)^2}{2} & \text{if } r > (1 - \tau) \kappa.
\end{cases} \]

\[
\min_{L, R} \rho_\tau(W - LR^T)
\]

Low-rank decomposition of the similarity matrix
Idea 2: Spectral Embedding with Quantile Loss Enables Robust Sparsification of Graphs

When the quantile parameter is large:
- Positive residuals will be penalized lesser, i.e. the approximation will underestimate the similarity function

When the quantile parameter is small:
- Returns a dense graph with overestimated edge strengths

\[
\min_{L,R} \rho_\tau(W - LR^T)
\]

Low-rank decomposition of the similarity matrix
Conditional Quantiles of the Similarity Function Reveal the Importance of Edges

\[ p(e_{ij}) = \max\left(\delta, 1 - \frac{\beta_{\hat{T}}}{\beta_{0.1}}\right) \]

- Minimum probability for any edge
- Highest quantile at which edge persists
- Number of edges at 0.1 quantile
Using the Locally-Scaled Graphs in Semi-Supervised Label Propagation

- Given a few labeled examples and a neighborhood graph, propagate labels to all samples – Greedy random walk

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Local Scaling</th>
<th>Path-Based</th>
<th>Proposed Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blood Transfusion</td>
<td>76.1</td>
<td>77.1</td>
<td>84.9</td>
</tr>
<tr>
<td>Breast Cancer</td>
<td>82.7</td>
<td>84.9</td>
<td>90.5</td>
</tr>
<tr>
<td>Echocardiogram</td>
<td>70.1</td>
<td>73.2</td>
<td>88.49</td>
</tr>
<tr>
<td>Kidney Disease</td>
<td>66.8</td>
<td>67.5</td>
<td>71.9</td>
</tr>
<tr>
<td>SPECT Heart</td>
<td>70</td>
<td>68.5</td>
<td>86</td>
</tr>
<tr>
<td>Thoracic Surgery</td>
<td>65.3</td>
<td>66.2</td>
<td>75.8</td>
</tr>
<tr>
<td>Arcene</td>
<td>59</td>
<td>61.4</td>
<td>71.3</td>
</tr>
</tbody>
</table>

Extreme Case – 1 Labeled Node
Conclusions

- Graph signal processing provides a convenient framework for a wide variety of data analysis problems.
- Can be highly effective in fusing imaging and non-imaging features in a neural network setting.
- Generalized deep networks for data on non-uniform grids – *the new kid on the block*
- Use of appropriate loss functions can lead to robust graph constructions.
- Applies to problems in supervised, unsupervised, semi-supervised, transfer learning...
Questions?

Contact
Jayaraman J. Thiagarajan
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Email: jjayaram@llnl.gov