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

=> 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:
    - Symmetrix 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 $\mathcal{F}(\ )$ denotes fourier transform operator.

- In graph theory, the fourier transformation usually refers transformation to the eigenvector dimension of the graph Laplacian L_{sym} , which shows in what directions and how smoothly the “energy” will diffuse over a graph.
- 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_{sym}) domain is defined as:

$$g_\theta \star x = U g_\theta U^\top x$$

  Where $L_{\text{sym}} = U \Lambda U^\top$ where U is the matrix of eigenvectors of L_{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, $\tilde{L} = \frac{2}{\lambda_{\max}} L_{\text{sym}} - I_N$ where λ_{max} is the largest eigenvalue of L_{sym}.

- Kipf and Welling (2017) simplified this by limitting K=1 and approximating λ_{max} by 2, resulting in:

$$g_{\theta'} \star x \approx \theta'_0 x + \theta'_1 \left(L - I_N\right) x = \theta'_0 x - \theta'_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x$$

- 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):

$$Z = f(X, A) = \mathrm{softmax}\Big(\hat{A}\ \mathrm{ReLU}\Big(\hat{A}XW^{(0)}\Big)\ W^{(1)}\Big)$$

  where $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$.

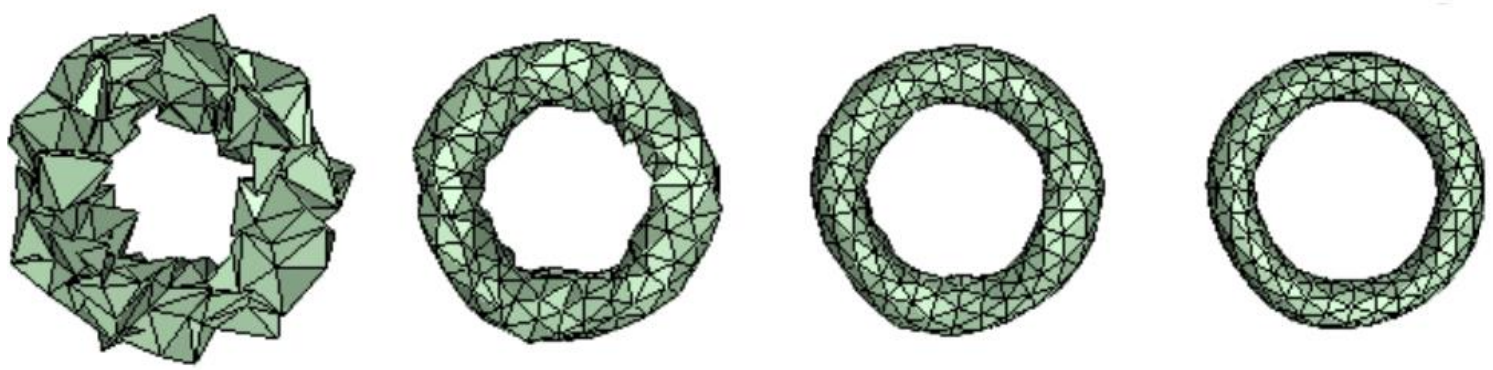
- Intepretation of $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$:
  + $\tilde{A} = A + I_N$ is the unnormalized adjacency matrix with self-loops.
  + $\tilde{A}_{row} = \tilde{D}^{-\frac{1}{2}}\tilde{A}$ is the row normalized matrix. What we usually use: $h_{ij}^{l+1} = \frac{\sum_{(i,j)\in\nu} a_{ik}h_{kj}^{l}}{\mathrm{d}_{i,i}}$

  + $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ normalized by both neighbor hood size of the current node and that of its current neighbor. The propagation rule is: $h_{ij}^{l+1} = \frac{\sum_{(i,j)\in\nu} a_{ik}h_{kj}^{l}}{\sqrt{\mathrm{d}_{k,k}\mathrm{d}_{i,i}}}$

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

Why GCNs work?

- Laplacian smoothing is a theoritically 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 theoritically 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 theoritically strong method called Laplacian smoothing (Taubin 1995).

$$\hat{\mathbf{y}}_i = (1-\gamma)\mathbf{x}_i + \gamma \sum_j \frac{\tilde{a}_{ij}}{d_i}\mathbf{x}_j \quad ; \quad \hat{Y} = X - \gamma \tilde{D}^{-1}\tilde{L}X = (I - \gamma \tilde{D}^{-1}\tilde{L})X$$

  where $\tilde{L} = \tilde{D} - \tilde{A}$.

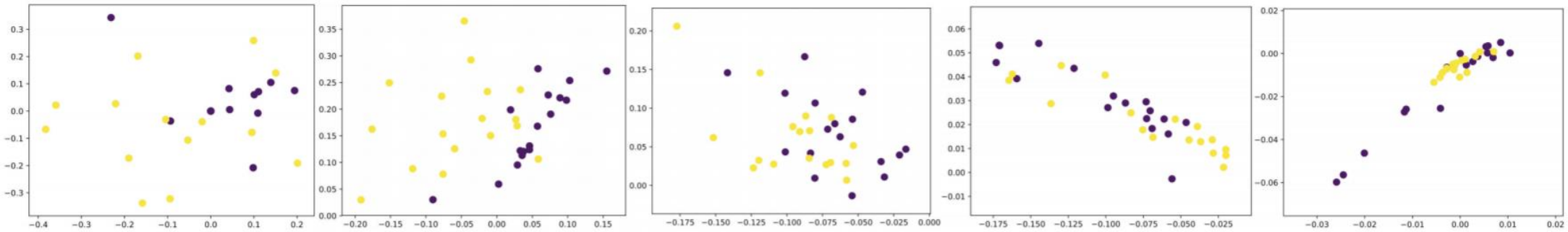
- Letting $\gamma = 1$ and replacing $\tilde{D}^{-1}\tilde{L}$ with $\tilde{D}^{-\frac{1}{2}}\tilde{L}\tilde{D}^{-\frac{1}{2}}$, we have:

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

Which is the propogation rule over a graph of GCNs.

When GCNs fail?

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



(a) 1-layer (b) 2-layer (c) 3-layer (d) 4-layer (e) 5-layer

When GCNs fail?

- Stacking many layers of GCNs can lead to over-smoothing issue.
- Theorem:
  - A graph $\mathcal{G}$ with k connected components $\{C_i\}_{i=1}^{k}$
  - Indication vector for the i-th component $\mathbf{1}^{(i)} \in \mathbb{R}^N$
  - If $\mathcal{G}$ has no bipartile component, then, for any $\mathbf{w} \in \mathbb{R}^N$ and $\alpha \in (0, 1]$, we have:

$$\lim_{m \to +\infty} (I - \alpha L_{rw})^m \mathbf{w} = [\mathbf{1}^{(1)}, \mathbf{1}^{(2)}, \ldots, \mathbf{1}^{(k)}]\theta_1,$$

$$\lim_{m \to +\infty} (I - \alpha L_{sym})^m \mathbf{w} = D^{-\frac{1}{2}}[\mathbf{1}^{(1)}, \mathbf{1}^{(2)}, \ldots, \mathbf{1}^{(k)}]\theta_2$$

Where $\theta_1 \in \mathbb{R}^k, \theta_2 \in \mathbb{R}^k$; and

+ $\{\mathbf{1}^{(i)}\}_{i=1}^{k}$ are the eigenvectors corresponding to eigenvalue 1 of $(I - \alpha L_{rw})$
+ $\{D^{-\frac{1}{2}}\mathbf{1}^{(i)}\}_{i=1}^{k}$ 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
- https://towardsdatascience.com/spectral-graph-convolution-explained-and-implemented-step-by-step-2e495b57f801