Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning

Qimai Li, Zhichao Han, Xiao-Ming Wu

AAAI 2018
Overview

- Graph-based Semi-Supervised Learning.
- Spectral Graph Convolutions.
- Graph Convolutional Networks (GCNs).
- Why GCNs work?
- When GCNs fail?
Graph-based Semi-Supervised Learning

• Problem: classifying nodes in a graph, where labels are only available for a small subset of nodes.
• Popular assumption: connected nodes in the graph are likely to share the same label.
• Training objective:

\[ \mathcal{L} = \mathcal{L}_0 + \lambda \mathcal{L}_{\text{reg}} \]

Where \[ \mathcal{L}_{\text{reg}} = \sum_{i,j} A_{i,j} \| f(X_i) - f(X_j) \|^2 \]

=> Limitation:
+ Structure information is weakly encoded.

• Graph Convolutional Networks (Kipf and Welling, 2017) directly operates on graphs, motivated from a first-order approximation of spectral graph convolutions.
Spectral Graph Convolutions

• Some notations:
  + Undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the vertex set and $\mathcal{E}$ is the edge set. $|\mathcal{V}| = N$ is the number of nodes.
  + Adjacency matrix $A$ (no self-loops).
  + Degree matrix $D$ where $D_{ii} = \sum_j A_{ij}$ is the degree of node $i$.
  + Graph Laplacian matrix $L := D - A$ with two normalized versions:
    - Symmetric normalization: $L_{\text{sym}} := D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$
    - Random walk normalization: $L_{\text{rw}} := D^{-1} L$
  + Self-loops added versions:
    - Adjacency matrix: $\tilde{A} = A + I$
    - Degree matrix: $\tilde{D} = \sum_j \tilde{A}_{ij}$
Spectral Graph Convolutions

• Convolution theorem states that convolution of two matrices is equivalent to pointwise multiplication in the fourier domain:

\[ \mathcal{F}(x * y) = \mathcal{F}(x) \odot \mathcal{F}(y) \]

Where \( \mathcal{F}( \cdot ) \) denotes fourier transform operator.

• In graph theory, the fourier transformation usually refers to the transformation to the eigenvector dimension of the graph Laplacian \( L_{\text{sym}} \).

• Spectral convolutions on a graph between a signal \( x \in \mathbb{R}^N \) in the vertex (spatial) domain, and a filter \( g_{\theta} = \text{diag}(\theta) \) parameterized by \( \theta \in \mathbb{R}^N \) in the spectral (eigenvector space of \( L_{\text{sym}} \)) domain is defined as:

\[ g_{\theta} * x = U g_{\theta} U^\top x \]

Where \( L_{\text{sym}} = U \Lambda U^\top \) where \( U \) is the matrix of eigenvectors of \( L_{\text{sym}} \).
Graph Convolutional Networks

- Hammond et al. (2011) shows that spectral convolutions can be well-approximated via Chebyshev polynomials $T_k( )$:
  \[ g_{\theta'} \star x \approx \sum_{k=0}^{K} \theta'_{k} T_{k}(\tilde{L})x \]
  In which, $\tilde{L} = 2\frac{\lambda_{\max}}{L_{\text{sym}}} I_N$ where $\lambda_{\max}$ is the largest eigenvalue of $L_{\text{sym}}$.

- Kipf and Welling (2017) simplified this by limiting $K=1$ and approximating $\lambda_{\max}$ by $2$, resulting in:
  \[ g_{\theta'} \star x \approx \theta'_0 x + \theta'_1 (L - I_N) x = \theta'_0 x - \theta'_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x \]
  They further simplified this by setting $\theta = \theta'_0 = -\theta'_1$, leading to:
  \[ g_{\theta} \star x \approx \theta \left( I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) x \]
  To avoid numerical instability, they replaced $I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \rightarrow \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$.
  \[ Z = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X \Theta \]
Graph Convolutional Networks

- Full model of Graph Convolutional Networks (2 layers):
  \[ Z = f(X, A) = \text{softmax}\left( \hat{A} \text{ReLU}\left( \hat{A}XW^{(0)} \right) W^{(1)} \right) \]
  where \( \hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \).

- Interpretation of \( \hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \):
  + \( \tilde{A} = A + I_N \) is the unnormalized adjacency matrix with self-loops.
  + \( \tilde{A}_{\text{row}} = \tilde{D}^{-1} \tilde{A} \) is the row normalized matrix. What we usually use:

\[
h_{ij}^{l+1} = \frac{\sum_{(i,j) \in \mathcal{V}} a_{ik} h_{kj}^l}{d_{i,i}^{\frac{1}{2}}}
\]

+ \( \hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \) normalized by both neighborhood size of the current node and that of its current neighbor. The propagation rule is:

\[
h_{ij}^{l+1} = \frac{\sum_{(i,j) \in \mathcal{V}} a_{ik} h_{kj}^l}{\sqrt{d_{k,k}d_{i,i}}}
\]

(less useful if getting information from a node that connects to so many nodes)
Why GCNs work?

- Laplacian smoothing is a theoretically strong method for mesh smoothing, which is the process of changing vertex positions in a mesh in order to improve the mesh quality for finite element analysis.

Source: Improved Laplacian Smoothing of Noisy Surface Meshes.
Why GCNs work?

- Laplacian smoothing is a theoretically strong method for mesh smoothing, which is the process of changing vertex positions in a mesh in order to improve the mesh quality for finite element analysis.

- Turns out GCNs' propagation rule is a special case of a theoretically strong method called Laplacian smoothing (Taubin 1995).

\[ \hat{y}_i = (1 - \gamma)x_i + \gamma \sum_j \tilde{a}_{ij} \frac{d_i}{d_j} x_j ; \quad \hat{Y} = X - \gamma \tilde{D}^{-1} \tilde{L} X = (I - \gamma \tilde{D}^{-1} \tilde{L}) X \]

where \( \tilde{L} = \tilde{D} - \tilde{A} \).

- Letting \( \gamma = 1 \) and replacing \( \tilde{D}^{-1} \tilde{L} \) with \( \tilde{D}^{-1/2} \tilde{L} \tilde{D}^{-1/2} \), we have:

\[ \hat{Y} = \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} X \]

Which is the propagation rule over a graph of GCNs.
When GCNs fail?

- Stacking many layers of GCNs can lead to over-smoothing issue:
When GCNs fail?

- Stacking many layers of GCNs can lead to over-smoothing issue.

- Theorem:
  - A graph $G$ with $k$ connected components $\{C_i\}_{i=1}^{k}$
  - Indication vector for the $i$-th component $1^{(i)} \in \mathbb{R}^N$
  - If $G$ has no bipartile component, then, for any $w \in \mathbb{R}^N$ and $\alpha \in (0, 1]$, we have:
    \[
    \lim_{m \to +\infty} (I - \alpha L_{rw})^m w = [1^{(1)}, 1^{(2)}, \ldots, 1^{(k)}] \theta_1,
    \]
    \[
    \lim_{m \to +\infty} (I - \alpha L_{sym})^m w = D^{-\frac{1}{2}} [1^{(1)}, 1^{(2)}, \ldots, 1^{(k)}] \theta_2
    \]

Where $\theta_1 \in \mathbb{R}^k$, $\theta_2 \in \mathbb{R}^k$; and
+ $\{1^{(i)}\}_{i=1}^{k}$ are the eigenvectors corresponding to eigenvalue 1 of $(I - \alpha L_{rw})$
+ $\{D^{-\frac{1}{2}} 1^{(i)}\}_{i=1}^{k}$ are the eigenvectors corresponding to eigenvalue 1 of $(I - \alpha L_{sym})$
References

• Semi-Supervised Learning with Graph Convolutional Networks (https://arxiv.org/pdf/1609.02907.pdf)
• Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning (https://arxiv.org/pdf/1801.07606.pdf)
• https://personal.utdallas.edu/~hkokel/articles/GraphConvolutionalNetwork.html