A brief introduction to Graph Convolutional Networks

Nhut-Nam Le

Computer Science Department, Information Technology, HCMUS, VNU

October 3, 2021

Table of contents

The motivation

- Convolutional Neural Network which can only operate on regular Euclidean data like images (2D grid) and text (sequences).
- Network Embedding achieved breakthoughs like Word Embeddings, DeepWalk, node2vec, LINE, TADW. However, they still suffer from two drawbacks:
  - No parameters are shared between nodes in the encoder
  - The direct embedding methods lack the ability of generalization

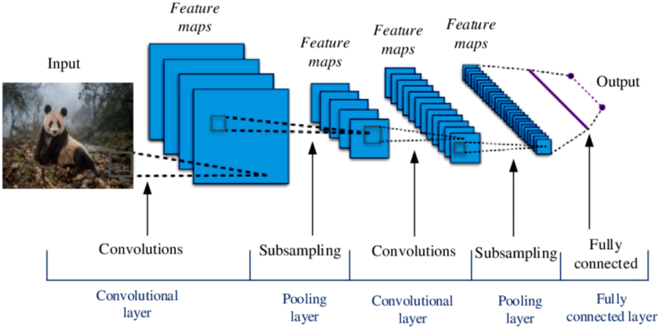
From Image with Convolutional Networks

Convolution operator in Digital Image Processing

$$\text{Output}(x, y) = (K * \text{Input})(x, y) = \sum_m \sum_n \text{Input}(x - m, y - m) K(m, n) \quad (1)$$

Cross-correlation in CNN

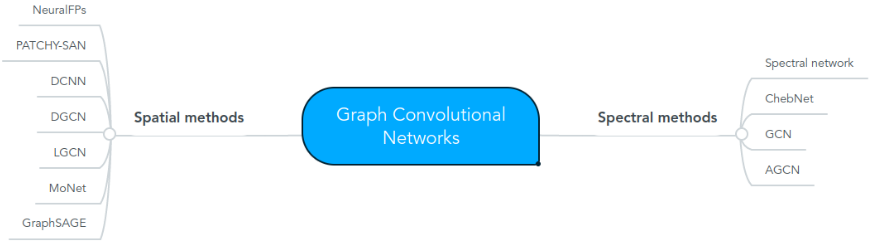
$$\text{Output}(x, y) = (K * \text{Input})(x, y) = \sum_m \sum_n \text{Input}(x + m, y + m) K(m, n) \quad (2)$$



Overview

We can categorize as

- Spectral approaches work with a spectral representation of the graphs, the learned filters depend on the Laplacian eigenbasis, which depends on the graph structure [2]
- Spatial approaches defines convolutions directly on the graph, operating on spatially close neighbors [2]



Problem statement

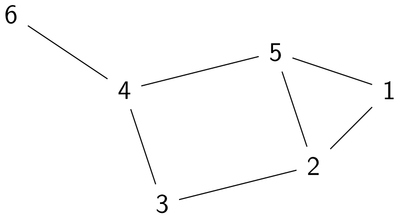
Similar to CNN, GCN is passing a filter over a graph, searching for important vertices and edges that can be used to classify nodes within the graph

Problem statement: Given a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, GCN

- Input: an input feature matrix $N \times F^0$ $\mathbf{X}$, where N is the number of nodes, F^0 is the number of input features; an $N \times N$ matrix representation of graph structure $\mathbf{A}$
- Output: feature of all neighbor nodes for each node, can be used for the next task like classification, link prediction, ...

Methodology

Consider a graph



$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \qquad \mathbf{X} = \begin{pmatrix} 1 & -1 \\ 2 & -2 \\ 3 & -3 \\ 4 & -4 \\ 5 & -5 \\ 6 & -6 \end{pmatrix}$$

Methodology

We can multiple **A** and **X** in order to obtain output

$$\mathbf{A} \times \mathbf{X} = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & -1 \\ 2 & -2 \\ 3 & -3 \\ 4 & -4 \\ 5 & -5 \\ 6 & -6 \end{pmatrix} = \begin{pmatrix} 7 & -7 \\ 9 & -9 \\ 6 & -6 \\ 14 & -14 \\ 7 & -7 \\ 4 & -4 \end{pmatrix}$$

Problem 1: Feature of the node itself

To addressing this problem, add an identity matrix **I** to **A** before perform the multiplication

$$\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_n$$

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

$$\widetilde{\mathbf{A}} \times \mathbf{X} = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & -1 \\ 2 & -2 \\ 3 & -3 \\ 4 & -4 \\ 5 & -5 \\ 6 & -6 \end{pmatrix} = \begin{pmatrix} 8 & -8 \\ 11 & -11 \\ 9 & -9 \\ 18 & -18 \\ 12 & -12 \\ 10 & -10 \end{pmatrix}$$

Problem 2: Normalization

To addressing this problem, we construct degree matrix **D** and use its inverse for multiplication

$$\mathbf{D} = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{D}^{-1} = \begin{pmatrix} \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{D}^{-1}\widetilde{\mathbf{A}}\mathbf{X} = \begin{pmatrix} \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 8 & -8 \\ 11 & -11 \\ 9 & -9 \\ 18 & -18 \\ 12 & -12 \\ 10 & -10 \end{pmatrix} = \begin{pmatrix} 4 & -4 \\ 11/3 & -11/3 \\ 9/2 & 9/2 \\ 6 & -6 \\ 4 & -4 \\ 10 & -10 \end{pmatrix}$$

Apply weights

Consider a simple weight matrix $\mathbf{W} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$

$$\mathbf{D}^{-1}\widetilde{\mathbf{A}}\mathbf{X}\mathbf{W} = \begin{pmatrix} 8 & -8 \\ 11 & -11 \\ 9 & -9 \\ 18 & -18 \\ 12 & -12 \\ 10 & -10 \end{pmatrix} \times \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \begin{pmatrix} -8 \\ -22/3 \\ -9 \\ -12 \\ -8 \\ -20 \end{pmatrix}$$

Finally, like other neural networks, we can apply an activation function σ

$$f(X, A) = \sigma(\mathbf{D}^{-1}\widetilde{\mathbf{A}}\mathbf{X}\mathbf{W})$$

In practice, we use a symmetric normalization $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$

$$f(X, A) = \sigma(\mathbf{D}^{-1/2}\widetilde{\mathbf{A}}\mathbf{D}^{-1/2}\mathbf{X}\mathbf{W})$$

Variations of GCNs

We have some variations of GCNs

- **Attention mechanisms**: capture the neighbor properties of the nodes, remember important nodes, give them higher weights
- **Graph Generative Networks**: similar to Generative Adversarial Network
- **Graph Spatial-Temporal Networks**: support the inputs that change over time

Application of GCNs

- Image Classification
- Community prediction
- Combinatorial Optimization

Conclusion

For this presentation in this week, we introduced some basic about Graph Convolutional Networks

- An overview about Graph Convolutional Approach: **Spectral method** and **Spatial method**
- How Graph Convolutional Networks work?
- Some variations of GCNs
- Applications of GCNs

Q&A

Thanks for your attention with this presentation!
Any question? :)

References

Thomas N. Kipf and Max Welling.
Semi-supervised classification with graph convolutional networks, 2017.

Zhiyuan Liu and Jie Zhou.
Introduction to graph neural networks.
Synthesis Lectures on Artificial Intelligence and Machine Learning, 14:1–127, 03 2020.