Graph Neural Networks

Lecture 6

Danila Biktimirov
Applied Computer Science
Neapolis University Pafos

21 March 2025

Overview

1. **Previously on...**

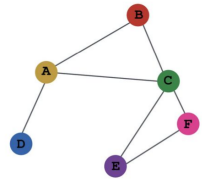
2. **Architectures**

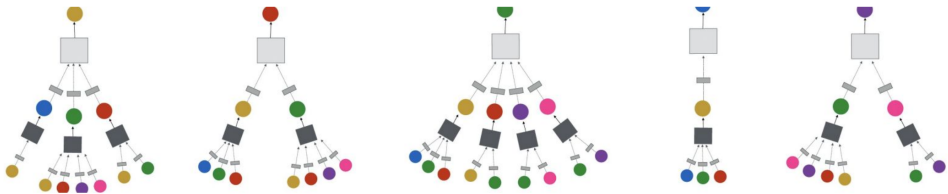
Previously on...

Recap: Label Propagation

- Information propagation in a semi-supervised format
- State updates for each unlabeled node until convergence
- Convergence is not guaranteed

Recap: Graph Convolutions





Recap: Deep Encoder of a GCN

Basic approach: Average neighbor messages and apply a neural network

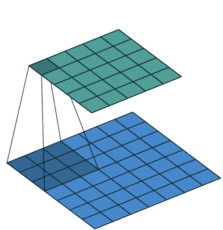
- Initial 0-th layer embeddings are equal to node features: $h_v^0 = x_v$
- Update rule:

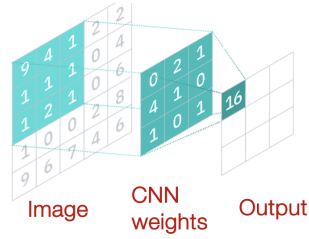
$$h_v^{(k+1)} = \sigma\left(W_k \sum_{u \in N(v)} \frac{h_u^{(k)}}{|N(v)|} + B_k h_v^{(k)}\right), \quad \forall k \in \{0, ..., K-1\}$$

- **Final embedding:** $z_v = h_v^{(K)}$

Convolutional Neural Network

Convolutional neural network (CNN) layer with 3x3 filter:





CNN formulation:

$$h_v^{(l+1)} = \sigma\left(\sum_{u \in N(v) \cup \{v\}} W_l^u h_u^{(l)}\right), \quad \forall l \in \{0, \ldots, L-1\}$$

GNN vs. CNN

GNN formulation:

$$h_v^{(l+1)} = \sigma\left(\mathbf{W_l} \sum_{u \in N(v)} \frac{h_u^{(l)}}{|N(v)|} + B_l h_v^{(l)}\right), \quad \forall l \in \{0, \ldots, L-1\}$$

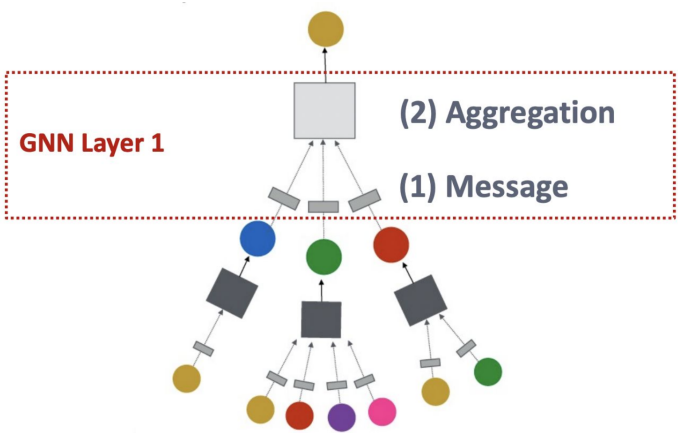
CNN formulation:

$$h_v^{(l+1)} = \sigma\left(\sum_{u \in N(v)} \mathbf{W_l^u} h_u^{(l)} + B_l h_v^{(l)}\right)$$

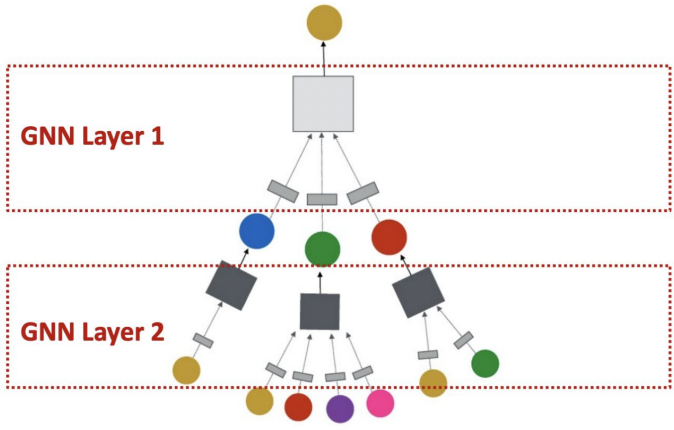
GNN/Transformer

Fully connected text graph

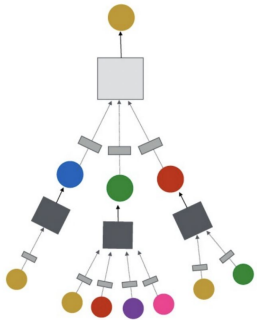
Parts of GNN



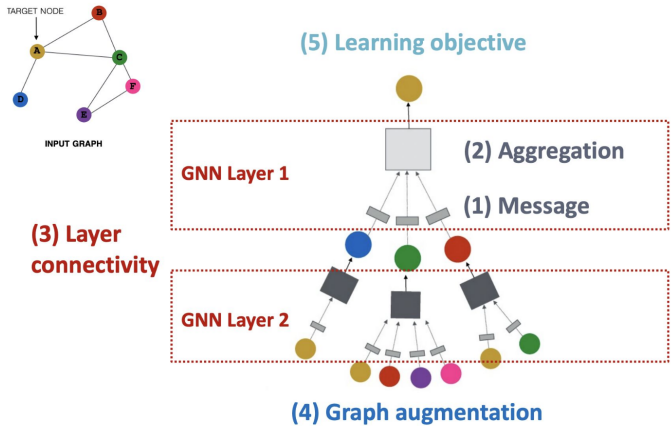
Parts of GNN



Parts of GNN



Parts of GNN


TARGET NODE
B
A
C
F
D
E
INPUT GRAPH
(5) Learning objective
(2) Aggregation
GNN Layer 1
(1) Message
(3) Layer connectivity
GNN Layer 2
(4) Graph augmentation

Message

$$\mathbf{m}_u^{(l)} = \mathrm{MSG}^{(l)}\left(\mathbf{h}_u^{(l-1)}\right)$$

Aggregation

$$\mathbf{h}_v^{(l)} = \mathrm{AGG}^{(l)}\left(\left\{\mathbf{m}_u^{(l)}, u \in N(v)\right\}\right)$$

Problems with the First Two Steps

The information at the top node might be lost. It must be preserved.

1)

$$\mathbf{m}_u^{(l)} = \mathbf{W}^{(l)}\mathbf{h}_u^{(l-1)}$$

$$\mathbf{m}_v^{(l)} = \mathbf{B}^{(l)}\mathbf{h}_v^{(l-1)}$$

2)

$$\mathbf{h}_v^{(l)} = \text{CONCAT}\left(\text{AGG}\left(\left\{\mathbf{m}_u^{(l)}, u \in N(v)\right\}\right), \mathbf{m}_v^{(l)}\right)$$

GCN (Graph Convolutional Network)

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

$$\mathbf{m}_u^{(l)} = \frac{1}{|N(v)|}\mathbf{W}^{(l)}\mathbf{h}_u^{(l-1)}$$

$$\mathbf{h}_v^{(l)} = \sigma\left(\text{Sum}\left(\left\{\mathbf{m}_u^{(l)}, u \in N(v)\right\}\right)\right)$$

GraphSAGE

$$\mathbf{h}_v^{(l)} = \sigma\left(\mathbf{W}^{(l)} \cdot \text{CONCAT}\left(\mathbf{h}_v^{(l-1)}, \text{AGG}\left(\left\{\mathbf{h}_u^{(l-1)}, \forall u \in N(v)\right\}\right)\right)\right)$$

1) Inside AGG

$$\mathbf{h}_{N(v)}^{(l)} \leftarrow \text{AGG}\left(\left\{\mathbf{h}_u^{(l-1)}, \forall u \in N(v)\right\}\right)$$

2) Full update

$$\mathbf{h}_v^{(l)} \leftarrow \sigma\left(\mathbf{W}^{(l)} \cdot \text{CONCAT}\left(\mathbf{h}_v^{(l-1)}, \mathbf{h}_{N(v)}^{(l)}\right)\right)$$

GraphSAGE AGG

$$\text{AGG} = \sum_{u \in N(v)} \frac{\mathbf{h}_u^{(l-1)}}{|N(v)|}$$

$$\text{AGG} = \text{Mean}\left(\left\{\text{MLP}(\mathbf{h}_u^{(l-1)}), \forall u \in N(v)\right\}\right)$$

$$\text{AGG} = \text{LSTM}\left(\left[\mathbf{h}_u^{(l-1)}, \forall u \in \pi(N(v))\right]\right)$$

L2 Normalization

$$\mathbf{h}_v^{(l)} \leftarrow \frac{\mathbf{h}_v^{(l)}}{\|\mathbf{h}_v^{(l)}\|_2} \quad \forall v \in V, \quad \text{where } \|\mathbf{u}\|_2 = \sqrt{\sum_i u_i^2} \quad (\ell_2\text{-norm})$$

GAT (Graph Attention Network)

$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \alpha_{vu} \mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)}\right)$$

$$\alpha_{vu} = \frac{1}{|N(v)|}$$

Issues:

- In GCN/GraphSAGE, α_{vu} is uniform.
- The problem: all nodes are equally important.

Attention Score Calculation

$$e_{vu} = a\left(\mathbf{W}^{(l)}\mathbf{h}_u^{(l-1)}, \mathbf{W}^{(l)}\mathbf{h}_v^{(l-1)}\right)$$

$$e_{AB} = a\left(\mathbf{W}^{(l)}\mathbf{h}_A^{(l-1)}, \mathbf{W}^{(l)}\mathbf{h}_B^{(l-1)}\right)$$

Attention Score Computation

$$\alpha_{vu} = \frac{\exp(e_{vu})}{\sum_{k \in N(v)} \exp(e_{vk})}$$

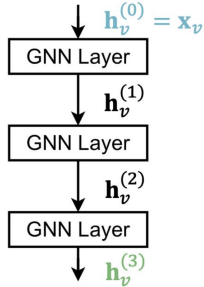
Weighted sum based on the **final attention weight**

$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \alpha_{vu} \mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)}\right)$$

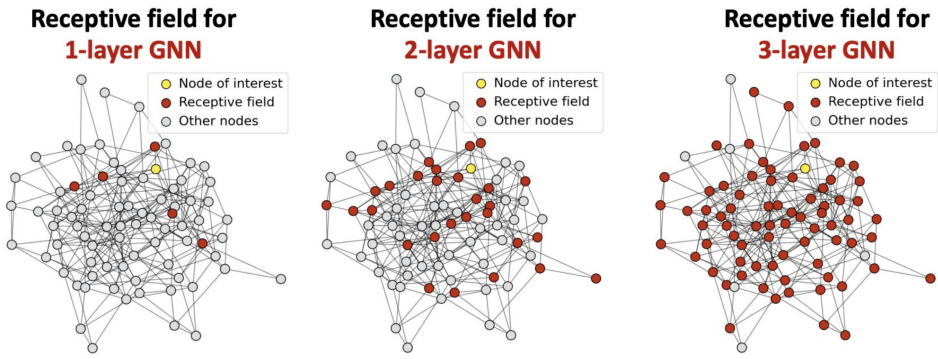
Weighted sum using α_{AB}, α_{AC}, α_{AD}:

$$\mathbf{h}_A^{(l)} = \sigma\left(\alpha_{AB}\mathbf{W}^{(l)}\mathbf{h}_B^{(l-1)} + \alpha_{AC}\mathbf{W}^{(l)}\mathbf{h}_C^{(l-1)} + \alpha_{AD}\mathbf{W}^{(l)}\mathbf{h}_D^{(l-1)}\right)$$

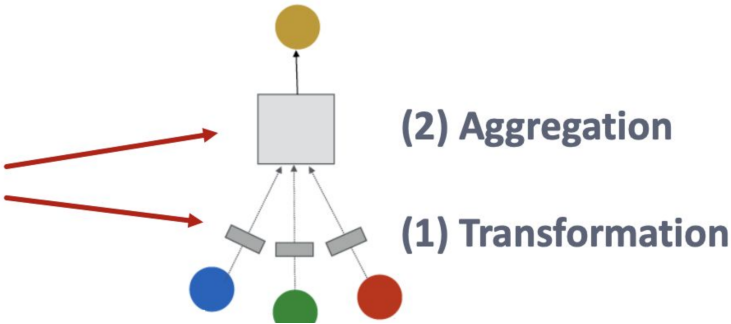
Sequential Processing



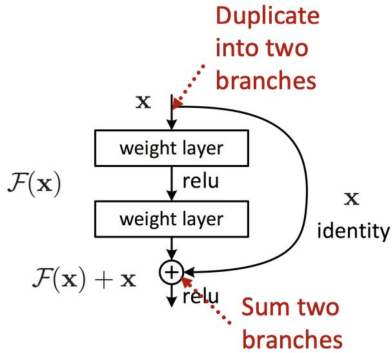
Receptive Field



Solution 1



Skip-Connections



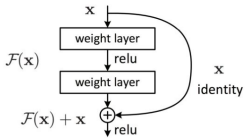
Example

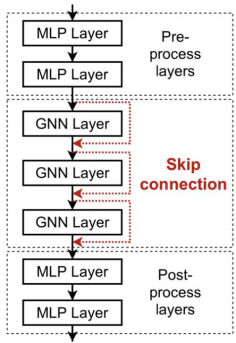
- **A standard GCN layer**
- $\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \mathbf{W}^{(l)} \frac{\mathbf{h}_u^{(l-1)}}{|N(v)|}\right)$

This is our $F(\mathbf{x})$

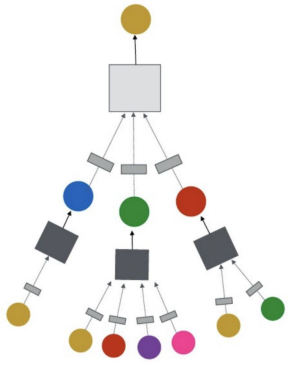
- **A GCN layer with skip connection**
- $\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \mathbf{W}^{(l)} \frac{\mathbf{h}_u^{(l-1)}}{|N(v)|} + \mathbf{h}_v^{(l-1)}\right)$

$F(\mathbf{x})$ + $\mathbf{x}$





Graph Augmentation



(4) Graph augmentation

Problems

- No features
- Sparse
- Dense
- Large

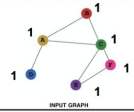
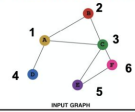
Solutions

- No features - augmentation
- Sparse - add connections
- Dense - use sampling
- Large - sample subgraphs

No Features

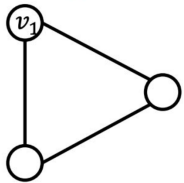
1. Add constant features to all nodes
2. Add node identifiers

Comparison

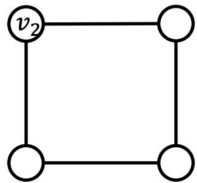
	Constant node feature	One-hot node feature
		
Expressive power	**Medium**. All the nodes are identical, but GNN can still learn from the graph structure	**High**. Each node has a unique ID, so node-specific information can be stored
Inductive learning (Generalize to unseen nodes)	**High**. Simple to generalize to new nodes: we assign constant feature to them, then apply our GNN	**Low**. Cannot generalize to new nodes: new nodes introduce new IDs, GNN doesn't know how to embed unseen IDs
Computational cost	**Low**. Only 1 dimensional feature	**High**. $O(\|V\|)$ dimensional feature, cannot apply to large graphs
Use cases	**Any graph, inductive settings** (generalize to new nodes)	**Small graph, transductive settings** (no new nodes)

Cycles

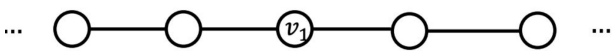
v_1 resides in a cycle with length 3



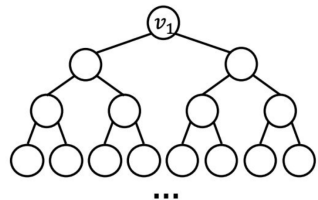
v_1 resides in a cycle with length 4



v_1 resides in a cycle with infinite length



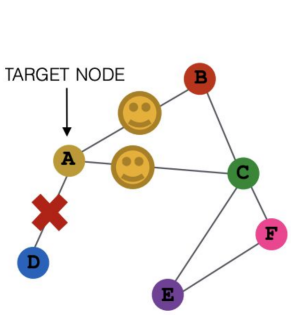
The computational graphs for node v_1 are always the same

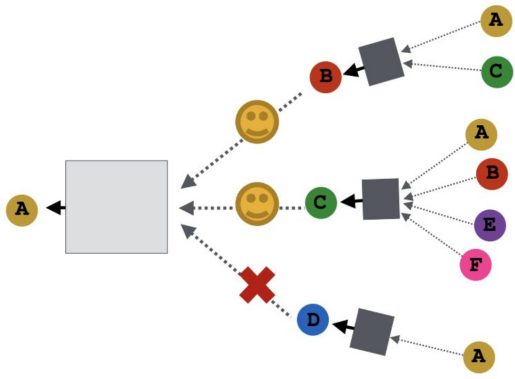


Adding Nodes/Edges

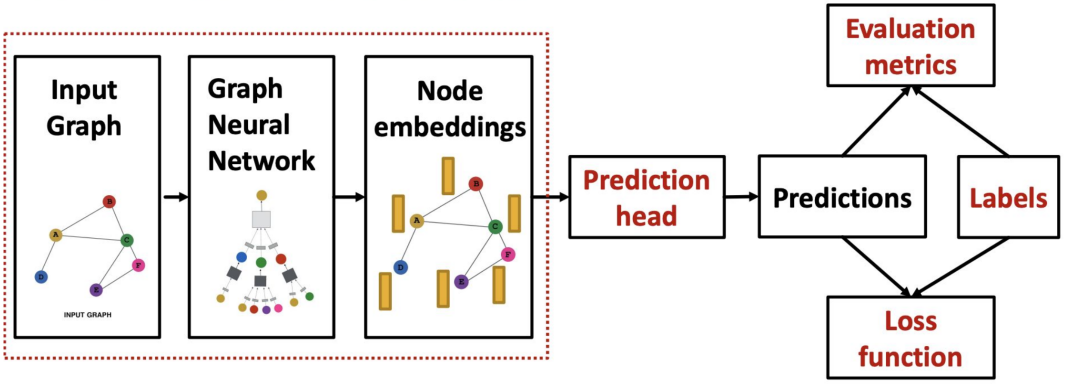
- Distance Reduction

Neighbor Sampling





Training



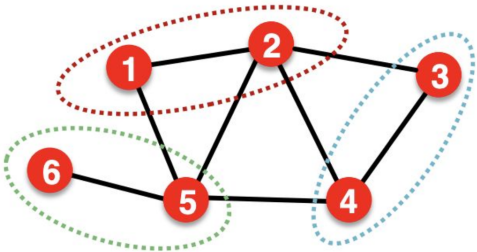
Output of a GNN: set of node embeddings

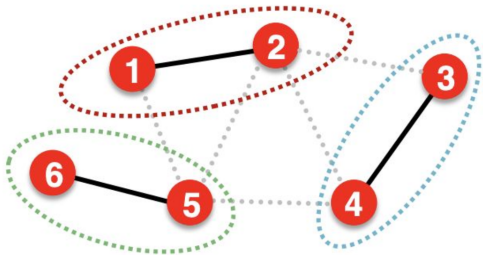
Hierarchical Global Pooling

$$\{-1, -2, 0, 1, 2\}$$

$$\{-10, -20, 0, 10, 20\}$$

Splitting





The End?