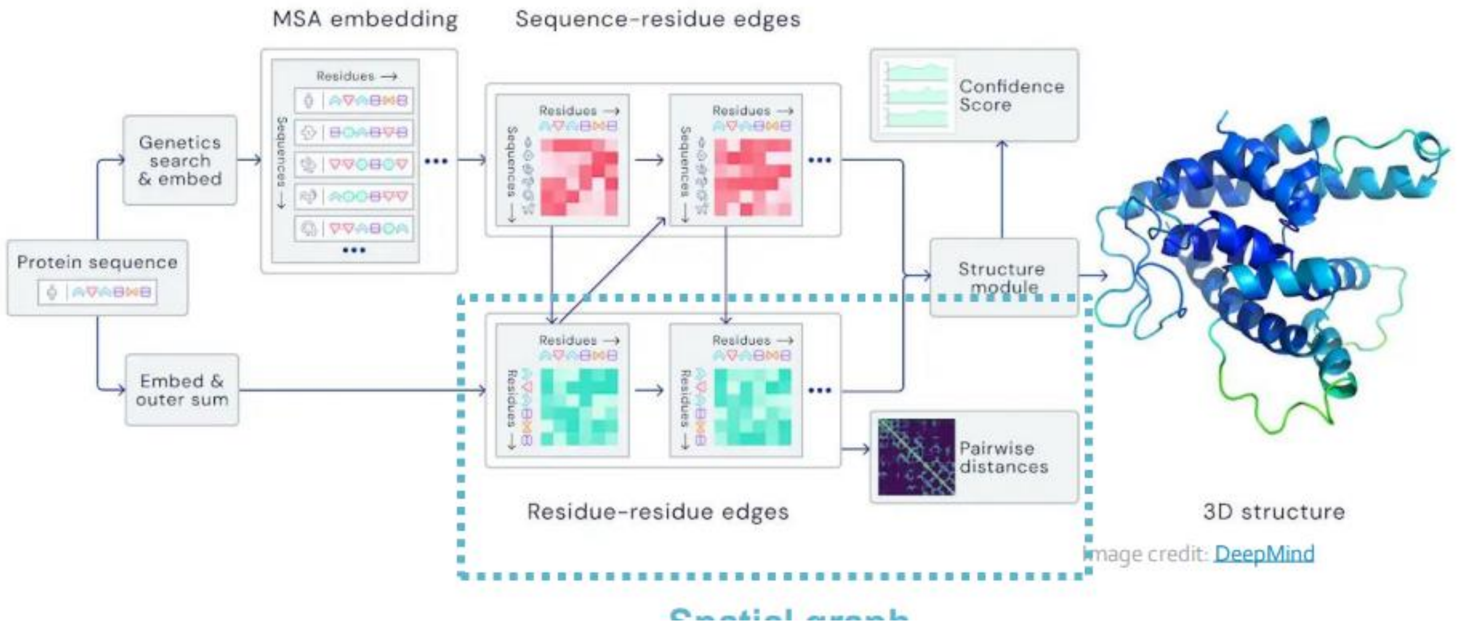
Deep Graph Networks

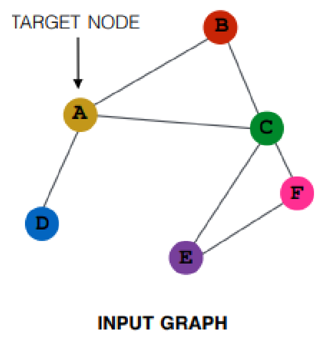


Slides from Jure Leskovec

ECE324, Winter 2022

Michael Guerzhoy

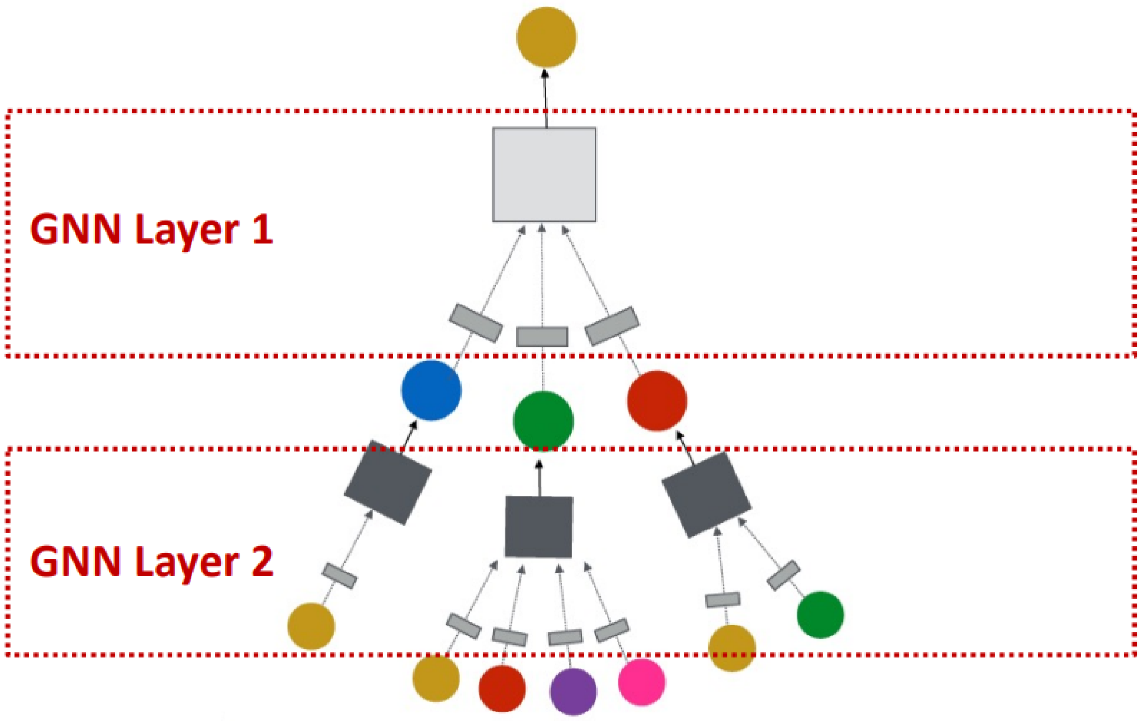
General GNN Framework



Connect GNN layers into a GNN

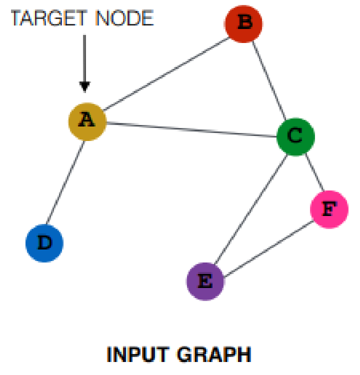
- **Stack layers sequentially**
- **Ways of adding skip connections**

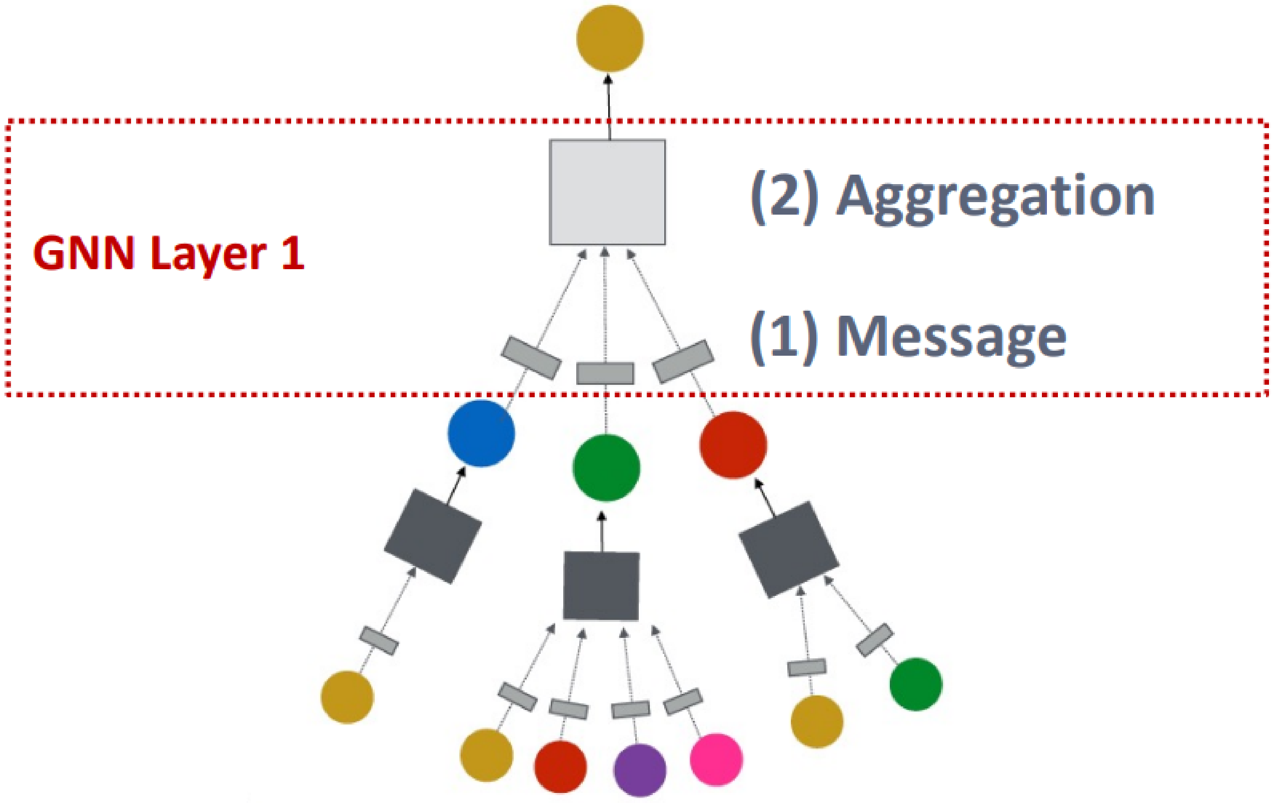
(3) Layer connectivity



GNN Layer = Message + Aggregation

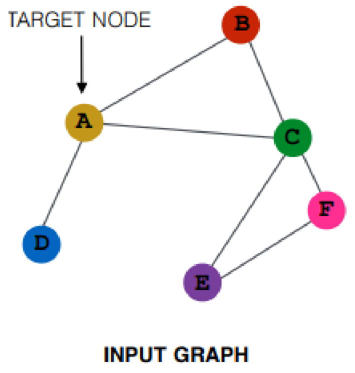
- Different instantiations under this perspective
- GCN, GraphSAGE, GAT, …

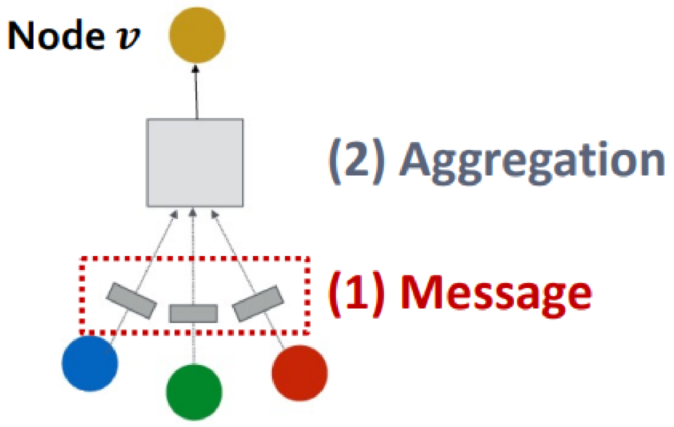




Message Computation

- Message function: $m_u^{(l)} = MSG^{(l)}\left(h_u^{(l-1)}\right)$
  - **Intuition**: each node will create a message, which will be sent to other nodes later
  - **Example:** a linear layer $m_u^{(l)} = W^{(l)} h_u^{(l-1)}$



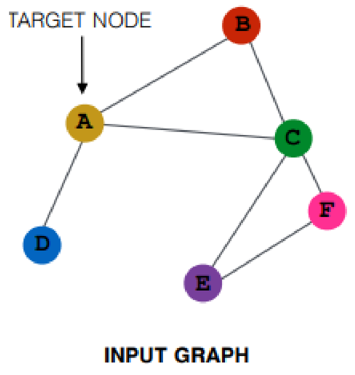


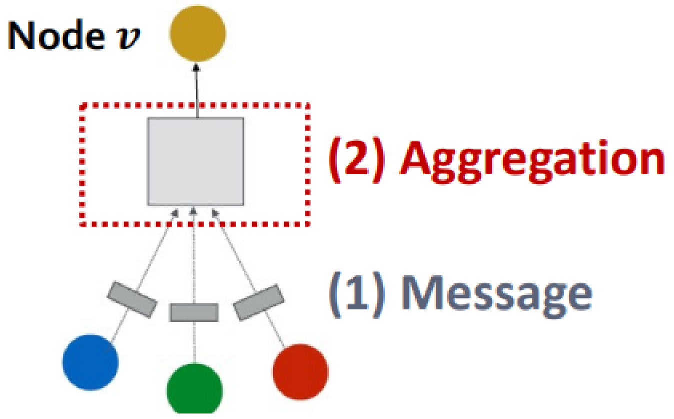
Aggregation

- **Intuition:** each node will aggregate the messages from node *v*'s neighbours

$$h_v^{(l)} = AGG^{(l)}\left(\left\{m_u^{(l)}, u \in N(b)\right\}\right)$$

- **Example aggregation functions:** sum, mean, max
  - $h_v^{(l)} = sum\left(\left\{m_u^{(l)}, u \in N(v)\right\}\right)$





Message Aggregation: Issue

- Want $h_v^{(l)}$ to encode information about *v*
- Options:
  - Include $h_v^{(l-1)}$ when computing $h_v^{(l)}$
    - Compute a message from node *v* to itself

    $m_u^{(l)} = W^{(l)} h_u^{(l-1)} \quad m_v^{(l)} = B^{(l)} h_v^{(l-1)}$
  - After aggregating from neighbours, aggregate the message from node *v* to itself, via concatenation or summation
    - $h_v^{(l)} = CONCAT\left(AGG\left(\left\{m_u^{(l)}, u \in N(v)\right\}\right), m_v^{(l)}\right)$

Putting things together: one layer

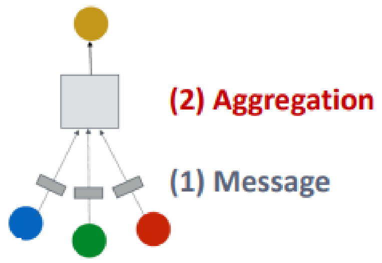
- Message: each node computes a message

$$\mathbf{m}_u^{(l)} = \mathrm{MSG}^{(l)}\left(\mathbf{h}_u^{(l-1)}\right), u \in \{N(v) \cup v\}$$

- Aggregation: aggregate messages from neighbours

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

- Apply nonlinearity



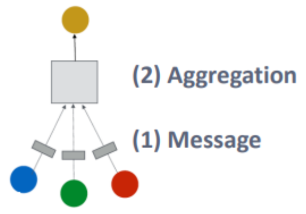
Graph Convolutional Networks

- Graph Convolution Network layer

Message

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

Aggregation



- Message from each neighbour: $\mathbf{m}_u^{(l)} = \frac{1}{|N(v)|} \mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)}$
- Aggregation: sum, then apply activation

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

In GCN graph is assumed to have self-edges that are included in the summation.

GraphSAGE (SAmple and AgreggatE)

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

- Message is computed within AGG()
- Two-stage aggregation:
  - Stage 1: aggregate from node neighbours

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

  - Stage 2: further aggregate over the node itself

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

GraphSAGE Neighbour Aggregation

- **Mean:** take a weighted average of neighbours

$$\mathrm{AGG} = \underbrace{\sum_{u\in N(v)}}_{\text{Aggregation}} \underbrace{\frac{\mathbf{h}_u^{(l-1)}}{|N(v)|}}_{\text{Message computation}}$$

- **Pool**: transform neighbour vectors and apply symmetric functions like Mean or Max

$$\mathrm{AGG} = \underbrace{\mathrm{Mean}}_{\text{Aggregation}}(\{\underbrace{\mathrm{MLP}}_{\text{Message computation}}(\mathbf{h}_u^{(l-1)}), \forall u \in N(v)\})$$

- **LSTM:** Apply LSTM to reshuffled neighbours

$$\mathrm{AGG} = \underbrace{\mathrm{LSTM}}_{\text{Aggregation}}([\mathbf{h}_u^{(l-1)}, \forall u \in \pi\big(N(v)\big)])$$

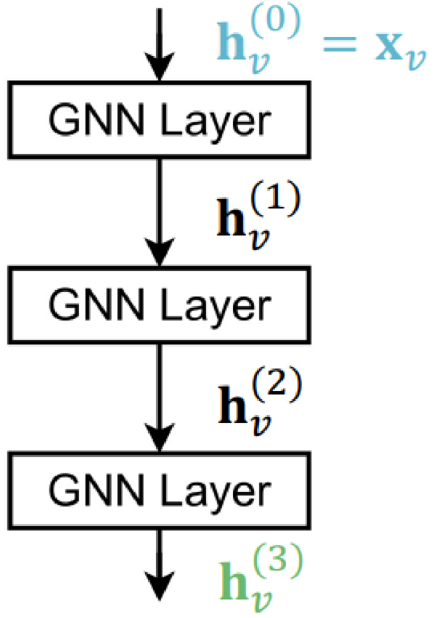
Graph Attention Networks (GAT)

$$\mathbf{h}_v^{(l)} = \sigma(\textstyle\sum_{u \in N(v)} \underbrace{\boxed{\alpha_{vu}}}_{\text{Attention weights}} \mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)})$$

- Can try $\alpha_{vu} = \frac{1}{|N(v)|}$ is the weighting factor of node *u's* message to node *v*
- Can learn an attention function:
  - $\alpha_{vu} = softmax(W_a[W^{(l)} h_u^{(l-1)}\ W^{(l)} h_v^{(l-1)}])$

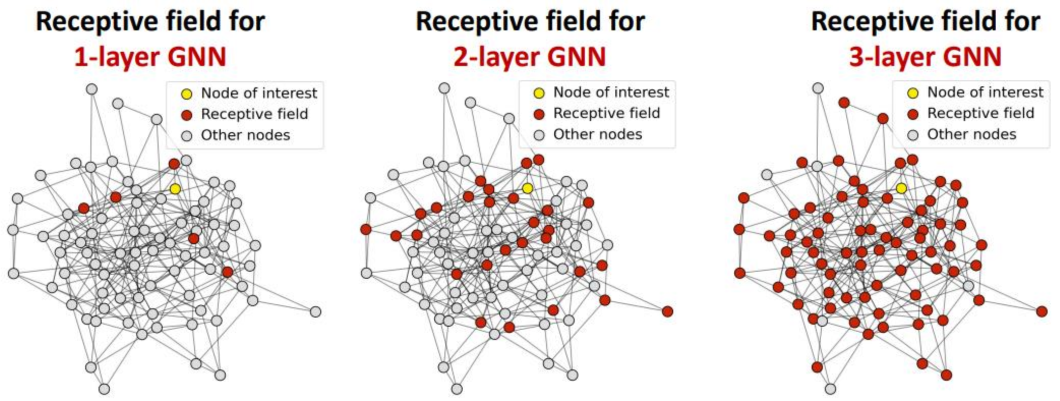
Stacking GNN layers

- Stack GNN layers sequentially
- Input: initial raw node feature x_v
- Output: node embeddings $h_v^{(l)}$ after *L* GNN layers



Oversmoothing Problem

- GNN suffers from over smoothing problem
  - All the node embeddings converge to the same value
    - Bad because we want to use node embeddings to differentiate nodes
- In a K-layer GNN, the receptive field of *v* (all the nodes that determine the value of $h(v)$) is the K-hop neighbourhood of *v*



GNN design

- Use fewer GNN layers
- Can make message/aggregation functions be deep networks
- Can add skip connections so that close neighbours are emphasized when computing *h(v)*