A Tutorial on Graph Neural Networks

Graph Convolution, Attention and SAmple and aggreGatE

Zhiming Xu

zhimingxu@smail.nju.edu.cn

THE HONG KONG POLYTECHNIC UNIVERSITY 香港理工大學

Data Exploring & Extracting @ PolyU (DEEP Lab)

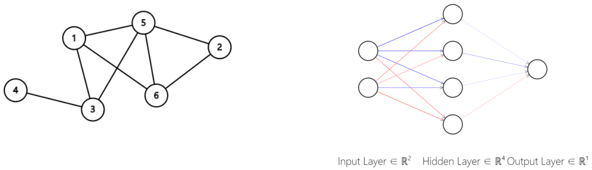
Department of Computing
The Hong Kong Polytechnic University

October 15, 2020

Overview

Recap

- Graph
  A data structure consists of *Vertices*[1] and *Edges*. Denoted by set $\mathcal{V}$ and $\mathcal{E}$, respectively, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.
- Neural Networks
  An interconnected group of neurons performing a series of computations.



(a) A graph with six vertices and eight edges.

(b) A neural network with one hidden layer.

Figure 2: Example of graph and neural network.

[1] The word "node" and "vertex" are used interchangeably in this tutorial.

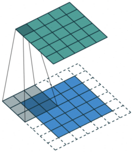
Graph Neural Networks (GNNs)

- A type of neural networks operating directly on graphs [1].
- To learn a state representation which contains information of each vertex's neighborhood.
- Notations in this tutorial

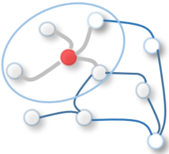
Notation	**Description**
$\mathbb{R}^m$	m-dimensional Euclidean space
$a, \vec{a}, A$	scalar, vector, matrix
A	adjacent matrix
X	(node) feature matrix
D	degree matrix, $D_{ii} = \sum_j A_{ij}$
I_N	N-dimensional identity matrix
$\vec{h}, H$	learned hidden vector, matrix
W	neural network weight matrix
$\sigma, \cdot^\top, \cdot \Vert \cdot$	non-linear activation, transpose, concatenation

Table 1: Notations used in this tutorial

- **Convolution**
  An operation on two functions f and g that produces a third function $f \star g$.
- **Convolutional Neural Network (CNN)**
  Neural networks with the operation of convolution, usually used on images where g is grid and f is called *filter*.
- **Convolution on Graphs**
  Graphs are not as regular as grids. New methods are needed to generalize convolution to them.



(a) An example of 2D convolution.



(b) Convolution on graphs?

Generalize Convolution to Graphs

- Spectral convolutions on graphs with signal $\vec{x} \in \mathbb{R}^n$ in the Fourier domain

$$g_\theta \star \vec{x} = U g_\theta U^\top \vec{x} \tag{1}$$

  where

  1. Normalized graph Laplacian $L = I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U \Lambda U^\top$
  2. U is the matrix of eigenvectors of normalized graph Laplacian
  3. $U^\top \vec{x}$ is the Fourier transformation on $\vec{x}$
  4. g_θ is the spectral convolutional filter. Can be seen as a function $g_\theta(\Lambda)$ on eigenvalues of L

- Equation 1 is computationally expensive and thus needed an efficient approximation.

Approximations

1. K^{th} order Chebyshev polynomial

$$g_{\theta'}(\Lambda) \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{\Lambda}) \tag{2}$$

2. In Equation 1, substitute g_θ with Equation 2

$$g_{\theta'} \star \vec{x} = \sum_{k=0}^{K} \theta'_k T_k(\tilde{L})\vec{x}, \tilde{L} = \frac{2}{\lambda_{\max}} L - I_N \tag{3}$$

3. Limit order K to 1, round λ_{max} to 2, and reduce parameters

$$\begin{aligned} g_{\theta'} \star \vec{x} &\approx \theta'_0 \vec{x} - \theta'_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \vec{x} \\ &\approx \theta \left(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) \vec{x} \end{aligned} \tag{4}$$

Renormalization

- In Equation 4, the $I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ term's eigenvalues are in $[0, 2]$. Stacking layers with this operation might cause vanishing/exploding gradients.
- The renormalization trick is thus introduced to alleviate this problem

$$I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \longrightarrow \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$$

where $\tilde{A} = A + I_N$, $\tilde{D}$ is $\tilde{A}$'s degree matrix

Fast Approximate Graph Convolution

- Generalize to vector signal nodes

$$\theta\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}\right)\vec{x} \longrightarrow \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\Theta$$

- Propagation rule

Multi-layer Graph Convolution Network [2]

$$H^{(l+1)}=\sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right), H^{(0)}=X$$

- Two-layer example (Calculate $\hat{A}=\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ in advance)

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

Goal

- Distill high-dimensional information and reduce it to a dense vector.
- Low-dimensional vector embeddings of nodes in large graphs are very useful in various downstream tasks.

Problem with Using GCNs

- Whole graph is large and computationally prohibitive. Mini-batch is slow to train and hard to converge.
- Full graph is needed, training in a transductive way.
- Difficult to use on real-world *dynamic* graphs.

Graph SAmple and aggreGatE (GraphSAGE)

Sample neighborhood and aggregate the information.

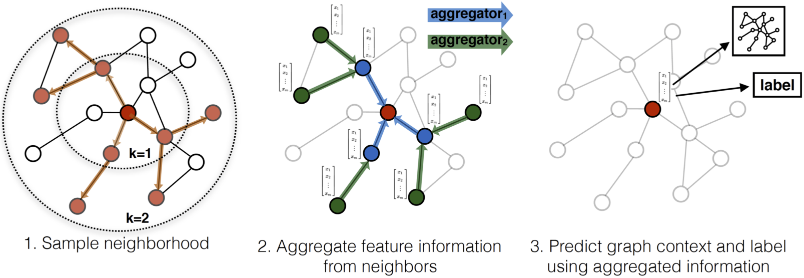


Figure 4: Illustration of GraphSAGE forward propagation.[2]

[2] http://snap.stanford.edu/graphsage/

GraphSAGE Embedding Generation

GraphSAGE Forward Propagation [3]

Result: Node i's representation z_i after K iterations

$\vec{h}_i^0 \leftarrow \vec{h}_i, \forall i \in \mathcal{V}$;

for $k = 1 \dots K$ **do**

  for $i \in \mathcal{V}$ **do**

    $\vec{h}_{N_i}^k \leftarrow \text{AGGREGATE}_k \left(\{ \vec{h}_j, \forall j \in \mathcal{N}_i \} \right)$;

    $\vec{h}_i^k \leftarrow \sigma \left(W^k \cdot \left[\vec{h}_i^{k-1} \| \vec{h}_{\mathcal{N}_i}^k \right] \right)$;

  end

  $\vec{h}_i^k \leftarrow \frac{\vec{h}_i^k}{\|\vec{h}_i^k\|_2}, \forall i \in \mathcal{V}$;

end

$\vec{z}_i \leftarrow \vec{h}_i^K, \forall i \in \mathcal{V}$

Parameter Learning of GraphSAGE

Graph-Based Loss Function [3]

$$L_G(\vec{h}_i) = -\log\left(\sigma\left(\vec{h}_i^\top \vec{h}_j\right)\right) - Q \cdot \left(\mathbb{E}_{v_i \sim P_{n(i)}} \log\left(\sigma\left(-\vec{h}_i^\top \vec{h}_{v_i}\right)\right)\right)$$

- j is a node that co-occurs near i on fixed-length random walk.
- σ is the sigmoid function, $\sigma(x) = \dfrac{1}{1+\exp(-x)}$
- P_n is a negative sampling distribution, Q is # of negative samples.

Based on loss L_G, the parameters in Algorithm 1 are optimized with stochastic gradient descend.

Choice of Aggregator Functions

- Mean Aggregator

$$\vec{h}_i^k \leftarrow \sigma\left(W \cdot \mathrm{MEAN}\left(\left\{\vec{h}_i^{k-1}\right\} \cup \left\{\vec{h}_j^{k-1}, \forall j \in \mathcal{N}_i\right\}\right)\right)$$

- LSTM Aggregator

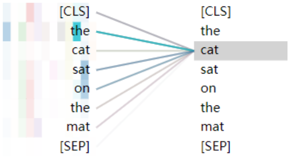
$$\vec{h}_i^k \leftarrow \mathrm{LSTM}\left(\pi\left\{\vec{h}_j, \forall j \in \mathcal{N}_i\right\}\right)$$

- Pooling Aggregator

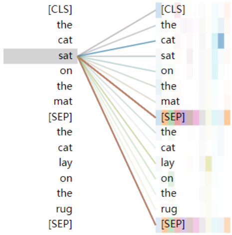
$$\vec{h}_i^k \leftarrow \max\left(\left\{\sigma\left(W_{\mathrm{pool}}\vec{h}_j^k + \vec{b}\right), \forall j \in \mathcal{N}_i\right\}\right)$$

Attention Mechanism

- Attention mechanism achieves great successes in sequence-based tasks.
- They can be used to deal with variable size inputs, and focus on the most relevant parts by assigning different *weights*.
- Attention used on a single sequence is called *self-attention*.



(a) Self-attention



(b) Sentence pair attention

Figure 5: Attention visualization, generated by bertviz

(Self-)Attention Mechanism on Graphs

- Input: Set of node features $H = \left\{ \vec{h}_1, \vec{h}_2, \cdots, \vec{h}_N \right\}, h_i \in \mathbb{R}^d$. N is the number of nodes and d is the feature dimension.
- Output: A new set of node features $H' = \left\{ \vec{h}'_1, \vec{h}'_2, \cdots, \vec{h}'_N \right\}, h'_i \in \mathbb{R}^d$.
- Attention $a : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ with *weight matrix* W

$$e_{ij} = a(W\vec{h}_i, W\vec{h}_j)$$

e_{ij} is called attention coefficients.

Details of Attention a

- Masked: Calculate e_{ij} for $j \in \mathcal{N}_i$, i.e., i's neighborhood.
- Normalized: Use softmax to normalize across all j's

$$\alpha_{ij} = \mathrm{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in N_i} \exp(e_{ik})}$$

- Attention a's implementation

$$\alpha_{ij} = \frac{\exp\left(\mathrm{LeakyReLU}(\vec{a}^\top [W\vec{h}_i || W\vec{h}_j])\right)}{\sum_{k \in N_i} \exp\left(\mathrm{LeakyReLU}(\vec{a}^\top [W\vec{h}_i || W\vec{h}_j])\right)}$$

$$\mathrm{LeakyReLU} = \begin{cases} \alpha \cdot x, & x < 0 \\ x, & x > 0 \end{cases}$$

Attention Acts on Hidden Representations

- Linear combination and activation

$$\vec{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} W \vec{h}_j\right)$$

- *Multi-head attention*
  - Concatenation

$$\vec{h}'_i = \Big\|_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} W^k \vec{h}_j\right)$$

  - Average

$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j \in \mathcal{N}_i} \alpha_{ij} W^k \vec{h}_j\right)$$

Graph Attention Network Propagation Rule and Illustration [4]

$$\vec{h}_i' = \sigma\left(\sum_{j\in N_i} \frac{\exp\left(\text{LeakyReLU}\left(\vec{a}^\top\left[W\vec{h}_i || W\vec{h}_j\right]\right)\right)}{\sum_{k\in N_i}\exp\left(\text{LeakyReLU}\left(\vec{a}^\top\left[W\vec{h}_i || W\vec{h}_j\right]\right)\right)} W\vec{h}_j\right)$$

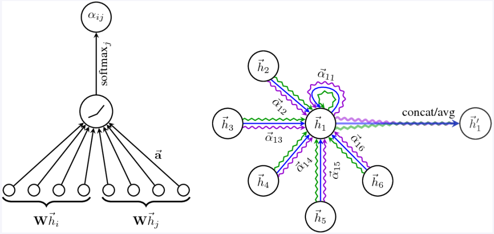


Figure 6: Left: Attention mechanism a. Right: Multi-head attention on a graph.

Thank You for Your Attention

Q & A

References

[1] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009. DOI: 10.1109/TNN.2008.2005605. [Online]. Available: https://doi.org/10.1109/TNN.2008.2005605.

[2] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=SJU4ayYgl.

References (cont.)

[3] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 1024–1034. [Online]. Available: http://papers.nips.cc/paper/6703-inductive-representation-learning-on-large-graphs.

[4] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, OpenReview.net, 2018. [Online]. Available: https://openreview.net/forum?id=rJXMpikCZ.