Revisiting GNN: Graph Filtering Perspective

Hoang NT, Takanori Maehara, and Tsuyoshi Murata
Tokyo Institute of Technology, RIKEN AIP

December 9, 2020
Outline

Vertex Classification Problem
  Problem Illustration
  Assumption on Benchmark Datasets

Understanding the SOTA for Vertex Classification
  Frequency Analysis Perspective
  Filter Then Classify

Next steps
  On Vertex Classification
Vertex Classification Problem
Community detection [PARS14, YCS16, KW17]; recommendation systems [YHC+18]; molecular discovery/generation [YLY+18]; weakly-supervised learning [KCL+19].
Most recent research have chosen Cora as the benchmark datasets.

What is their assumptions for dataset like Cora?
My answer: Low-frequency assumption!
Understanding the SOTA for Vertex Classification
Given a symmetric Laplacian matrix $L \in \mathbb{R}^{n \times n}$ of a graph $G$, the Rayleigh quotient $R(L, f)$ for $f \in \mathbb{R}^n$ is given as:

$$R(L, f) = \frac{f^\top L f}{f^\top f} = \frac{1}{f^\top f} \sum_{u \sim v} (f(u) - f(v))^2 \quad (1)$$

**Figure:** $R(L, f) = 1/3$. $f$ is called “low-frequency”.

**Figure:** $R(L, f) = 5/3$. $f$ is called “high-frequency”.
Figure: Rayleigh quotient of $\mathcal{Y}$ in benchmark datasets
The classification accuracy *increases* in the low-frequency regions for the benchmark datasets. In addition, this low-frequency regions (green boxes) are relatively noise tolerant.
Two previous experiments show:

- Information is concentrated in the low-frequency regions.
- Rayleigh quotient can be used to predict the useful frequency regions.

**Assumption**

*In the vertex classification problem, we assume $R(L, y)$ to be sufficiently small. If $R(L, y)$ is large, the performances of most SOTA models are not guaranteed.*

This is the “low-frequency” assumption. This assumption is also made for the feature $\mathcal{X}$. 
Most recent models can be generalized to “filter-then-classify” approach. The proposal of SGC [WZSJ\textsuperscript{+}19] and the work by [LWL\textsuperscript{+}19] support this observation.

\[
h_{	ext{GCN}} = W_2 \times \text{gf}(A) \times \sigma[W_1 \times \text{gf}(A) \times X]
\]

\[
h_{	ext{SGC}} = W_1 \times \text{gf}(A)^k \times X
\]

\[
h_{	ext{gfNN}} = W_2 \times \sigma[W_1 \times \text{gf}(A)^k \times X]
\]

We will see that “filter-then-classify” has a few advantages to the feature propagation understanding.

**Figure:** Toy models.
GCN and other multi-layers model might overfit to noisy data.

Figure: Add gaussian noise to features.
Claim: Graph filters cannot “learn” manifolds!

Figure: Results for donuts case.
We have to learn the filters, not only the neural network’s weights!

Figure: Results for high frequency case

In this setting, $R(\mathcal{L}, y) \approx 2$ (maximum value).
High Frequency: Feature Shift

\[ \text{softmax} \]

\[ W_2 \]

\[ \text{gf}(A) \ Stars \]

\[ W_1 \]

\[ \text{gf}(A) \ Stars \]

\[ X \]

GCN
1. Most benchmark datasets are community detection in nature, hence the designs for SOTA Graph Neural Networks are biased toward the low-frequency characteristics of these datasets.

2. A tool like Rayleigh quotient and more flexible models like gfNN are needed in solving real-world vertex classifications.

3. High frequency cases (or different frequency case) are interesting because they can be used for constructing adversarial examples for graphs.

4. Disadvantage of filter-then-classify is that it doesn't provide an immediate intuition for the spacial domain. Also, currently selecting the appropriate graph filter for a problem beyond Decision Trees and Random Forest remains an open problem.
Next steps
Viewing graph simply as a filter allows several directions:

- **Statistical learning analysis:** Quantify model complexity, number of samples for optimal training (somewhat similar to the Nyquist rate in SP and CS).

- **Practical models:** Adaptive filters with trade-off of data efficiency\(^1\).


[LWL⁺19] Qimai Li, Xiao-Ming Wu, Han Liu, Xiaotong Zhang, and Zhichao Guan.
Label efficient semi-supervised learning via graph filtering.

[NJW02] Andrew Y Ng, Michael I Jordan, and Yair Weiss.
On spectral clustering: Analysis and an algorithm.

Deepwalk: Online learning of social representations.

Simplifying graph convolutional networks.

Revisiting semi-supervised learning with graph embeddings.
Graph convolutional neural networks for web-scale recommender systems.

Graph convolutional policy network for goal-directed molecular graph generation.
Baseline vertex classification models such as spectral clustering [NJW02] often use first few eigenvectors to make feature vectors for vertices. More recent models such as Deepwalk [PARS14] or Planetoid [YCS16] relies on embedding neighbors “close” together.

Recent neural network based models such as ChebNet [DBV16], GCN [KW17], and GraphSAGE [HYL17] combine vertex features with graph structure by averaging neighbors (similar to feature propagation). **Common theme:** Low-frequency design!
Multiplying the feature vectors to $\Delta_{\text{sym}} = I - D^{-1/2}LD^{-1/2}$ is similar to applying the $1 - \lambda$ filter. Furthermore, adding loops to the graph truncates the largest eigenvector.

**Theorem 1 ([WZSJ+19])**

*Let $A$ be the adjacency matrix of an undirected, weighted, simple graph $G$ without isolated nodes and with corresponding degree matrix $D$. Let $\tilde{A} = A + \gamma I$, such that $\gamma > 0$, be the augmented adjacency matrix with corresponding degree matrix $D$. Also, let $\lambda_1$ and $\lambda_n$ denote the smallest and largest eigenvalues of $\Delta_{\text{sym}} = I - D^{-1/2}AD^{-1/2}$; similarly, let $\tilde{\lambda}_1$ and $\tilde{\lambda}_n$ be the smallest and largest eigenvalues of $\tilde{\Delta}_{\text{sym}}$. We have that*

$$0 = \lambda_1 = \tilde{\lambda}_1 < \tilde{\lambda}_n < \lambda_n$$
Since \([WZSJ^{+}19]\) only proved for the largest eigenvalues, we do not know the relation between \(\lambda_i\) and \(\tilde{\lambda}_i\) for \(0 < i < n\).

Solution: Use the Courant–Fisher–Weyl’s min-max principle to argue about other pairs of eigenvalues!

**Theorem 2 (NTMM, 2019)**

Let \(\lambda_i(\gamma)\) be the \(i\)-th smallest generalized eigenvalue of \((\tilde{D}, L) = (D + \gamma I)\). Then, \(\lambda_i(\gamma)\) is a non-negative number, and monotonically non-increasing in \(\gamma \geq 0\). Moreover, \(\lambda_i(\gamma)\) is strictly monotonically decreasing if \(\lambda_i(0) \neq 0\).
It is trivial to see that $\lambda_1 = \tilde{\lambda}_1 = 0$:

$$x^\top \tilde{\Delta}_{\text{sym}} x = \sum_i x_i^2 - \sum_i \sum_j \frac{\tilde{a}_{ij}x_i x_j}{\sqrt{(d_i + \gamma)(d_j + \gamma)}} \leq 0 \quad (2)$$

Let $\beta_1 \leq \beta_2 \leq \ldots \leq \beta_n$ be the eigenvalues of $D^{-1/2} A D^{-1/2}$ and $\alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_n$ be the eigenvalues of $\tilde{D}^{-1/2} A \tilde{D}^{-1/2}$. We see that $\beta_1 < 0$. Choose $x$ such that $||x|| = 1$ and $y = D^{1/2} \tilde{D}^{-1/2} x$, 
see that \( \|y\|^2 = \sum_i \frac{d_i}{d_i + \gamma} x_i^2 \) and \( \frac{\min_i d_i}{\gamma + \min_i d_i} \leq \|y\|^2 \leq \frac{\max_i d_i}{\gamma + \max_i d_i} \).

Using Rayleigh quotient to look at \( \alpha_1 \):

\[
\alpha_1 = \min_x \left( x^\top \tilde{D}^{-1/2} A \tilde{D}^{-1/2} x \right) = \min_x \left( y^\top D^{-1/2} AD^{-1/2} y \frac{\|y\|^2}{\|y\|^2} \right)
\]

\[
\geq \min_x \left( y^\top D^{-1/2} AD^{-1/2} y \frac{\|y\|^2}{\|y\|^2} \right) \max_x (\|y\|^2) \geq \beta_1 \max_x \|y\|^2 \geq \frac{\max_i d_i}{\gamma + \max_i d_i}
\]
Proof by [WZSJ$^+$19] III

Note that $\tilde{\Delta}_{\text{sym}} = I - \gamma \tilde{D}^{-1} - \tilde{D}^{-1/2} A \tilde{D}^{-1/2}$. Using the result above we have:

$$\lambda_n = \max_x x^\top (I - \gamma \tilde{D}^{-1} - \tilde{D}^{-1/2} A \tilde{D}^{-1/2}) x$$  \hspace{1cm} (8)  

$$\leq 1 - \min_x \gamma x^\top \tilde{D}^{-1} x - \min_x x^\top \tilde{D}^{-1/2} A \tilde{D}^{-1/2} x$$  \hspace{1cm} (9)  

$$= 1 - \frac{\gamma}{\gamma + \max_i d_i} - \alpha_1$$  \hspace{1cm} (10)  

$$< 1 - \beta_1 = \lambda_n$$  \hspace{1cm} (11)
Since the generalized eigenvalues of \((D + \gamma I, L)\) are the eigenvalues of a positive semidefinite matrix 
\((D + \gamma I)^{1/2} L (D + \gamma I)^{1/2}\), these are non-negative real numbers. 
To obtain the shrinking result, we use the Courant–Fisher–Weyl’s min-max principle [Bha13, Corollary III. 1.2]: For any \(0 \leq \gamma_1 < \gamma_2\),

\[
\lambda_i(\gamma_2) = \min_{H:\text{subspace}, \dim(H) = i} \max_{x \in H, x \neq 0} \frac{x^\top L x}{x^\top (D + \gamma_2 I) x} \leq \min_{H:\text{subspace}, \dim(H) = i} \max_{x \in H, x \neq 0} \frac{x^\top L x}{x^\top (D + \gamma_1 I) x} = \lambda_i(\gamma_1). 
\]

Here, the second inequality follows because
\(x^\top (D + \gamma_1) x < x^\top (D + \gamma_2) x\) for all \(x \neq 0\) Hence, the inequality is strict if \(x^\top L x \neq 0\), i.e., \(\lambda_i(\gamma_1) \neq 0\).
Table: Real-world benchmark datasets and synthetic datasets for vertex classification

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Nodes</th>
<th>Edges</th>
<th>Features (X)</th>
<th>(µ_X, σ_X)</th>
<th>Classes</th>
<th>Train/Val/Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>2,708</td>
<td>5,278</td>
<td>1,433</td>
<td>(0.0007, 0.0071)</td>
<td>7</td>
<td>140/500/1,000</td>
</tr>
<tr>
<td>Citeseer</td>
<td>3,327</td>
<td>4,732</td>
<td>3,703</td>
<td>(0.0003, 0.0029)</td>
<td>6</td>
<td>120/500/1,000</td>
</tr>
<tr>
<td>Pubmed</td>
<td>19,717</td>
<td>44,338</td>
<td>500</td>
<td>(0.0019, 0.0087)</td>
<td>3</td>
<td>60/500/1,000</td>
</tr>
<tr>
<td>Reddit</td>
<td>231,443</td>
<td>11,606,919</td>
<td>602</td>
<td>-</td>
<td>41</td>
<td>151,708/23,699/55,334</td>
</tr>
<tr>
<td>PPI</td>
<td>56,944</td>
<td>818,716</td>
<td>50</td>
<td>-</td>
<td>121</td>
<td>44,906/6,514/5,524</td>
</tr>
<tr>
<td>Two Circles</td>
<td>4,000</td>
<td>10,000</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>80/80/3,840</td>
</tr>
<tr>
<td>BA-High</td>
<td>200</td>
<td>2000</td>
<td>50</td>
<td>(0,1)</td>
<td>2</td>
<td>10/10/180</td>
</tr>
</tbody>
</table>
High Frequency Artificial Data

In this setting, $R(L, y) \approx 2$ (maximum value).

Figure: Artificial BA with high-freq labels.
### Classification results

Table: Average test accuracy on original train/val/test splits (50 times)

<table>
<thead>
<tr>
<th>Method</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
<th>Reddit</th>
<th>PPI</th>
<th>2Circles</th>
<th>BA-High</th>
</tr>
</thead>
<tbody>
<tr>
<td>DGI</td>
<td>83.1 ± 0.2</td>
<td>72.1 ± 0.1</td>
<td>80.1 ± 0.2</td>
<td>94.5 ± 0.3</td>
<td>99.2 ± 0.1</td>
<td>85.2 ± 0.6</td>
<td>54.6 ± 1.8</td>
</tr>
<tr>
<td>GCN</td>
<td>80.0 ± 1.8</td>
<td>69.6 ± 1.1</td>
<td>79.3 ± 1.3</td>
<td>-</td>
<td>-</td>
<td>84.9 ± 0.8</td>
<td>58.9 ± 2.2</td>
</tr>
<tr>
<td>SGC</td>
<td>77.6 ± 2.2</td>
<td>65.6 ± 0.1</td>
<td>78.4 ± 1.1</td>
<td>94.9 ± 0.2</td>
<td>89.0 ± 0.1</td>
<td>53.5 ± 1.4</td>
<td>55.5 ± 1.3</td>
</tr>
<tr>
<td>gfNN-low</td>
<td>82.3 ± 0.2</td>
<td>71.8 ± 0.1</td>
<td>79.2 ± 0.2</td>
<td>94.8 ± 0.2</td>
<td>89.3 ± 0.5</td>
<td>85.6 ± 0.8</td>
<td>55.4 ± 2.3</td>
</tr>
<tr>
<td>gfNN-high</td>
<td>24.2 ± 1.9</td>
<td>22.5 ± 2.2</td>
<td>43.6 ± 1.3</td>
<td>10.5 ± 2.6</td>
<td>86.6 ± 0.1</td>
<td>48.3 ± 3.5</td>
<td>96.2 ± 1.0</td>
</tr>
<tr>
<td>gf-Ensemble</td>
<td>82.9 ± 1.2</td>
<td>72.3 ± 1.2</td>
<td>81.5 ± 1.3</td>
<td>94.8 ± 0.2</td>
<td>88.2 ± 0.4</td>
<td>83.5 ± 0.3</td>
<td>95.7 ± 1.2</td>
</tr>
</tbody>
</table>
Compute Rayleigh quotient for $\mathcal{Y}$

We use the symmetric normalized Laplacian $\mathcal{L} = D^{-1/2}(D - A)D^{-1/2}$ and create a one-hot vector to indicate the label on each vertex.

For example, suppose we have a simple graph $G = (V, E)$, $|V| = n$, $|E| = m$, $\mathcal{Y}: V \mapsto \mathcal{C}$, and $|\mathcal{C}| = 3$. We construct the one-hot matrix: $Y \in 0, 1^{n \times |\mathcal{C}|}$. We denote $Y_i \in 0, 1^n$ as the column of the binary matrix $Y$. The Rayleight quotient for label $i$ is given by:

$$R(\mathcal{L}, Y_i) = \frac{Y_i^\top \mathcal{L} Y_i}{Y_i^\top Y_i} = \frac{1}{Y_i^\top Y_i} \sum_{u \sim v} (f(u) - f(v))^2$$
Graph Low-pass Filters: Hard filter

1. Compute the graph Fourier basis $U$ from $\mathcal{L}$
2. Add Gaussian noise to the input features: $\mathcal{X} \leftarrow \mathcal{X} + \mathcal{N}(0, \sigma^2)$ for $\sigma = \{0, 0.01, 0.05\}$
3. Compute the first $k$-frequency component:
   $\hat{\mathcal{X}}_k = U[:, k]^{\top} D^{1/2} \mathcal{X}$
4. Reconstruct the features: $\tilde{\mathcal{X}}_k = D^{-1/2} U[:, k] \hat{\mathcal{X}}_k$
5. Train and report test accuracy of a 2-layers neural net on the reconstructed features $\tilde{\mathcal{X}}_k$