Deep learning

Graph neural networks[1]

Hamid Beigy

Sharif University of Technology

December 30, 2024

[1] Some slides and figures are taken from Prof. Leskovec's slides

Table of contents

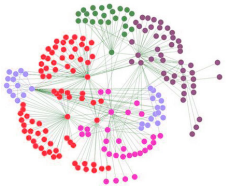
1. Introduction
2. Graph neural networks
3. Graph convolutional networks
4. Reading

Introduction

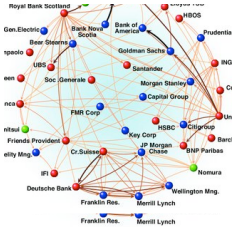
Introduction

1. Networks are a general language for describing and modeling complex systems.
2. Many data are networks such as

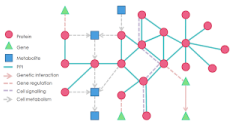
Social networks



Economic networks

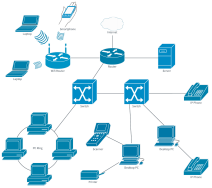


Biological networks

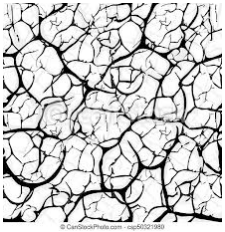


Citation networks

Internet

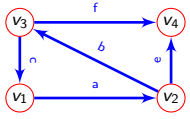


Networks of neurons

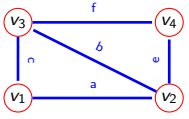


Graphs

1. Graph $G = (V, E)$ is a data structure consisting of two components:
   - the set of vertices/nodes V and
   - and the set of edges E.
2. Edges can be either directed or undirected.

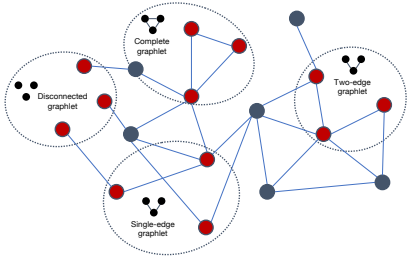


$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$



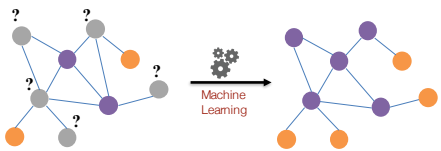
$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Graphlets

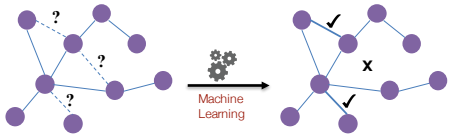


Machine learning tasks on graphs

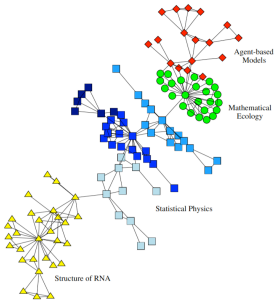
Node classification



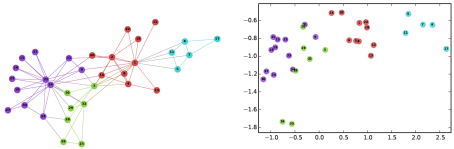
Link prediction



Community detection



Node embedding

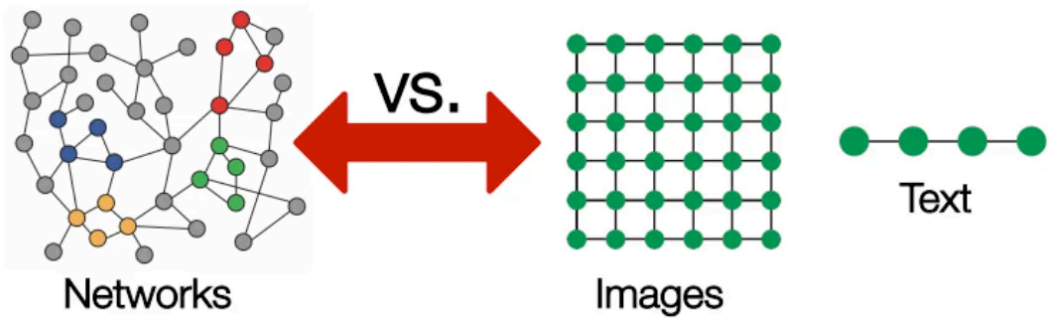


Node embedding

1. Goal is to encode nodes so that similarity in the embedding space (e.g., dot product) approximates similarity in the original network.
2. Let $\mathbf{z}_u$ be the embedding of node u.
3. Goal is to find the encoder function f such that $similarity(u, v) \approx \mathbf{z}_u^\top \mathbf{z}_v$.
4. Learning node embedding
   - Define an encoder
   - Define a node similarity function
   - Optimize the parameters of the encoder so that $similarity(u, v) \approx \mathbf{z}_u^\top \mathbf{z}_v$.
5. Two key components
   - Encoder function $f(u) = \mathbf{z}_u$.
   - Similarity measure $similarity(u, v) \approx \mathbf{z}_u^\top \mathbf{z}_v$.

Why is it hard to analyze a graph?

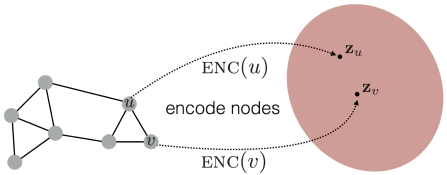
1. Graph data is so complex that it's created a lot of challenges for existing machine learning algorithms.
2. Images with the same structure and size can be considered as fixed-size grid graphs.
3. Text and speech are sequences, so they can be considered as line graphs.
4. Graphs have arbitrary size and complex topological structure.
5. In graphs, there is no fixed node ordering or reference point.
6. Graphs are often dynamic and have multimodal features.



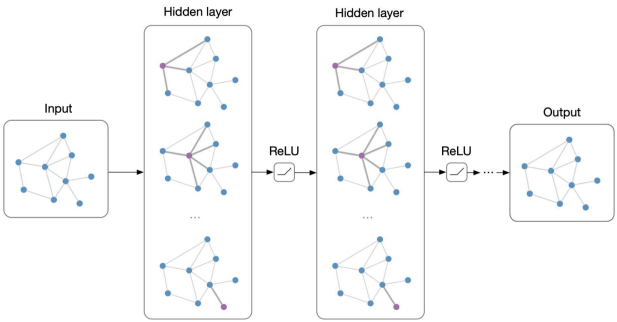
Graph neural networks

Node embedding

1. Goal is to encode nodes so that similarity in the embedding space (e.g., dot product) approximates similarity in the original network.

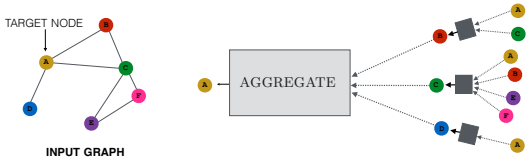


2. Graph neural network is a neural network architecture that learns embeddings of nodes in a graph by looking at its nearby nodes.

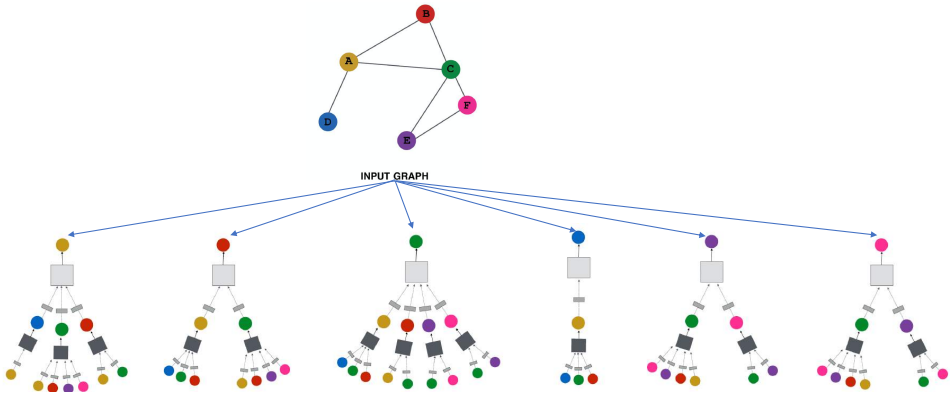


Graph neural networks (GNNs)

1. The idea is to generate node embeddings based on local neighborhoods.
2. The intuition is nodes aggregate information from their neighbors using neural networks.

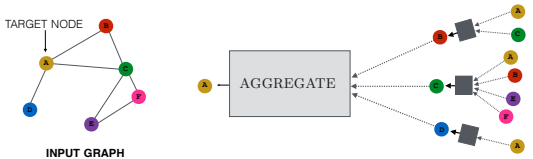


3. Network neighborhood defines a computation graph.



Neural message passing

1. GNN uses a form of neural message passing in which vector messages are exchanged between nodes and updated using neural networks.
2. During each message-passing iteration, a hidden embedding $\mathbf{h}_u^k$ corresponding to each node $u \in U$ is updated according to information aggregated from its neighborhood $N(u)$.



3. This message-passing update can be expressed as follows:

$$\begin{aligned}\mathbf{h}_u^{k+1} &= Update^k\left(\mathbf{h}_u^k, Aggregate(\mathbf{h}_v^k \mid \forall v \in N(u))\right) \\ &= Update^k\left(\mathbf{h}_u^k, \mathbf{m}_{N(u)}^k\right)\end{aligned}$$

   where

   - *Update* and *Aggregate* are arbitrary differentiable functions and
   - $\mathbf{m}_{N(u)}^k$ is the message aggregated from neighborhoods of u.

4. The initial embeddings at $k = 0$ are set to the input features for all the nodes, i.e.,

$$\mathbf{h}_u^{(0)} = \mathbf{x}_u \qquad \forall u \in V$$

The basic GNN

1. The basic GNN message passing is defined as

$$\mathbf{h}_u^k = \sigma\left(\mathbf{W}_{self}^k \mathbf{h}_u^{k-1} + \mathbf{W}_{neigh}^k \sum_{v \in N(u)} \mathbf{h}_v^{k-1} + \mathbf{b}^k\right).$$

2. As a simplification of the neural message passing approach, it is common to add self-loops to the input graph and omit the explicit update step.

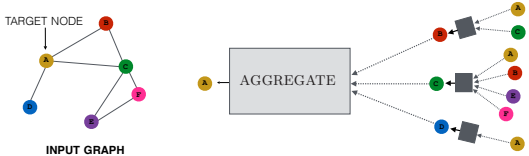
$$\mathbf{h}_u^k = Aggregate\left(\left\{\mathbf{h}_v^{k-1} \mid \forall v \in N(u) \cup \{u\}\right\}\right)$$

3. A benefit of this approach is that we no longer need to define an explicit update function.
4. Simplifying message passing in this way limits the expressiveness of GNN, because we can't distinguish the information coming from neighboring nodes from the node itself.
5. Adding self-loops is equivalent to sharing parameters between $\mathbf{W}_{self}^k$ and $\mathbf{W}_{neigh}^k$ matrices.

$$\mathbf{H}^t = \sigma\left((\mathbf{A} + \mathbf{I})\mathbf{H}^{t-1}\mathbf{W}^t\right)$$

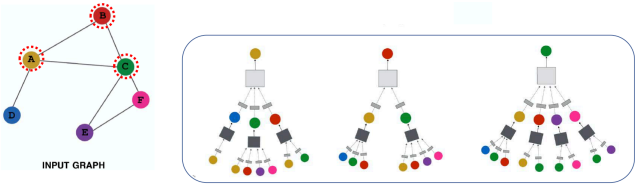
Training GNNs

1. How do we train the model to generate high-quality embeddings?



2. We need to define a loss function on the embeddings, $\ell(\mathbf{z}_A)$.

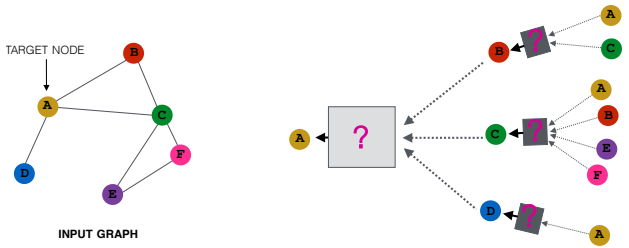
3. Train on a set of nodes, i.e., a batch of compute graphs.



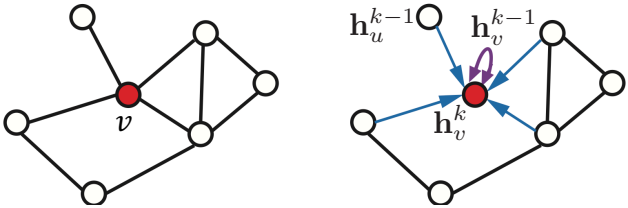
Graph convolutional networks

Graph convolutional networks (GCNs)

1. In GNN, we have aggregated the neighbor messages by taking their weighted average. Can we do better?



2. Any differentiable function that maps set of vectors in $N(u)$ to a single vector can be used as the *Aggregate* function.



3. GCN defines the message passing function as

$$\mathbf{h}_v^k = \sigma\left(\left[\mathbf{W}_k.Aggregate\left(\{\mathbf{h}_u^{k-1} \mid u \in N(v)\}\right), \mathbf{B}_k\mathbf{h}_v^{k-1}\right]\right)$$

Neighborhood aggregation

1. The basic GNN can be improved upon and generalized in many ways: improving *Aggregate* and *Update* functions.
2. GNN has the two following limitations:
   - Multiplication with $\mathbf{A}$ means that, for every node, we sum up all the feature vectors of all neighboring nodes.
   - $\mathbf{A}$ is typically not normalized and therefore the multiplication with $\mathbf{A}$ will completely change the scale of feature vectors.
3. The most basic neighborhood aggregation operation takes sum of neighboring embeddings.
4. One issue is that it can be unstable and highly sensitive to node degrees.
5. Let $|N(u)| \gg |N(v)|$, then we would reasonably expect that

$$\left\| \sum_{u' \in N(u)} \mathbf{h}'_u \right\| \gg \left\| \sum_{v' \in N(v)} \mathbf{h}'_v \right\|$$

   for any reasonable vector norm.
6. This drastic difference in magnitude can lead to numerical instabilities as well as difficulties for optimization.

Neighborhood aggregation

1. This drastic difference in magnitude can lead to numerical instabilities as well as difficulties for optimization.
2. One solution is to normalize the aggregation operation based upon the degrees of the given node.
3. The simplest approach is to just take an average rather than sum

$$\mathbf{m}_{N(u)} = \frac{\sum_{v \in N(u)} \mathbf{h}_v}{|N(u)|}$$

4. Other normalization is symmetric normalization.

$$\mathbf{m}_{N(u)} = \sum_{v \in N(u)} \frac{\mathbf{h}_v}{\sqrt{|N(u)||N(v)|}}$$

5. GCN employs the symmetric-normalized aggregation as well as the self-loop update approach.
6. GCN defines the message passing function as

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}^k \sum_{u \in N(v) \cup \{v\}} \frac{\mathbf{h}_u}{\sqrt{|N(u)||N(v)|}}\right)$$

Neighborhood aggregation

1. Simple neighborhood aggregation

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}^k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right)$$

2. GraphSAGE concatenates neighbor embedding and self-embedding.

$$\mathbf{h}_v^k = \sigma\left(\left[\mathbf{W}_k . Aggregate\left(\{\mathbf{h}_u^{k-1} \mid u \in N(v)\}\right), \mathbf{B}_k \mathbf{h}_v^{k-1}\right]\right)$$

3. Pool aggregate function transforms neighbor vectors and apply symmetric vector function.

$$Aggregate = \gamma\left(\{\mathbf{Q}\mathbf{h}_u^{k-1} \mid u \in N(v)\}\right)$$

   where γ is element-wise min / max.

4. LSTM aggregate function applies LSTM to the reshuffles neighbors.

$$Aggregate = LSTM\left(\left[\mathbf{h}_u^{k-1} \mid u \in \pi(N(v))\right]\right)$$

5. Many aggregations can be performed efficiently by (sparse) matrix operations.

Reading

Readings

1. Chapter 5 of Graph Representation Learning[2].
2. Paper A Comprehensive Survey on Graph Neural Networks[3].
3. Paper Deep Learning on Graphs: A Survey[4].

[2] William L. Hamilton (2020). *Graph Representation Learning*. Morgan and Claypool.
[3] Zonghan Wu et al. (2021). "A Comprehensive Survey on Graph Neural Networks". In: *IEEE Trans. Neural Networks Learn. Syst.* 32.1, pp. 4–24.
[4] Ziwei Zhang, Peng Cui, and Wenwu Zhu (2018). "Deep Learning on Graphs: A Survey". In: *CoRR* abs/1812.04202. URL: http://arxiv.org/abs/1812.04202.

References i

- Hamilton, William L. (2020). *Graph Representation Learning*. Morgan and Claypool.
- Wu, Zonghan et al. (2021). "A Comprehensive Survey on Graph Neural Networks". In: *IEEE Trans. Neural Networks Learn. Syst.* 32.1, pp. 4–24.
- Zhang, Ziwei, Peng Cui, and Wenwu Zhu (2018). "Deep Learning on Graphs: A Survey". In: *CoRR* abs/1812.04202. URL: http://arxiv.org/abs/1812.04202.

Questions?