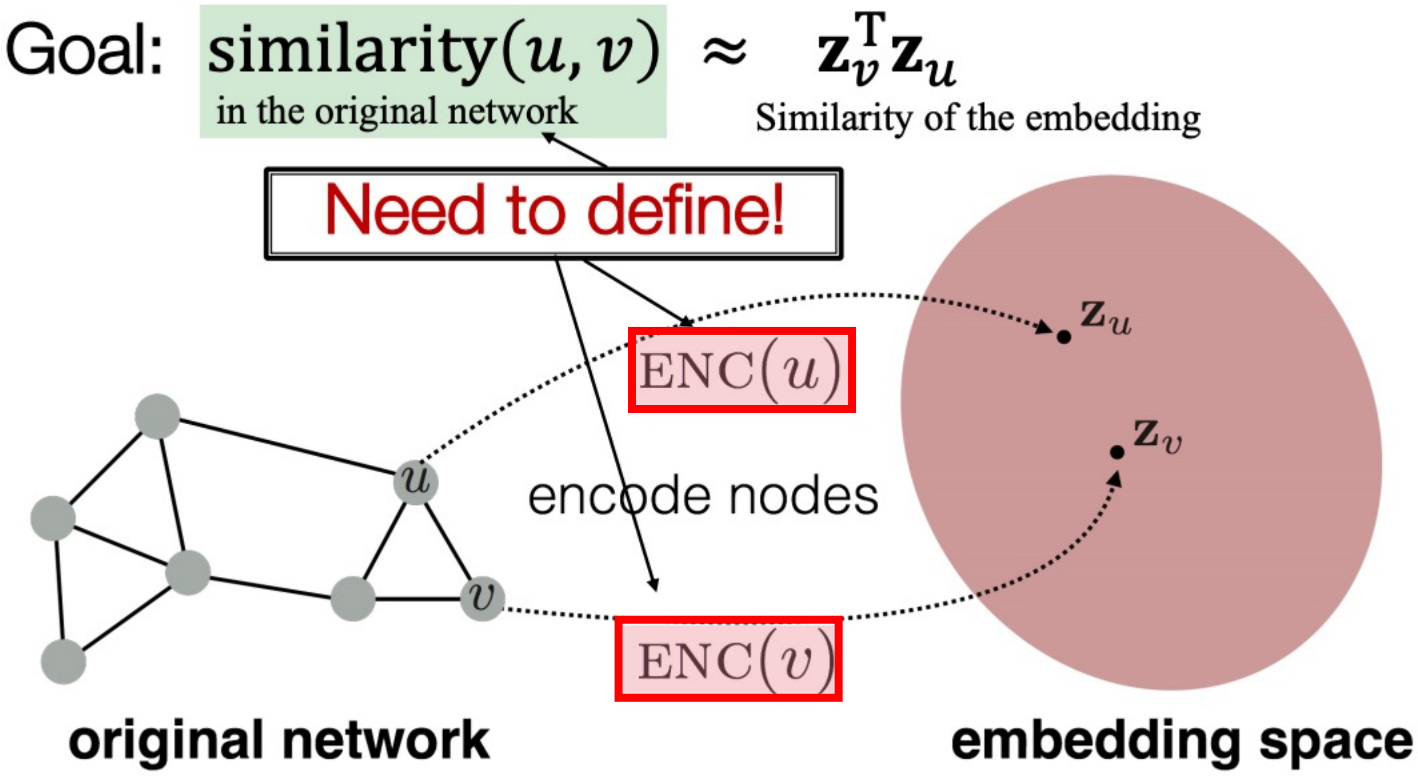
# Graph Neural Networks

Neural Networks Design And Application

# Encoder-decoder for graph data



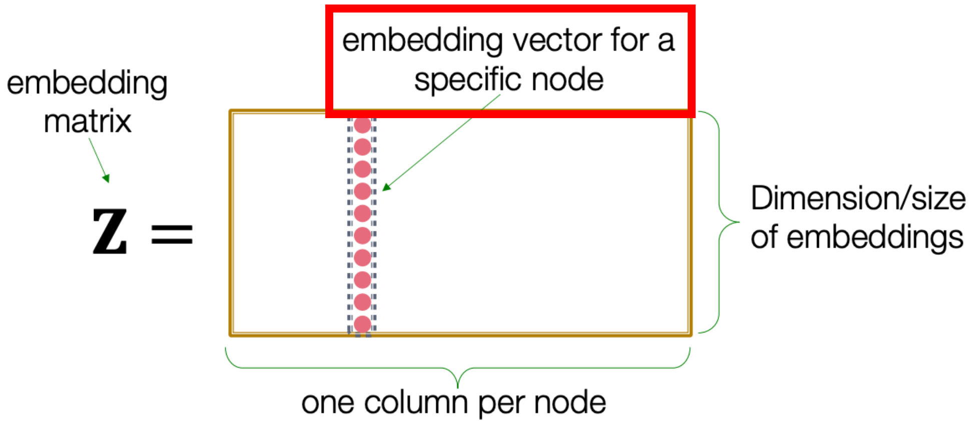
Goal: similarity$(u, v)$ $\approx$ $\mathbf{z}_v^{\mathrm{T}} \mathbf{z}_u$

in the original network

Similarity of the embedding

Need to define!

Q: how to learn ENC?



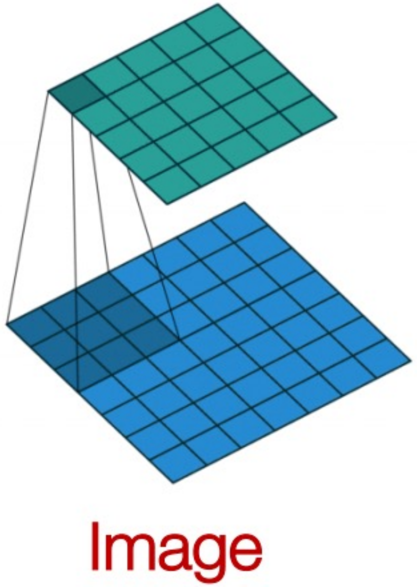
embedding vector for a specific node
embedding matrix
$\mathbf{Z} =$
Dimension/size of embeddings
one column per node

Learnable parameters

Q: how can we replace linear node embedding with nonlinear functions?

Image credit http://web.stanford.edu/class/cs224w/slides/03-nodeemb.pdf

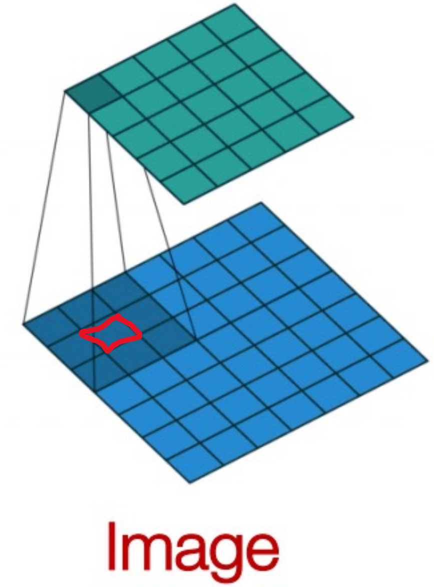
# Graph neural networks



Q: can we use convolution operation on graph?

Graph convolutional neural networks

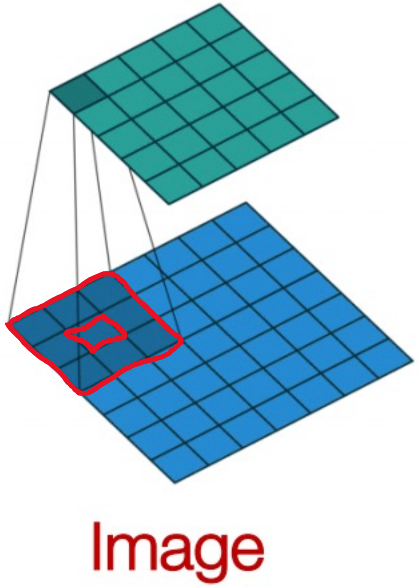
# Graph neural networks



Q: can we use convolution operation on graph?

Graph convolutional neural networks

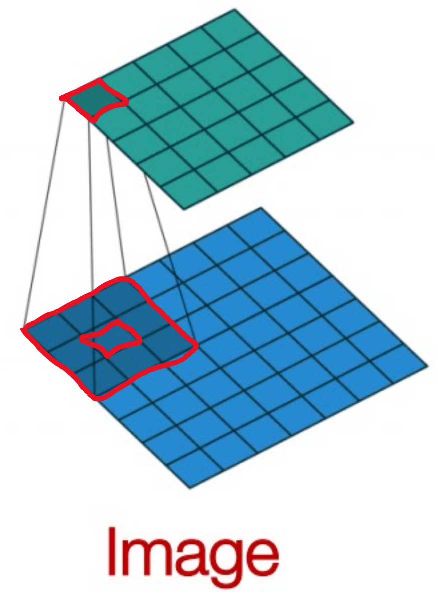
# Graph neural networks



Q: can we use convolution operation on graph?

Graph convolutional neural networks

# Graph neural networks

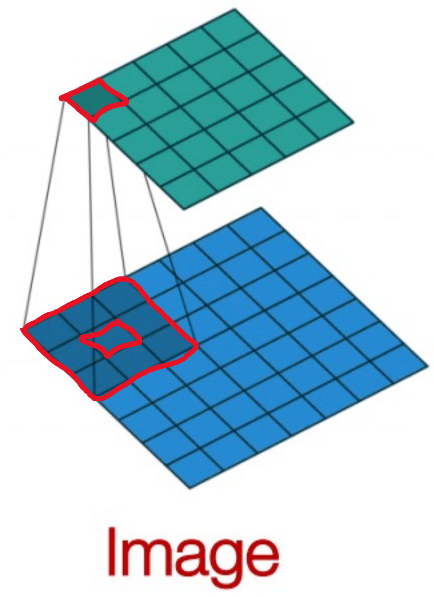


Q: can we use convolution operation on graph?

Graph convolutional neural networks

# Graph neural networks

Transform information at neighbors

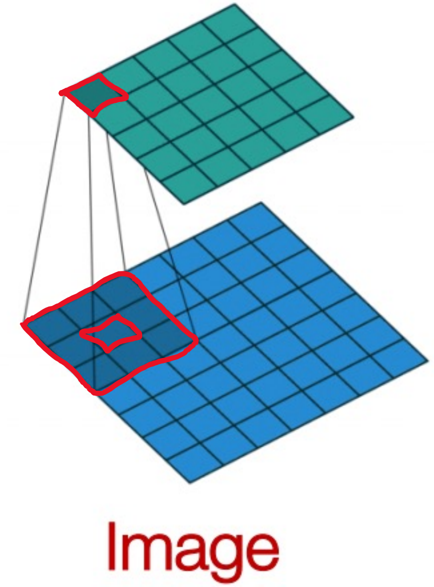


Q: can we use convolution operation on graph?

Graph convolutional neural networks

# Graph neural networks

Transform information at neighbors
Combine them



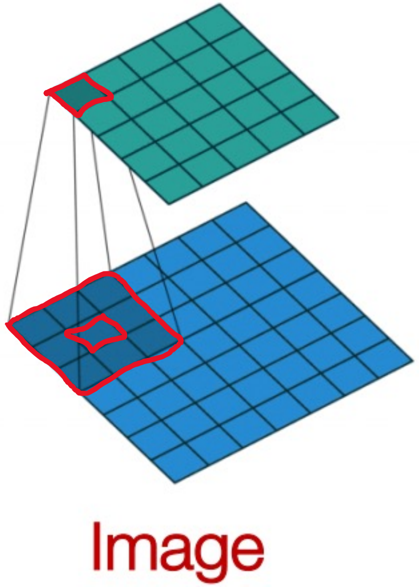
Q: can we use convolution operation on graph?

Graph convolutional neural networks

# Graph neural networks

Q: What is information?

Transform information at neighbors
Combine them



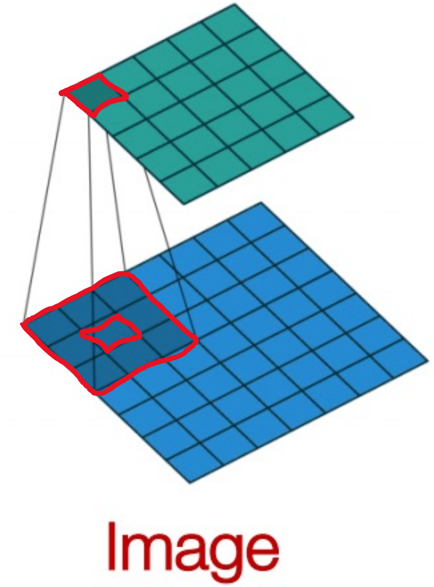
Q: can we use convolution operation on graph?

Graph convolutional neural networks

# Graph neural networks

Q: What is information? → Raw features (pixel values) or previous feature maps

Transform information at neighbors
Combine them



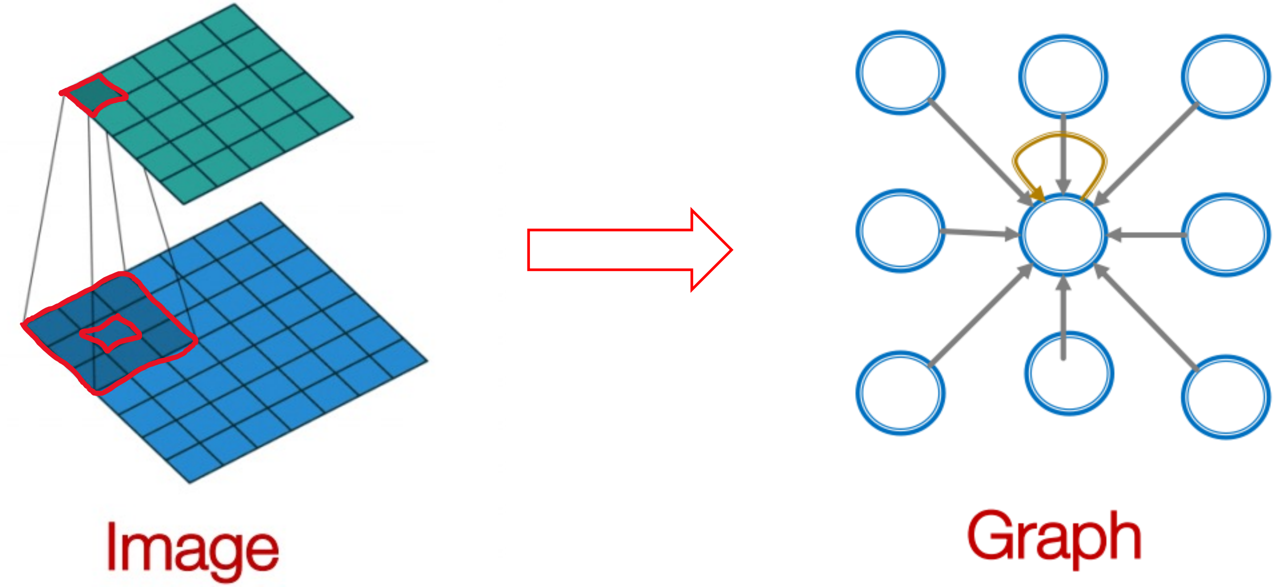
Q: can we use convolution operation on graph?

Graph convolutional neural networks

# Graph neural networks

Q: What is information? → Raw features (pixel values) or previous feature maps

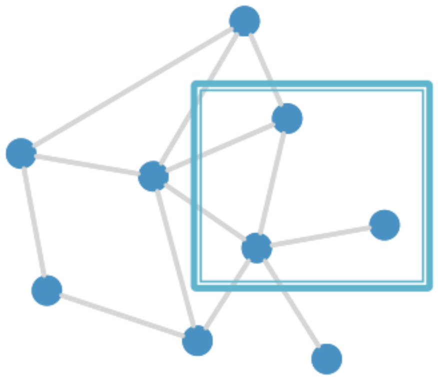
Transform information at neighbors
Combine them



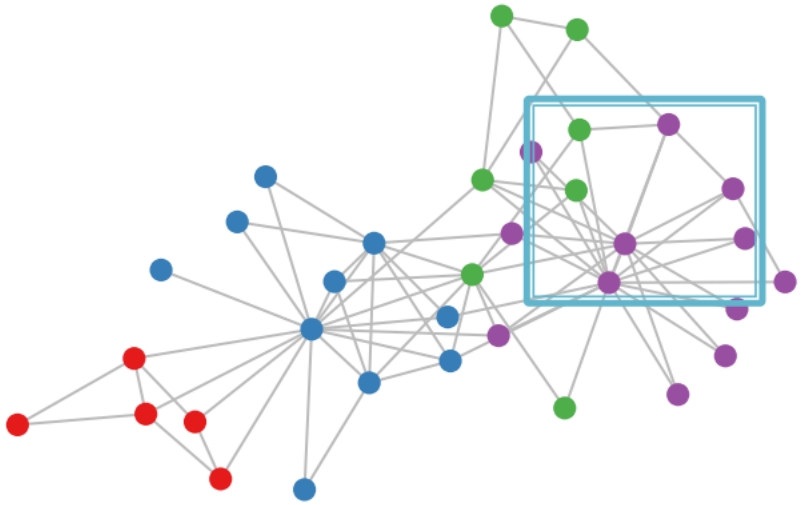
Q: can we use convolution operation on graph?

Graph convolutional neural networks

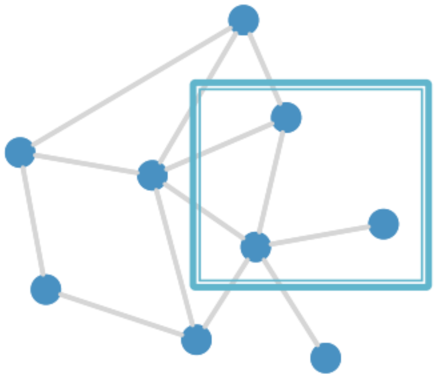
# General graph data



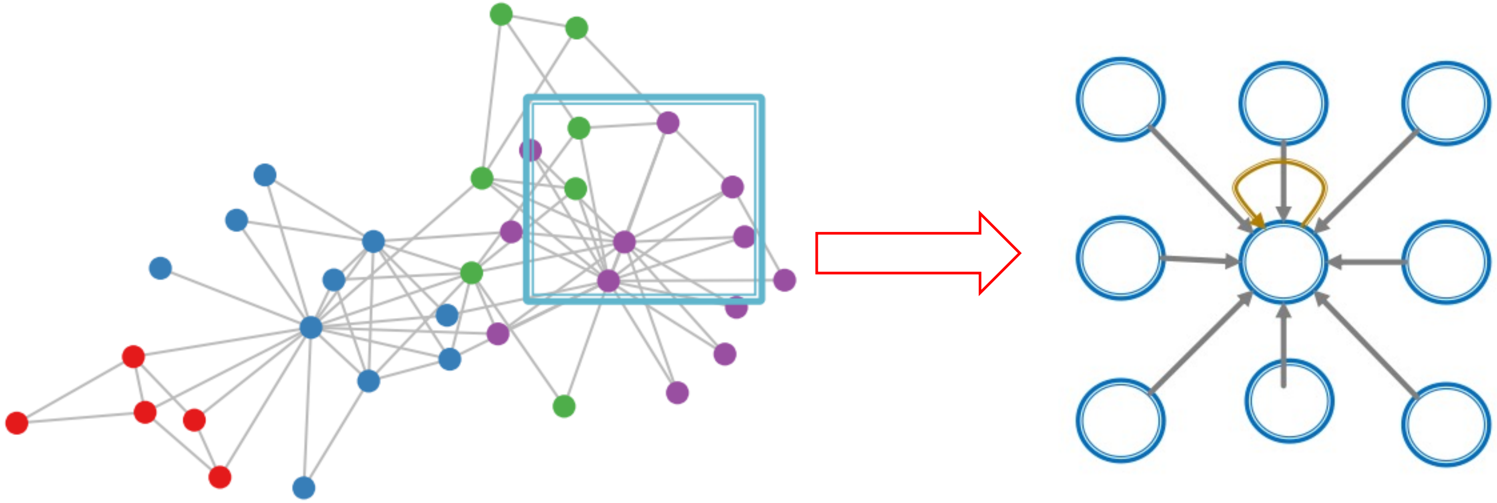
or this:



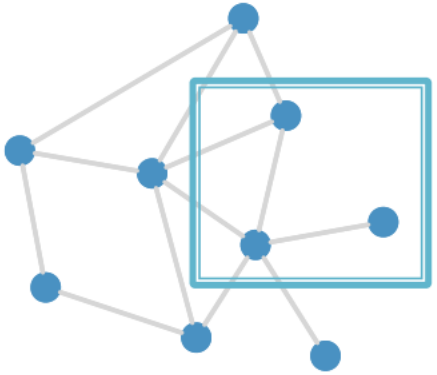
# General graph data



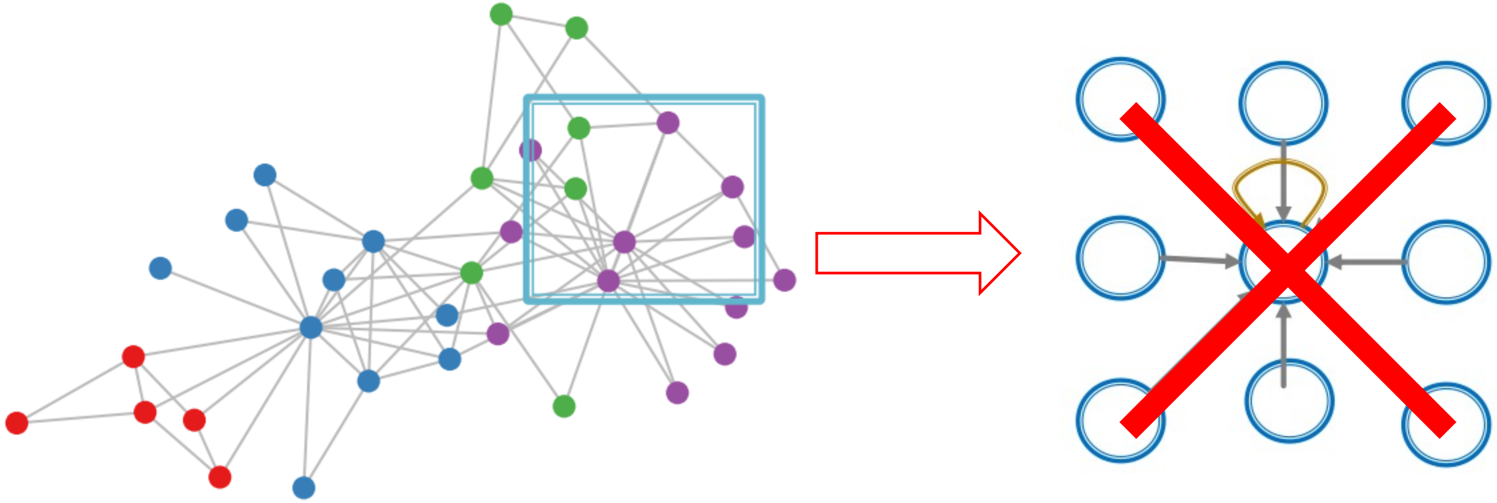
or this:



# General graph data

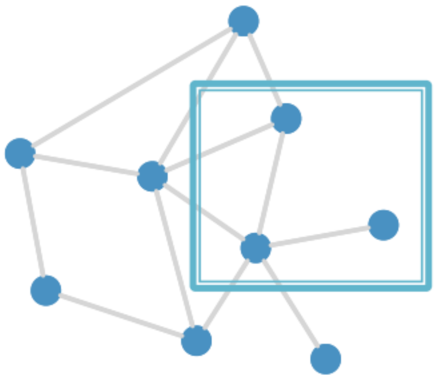


or this:

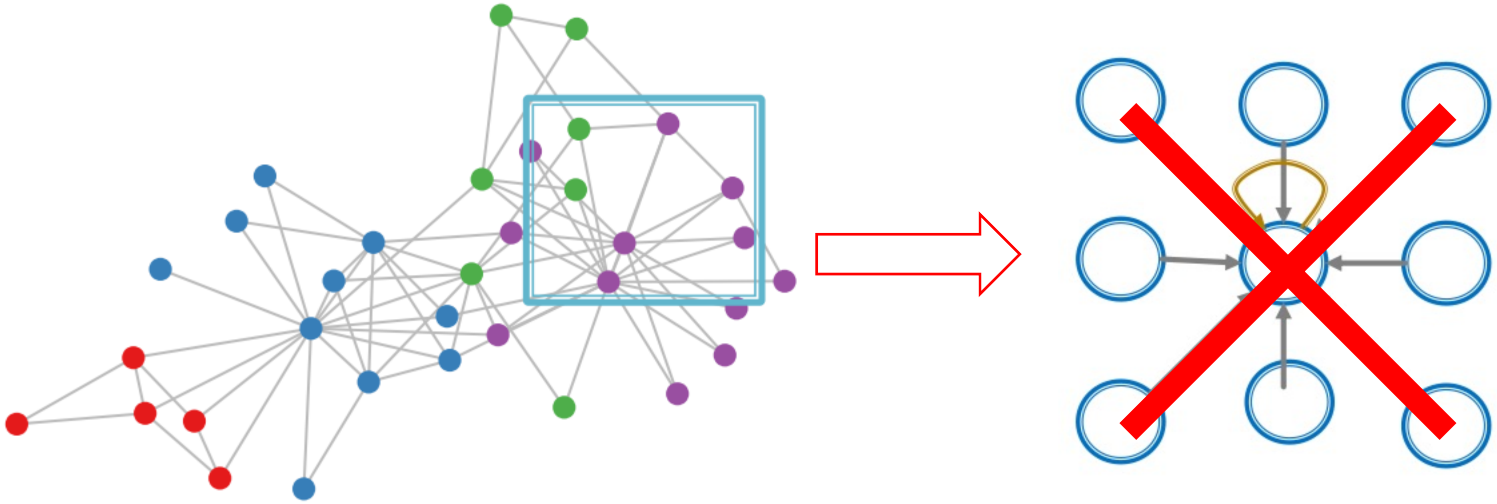


Q: can we extend similar operation to general graph?

# General graph data



or this:



Q: can we extend similar operation to general graph?

Key: aggregate information from neighbors

# Node degree

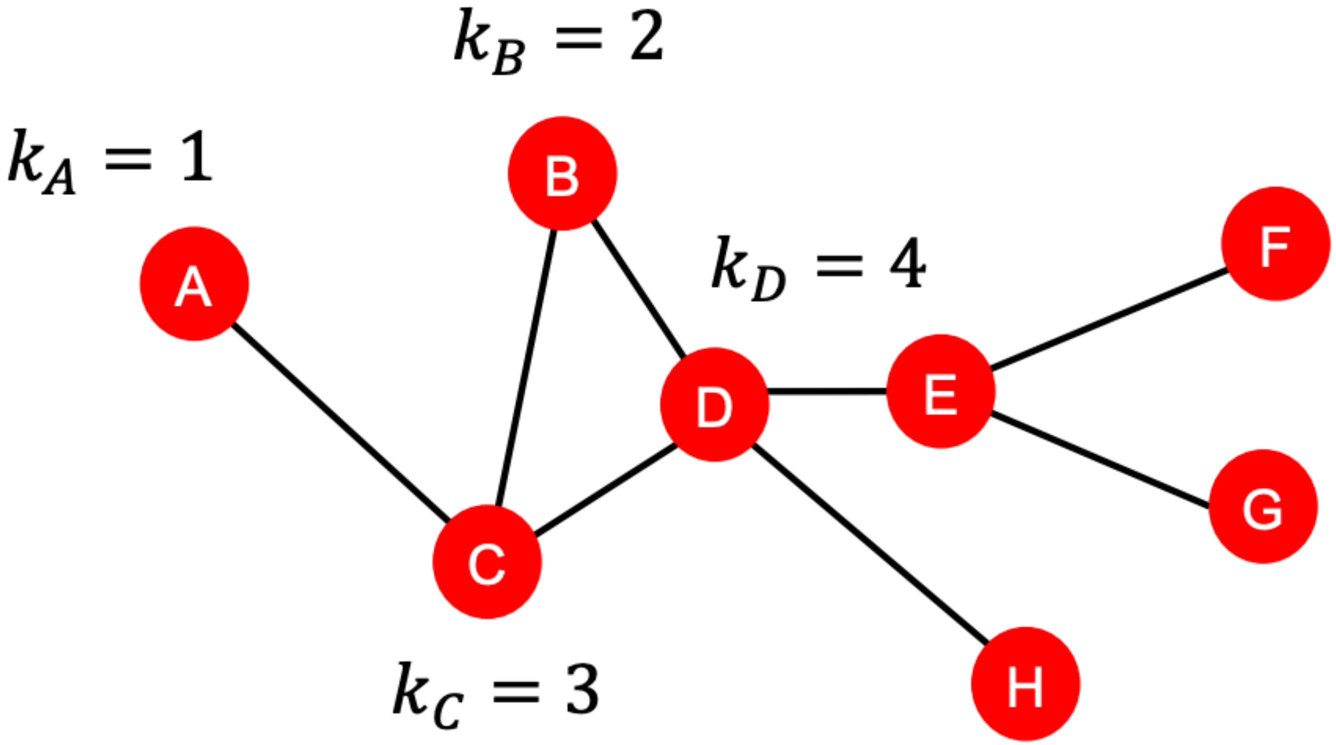
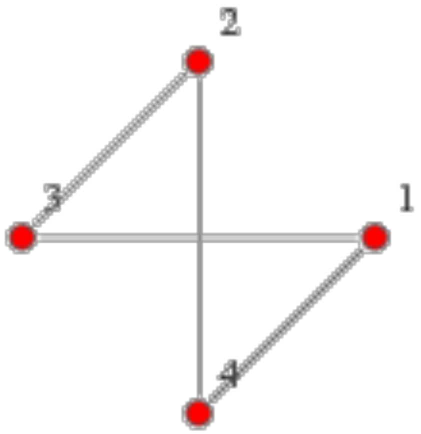
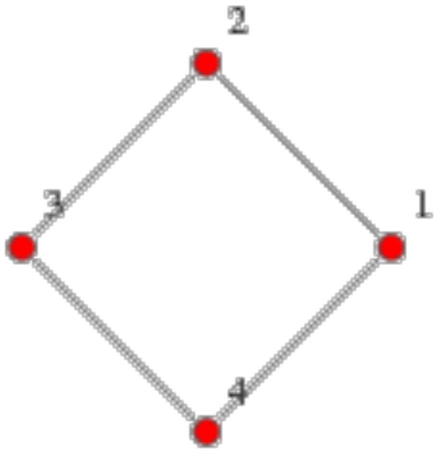
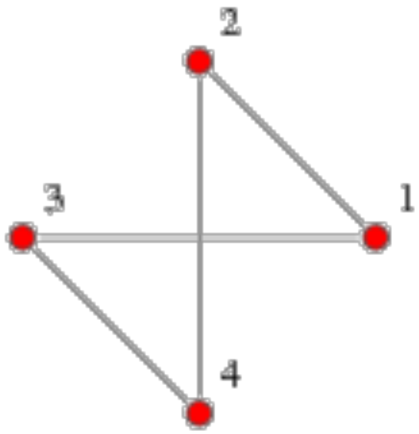


Image credit http://web.stanford.edu/class/cs224w/slides/03-nodeemb.pdf

# Adjacency matrix

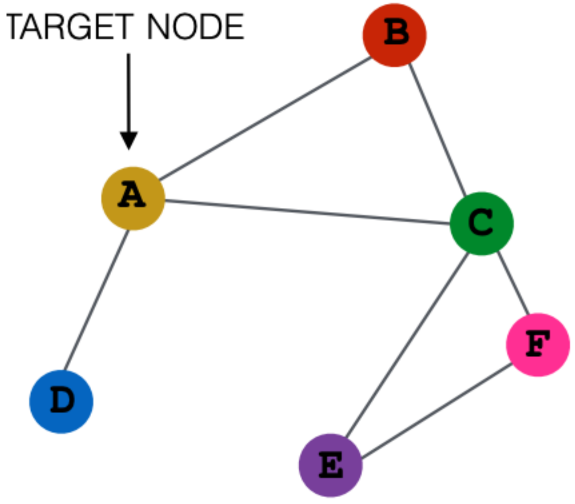






$$\begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

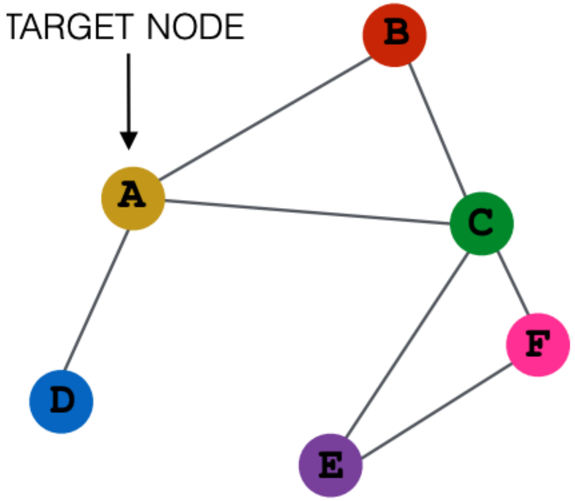
# Graph networks: aggregate neighbors



INPUT GRAPH

# Graph networks: aggregate neighbors

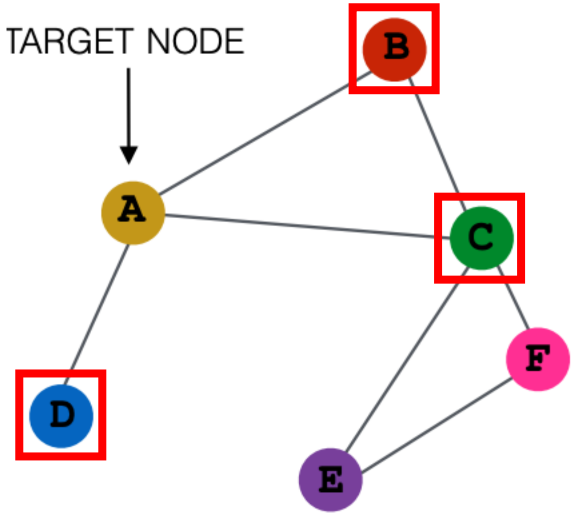
Nearest neighbors of A?



INPUT GRAPH

# Graph networks: aggregate neighbors

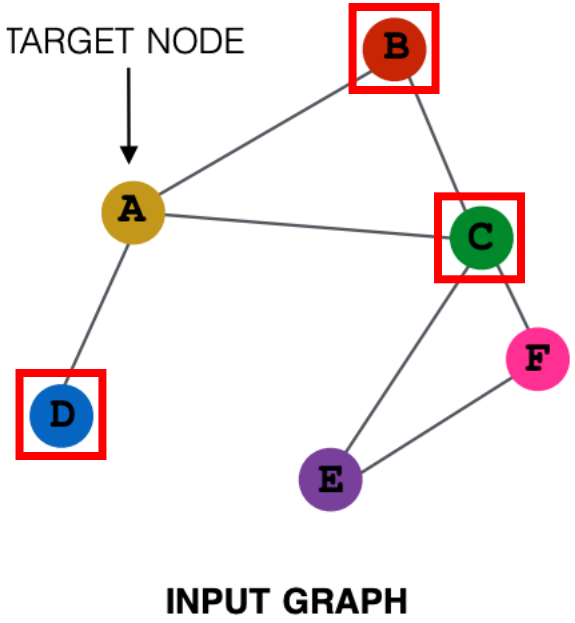
Nearest neighbors of A:
B, C, D

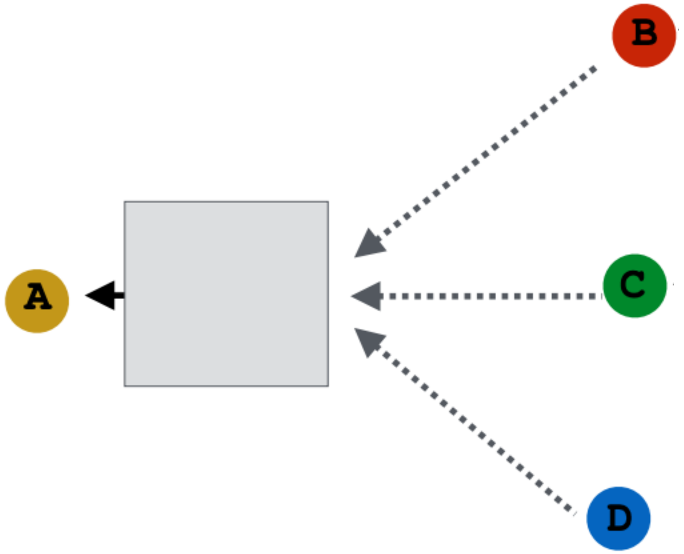


INPUT GRAPH

# Graph networks: aggregate neighbors

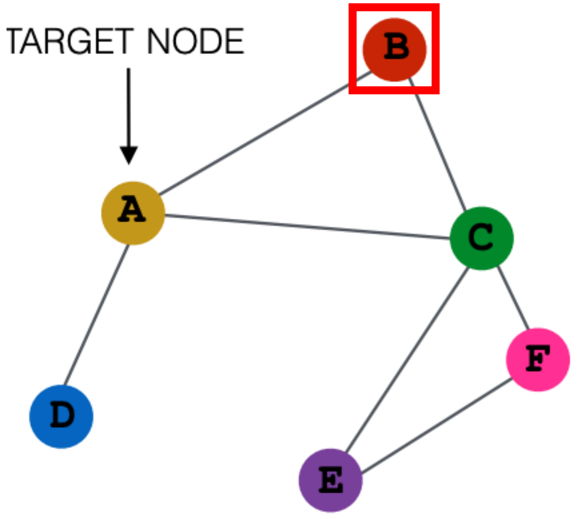
Nearest neighbors of A:
B, C, D





# Graph networks: aggregate neighbors

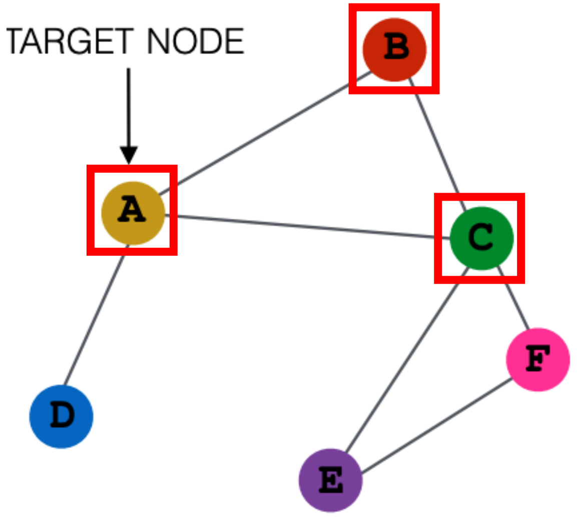
Nearest neighbors of B?



INPUT GRAPH

# Graph networks: aggregate neighbors

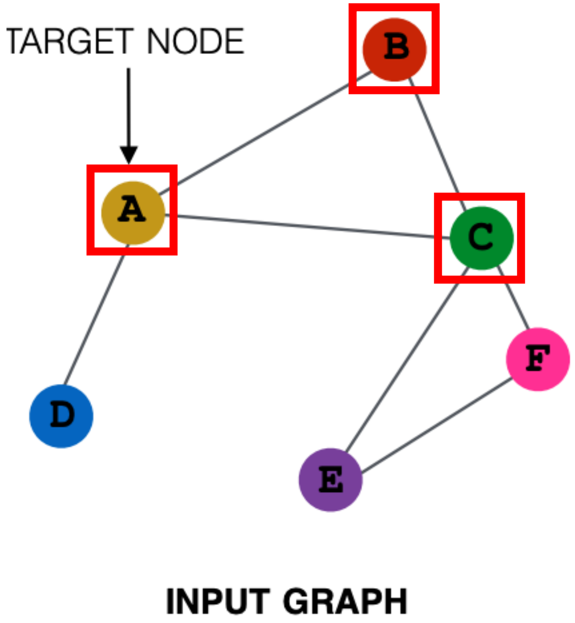
Nearest neighbors of B: A, C

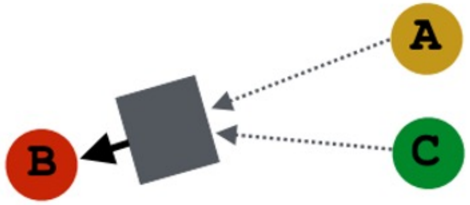


INPUT GRAPH

# Graph networks: aggregate neighbors

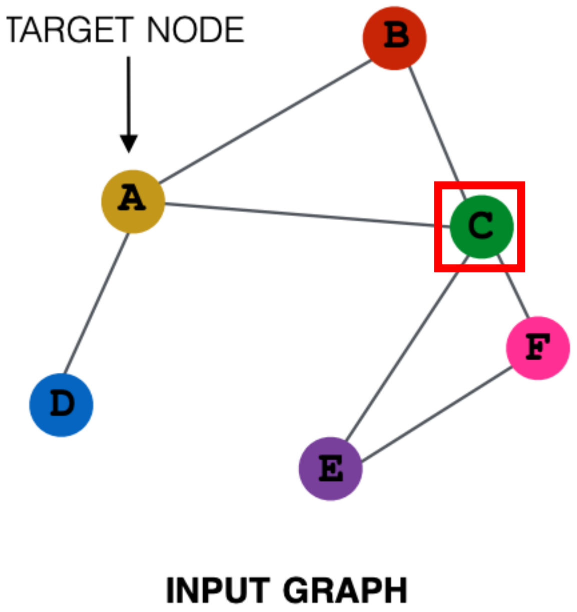
Nearest neighbors of B: A, C

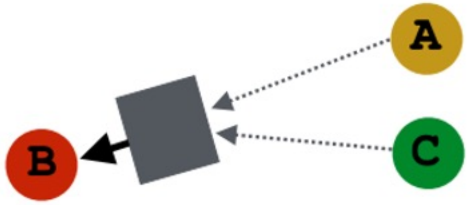




# Graph networks: aggregate neighbors

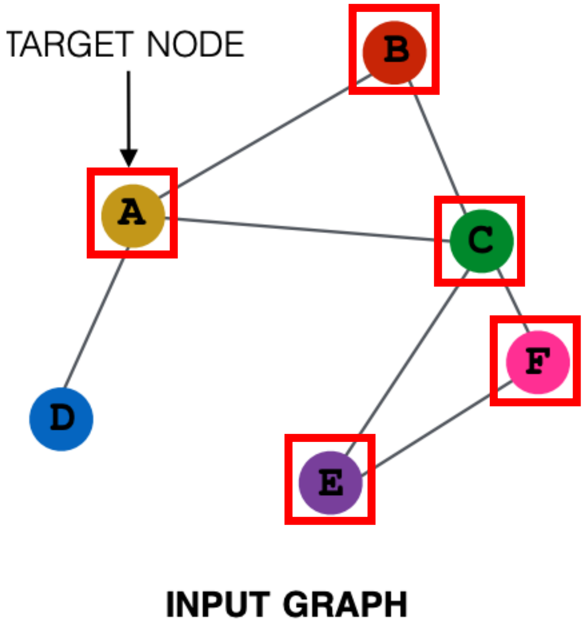
Nearest neighbors of C?

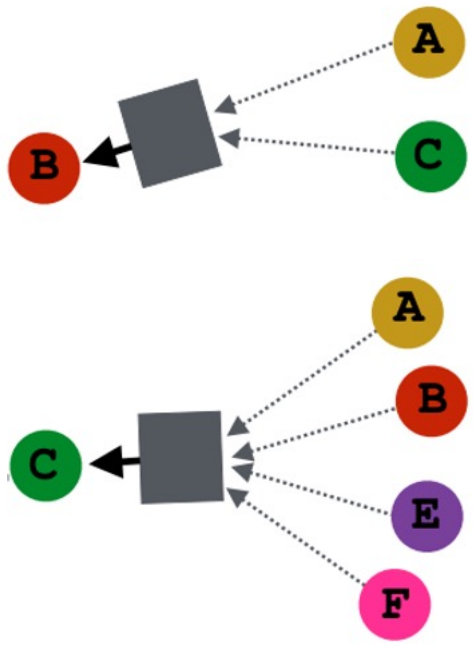




# Graph networks: aggregate neighbors

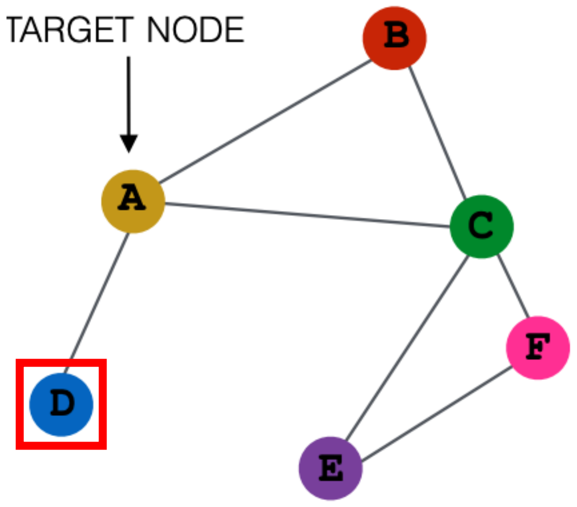
Nearest neighbors of C: A, B, E, F



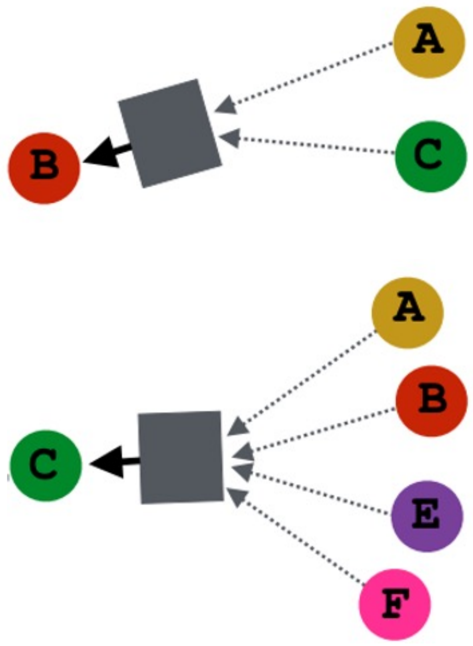


# Graph networks: aggregate neighbors

Nearest neighbors of D?

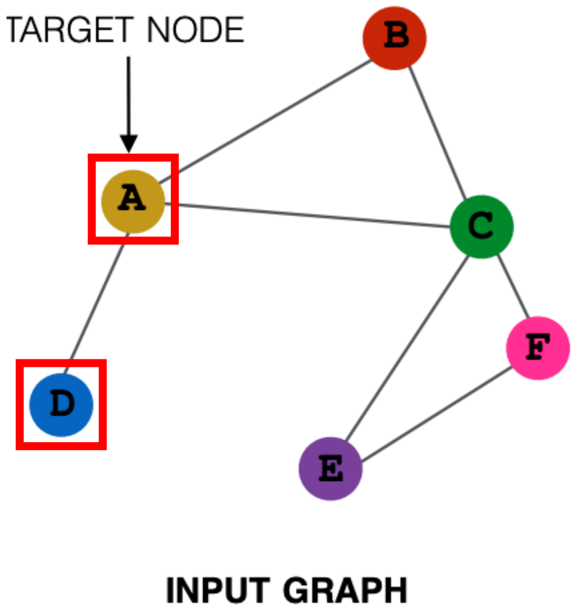


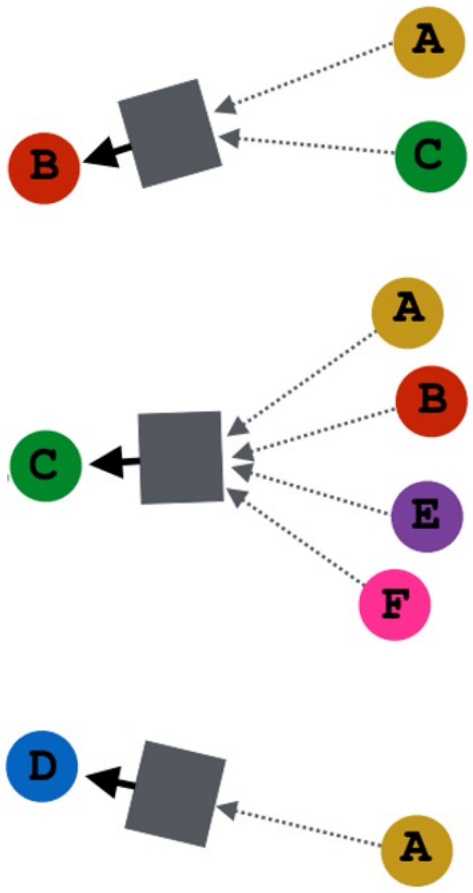
INPUT GRAPH



# Graph networks: aggregate neighbors

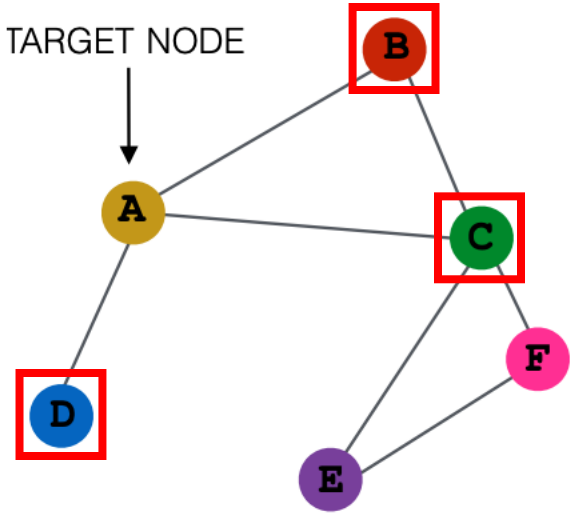
Nearest neighbors of D: A



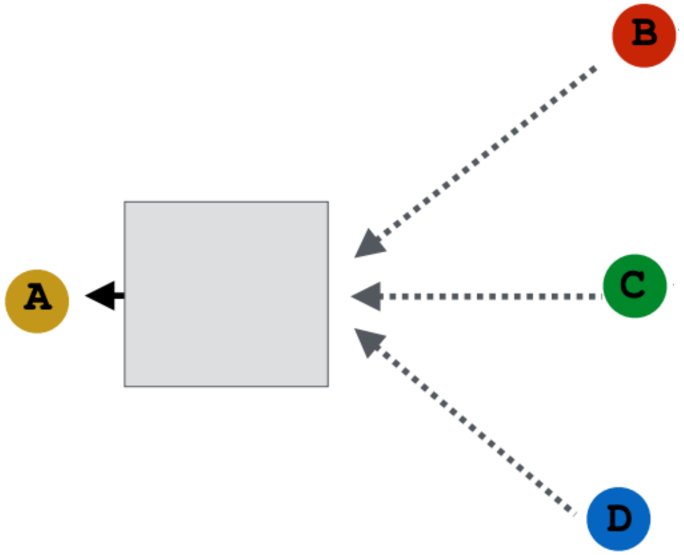


# Graph networks: aggregate neighbors

Nearest neighbors of A:
B, C, D

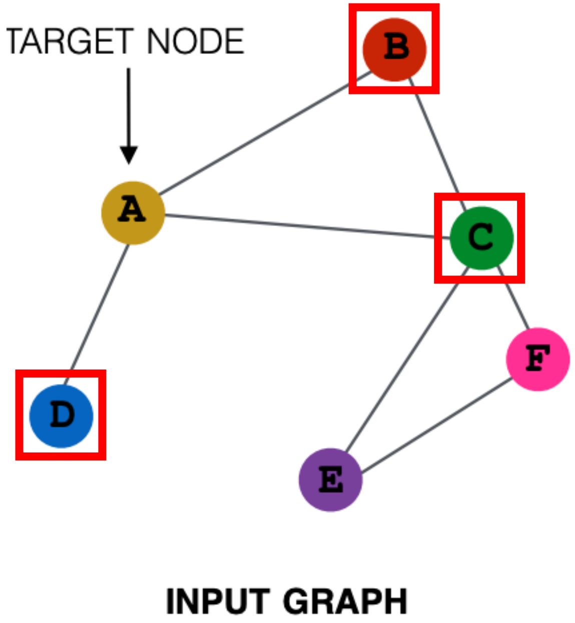


INPUT GRAPH

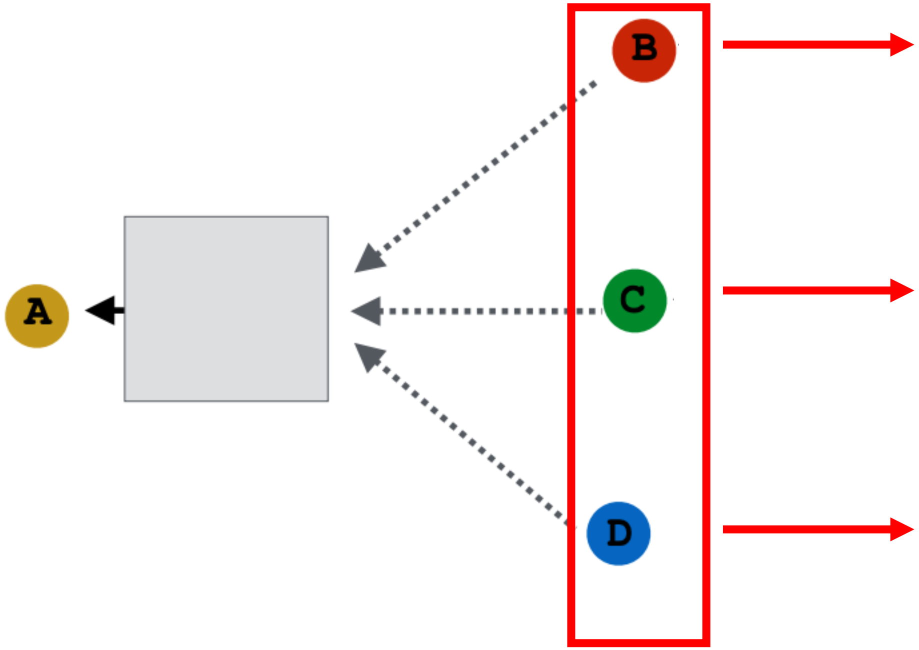


# Graph networks: aggregate neighbors

Nearest neighbors of A:
B, C, D

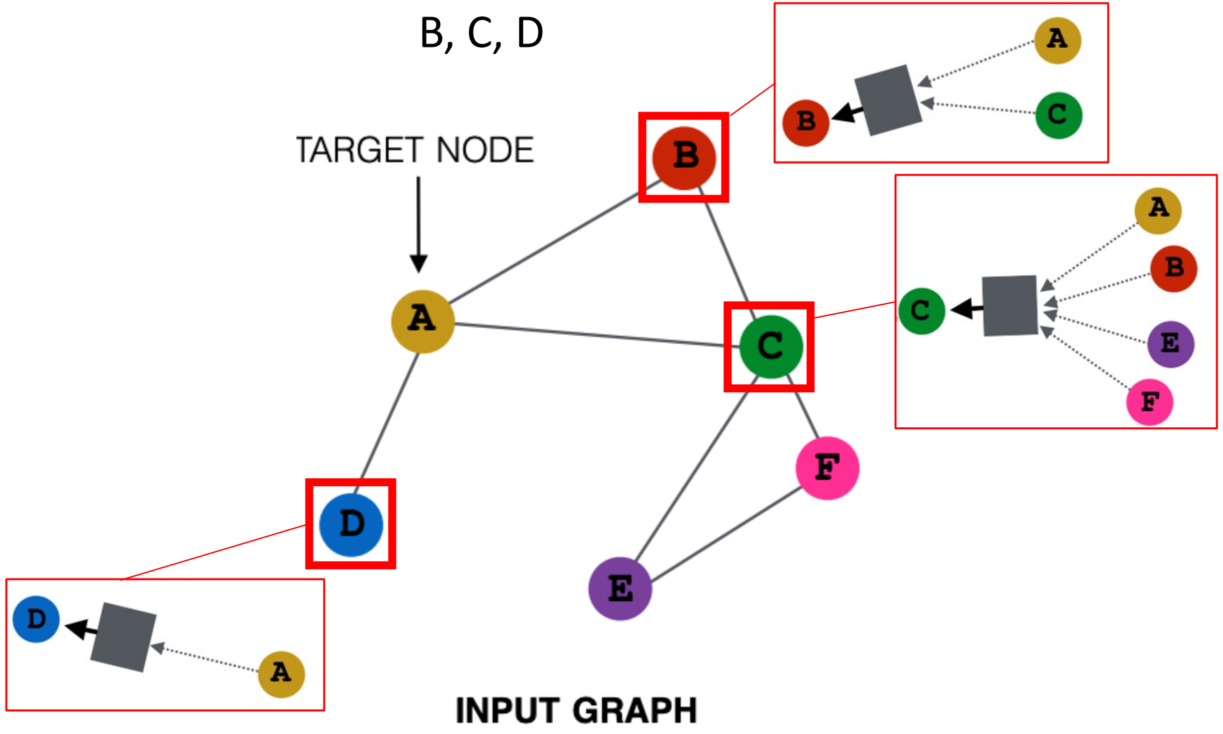


Q: Deeper structure?

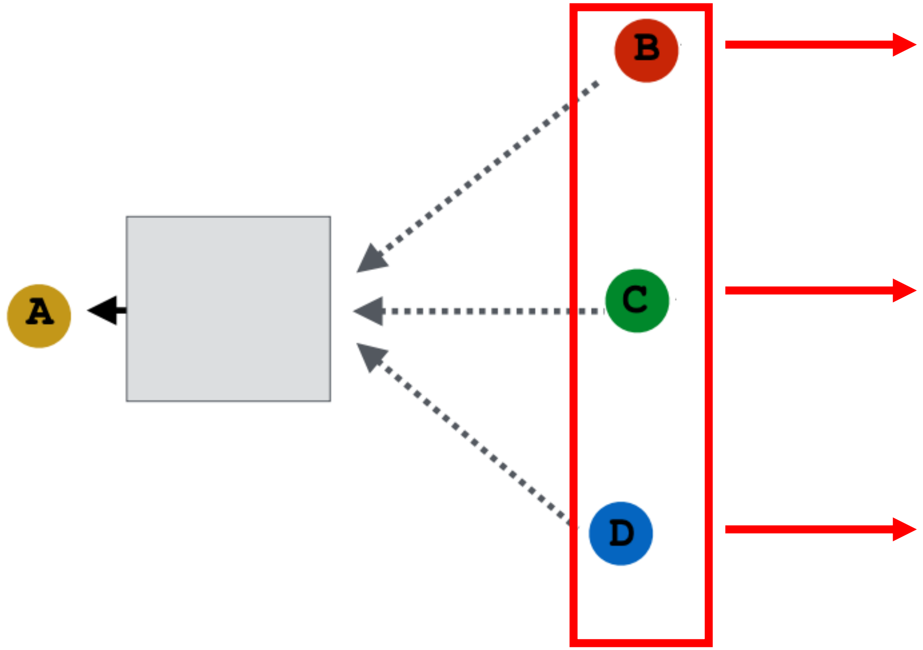


# Graph networks: aggregate neighbors

Nearest neighbors of A:
B, C, D

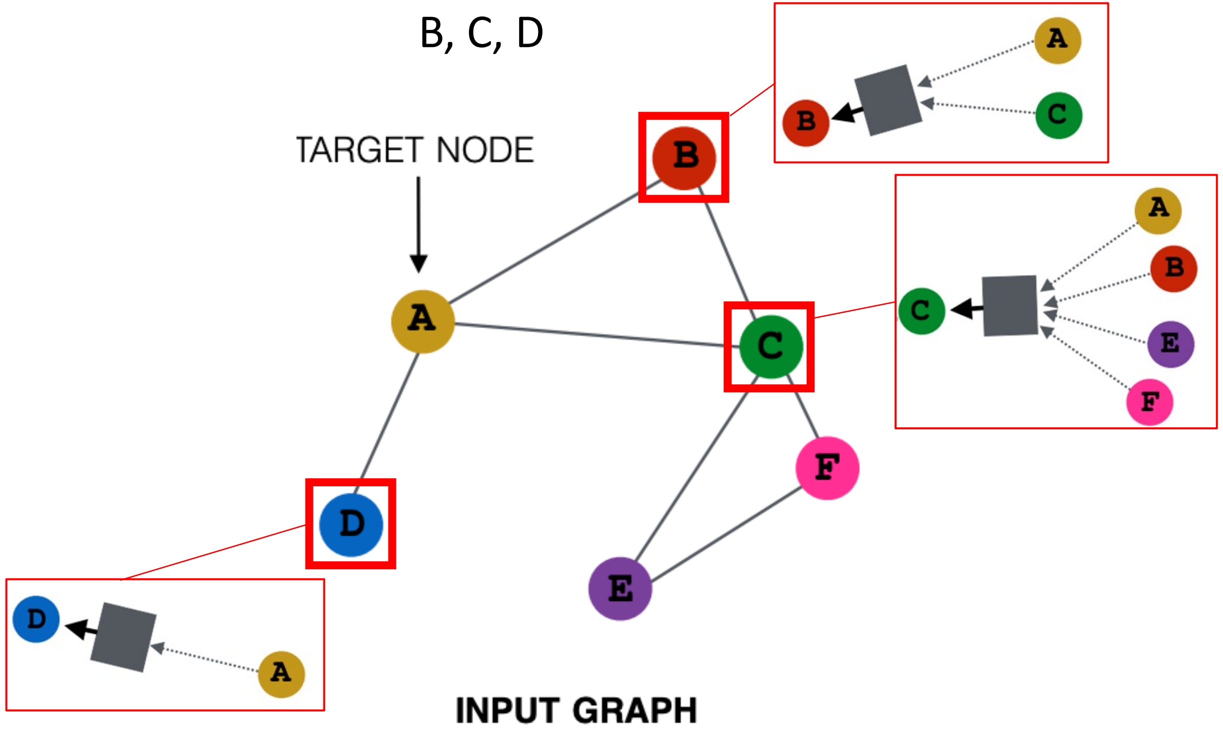


Q: Deeper structure?

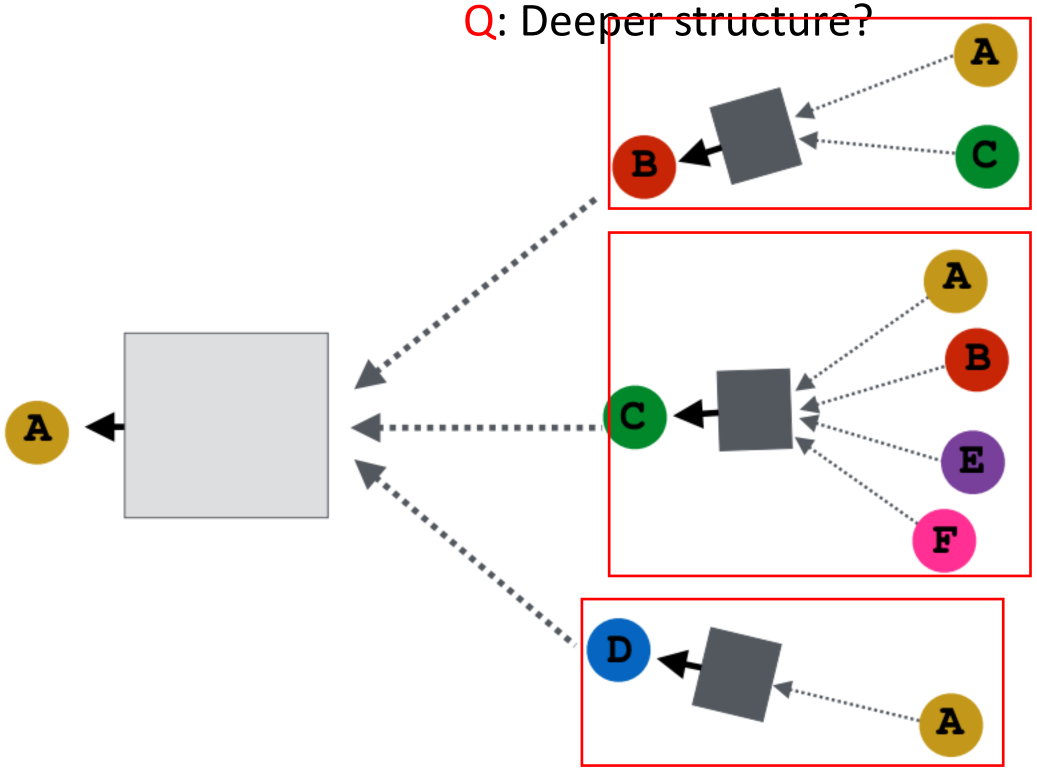


# Graph networks: aggregate neighbors

Nearest neighbors of A:
B, C, D

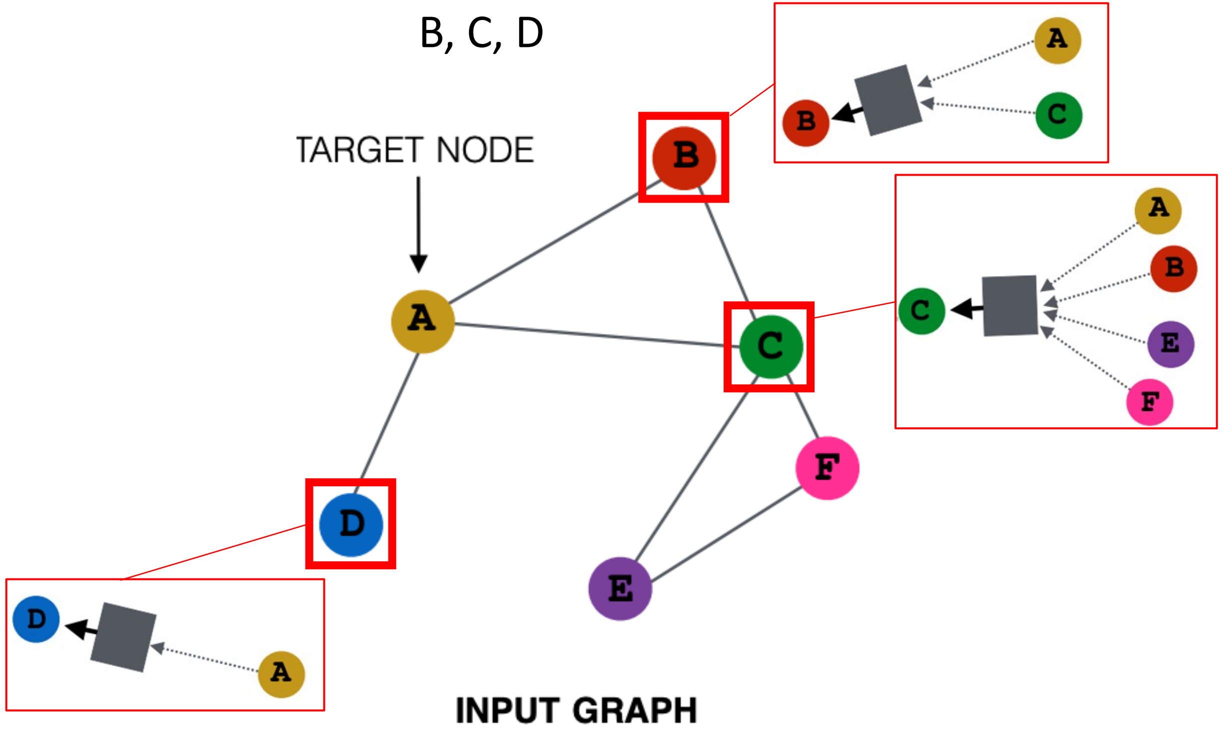


Q: Deeper structure?

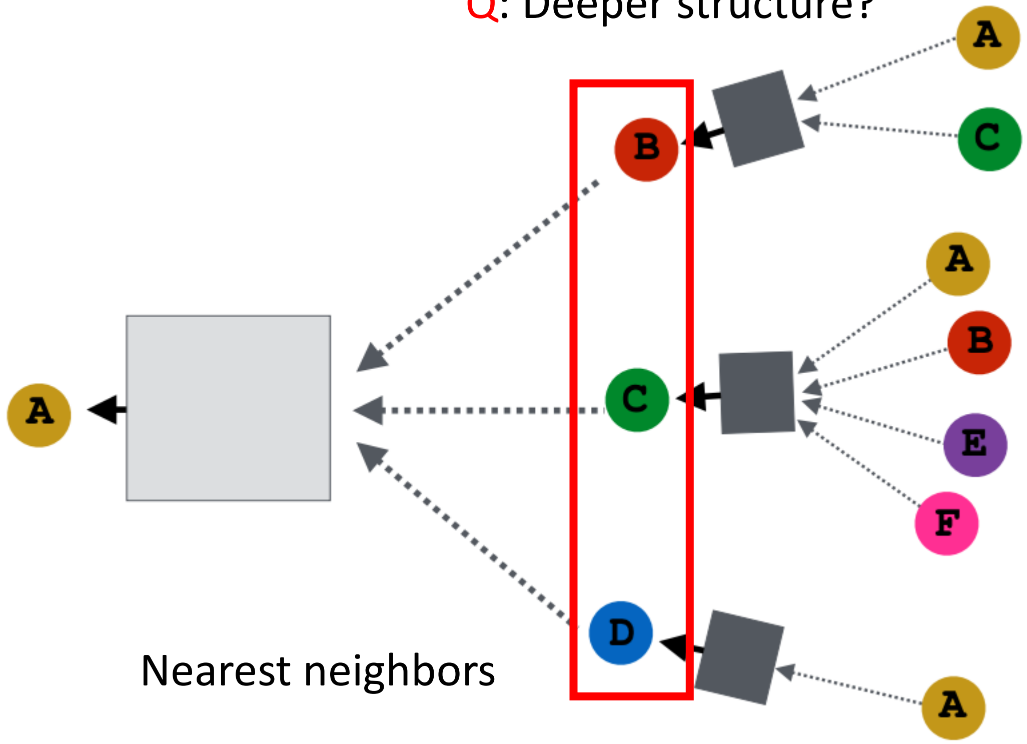


# Graph networks: aggregate neighbors

Nearest neighbors of A:
B, C, D



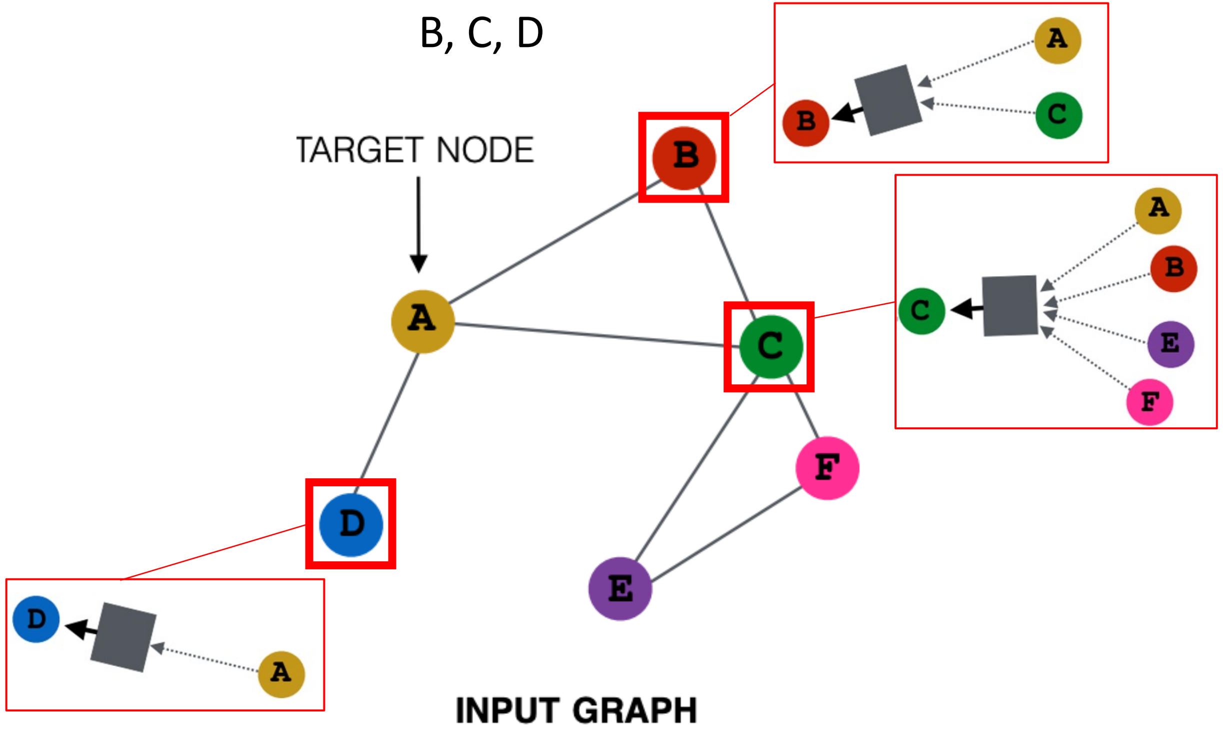
Q: Deeper structure?



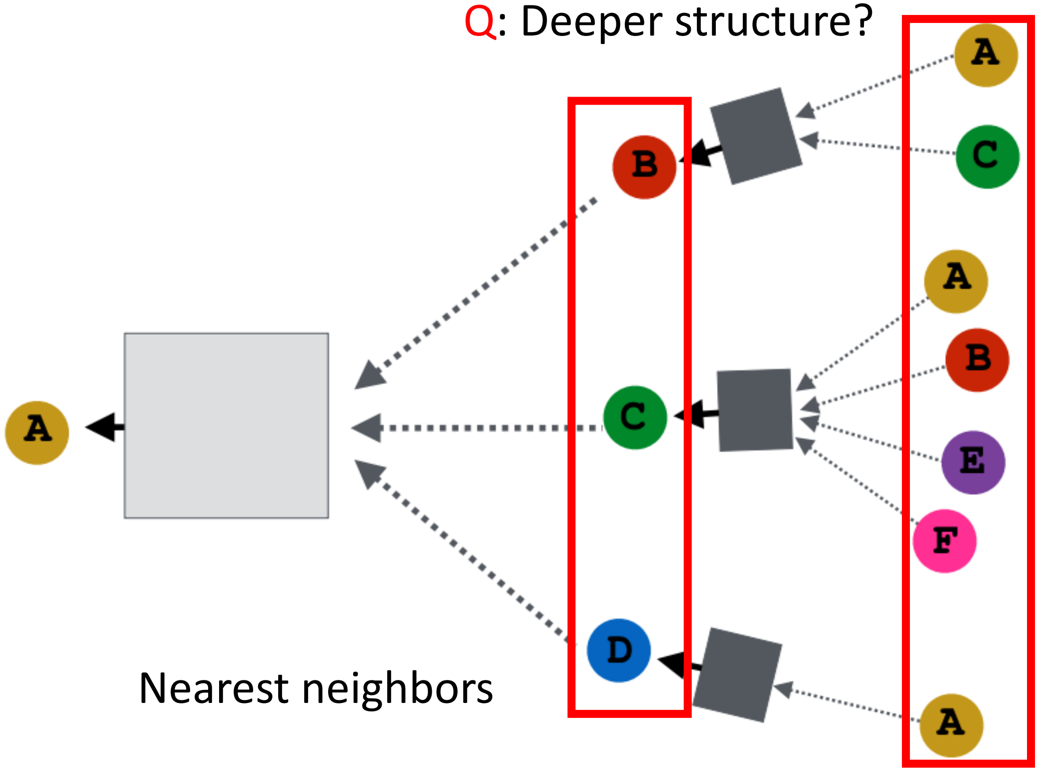
Nearest neighbors

# Graph networks: aggregate neighbors

Nearest neighbors of A:
B, C, D



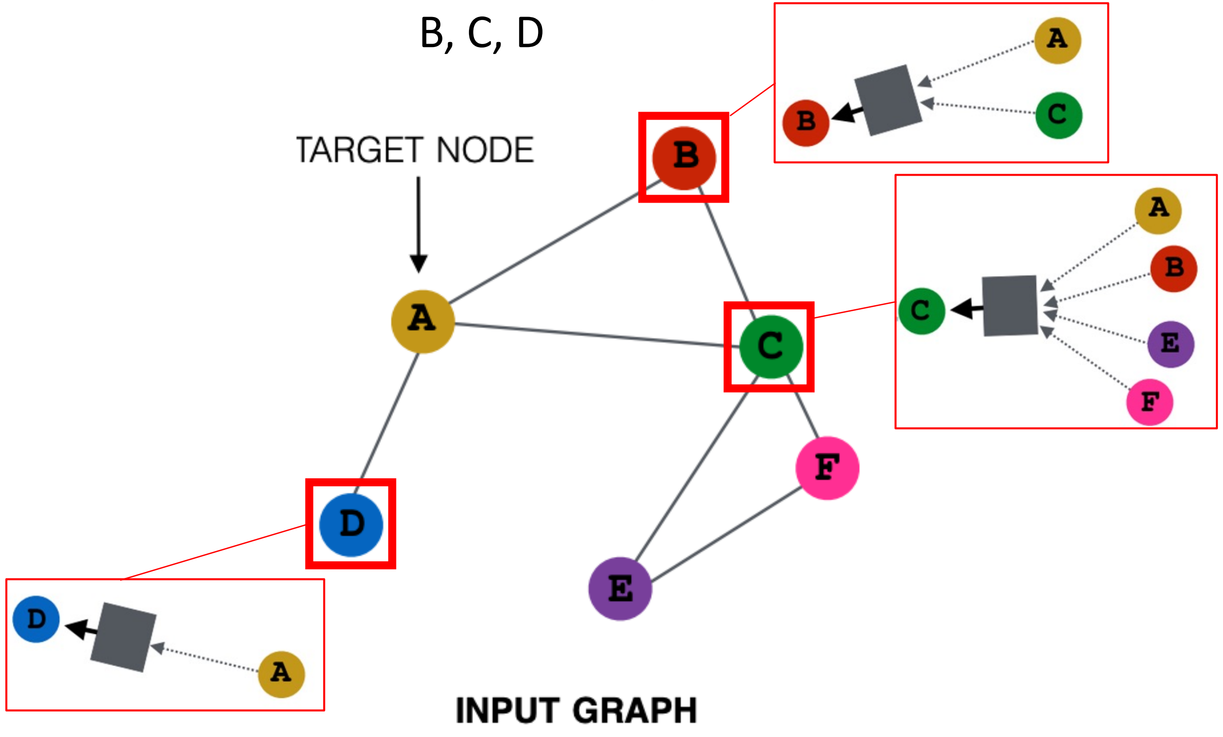
Q: Deeper structure?



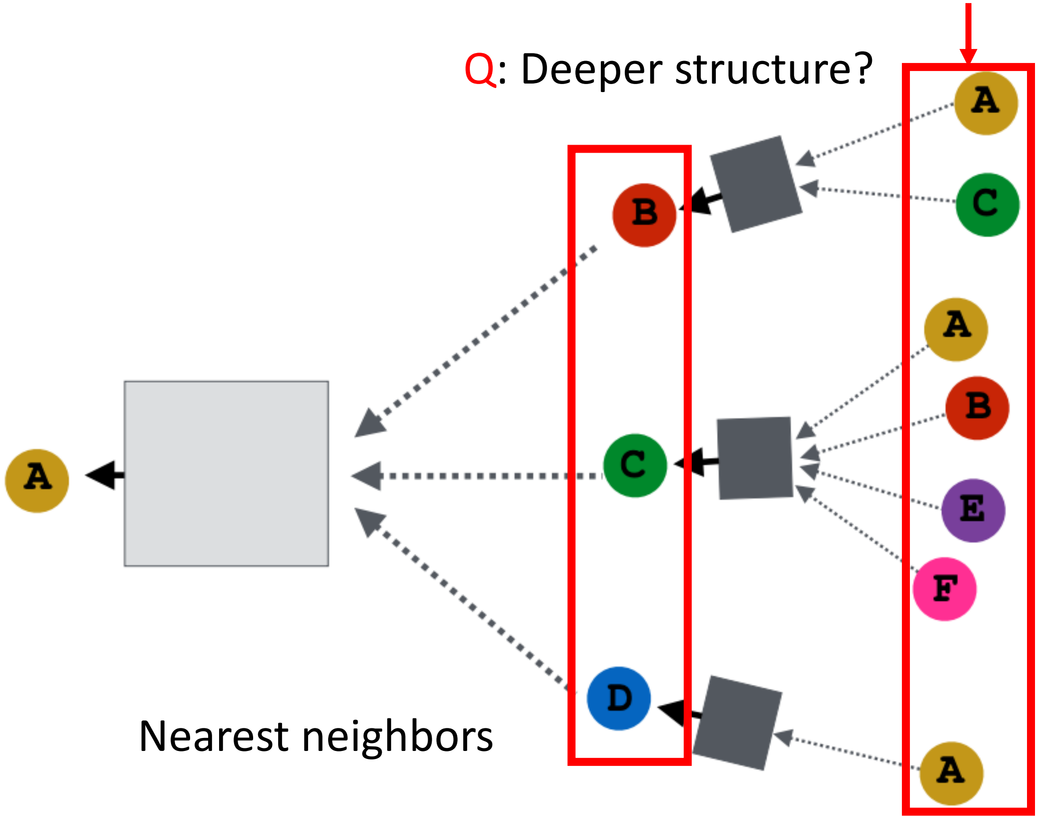
Nearest neighbors

Two hops away

# Graph networks: aggregate neighbors



Nearest neighbors of A:
B, C, D



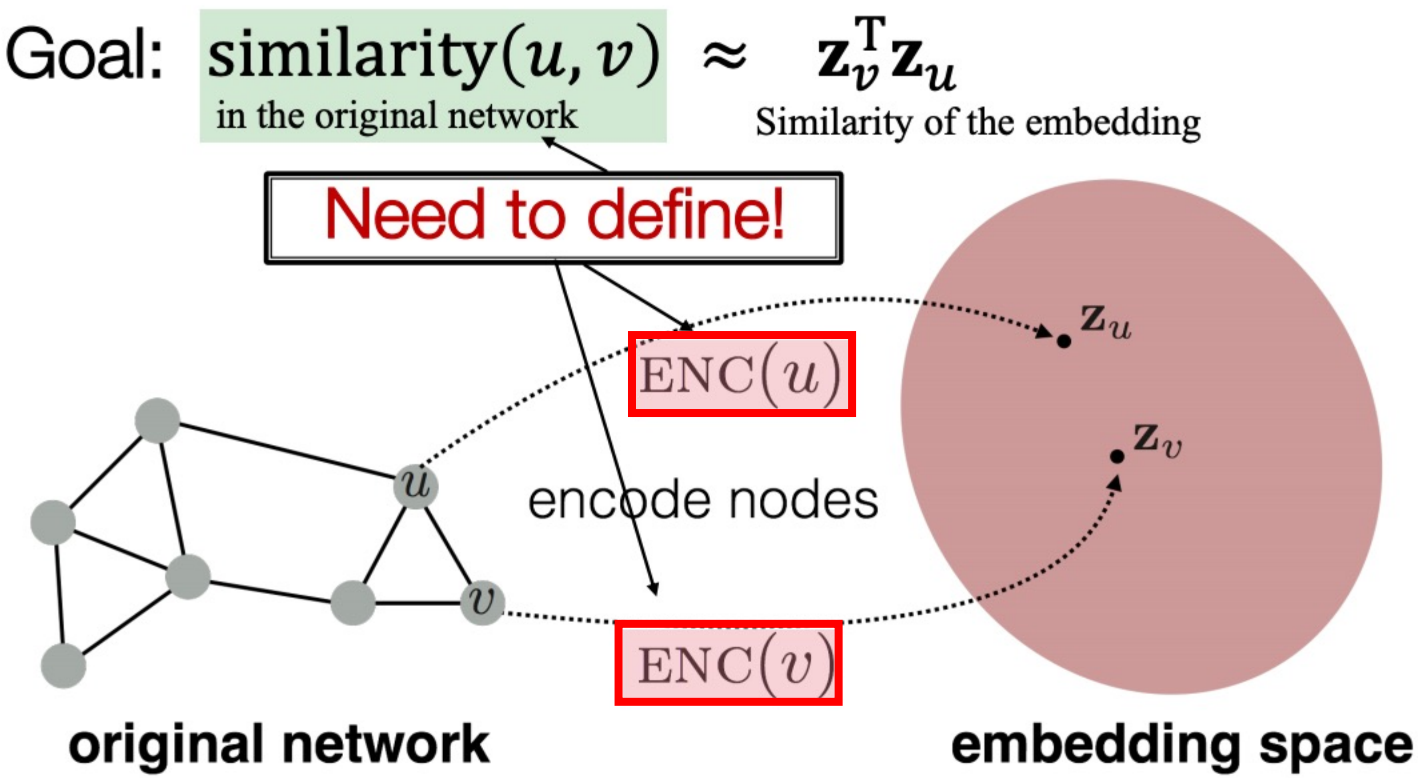
Raw feature:
Node embeddings

Q: Deeper structure?

Nearest neighbors

Two hops away

# Encoder-decoder for graph data



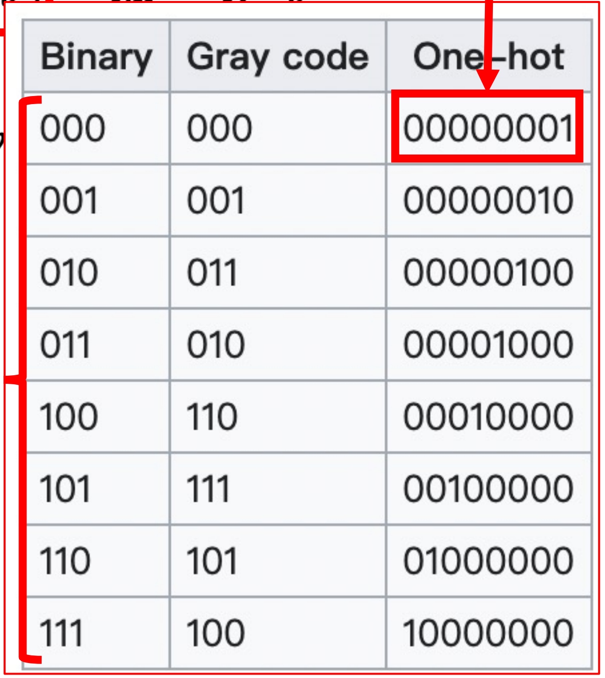
Q: how to learn ENC?
Linear transformation

$\boxed{\mathrm{ENC}(v)} = \mathbf{z}_v = \mathbf{Z} \cdot v$

$\boxed{\mathbf{Z}} \in \mathbb{R}^{d\times|\mathcal{V}}$

$\boxed{v} \in \mathbb{I}^{\boxed{|\mathcal{V}|}}$

Each node

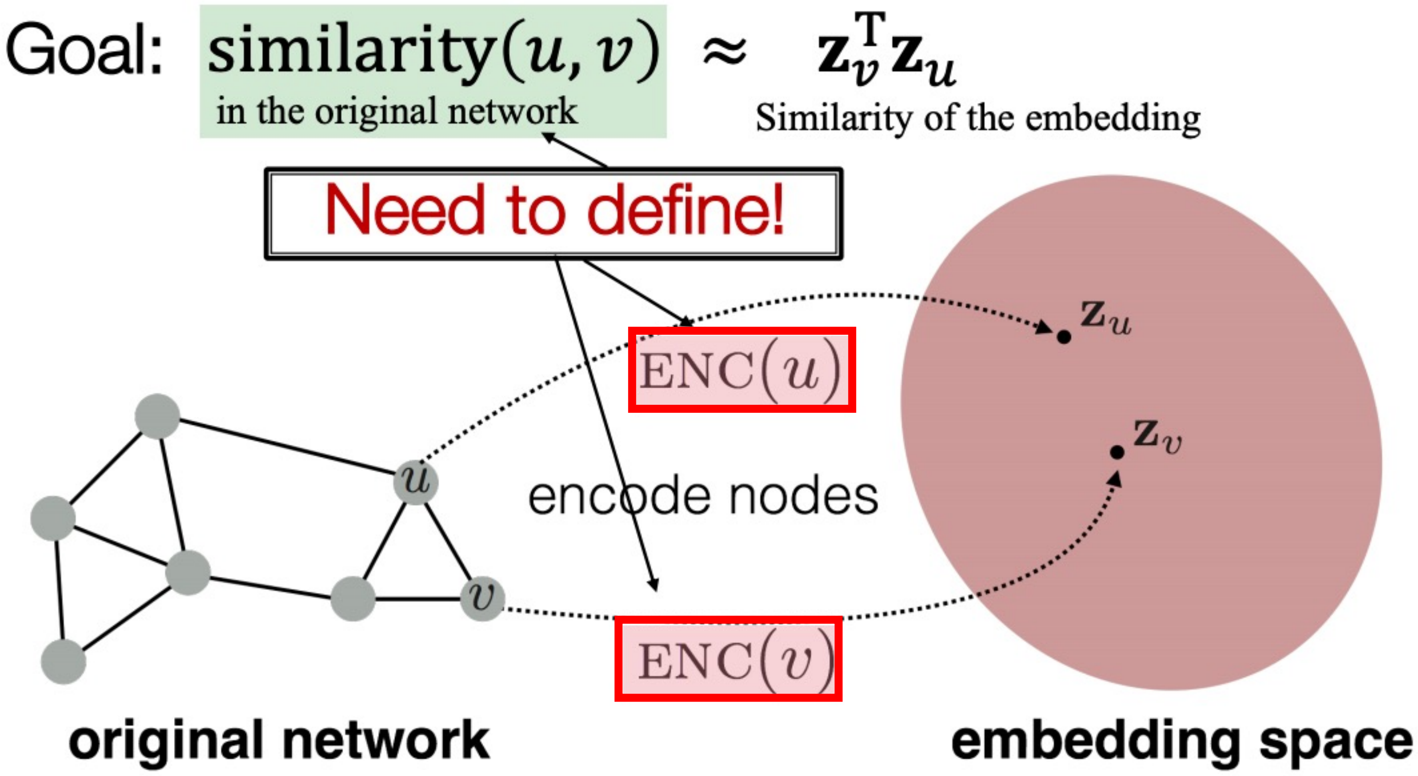


| Binary | Gray code | One-hot |
|---|---|---|
| 000 | 000 | 00000001 |
| 001 | 001 | 00000010 |
| 010 | 011 | 00000100 |
| 011 | 010 | 00001000 |
| 100 | 110 | 00010000 |
| 101 | 111 | 00100000 |
| 110 | 101 | 01000000 |
| 111 | 100 | 10000000 |

Image credit http://web.stanford.edu/class/cs224w/slides/03-nodeemb.pdf

# Encoder-decoder for graph data

Other embedding methods:
random walk embedding
node2vec



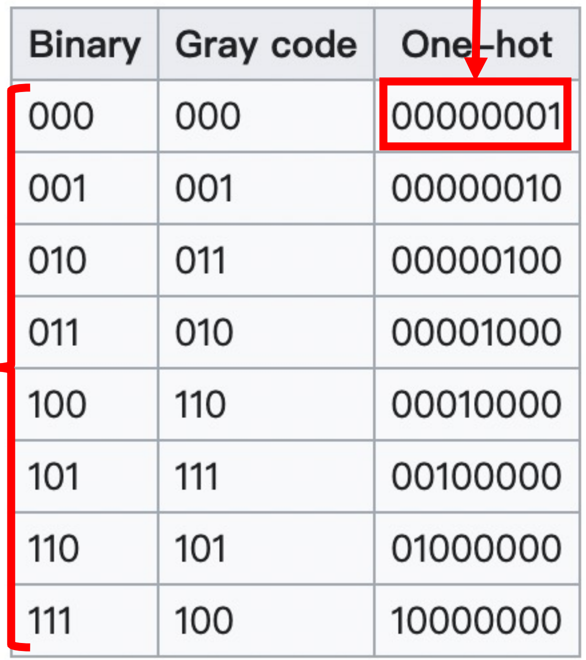
Q: how to learn ENC?
Linear transformation

$$\mathrm{ENC}(v) = \mathbf{z}_v = \mathbf{Z} \cdot v$$

$\mathbf{Z} \in \mathbb{R}^{d\times|\mathcal{V}|}$

$v \in \mathbb{I}^{|\mathcal{V}|}$

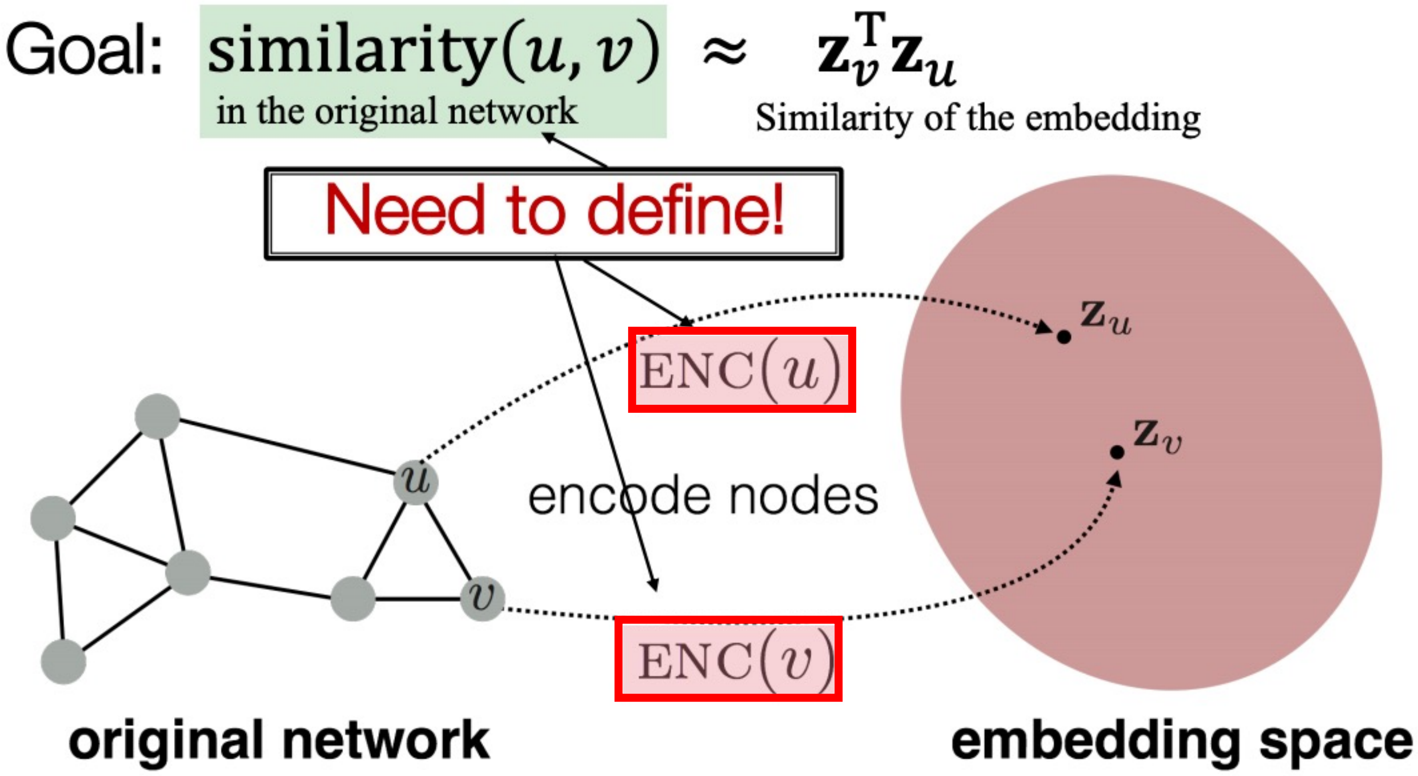
Each node



| Binary | Gray code | One-hot |
|---|---|---|
| 000 | 000 | 00000001 |
| 001 | 001 | 00000010 |
| 010 | 011 | 00000100 |
| 011 | 010 | 00001000 |
| 100 | 110 | 00010000 |
| 101 | 111 | 00100000 |
| 110 | 101 | 01000000 |
| 111 | 100 | 10000000 |

Image credit http://web.stanford.edu/class/cs224w/slides/03-nodeemb.pdf

# Encoder-decoder for graph data



Other embedding methods:
random walk embedding
node2vec

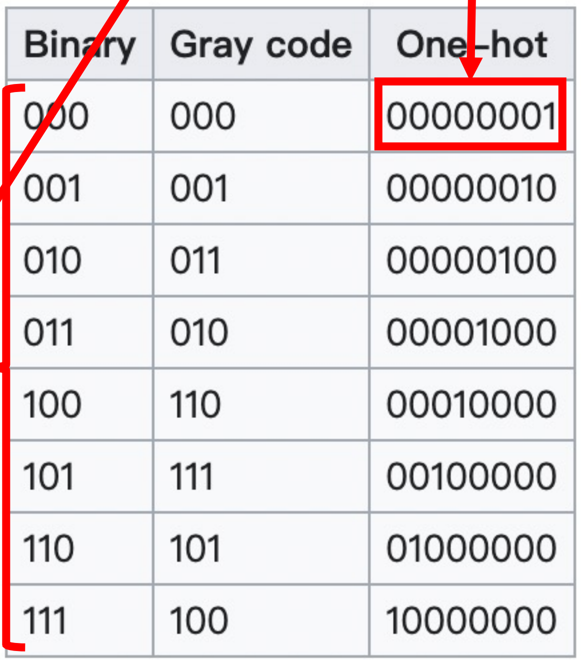
Q: how to learn ENC?
Linear transformation

$$\mathrm{ENC}(v) = \mathbf{z}_v = \mathbf{Z} \cdot v$$

$\mathbf{Z} \in \mathbb{R}^{d\times|\mathcal{V}|}$

$v \in \mathbb{I}^{|\mathcal{V}|}$

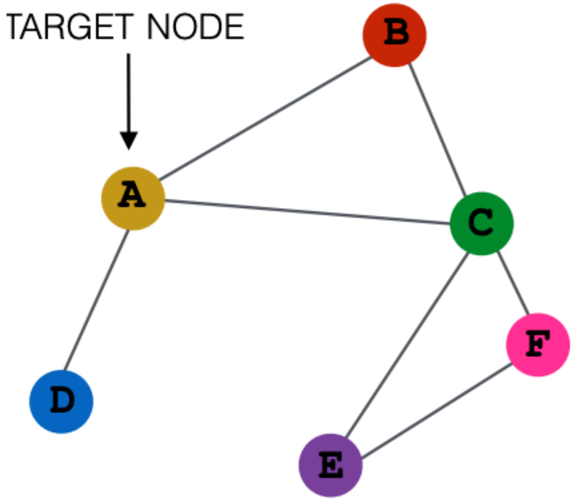
Each node



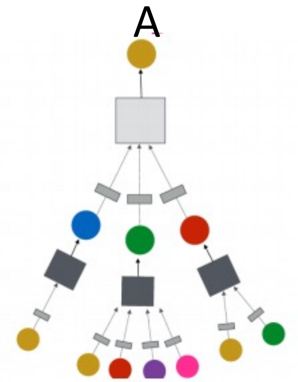
| Binary | Gray code | One-hot |
|---|---|---|
| 000 | 000 | 00000001 |
| 001 | 001 | 00000010 |
| 010 | 011 | 00000100 |
| 011 | 010 | 00001000 |
| 100 | 110 | 00010000 |
| 101 | 111 | 00100000 |
| 110 | 101 | 01000000 |
| 111 | 100 | 10000000 |

Image credit http://web.stanford.edu/class/cs224w/slides/03-nodeemb.pdf

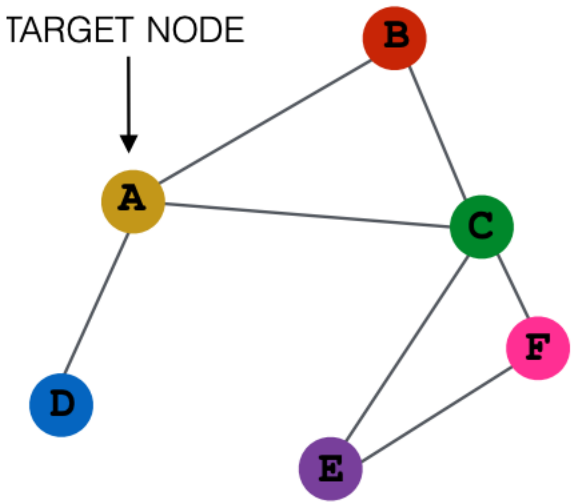
# Graph networks: aggregate neighbors



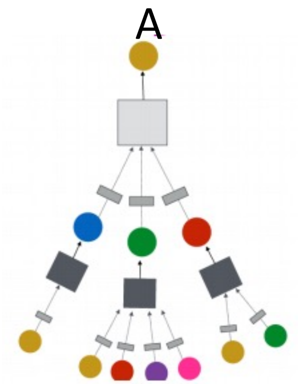
INPUT GRAPH

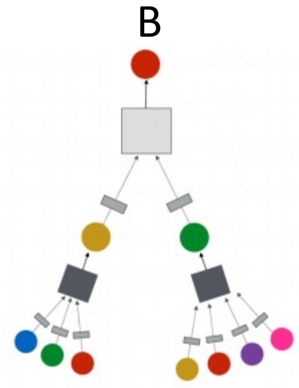


# Graph networks: aggregate neighbors

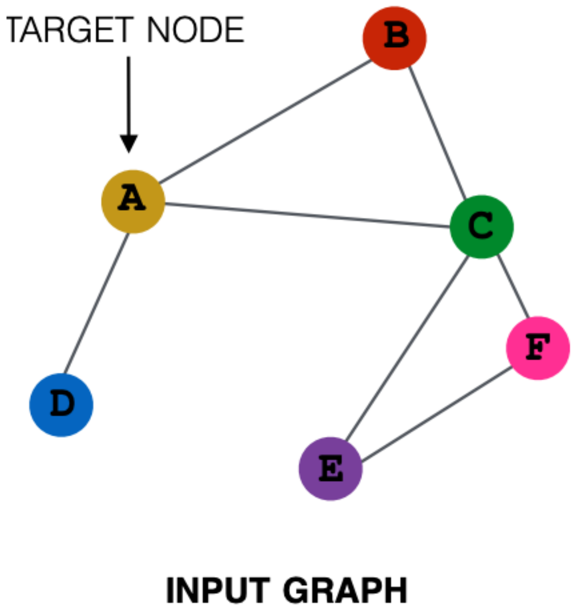


INPUT GRAPH

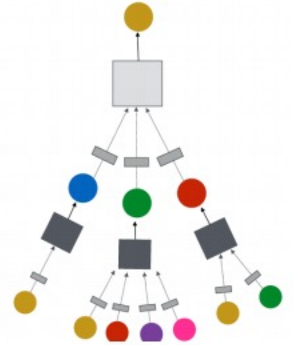




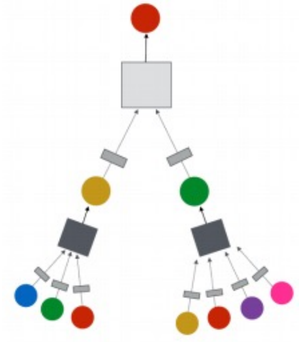
# Graph networks: aggregate neighbors



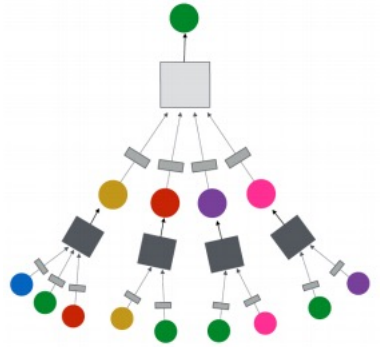
A



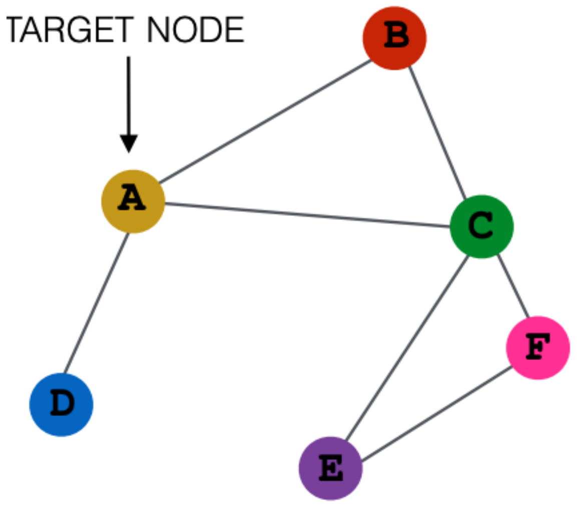
B



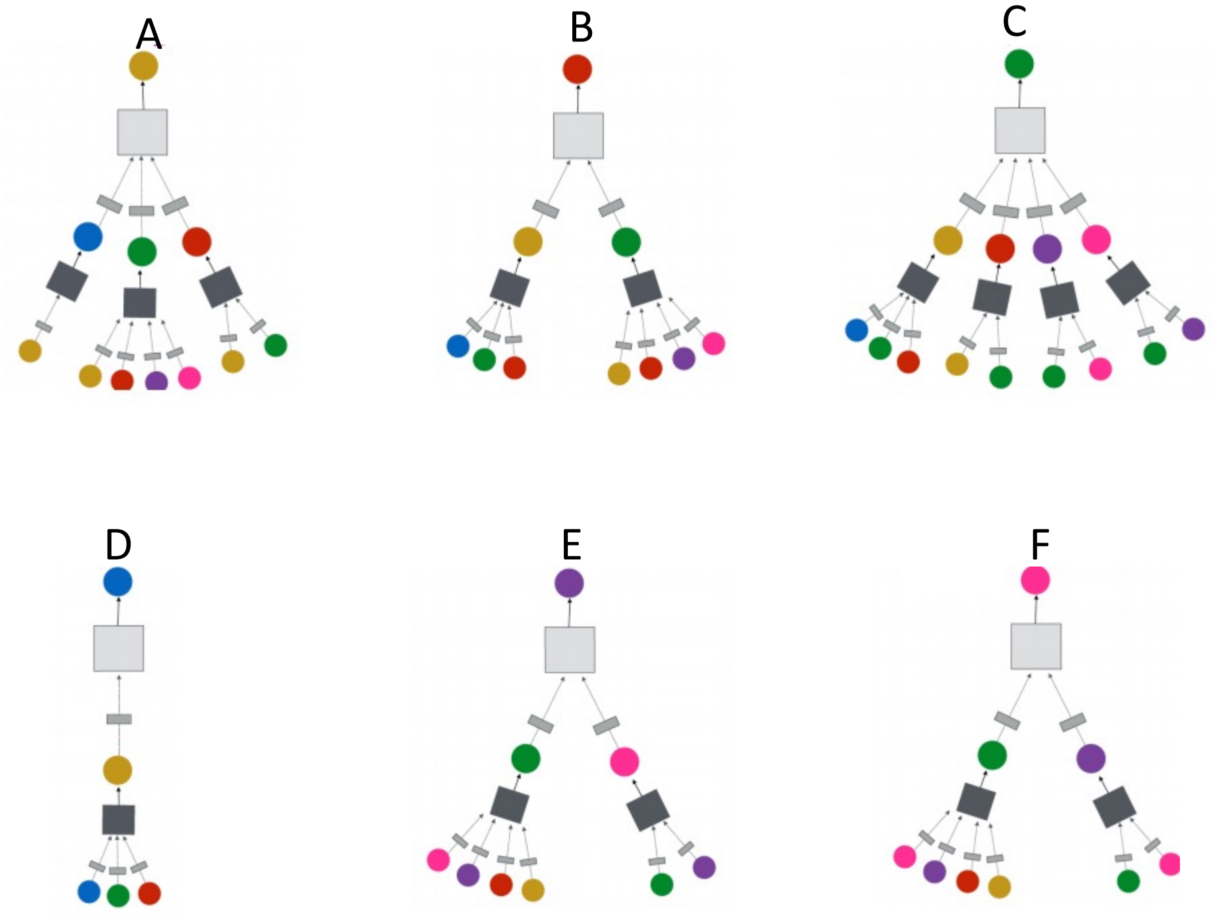
C



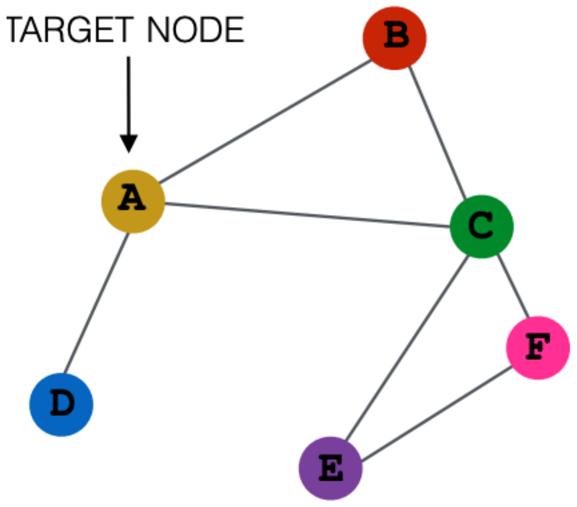
# Graph networks: aggregate neighbors



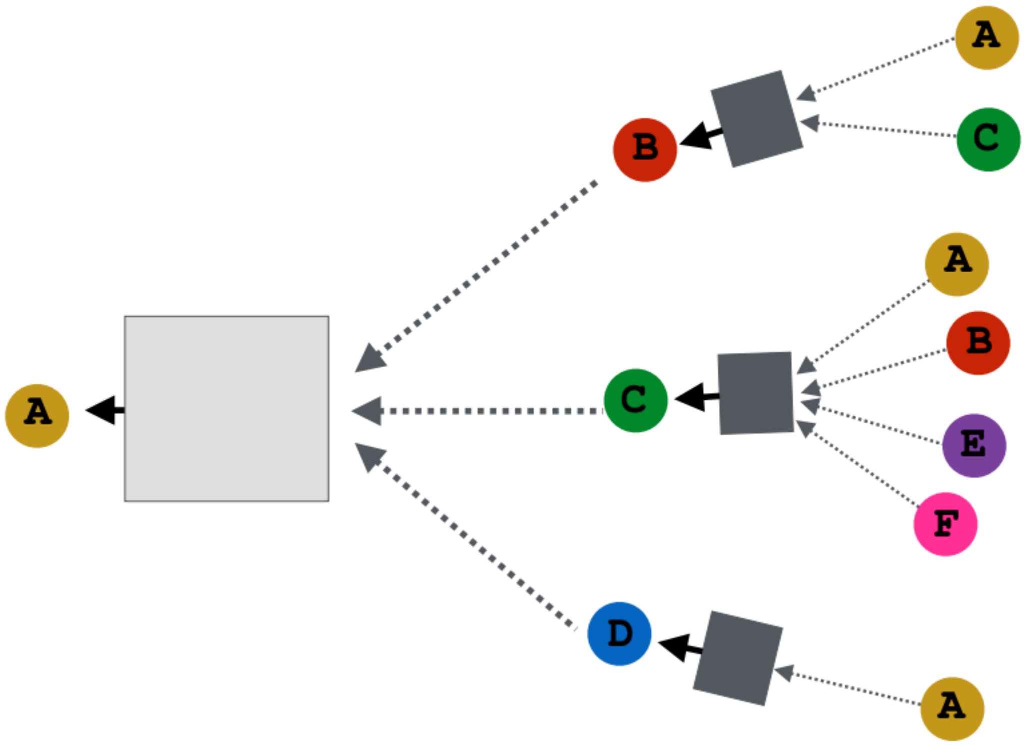
INPUT GRAPH



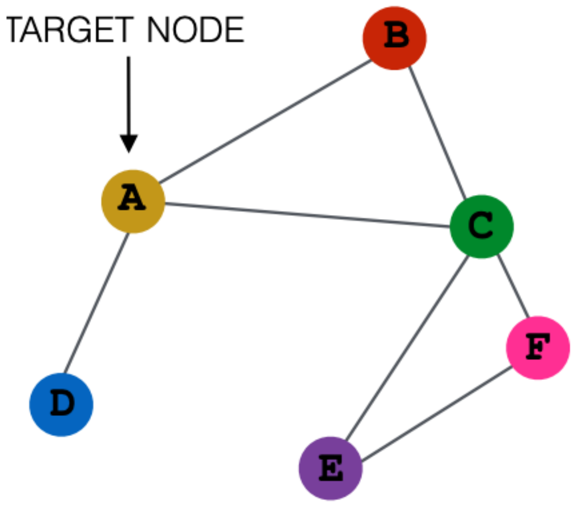
# Graph networks: aggregate neighbors



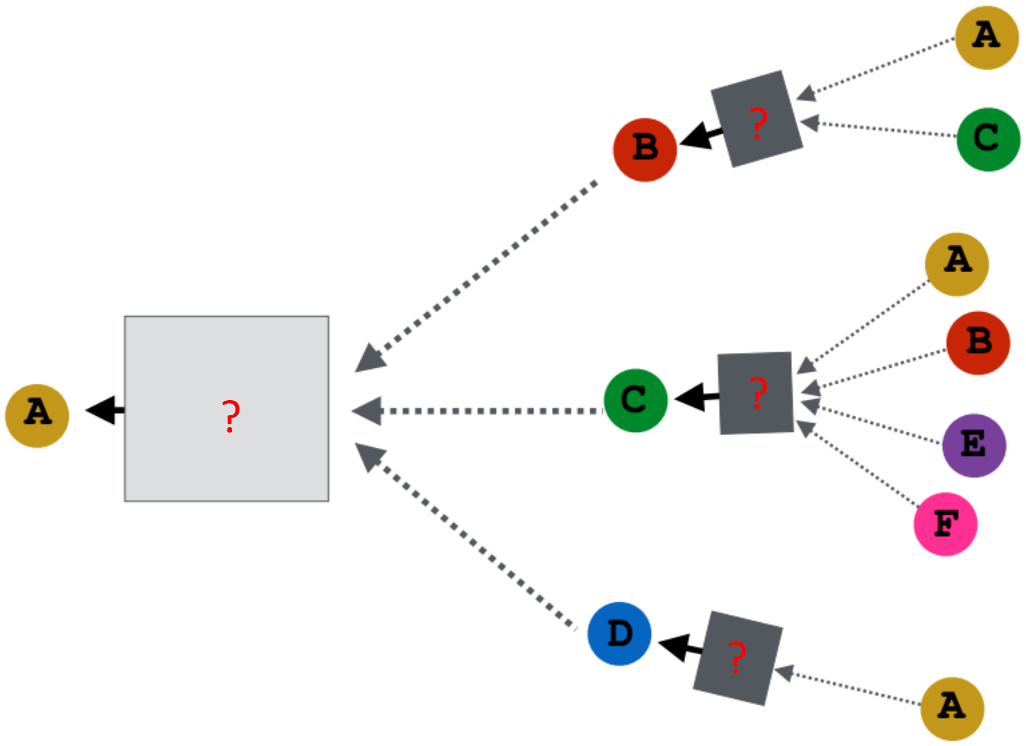
INPUT GRAPH



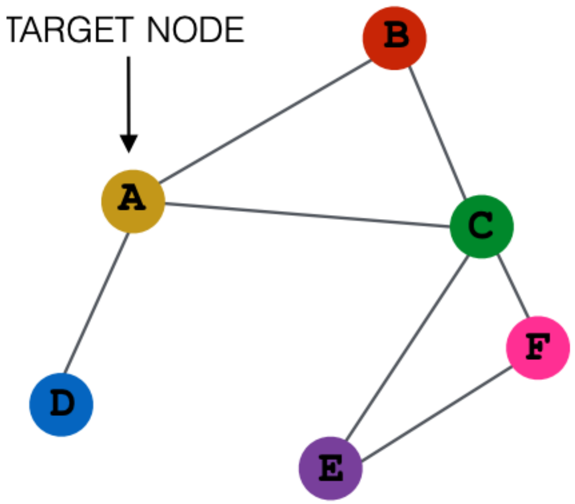
# Graph networks: aggregate neighbors



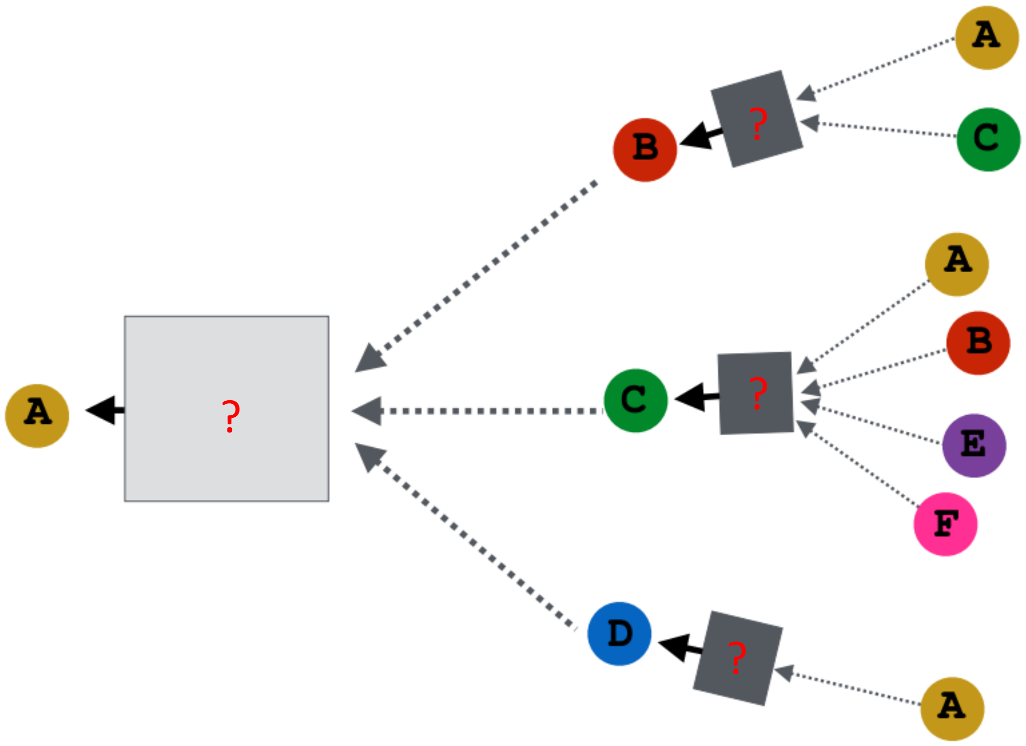
INPUT GRAPH



# Graph networks: aggregate neighbors

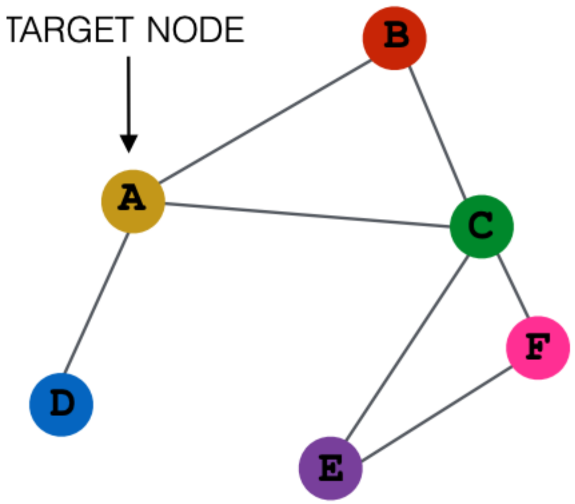


INPUT GRAPH

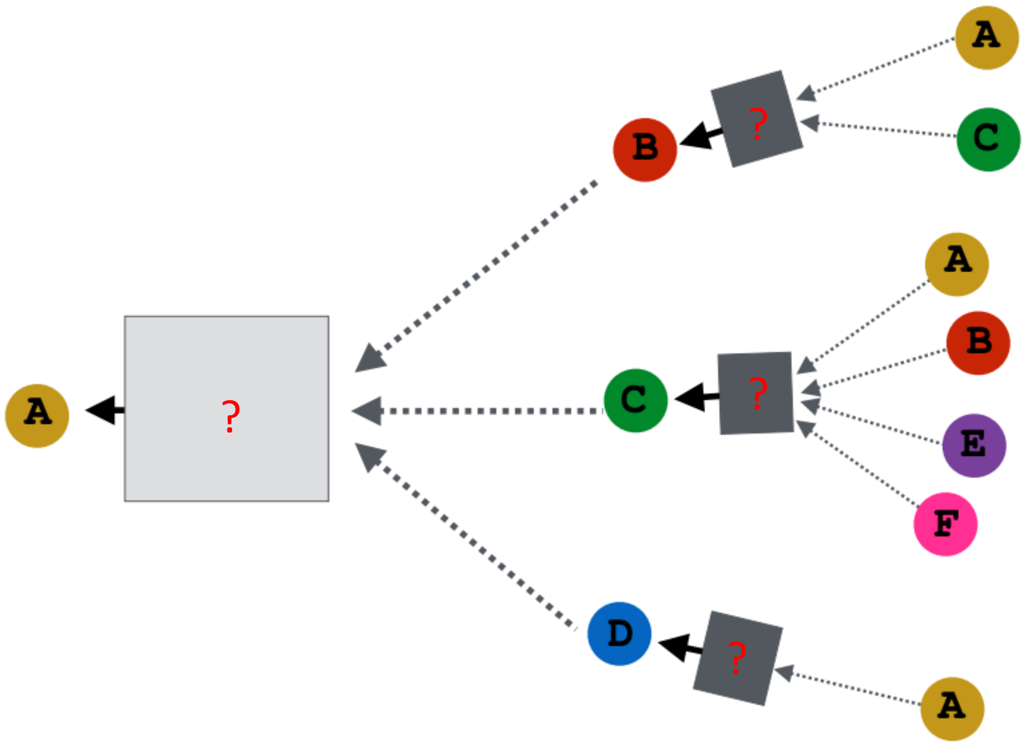


Q: what can we do in the box to aggregate information?

# Graph networks: aggregate neighbors

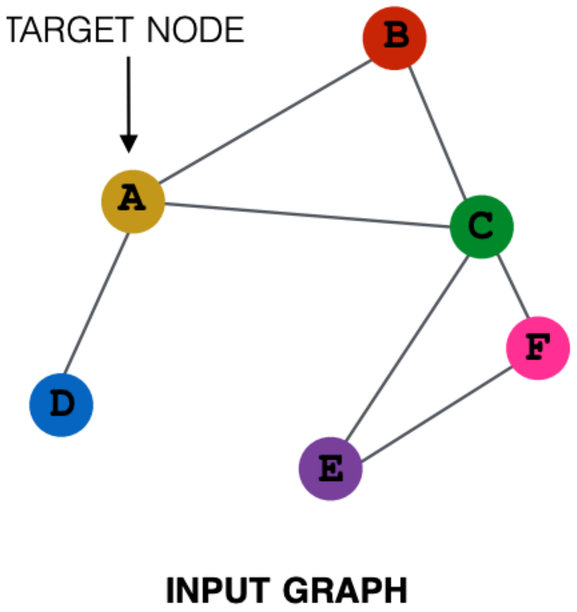


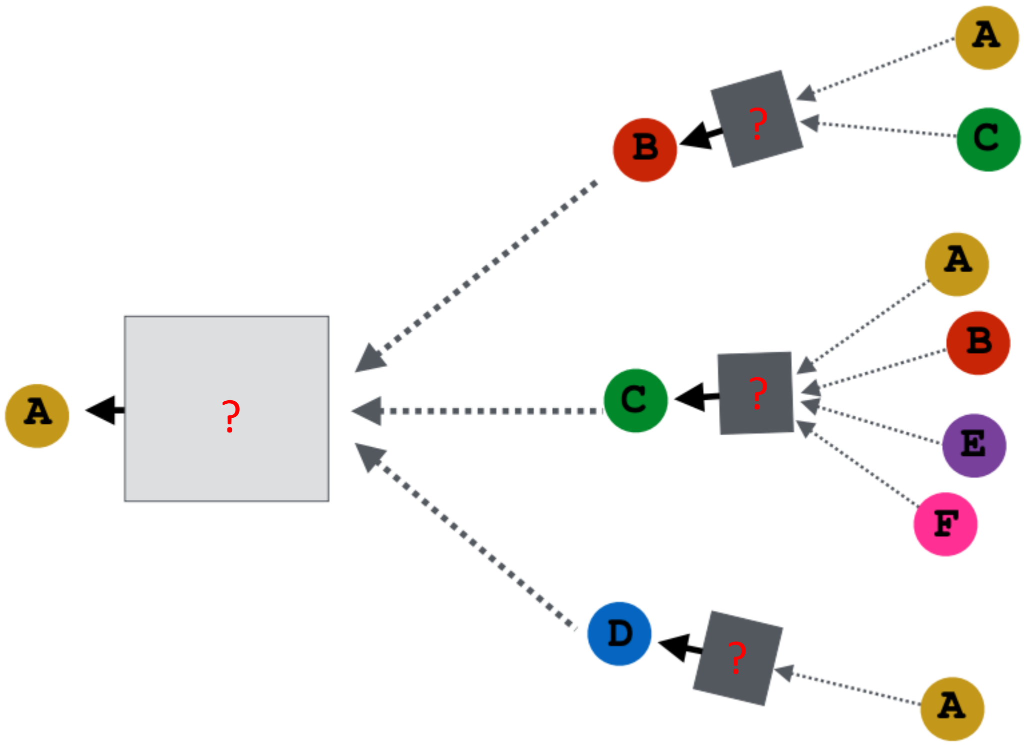
INPUT GRAPH



Q: what can we do in the box to aggregate information?
Average/summation?

# Graph networks: aggregate neighbors

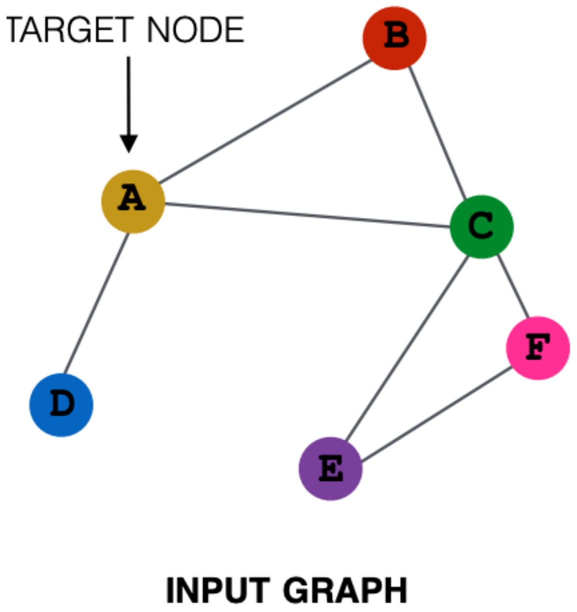


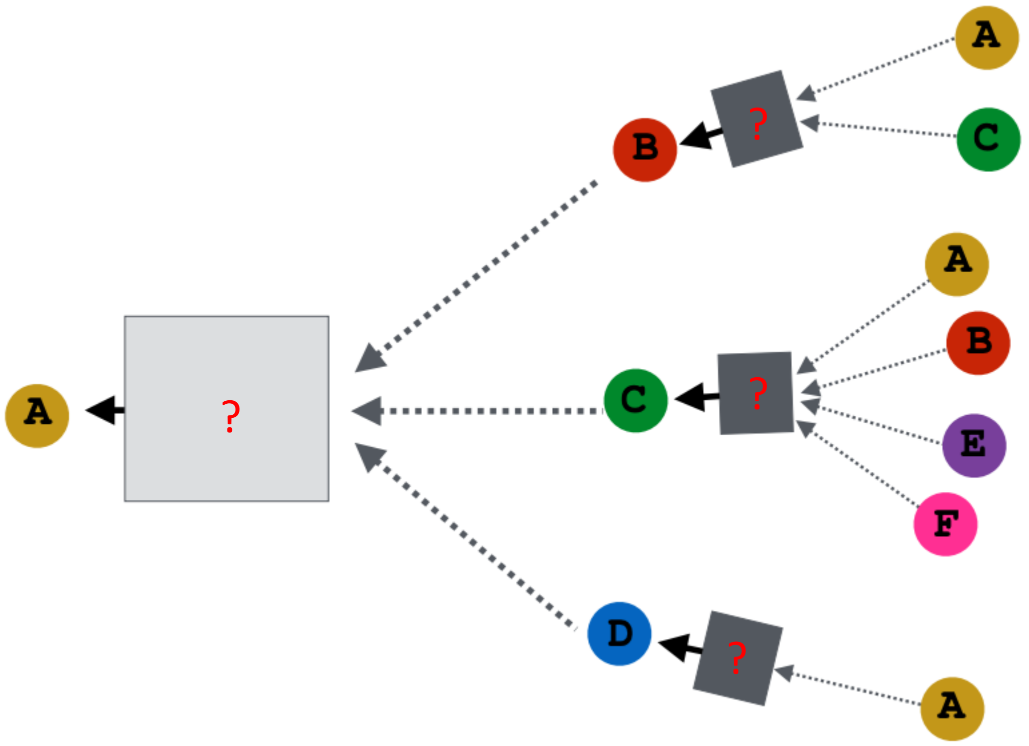


Q: what can we do in the box to aggregate information?

Average/summation → linear model

# Graph networks: aggregate neighbors





Q: what can we do in the box to aggregate information?
Average/summation → linear model
A neural network nonlinear layer?

# Graph networks: aggregate neighbors

Initial 0-th layer embeddings are equal to node features

$$\boxed{h_v^0 = x_v}$$

embedding of $v$ at layer $l$

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

$$\boxed{z_v} = h_v^{(L)}$$

Average of neighbor’s previous layer embeddings

Total number of layers

Embedding after L layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

# Graph networks: aggregate neighbors

Node $v$

$$\boxed{\mathrm{h}_v^0 = \boxed{\mathrm{x}_v}}$$

Initial 0-th layer embeddings are equal to node features

embedding of $v$ at layer $l$

$$\mathrm{h}_v^{(l+1)} = \boxed{\sigma}\left(\mathrm{W}_l \boxed{\sum_{u \in \mathrm{N}(v)} \frac{\mathrm{h}_u^{(l)}}{|\mathrm{N}(v)|}} + \mathrm{B}_l \boxed{\mathrm{h}_v^{(l)}}\right), \forall l \in \{0, \ldots, \boxed{L} - 1\}$$

$$\boxed{\mathrm{z}_v} = \mathrm{h}_v^{(L)}$$

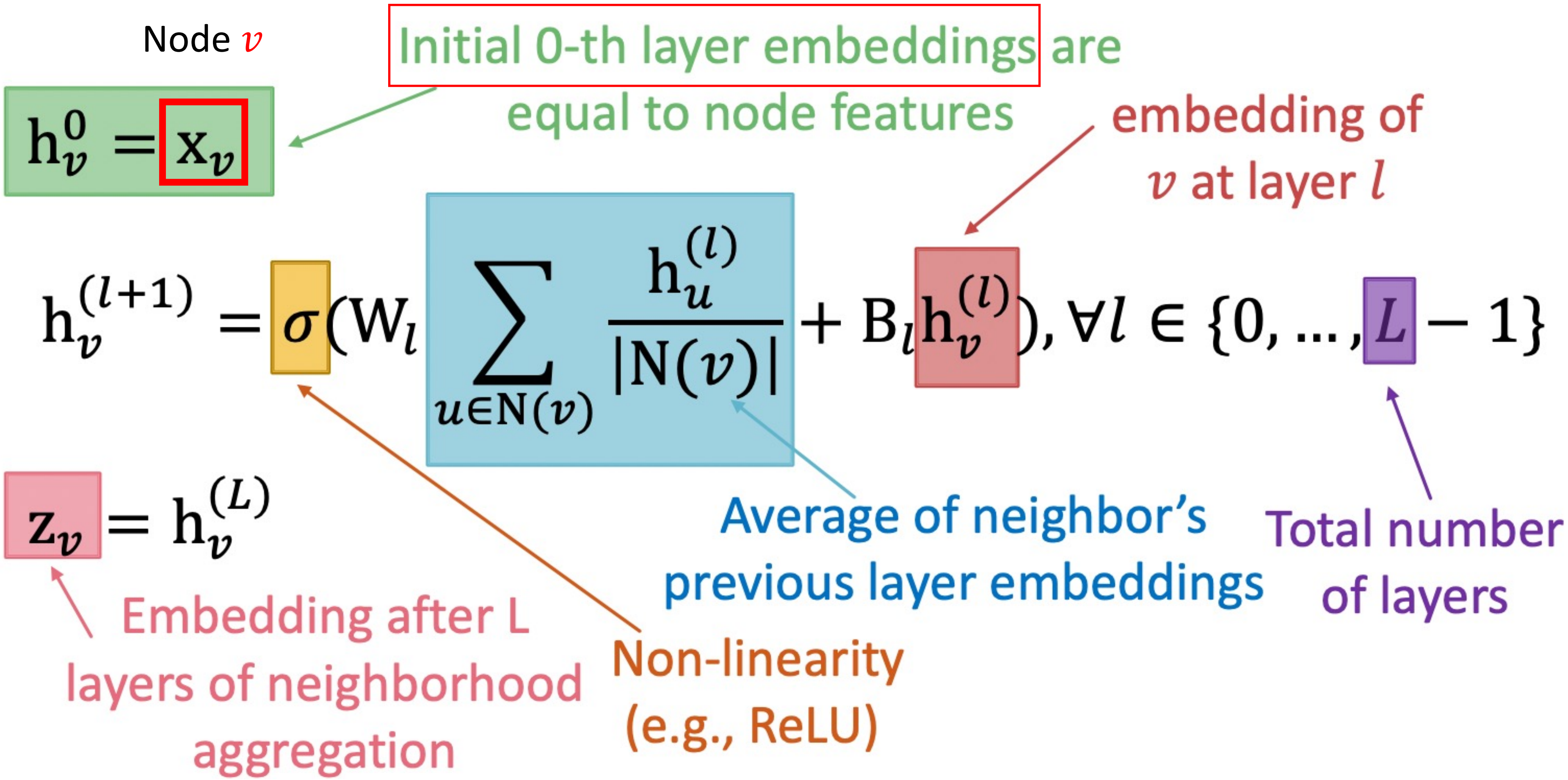
Average of neighbor's previous layer embeddings

Total number of layers

Embedding after L layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

# Graph networks: aggregate neighbors



Node $v$

$$h_v^0 = \boxed{x_v}$$

Initial 0-th layer embeddings are equal to node features

embedding of $v$ at layer $l$

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

$$z_v = h_v^{(L)}$$

Average of neighbor's previous layer embeddings

Total number of layers

Embedding after L layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

# Graph networks: aggregate neighbors

Node $v$

Initial 0-th layer embeddings are equal to node features

$$\boxed{h_v^0 = x_v}$$

embedding of $v$ at layer $l$

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

$$\boxed{z_v} = h_v^{(L)}$$

Average of neighbor's previous layer embeddings

Total number of layers

Embedding after L layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

# Graph networks: aggregate neighbors

Node $v$

Initial 0-th layer embeddings are equal to node features

$$\boxed{h_v^0 = x_v}$$

embedding of $v$ at layer $l$

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

$$z_v = h_v^{(L)}$$

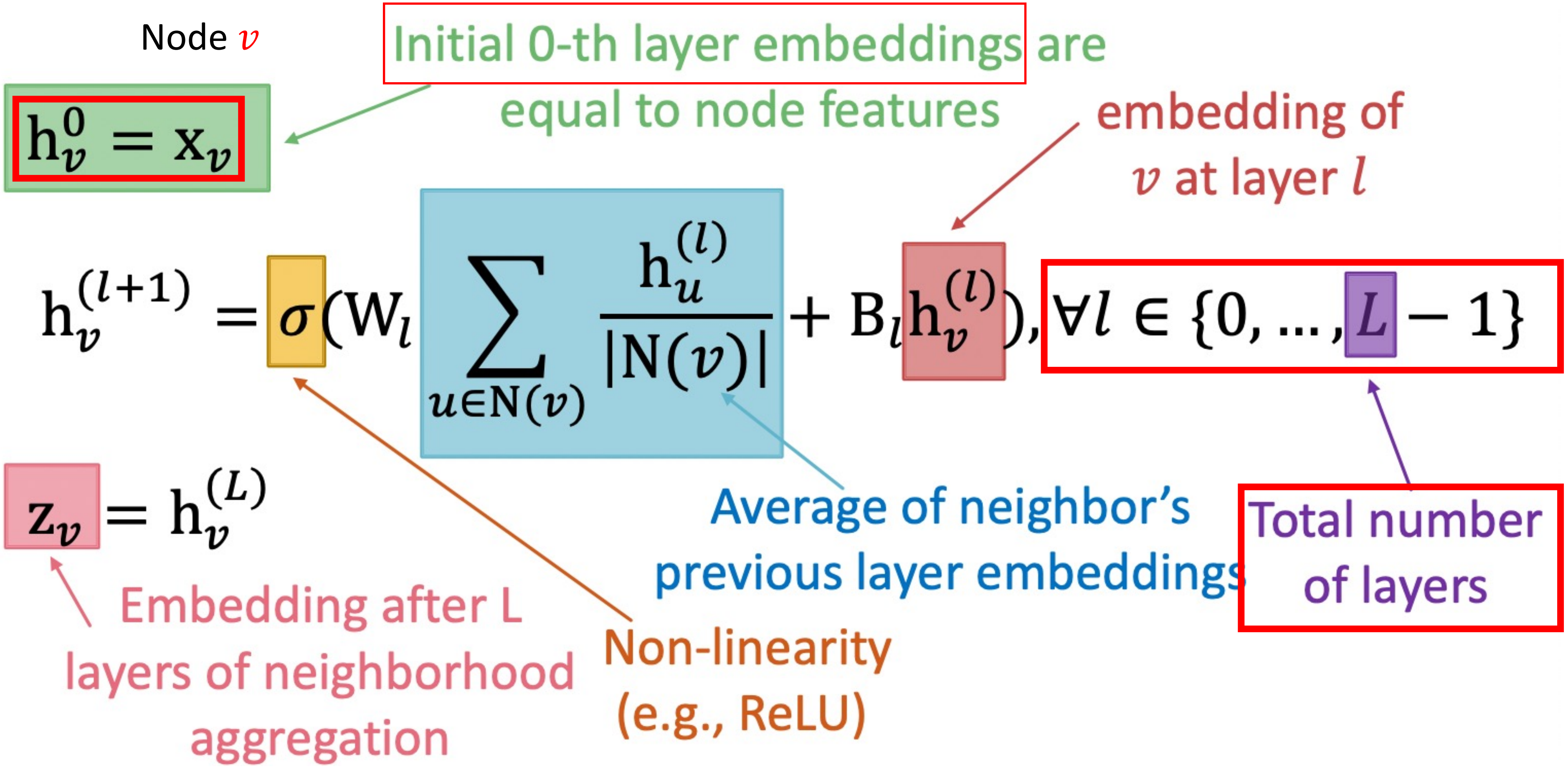
Average of neighbor's previous layer embeddings

Total number of layers

Embedding after L layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

# Graph networks: aggregate neighbors



Node $v$

$$h_v^0 = x_v$$

Initial 0-th layer embeddings are equal to node features

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

embedding of $v$ at layer $l$

$$z_v = h_v^{(L)}$$

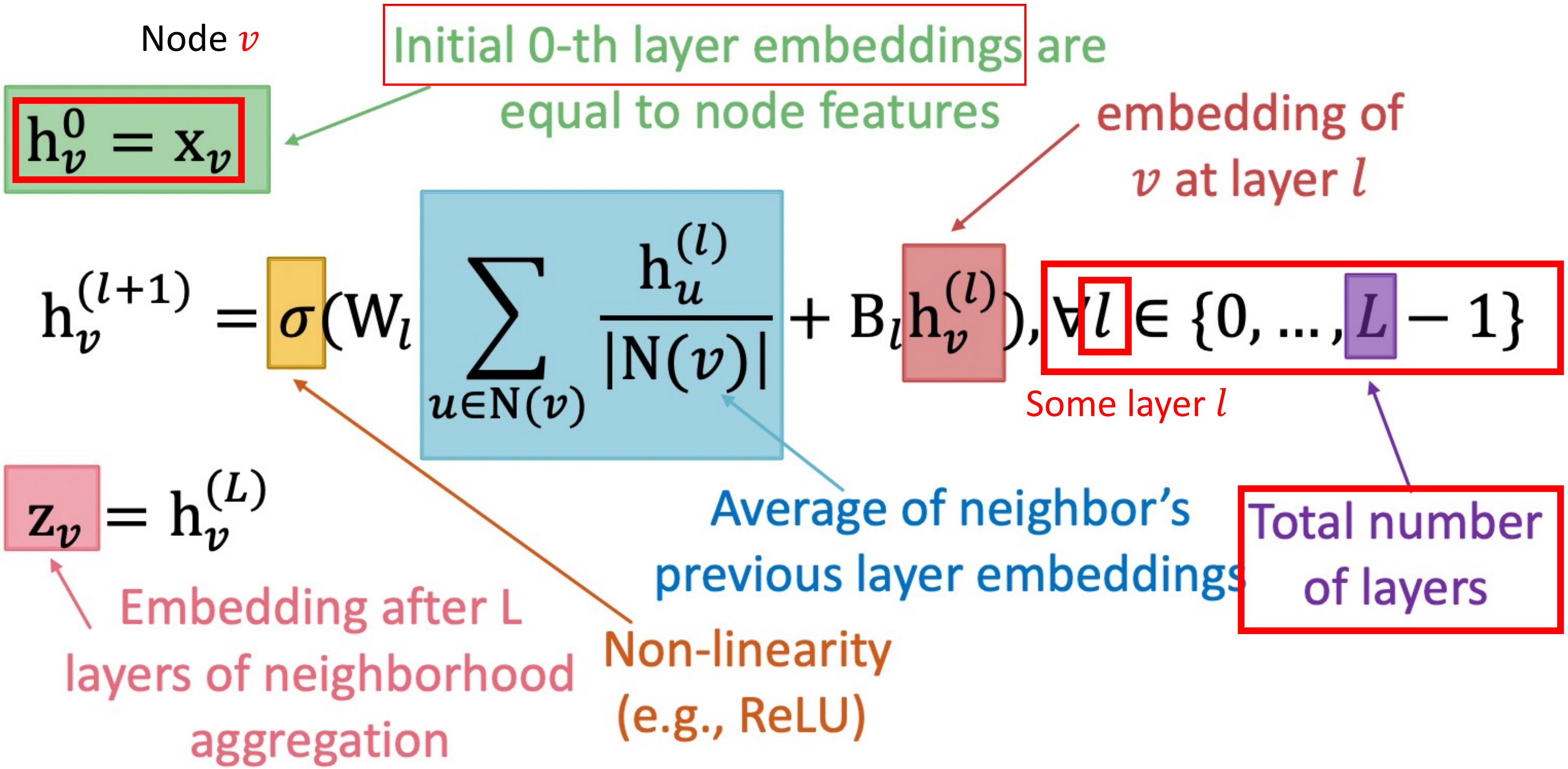
Embedding after L layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

Average of neighbor's previous layer embeddings

Total number of layers

# Graph networks: aggregate neighbors



Node $v$

$$\mathrm{h}_v^0 = \mathrm{x}_v$$

Initial 0-th layer embeddings are equal to node features

$$\mathrm{h}_v^{(l+1)} = \sigma\left(\mathrm{W}_l \sum_{u \in \mathrm{N}(v)} \frac{\mathrm{h}_u^{(l)}}{|\mathrm{N}(v)|} + \mathrm{B}_l \mathrm{h}_v^{(l)}\right), \forall l \in \{0, \dots, L - 1\}$$

embedding of $v$ at layer $l$

Some layer $l$

Total number of layers

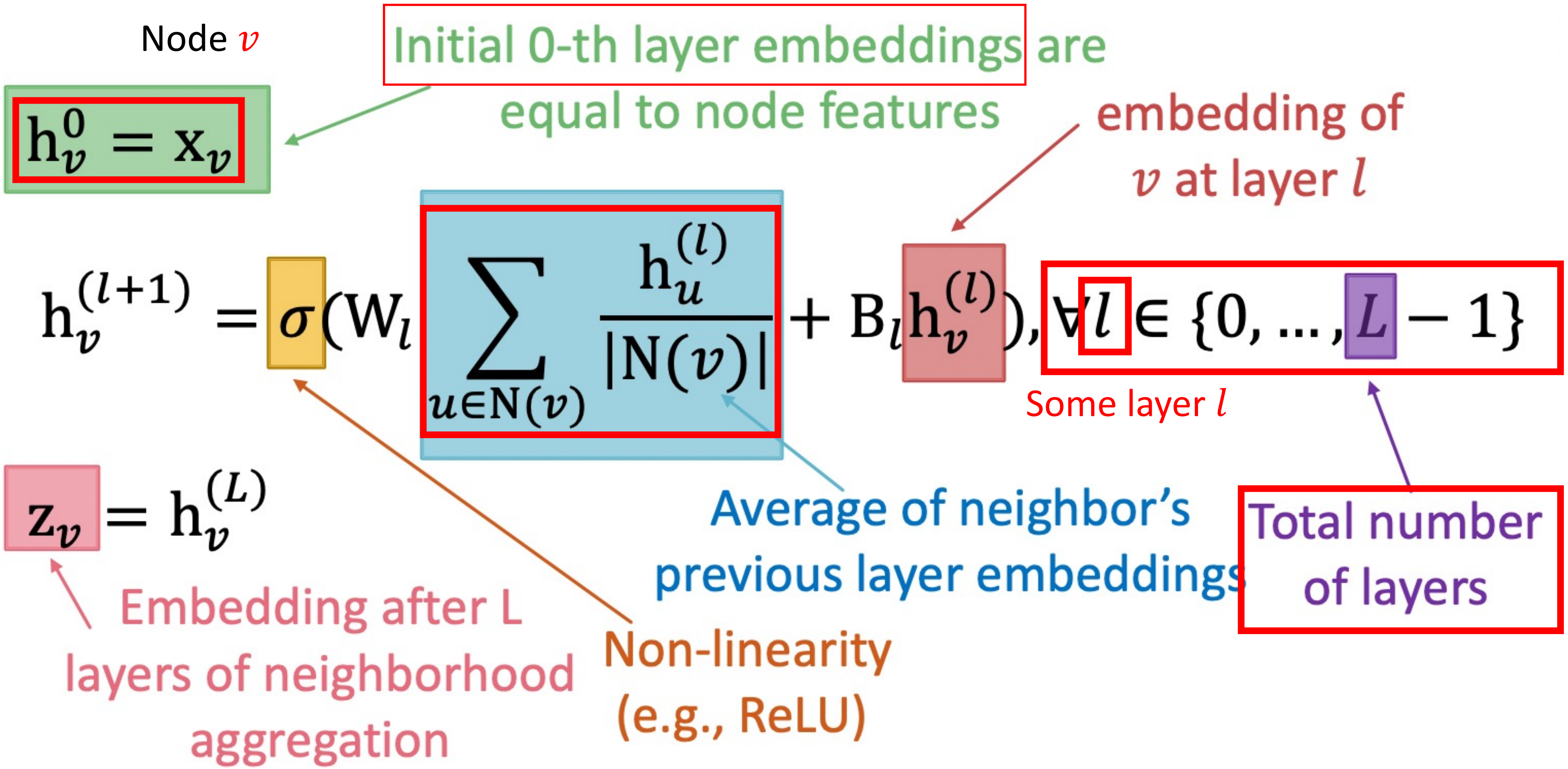
Average of neighbor's previous layer embeddings

Non-linearity (e.g., ReLU)

$$\mathrm{z}_v = \mathrm{h}_v^{(L)}$$

Embedding after L layers of neighborhood aggregation

# Graph networks: aggregate neighbors



Node $v$

Initial 0-th layer embeddings are equal to node features

$$h_v^0 = x_v$$

embedding of $v$ at layer $l$

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

Some layer $l$

$$z_v = h_v^{(L)}$$

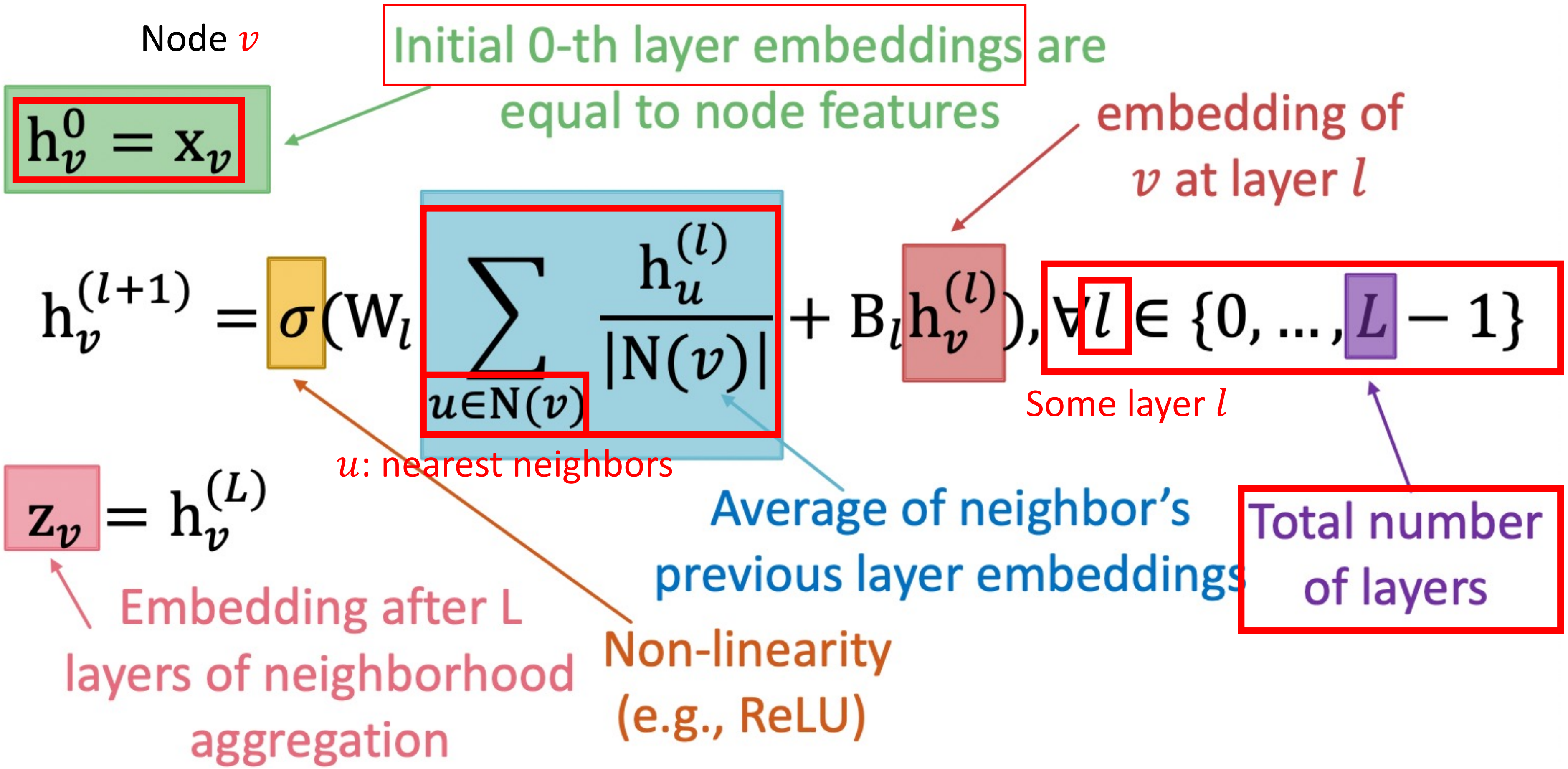
Embedding after L layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

Average of neighbor's previous layer embeddings

Total number of layers

# Graph networks: aggregate neighbors



Node $v$

$$\boxed{h_v^0 = x_v}$$

Initial 0-th layer embeddings are equal to node features

embedding of $v$ at layer $l$

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

Some layer $l$

$u$: nearest neighbors

$$z_v = h_v^{(L)}$$

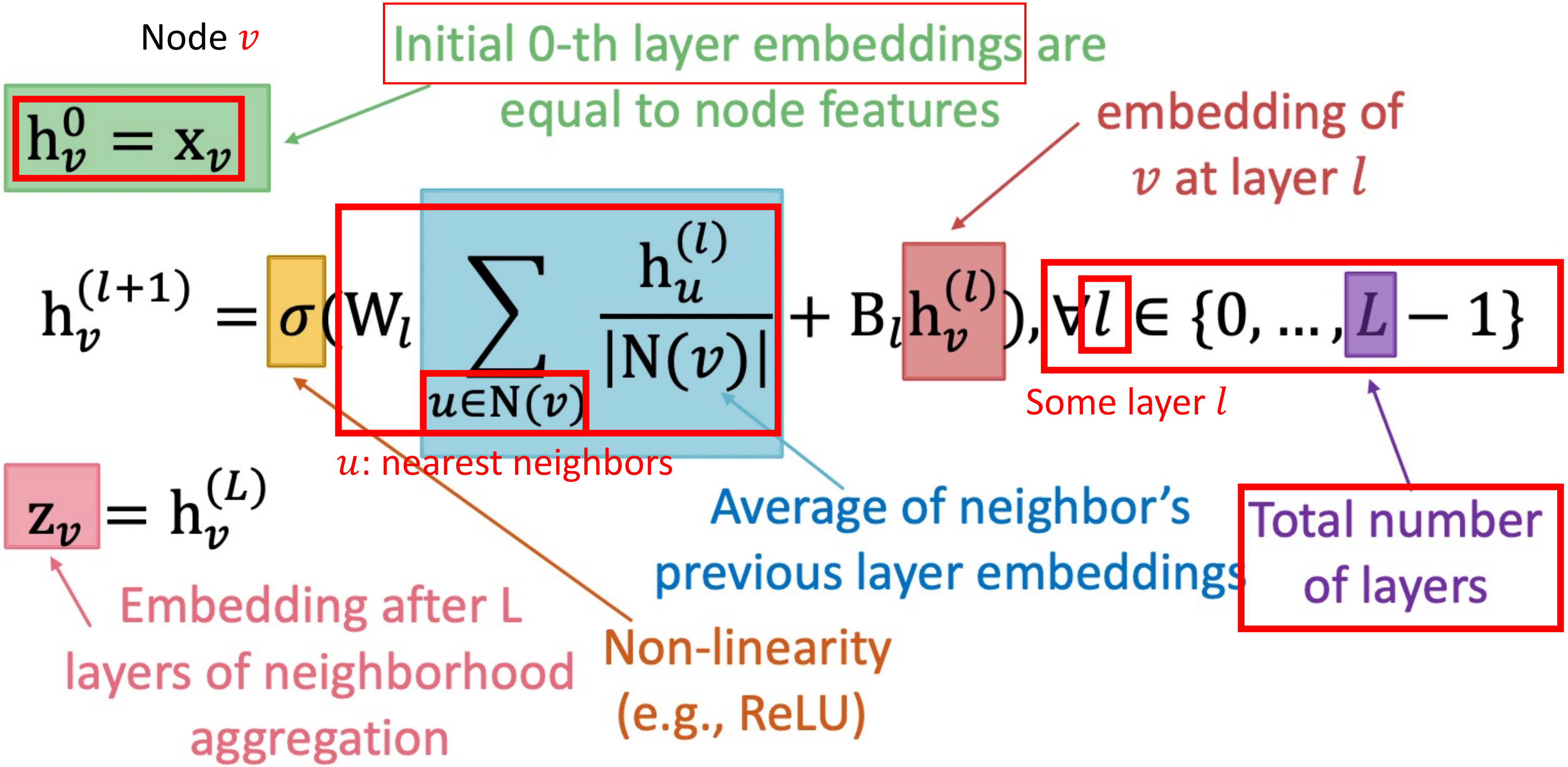
Average of neighbor's previous layer embeddings

Total number of layers

Embedding after L layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

# Graph networks: aggregate neighbors



Node $v$

$$h_v^0 = x_v$$

Initial 0-th layer embeddings are equal to node features

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

embedding of $v$ at layer $l$

Some layer $l$

$u$: nearest neighbors

Average of neighbor's previous layer embeddings

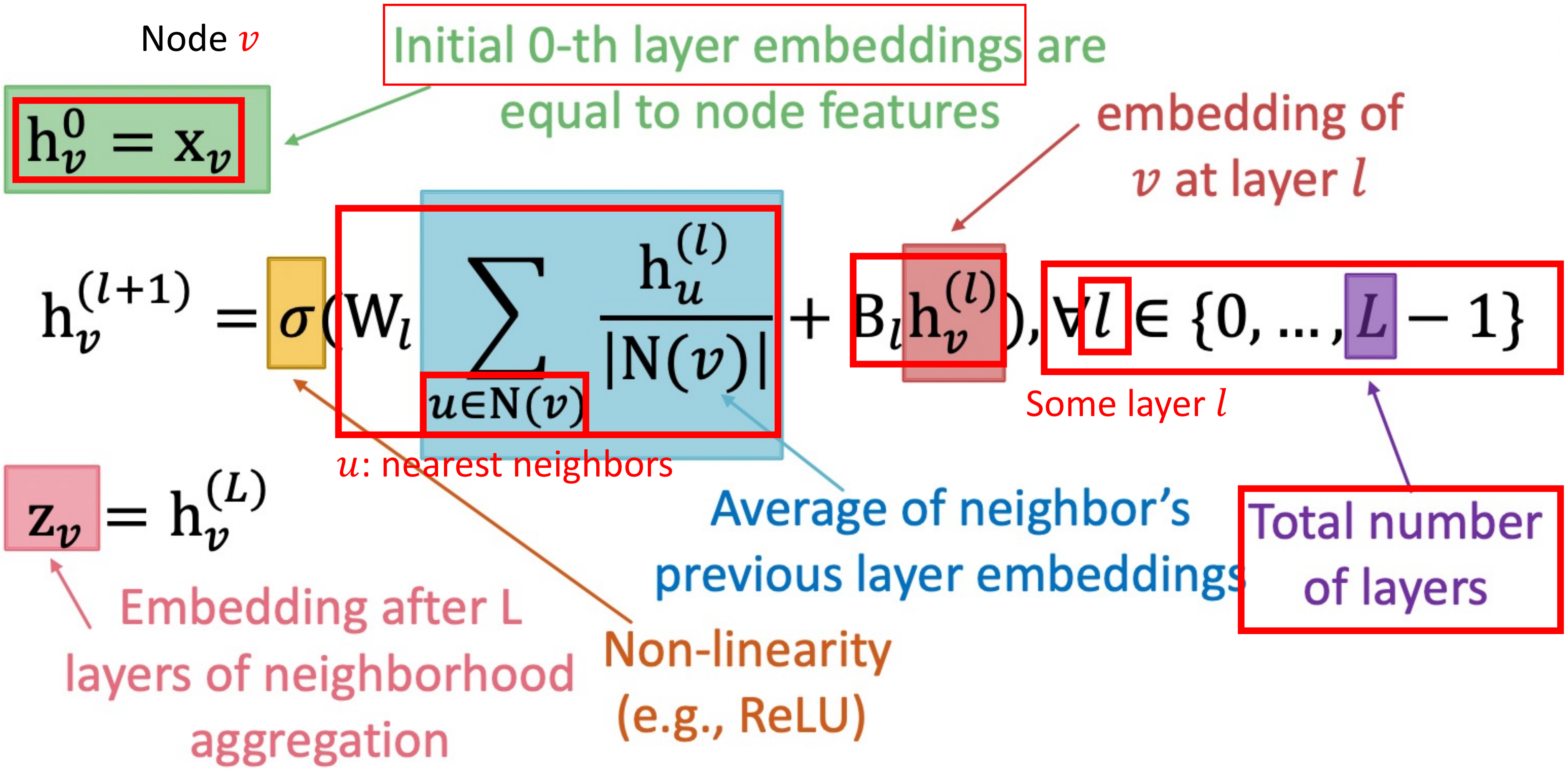
Total number of layers

$$z_v = h_v^{(L)}$$

Embedding after L layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

# Graph networks: aggregate neighbors



Node $v$

$$h_v^0 = x_v$$

Initial 0-th layer embeddings are equal to node features

embedding of $v$ at layer $l$

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

Some layer $l$

$u$: nearest neighbors

$$z_v = h_v^{(L)}$$

Average of neighbor's previous layer embeddings

Total number of layers

Embedding after L layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

# Graph networks: aggregate neighbors

Node $v$

$$h_v^0 = x_v$$

Initial 0-th layer embeddings are equal to node features

embedding of $v$ at layer $l$

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

Some layer $l$

$u$: nearest neighbors

$$z_v = h_v^{(L)}$$

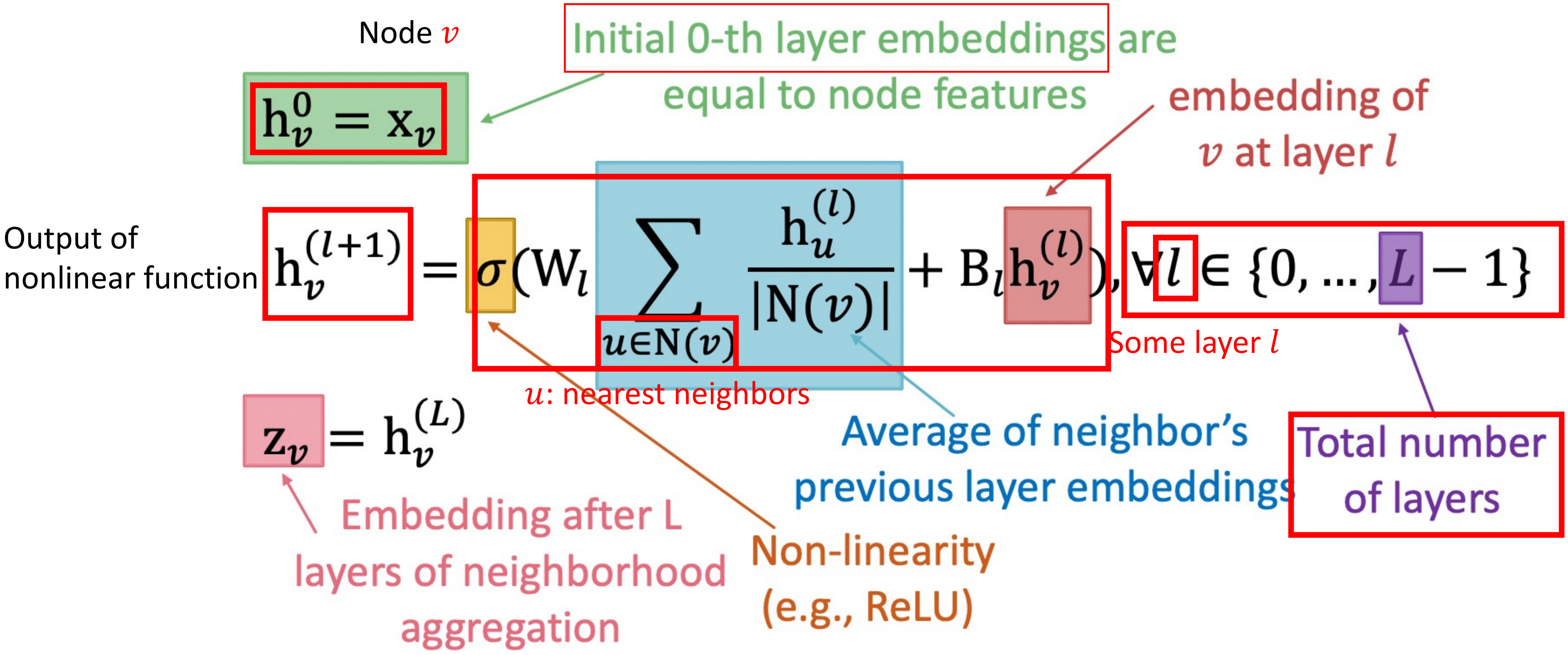
Average of neighbor's previous layer embeddings

Total number of layers

Embedding after L layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

# Graph networks: aggregate neighbors



Node $v$

Initial 0-th layer embeddings are equal to node features

$$h_v^0 = x_v$$

embedding of $v$ at layer $l$

Output of nonlinear function

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

Some layer $l$

$u$: nearest neighbors

$$z_v = h_v^{(L)}$$

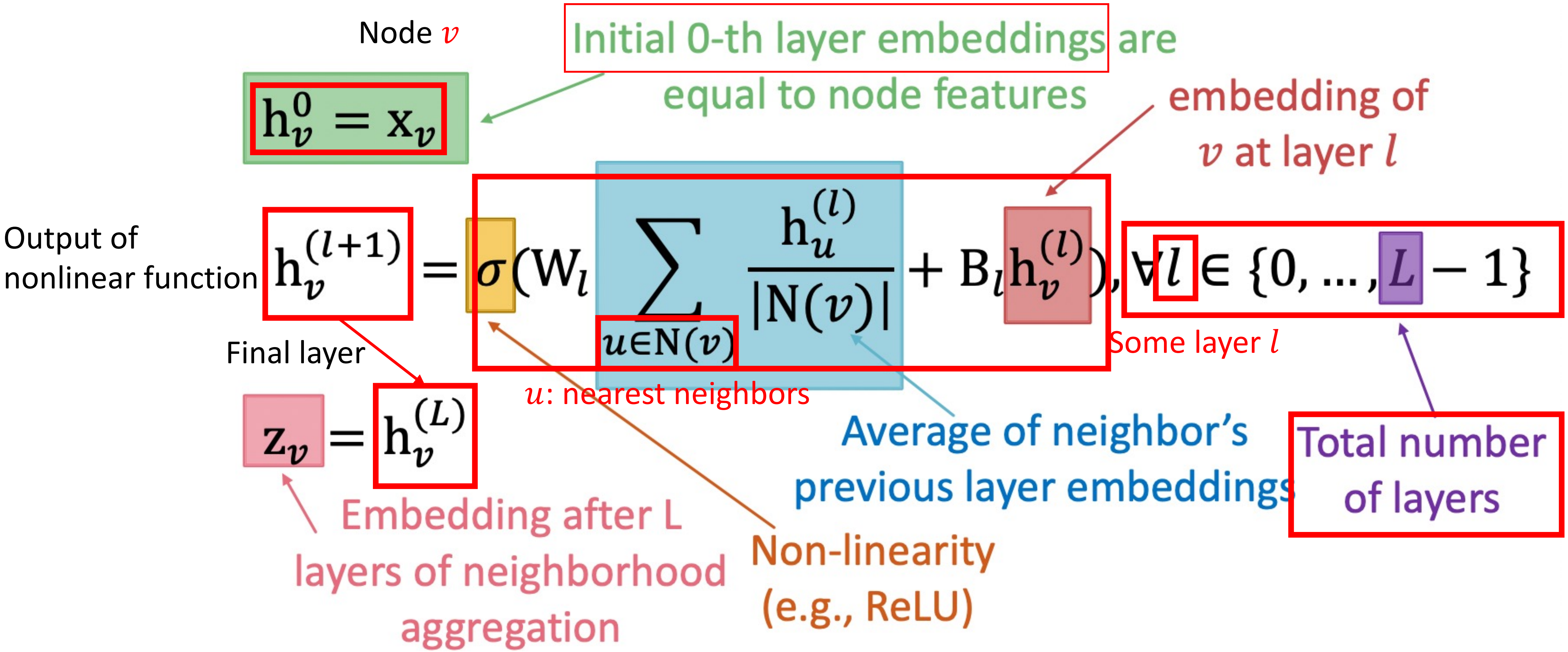
Average of neighbor's previous layer embeddings

Total number of layers

Embedding after L layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

# Graph networks: aggregate neighbors



Node $v$

Initial 0-th layer embeddings are equal to node features

$$h_v^0 = x_v$$

embedding of $v$ at layer $l$

Output of nonlinear function

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

Some layer $l$

Final layer

$$z_v = h_v^{(L)}$$

$u$: nearest neighbors

Average of neighbor's previous layer embeddings

Total number of layers

Embedding after L layers of neighborhood aggregation

Non-linearity (e.g., ReLU)