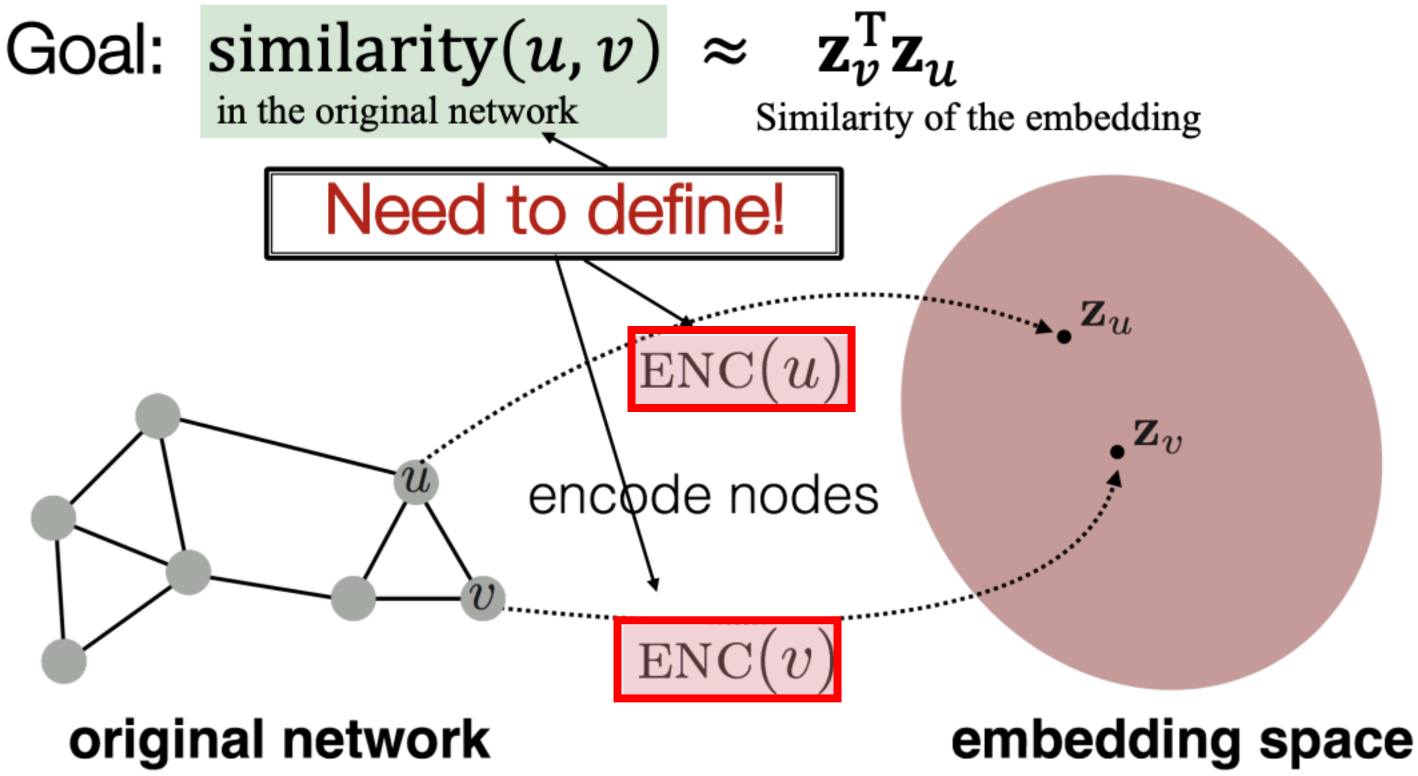
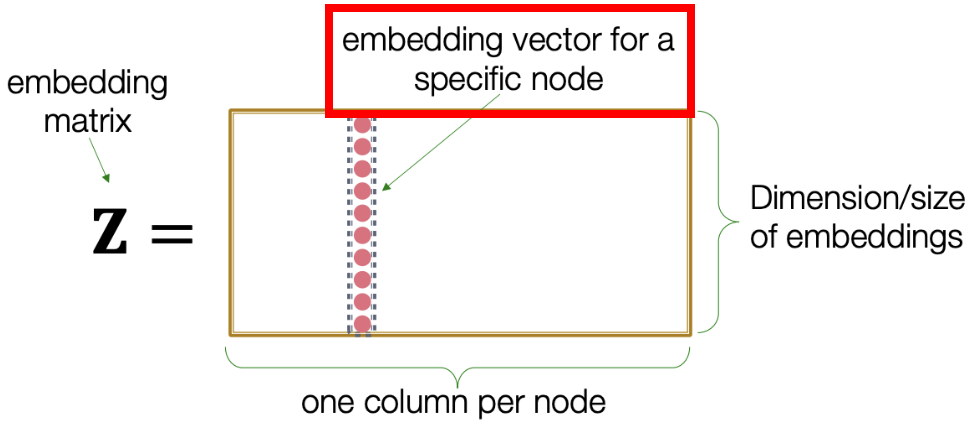
Graph Neural Networks

Neural Networks Design And Application

Encoder-decoder for graph data



Q: how to learn ENC?

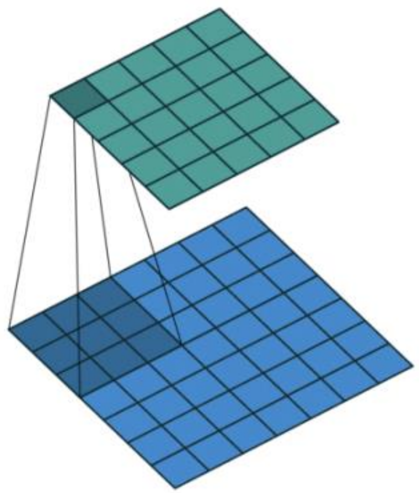


Learnable parameters

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

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

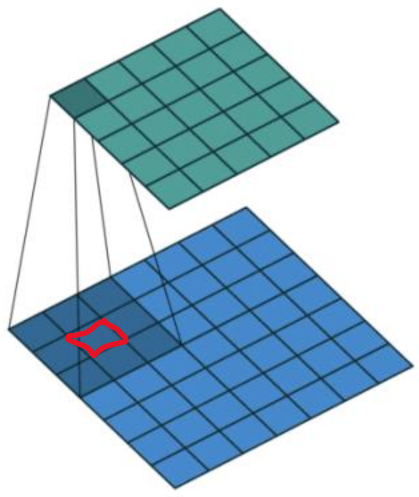
Graph neural networks



Q: can we use convolution operation on graph?

Graph convolutional neural networks

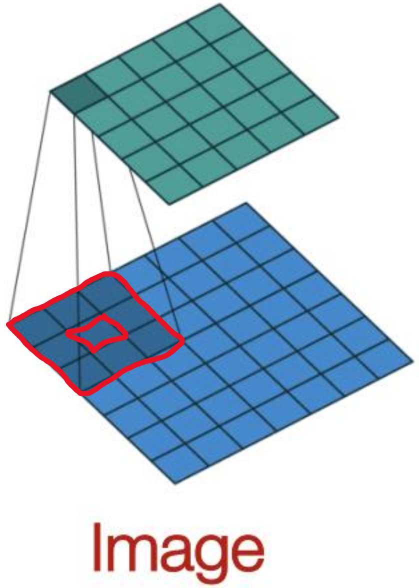
Graph neural networks



Q: can we use convolution operation on graph?

Graph convolutional neural networks

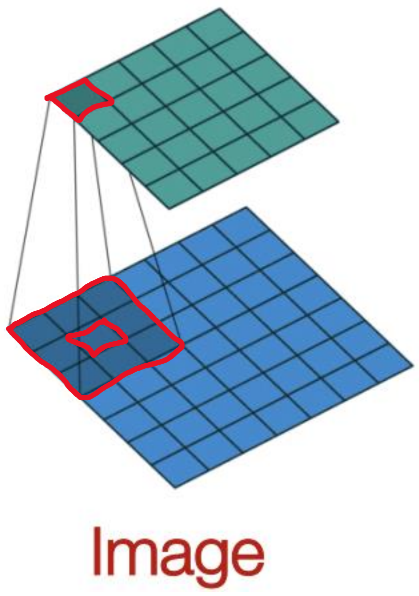
Graph neural networks



Q: can we use convolution operation on graph?

Graph convolutional neural networks

Graph neural networks

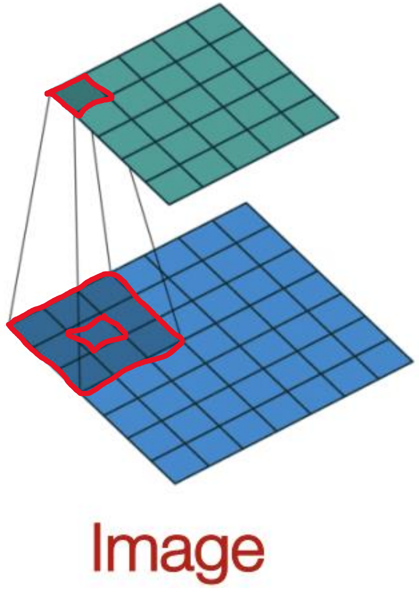


Q: can we use convolution operation on graph?

Graph convolutional neural networks

Graph neural networks

Transform information at neighbors

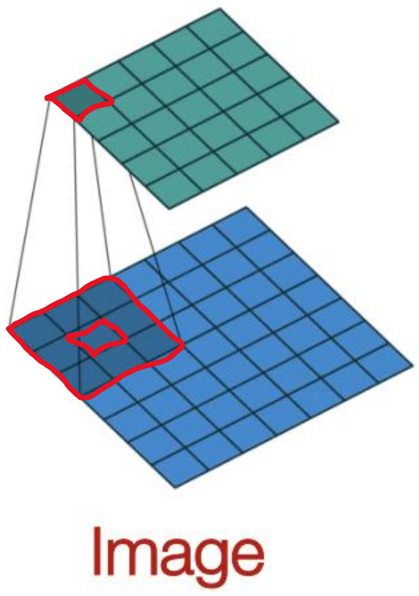


Q: can we use convolution operation on graph?

Graph convolutional neural networks

Graph neural networks

Transform information at neighbors
Combine them



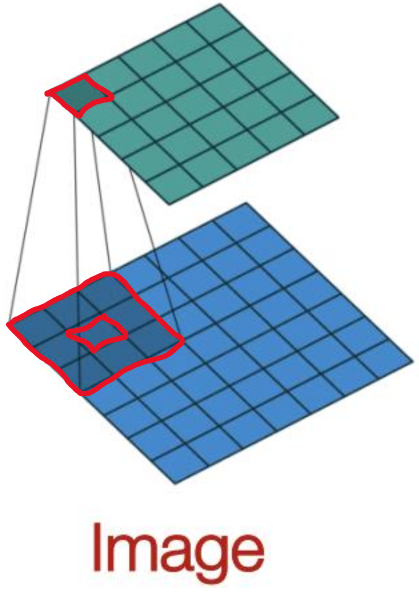
Q: can we use convolution operation on graph?

Graph convolutional neural networks

Graph neural networks

Q: What is information?

Transform information at neighbors
Combine them



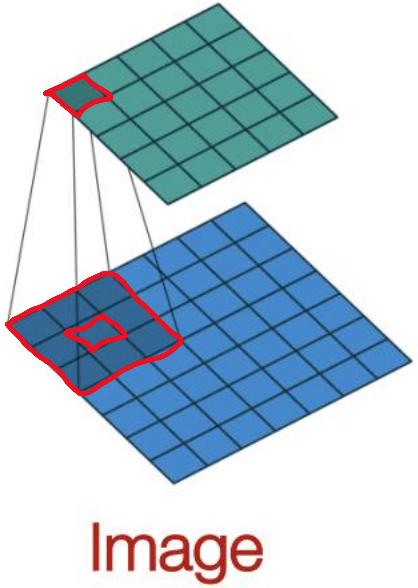
Q: can we use convolution operation on graph?

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



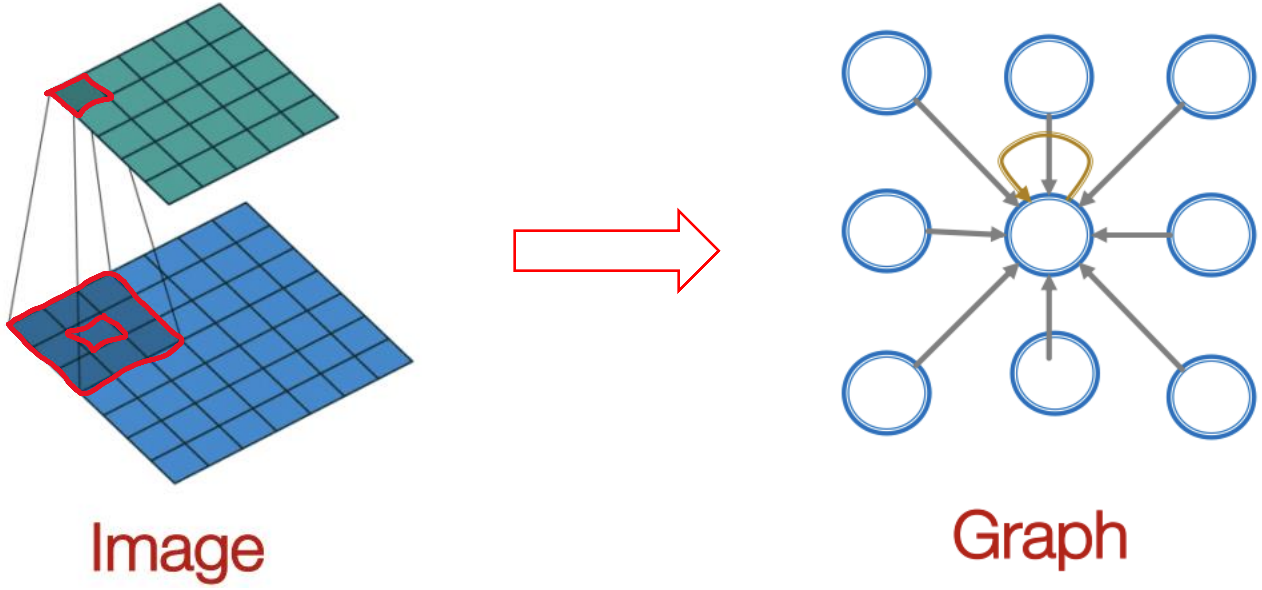
Q: can we use convolution operation on graph?

Graph convolutional neural networks

Graph neural networks

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

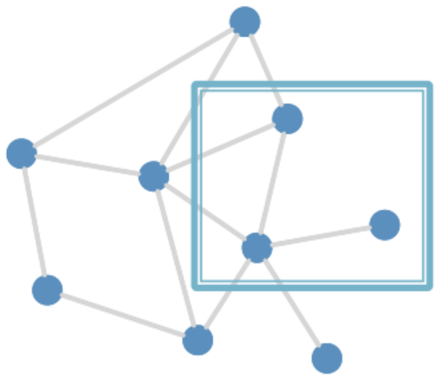
Transform information at neighbors
Combine them



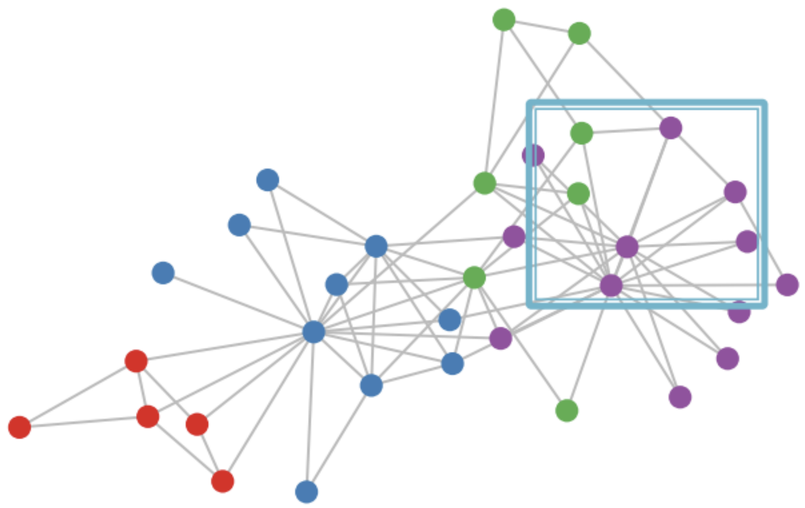
Q: can we use convolution operation on graph?

Graph convolutional neural networks

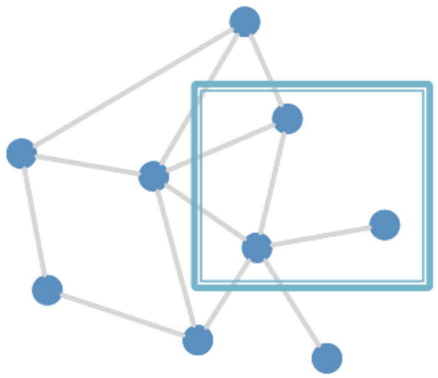
General graph data



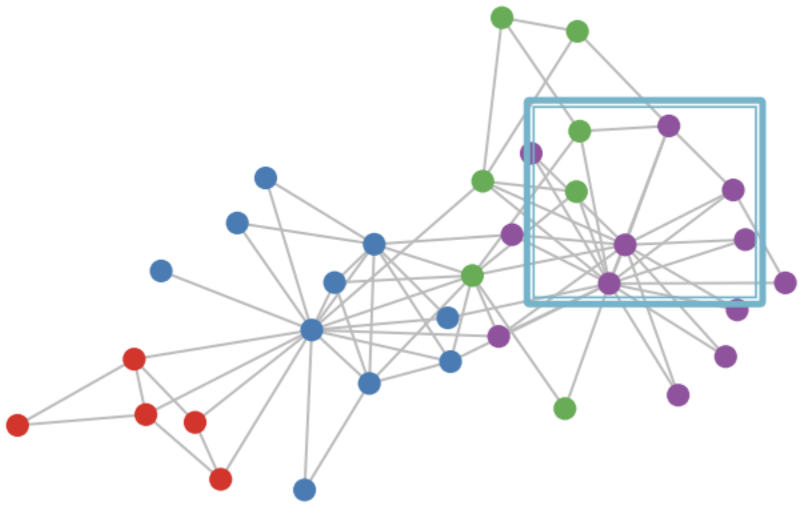
or this:

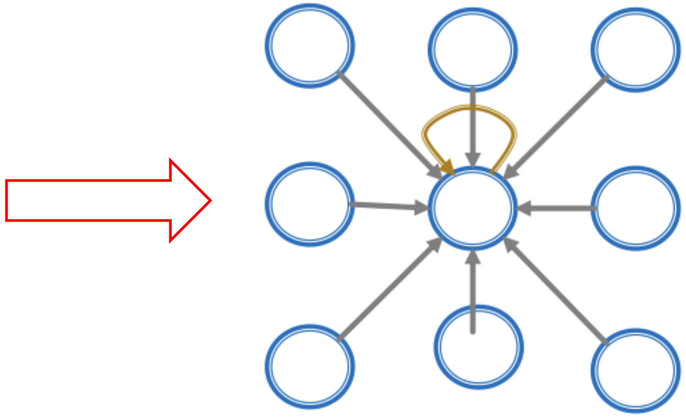


General graph data

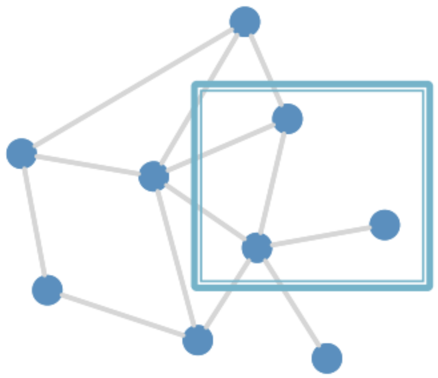


or this:

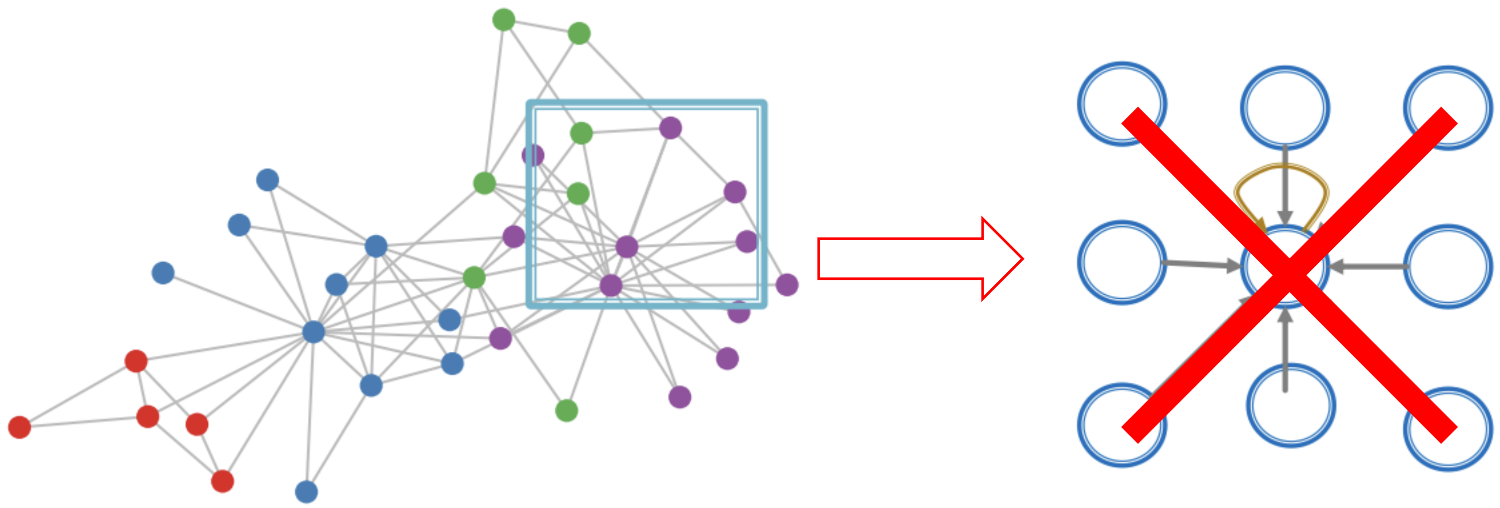




General graph data

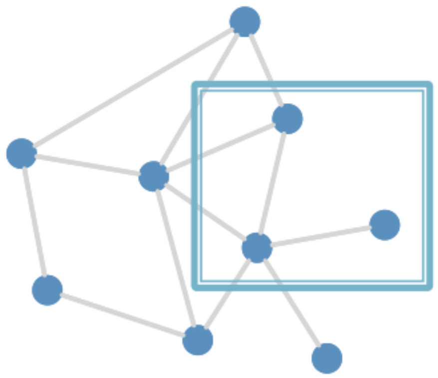


or this:

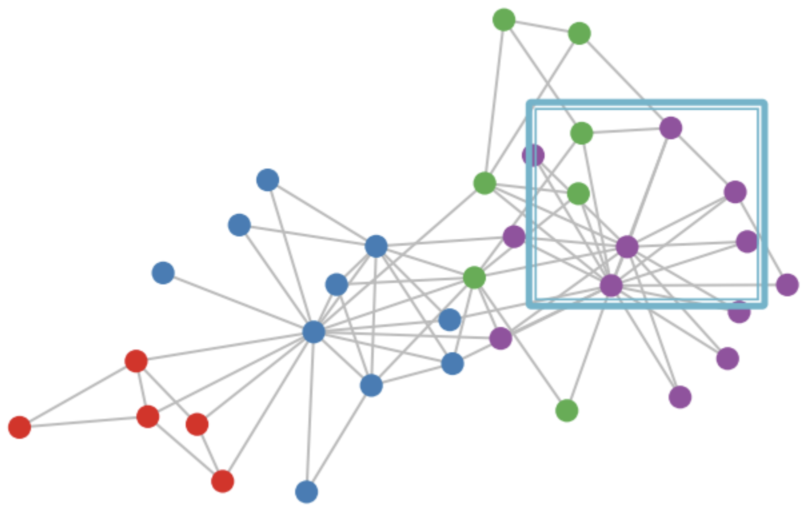


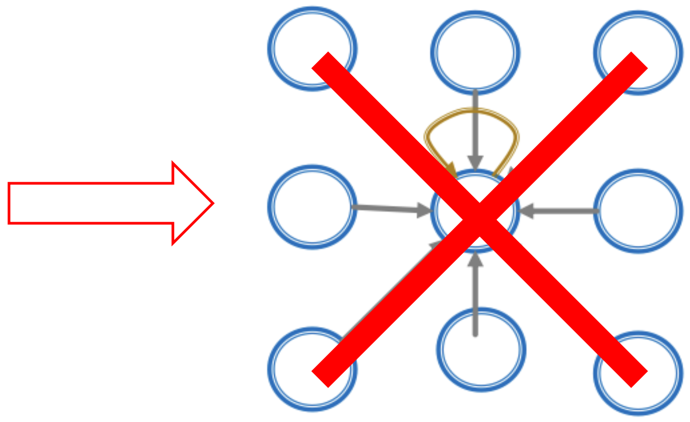
Q: can we extend similar operation to general graph?

General graph data



or this:





Q: can we extend similar operation to general graph?

Key: aggregate information from neighbors

Node degree

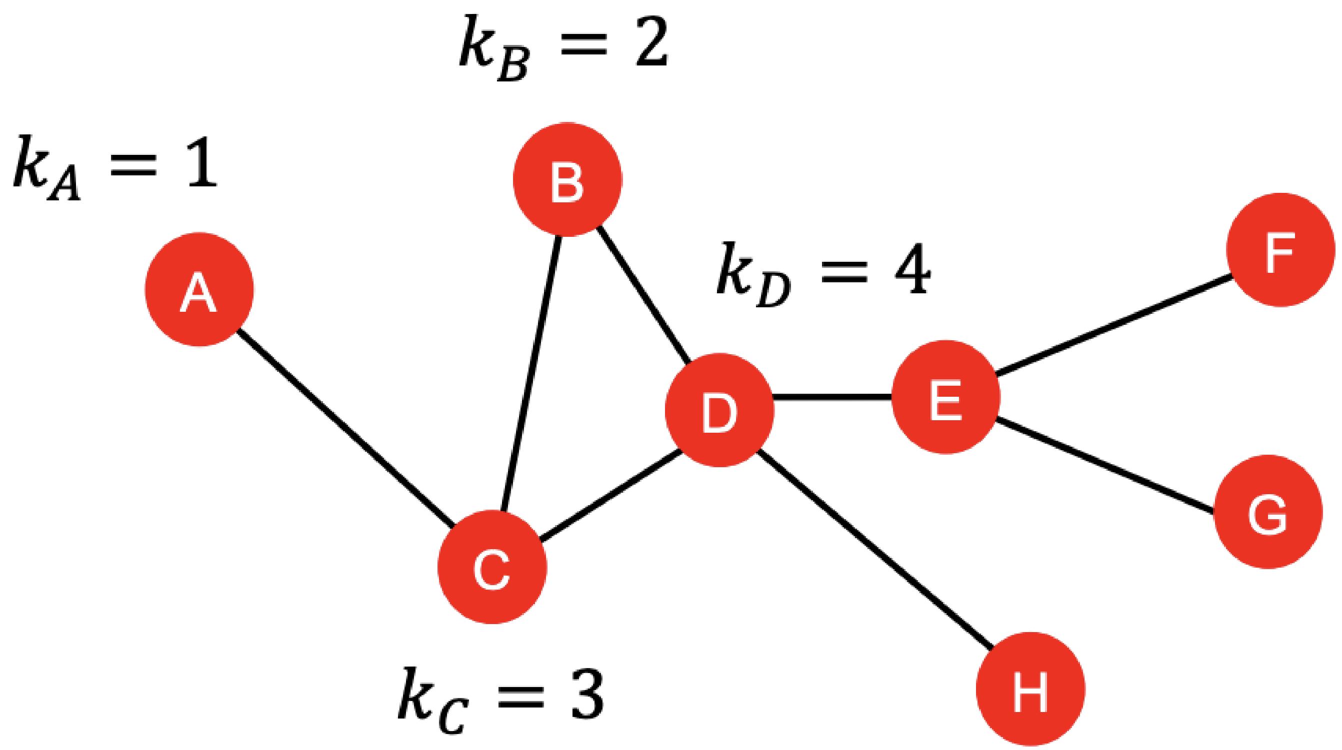
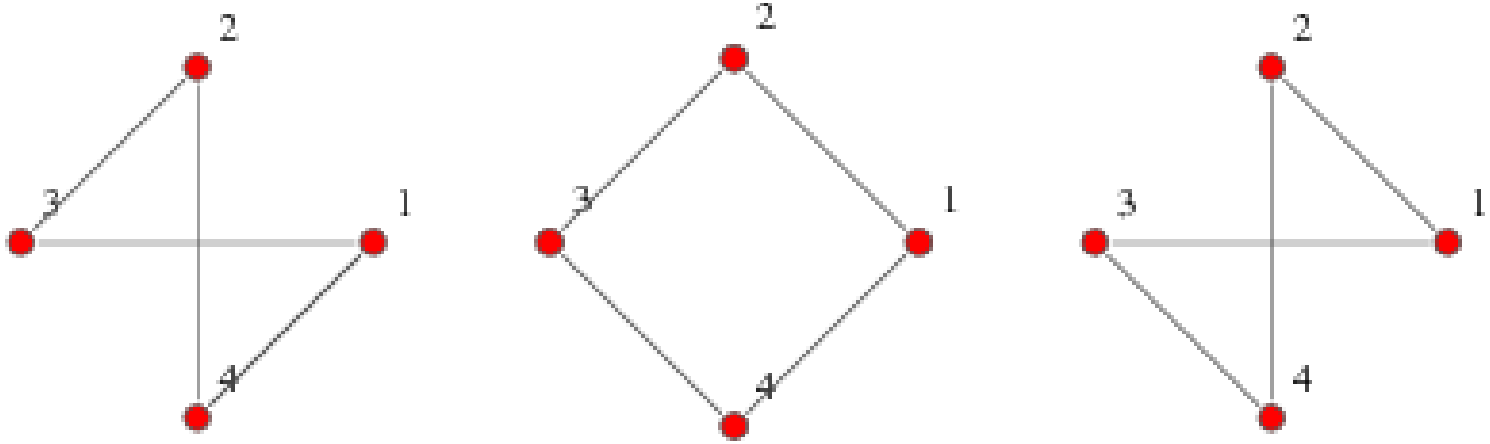


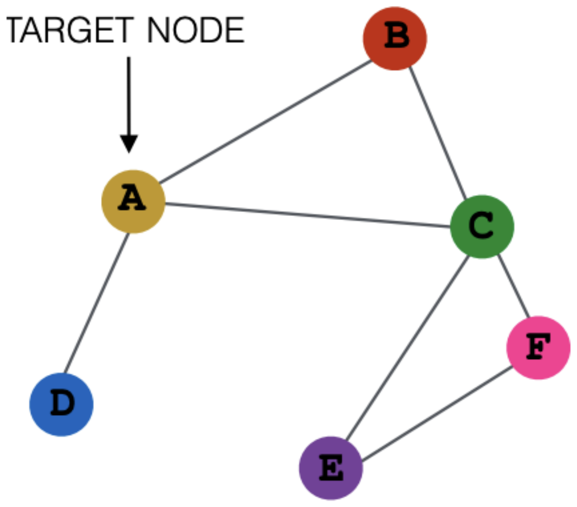
Image credit http://web.stanford.edu/class/cs224w/slides/03-nodeemb.pdf

Adjacency matrix



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

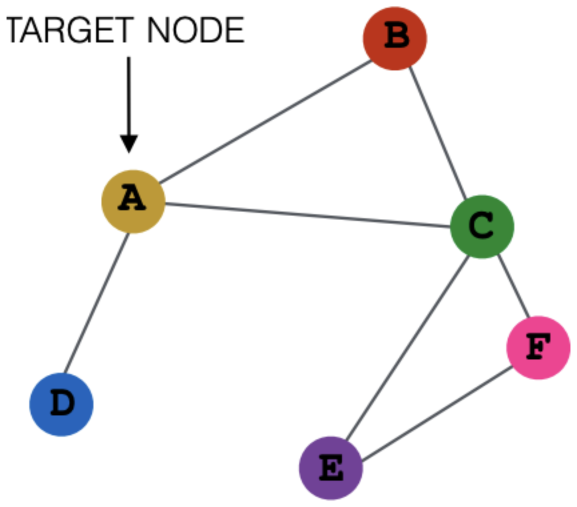
Graph networks: aggregate neighbors



INPUT GRAPH

Graph networks: aggregate neighbors

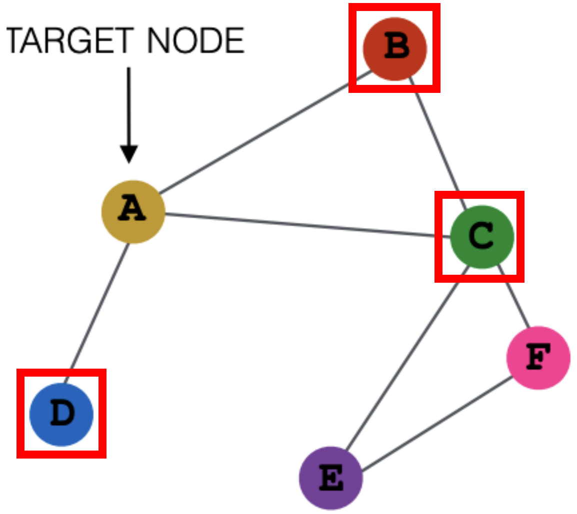
Nearest neighbors of A?



INPUT GRAPH

Graph networks: aggregate neighbors

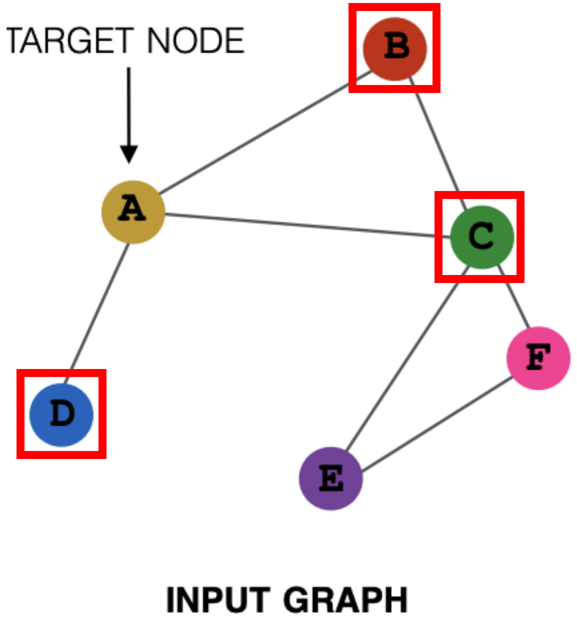
Nearest neighbors of A:
B, C, D

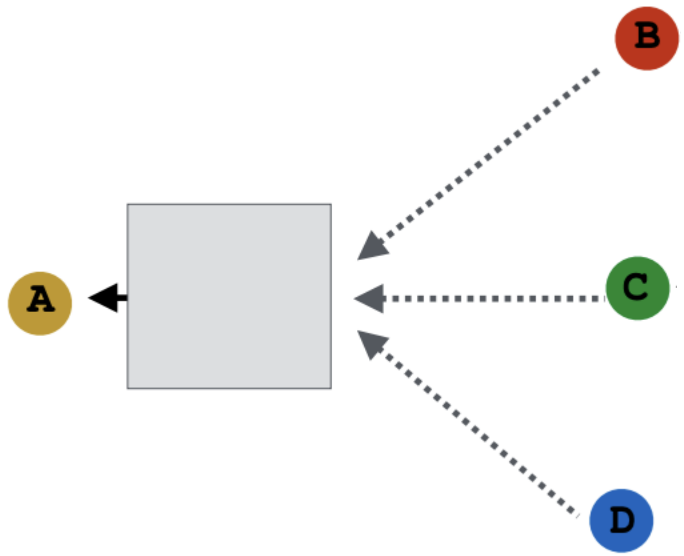


INPUT GRAPH

Graph networks: aggregate neighbors

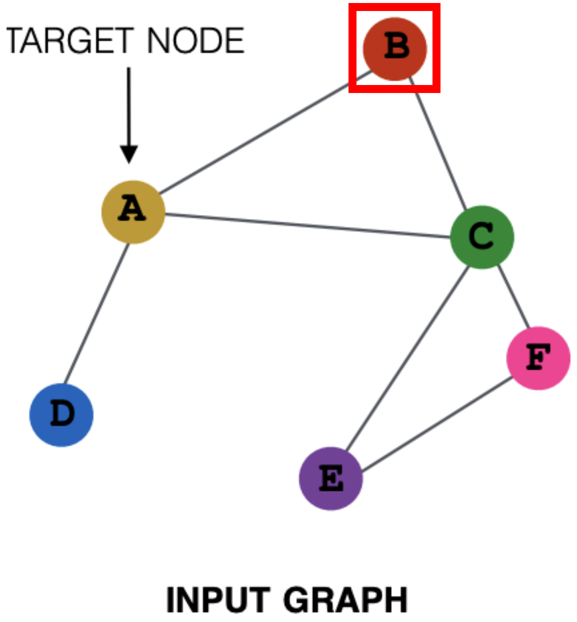
Nearest neighbors of A:
B, C, D





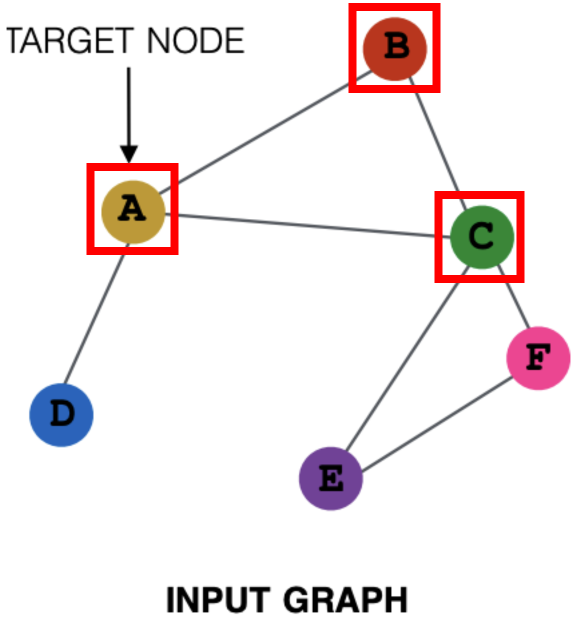
Graph networks: aggregate neighbors

Nearest neighbors of B?



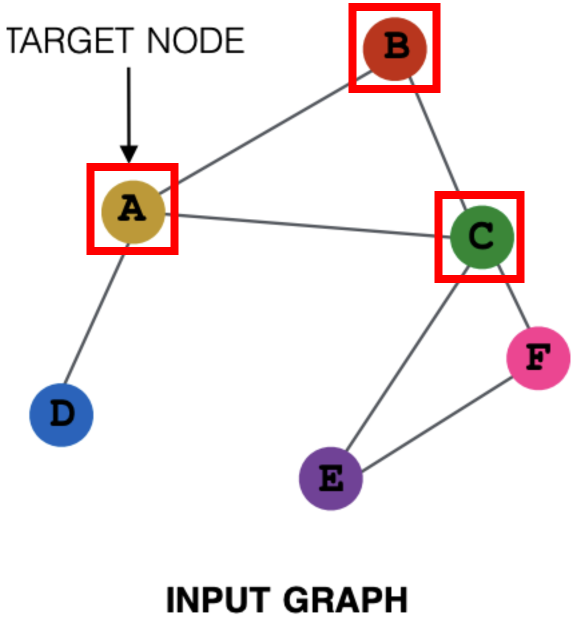
Graph networks: aggregate neighbors

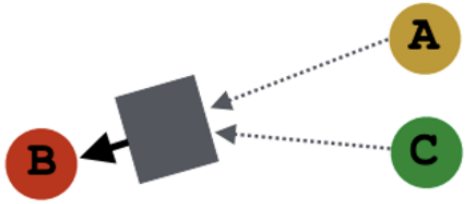
Nearest neighbors of B: A, C



Graph networks: aggregate neighbors

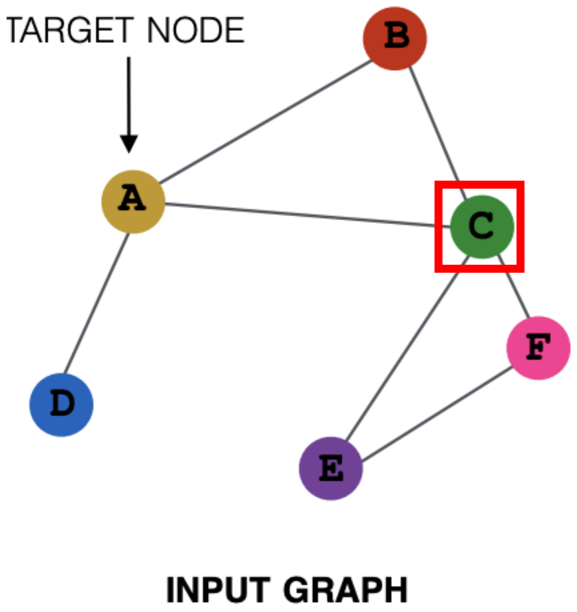
Nearest neighbors of B: A, C

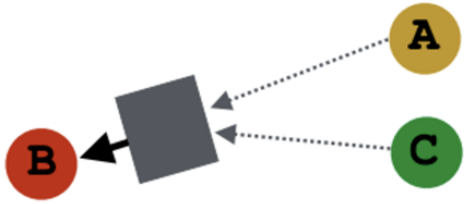




Graph networks: aggregate neighbors

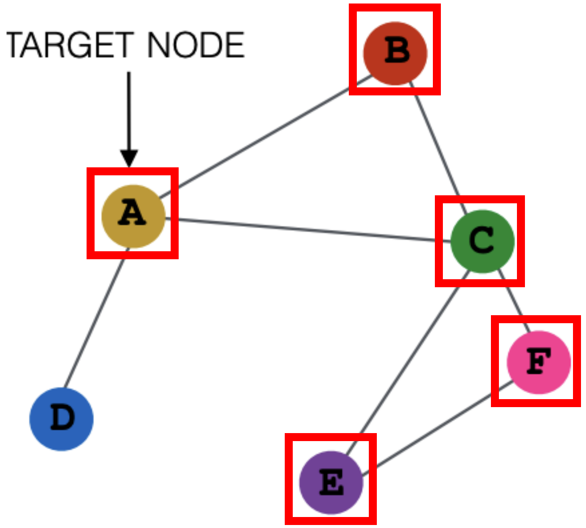
Nearest neighbors of C?



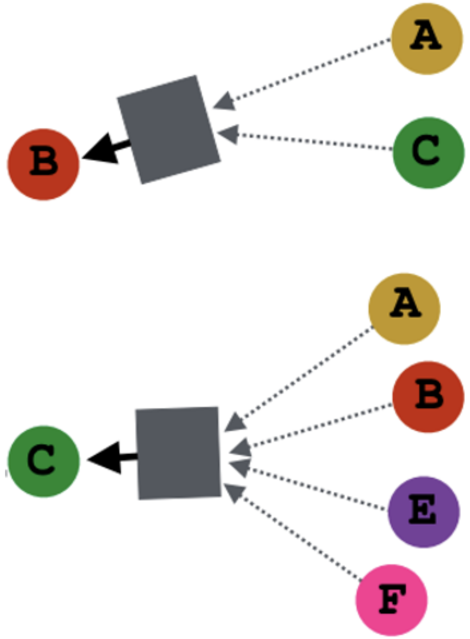


Graph networks: aggregate neighbors

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

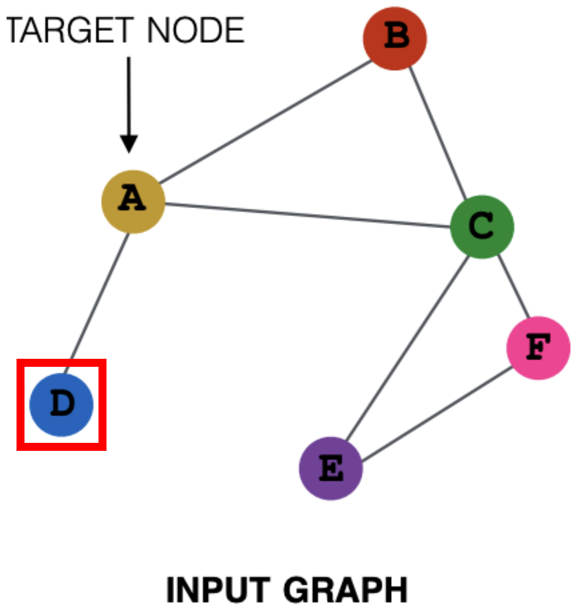


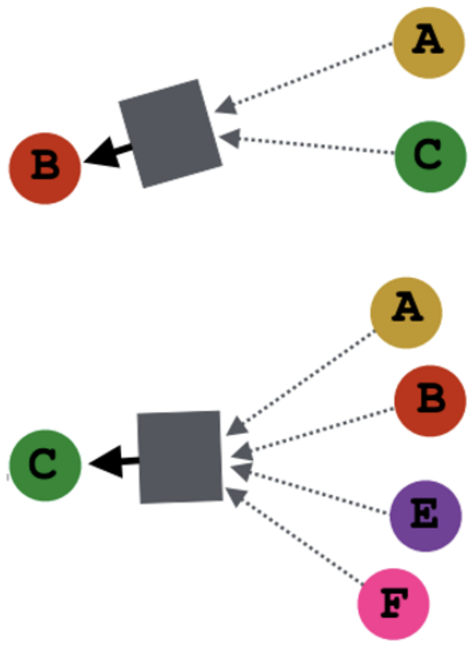
INPUT GRAPH



Graph networks: aggregate neighbors

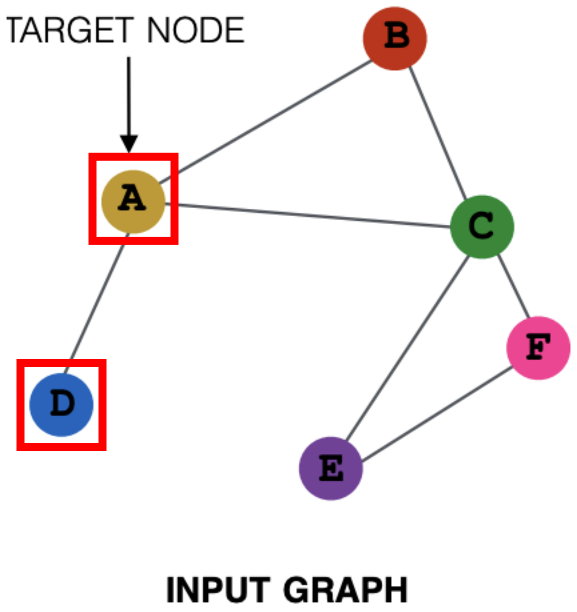
Nearest neighbors of D?

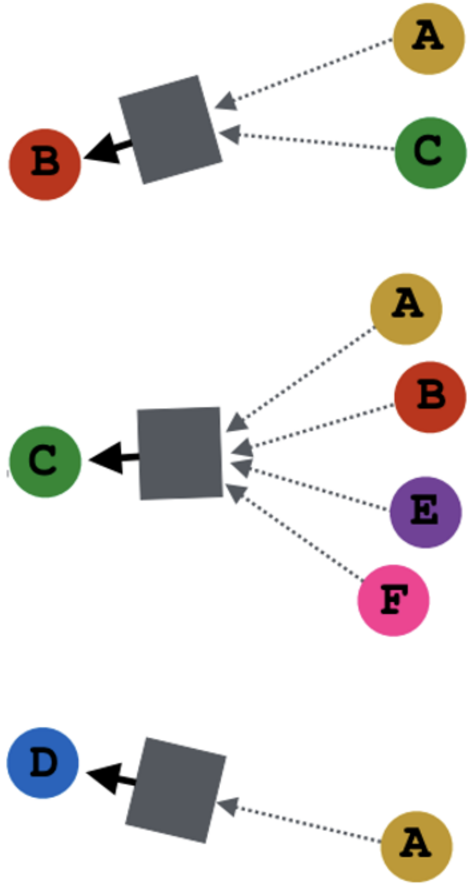




Graph networks: aggregate neighbors

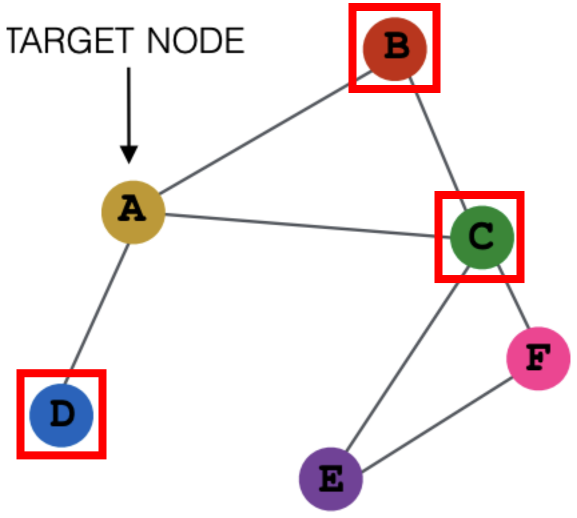
Nearest neighbors of D: A



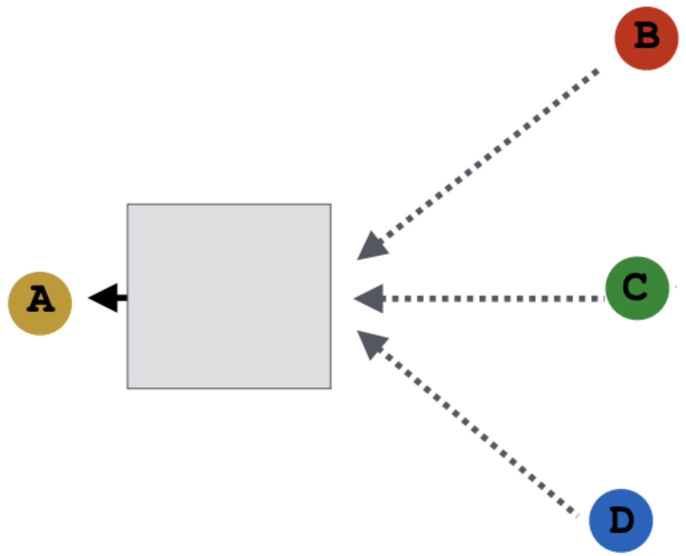


Graph networks: aggregate neighbors

Nearest neighbors of A:
B, C, D

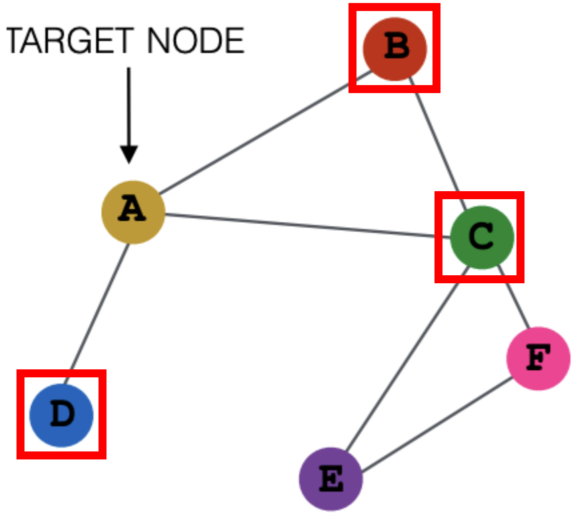


INPUT GRAPH



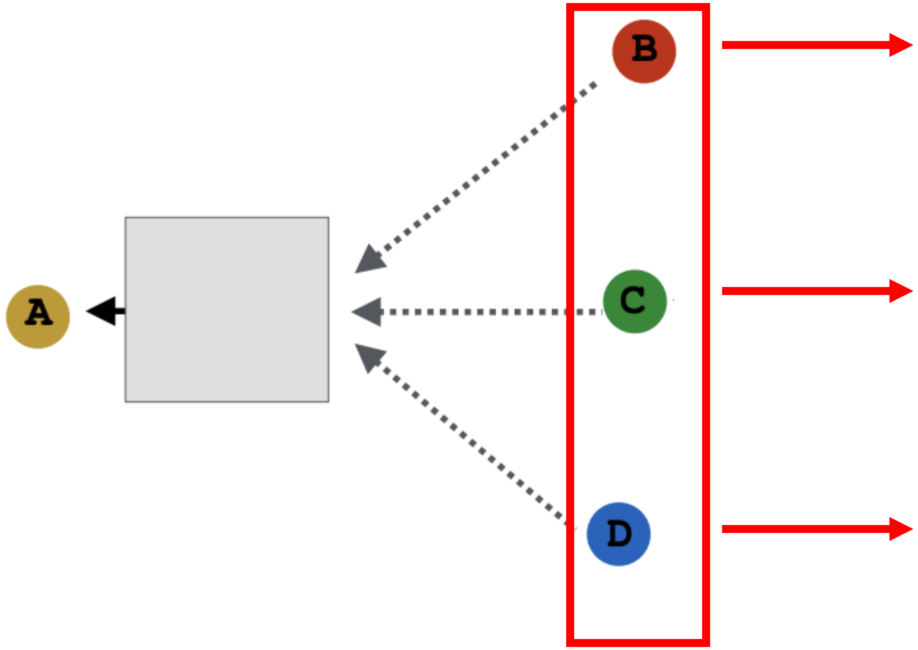
Graph networks: aggregate neighbors

Nearest neighbors of A:
B, C, D



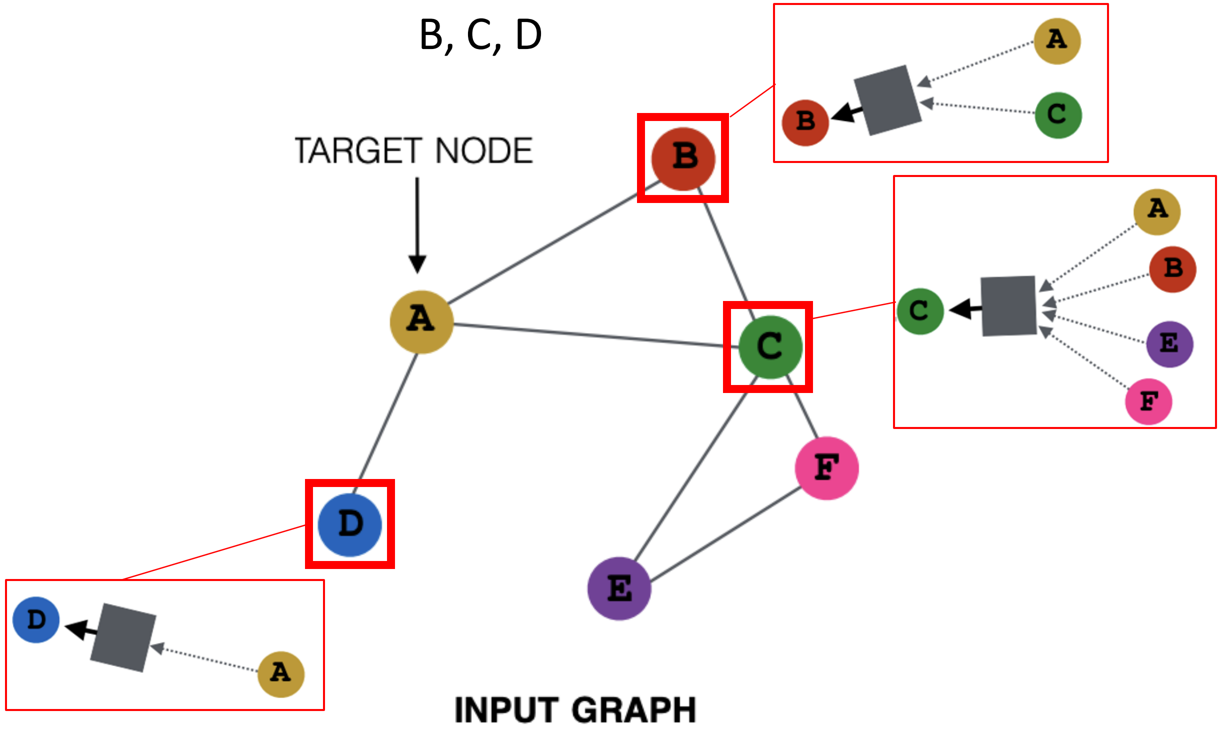
INPUT GRAPH

Q: Deeper structure?

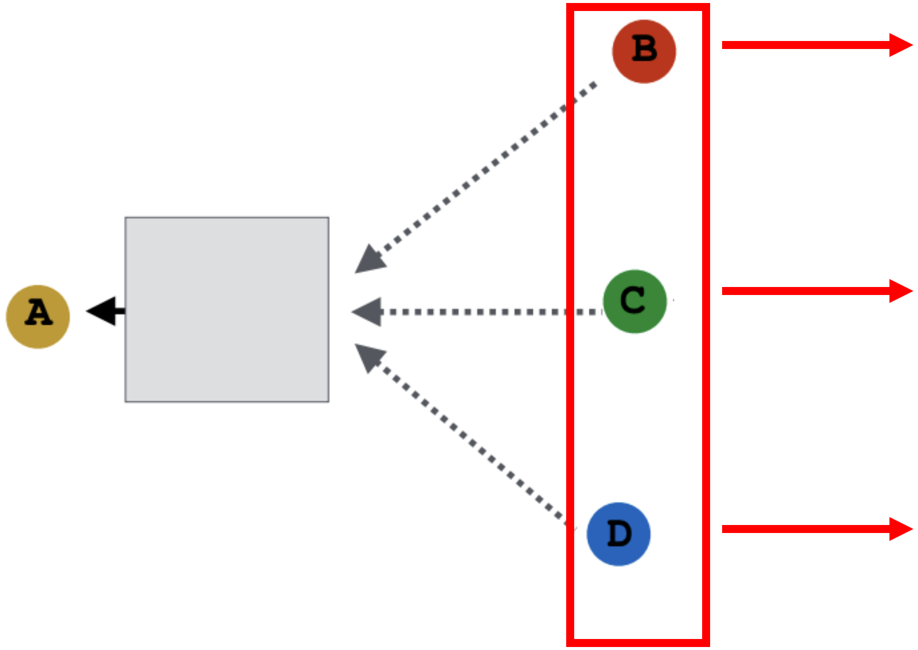


Graph networks: aggregate neighbors

Nearest neighbors of A:
B, C, D

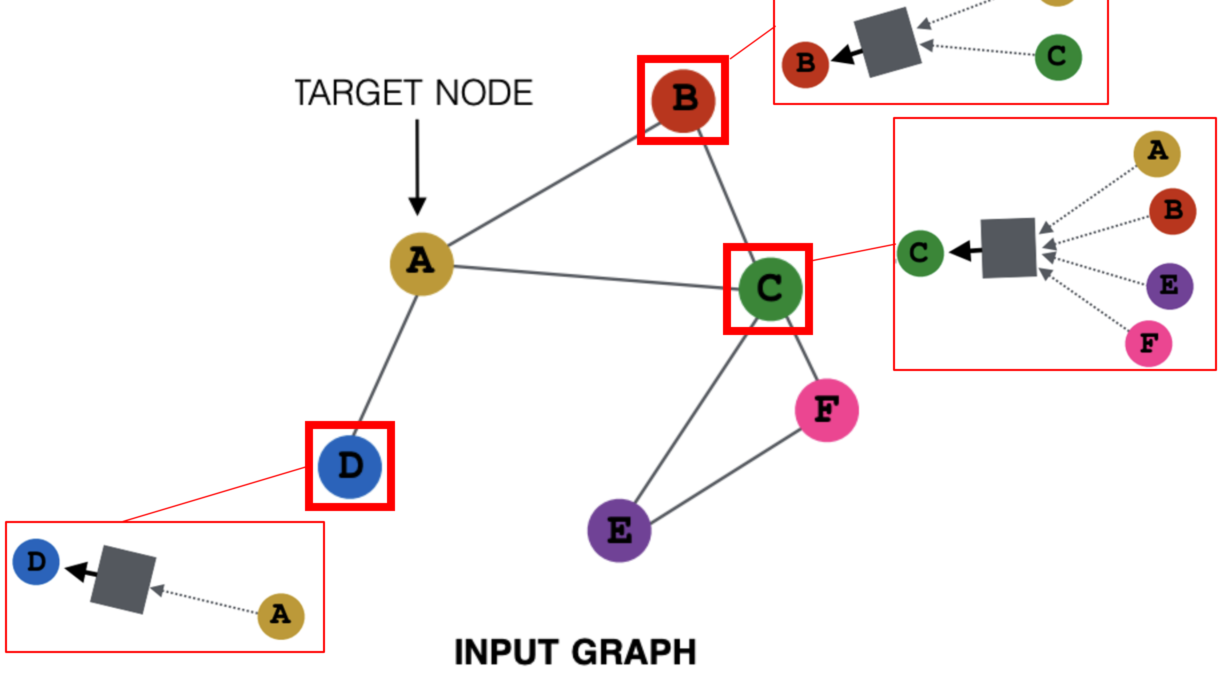


Q: Deeper structure?

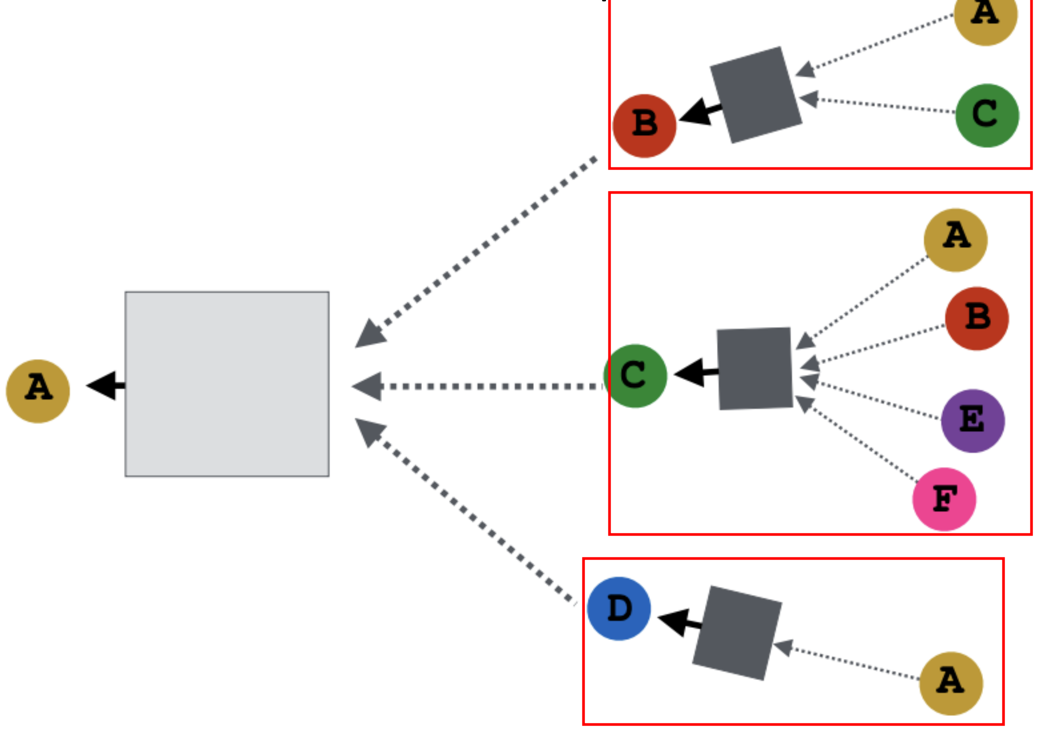


Graph networks: aggregate neighbors

Nearest neighbors of A:
B, C, D

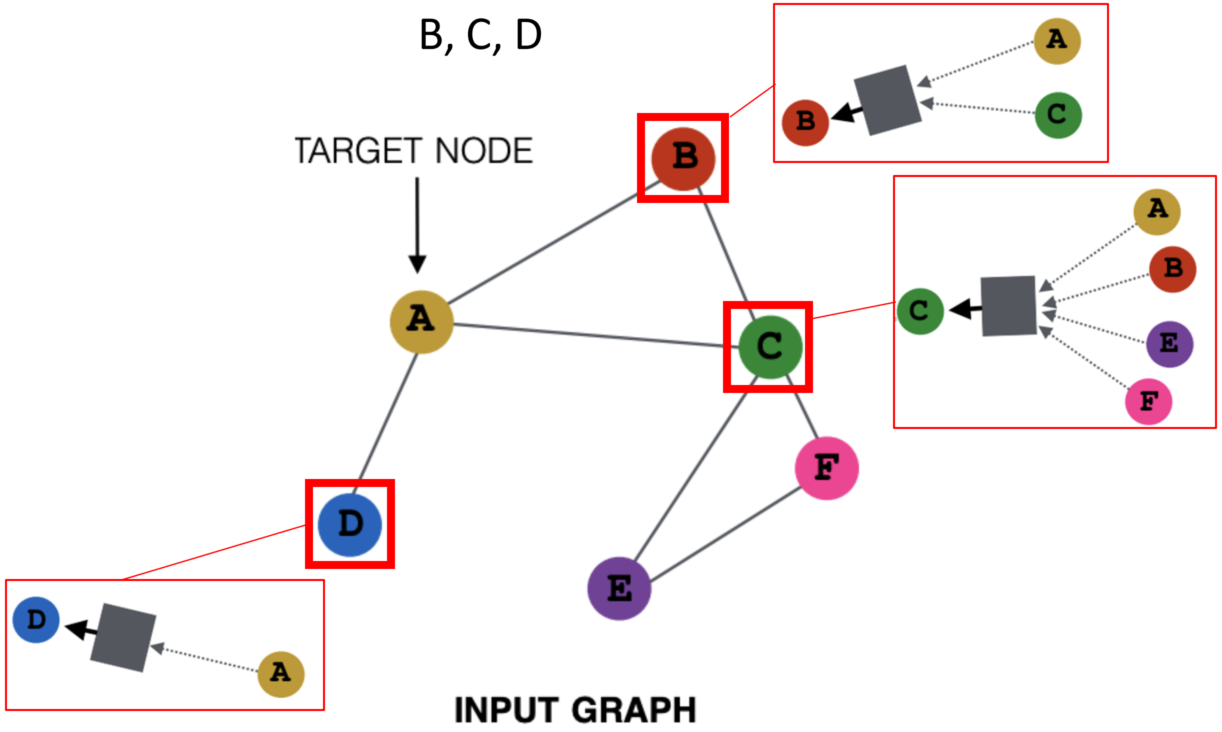


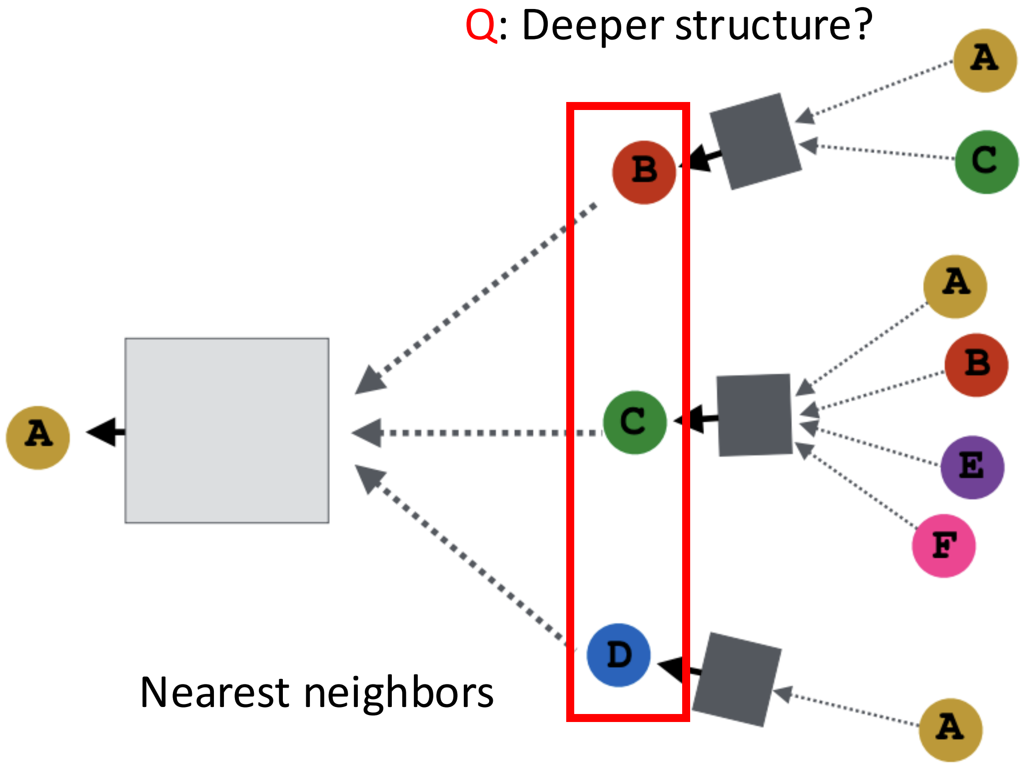
Q: Deeper structure?



Graph networks: aggregate neighbors

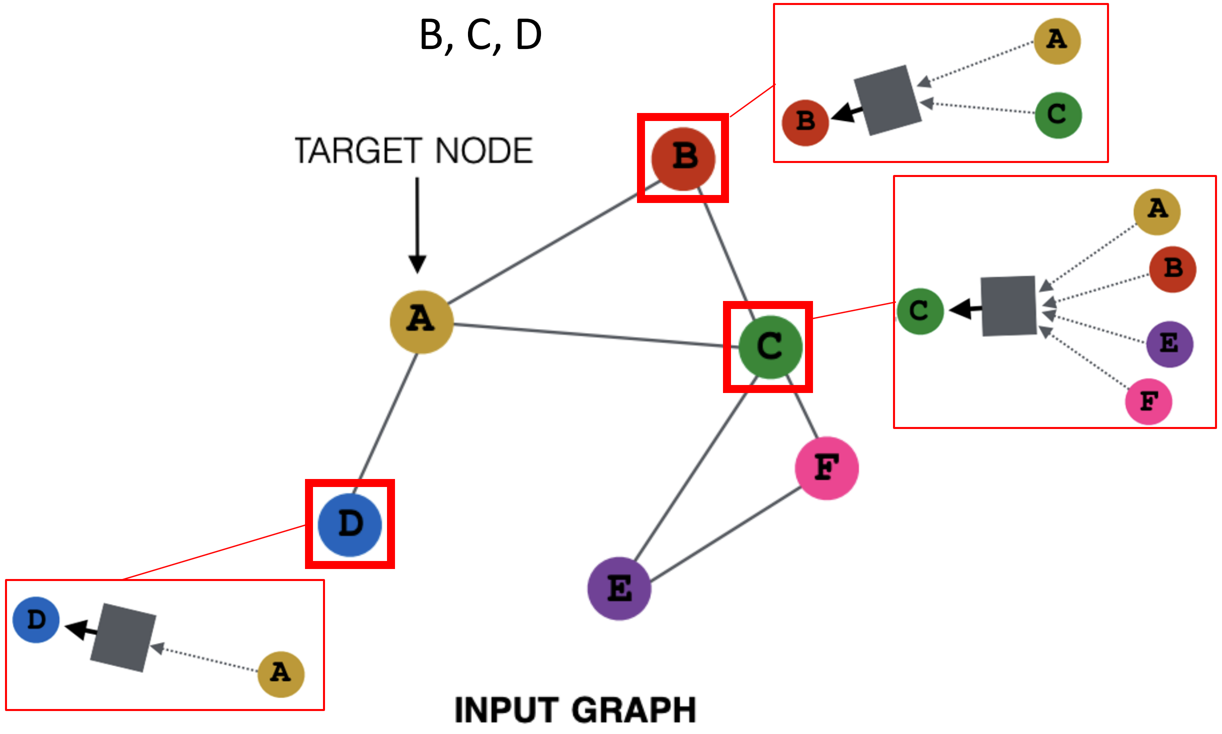
Nearest neighbors of A:
B, C, D

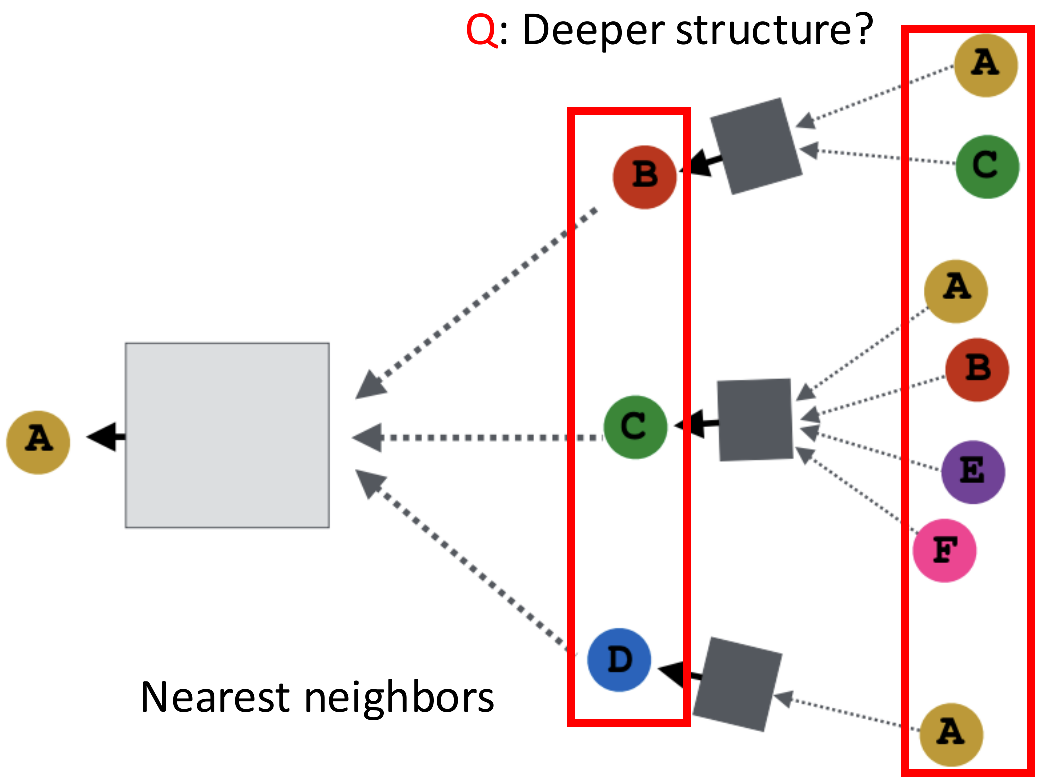




Graph networks: aggregate neighbors

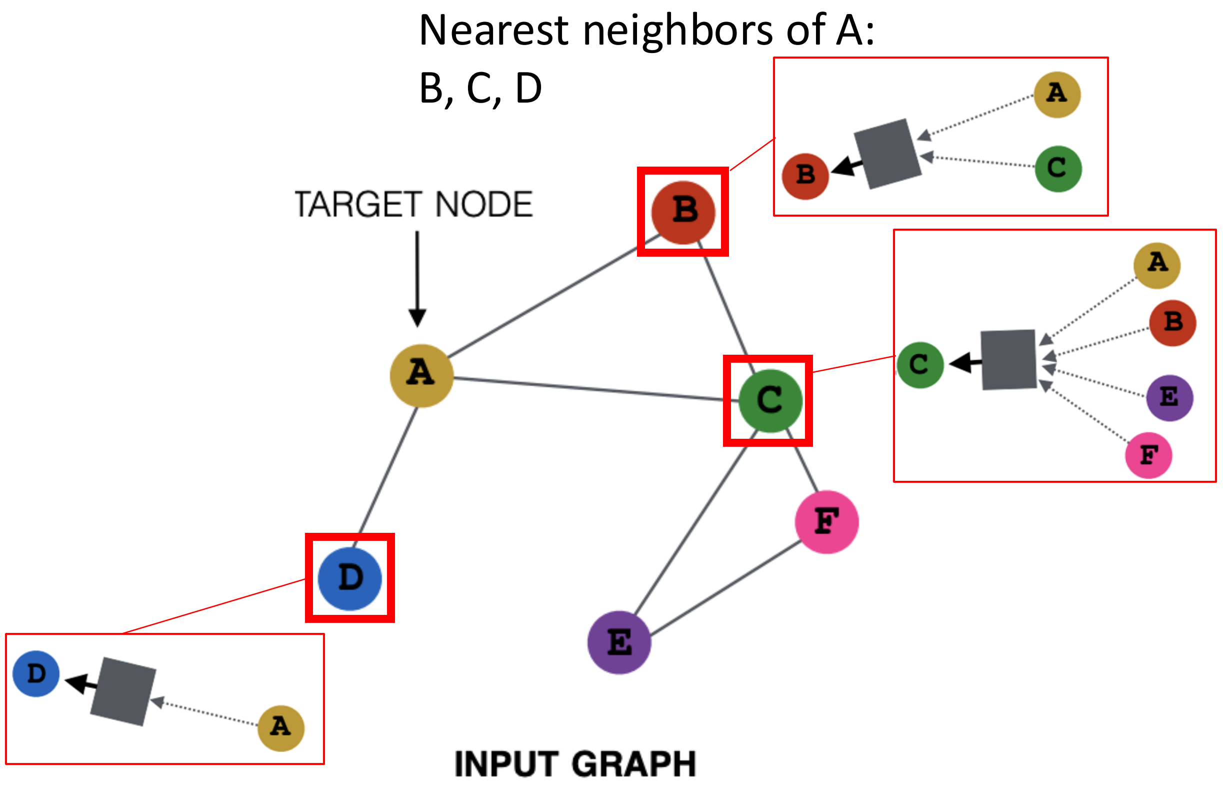
Nearest neighbors of A:
B, C, D

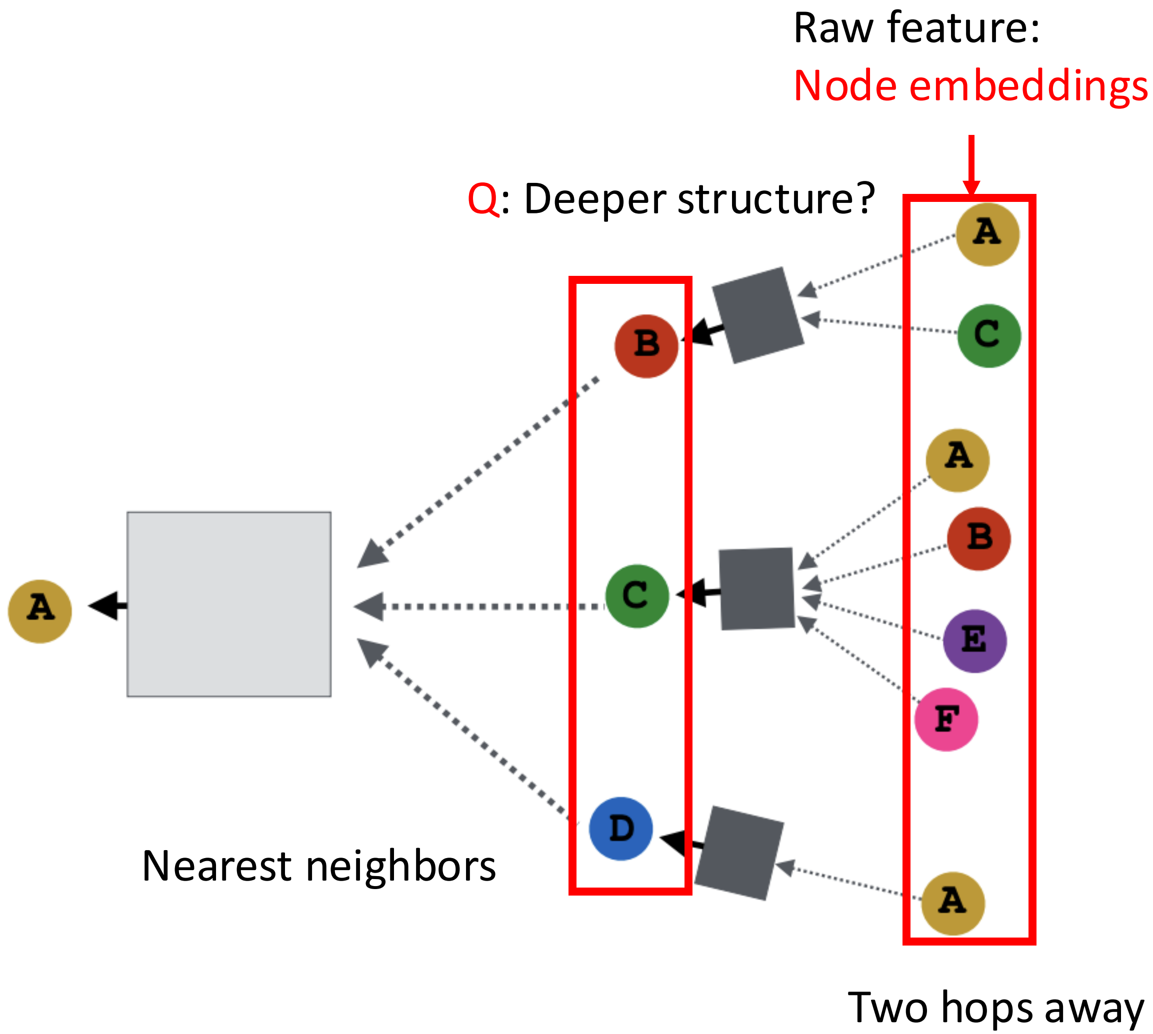




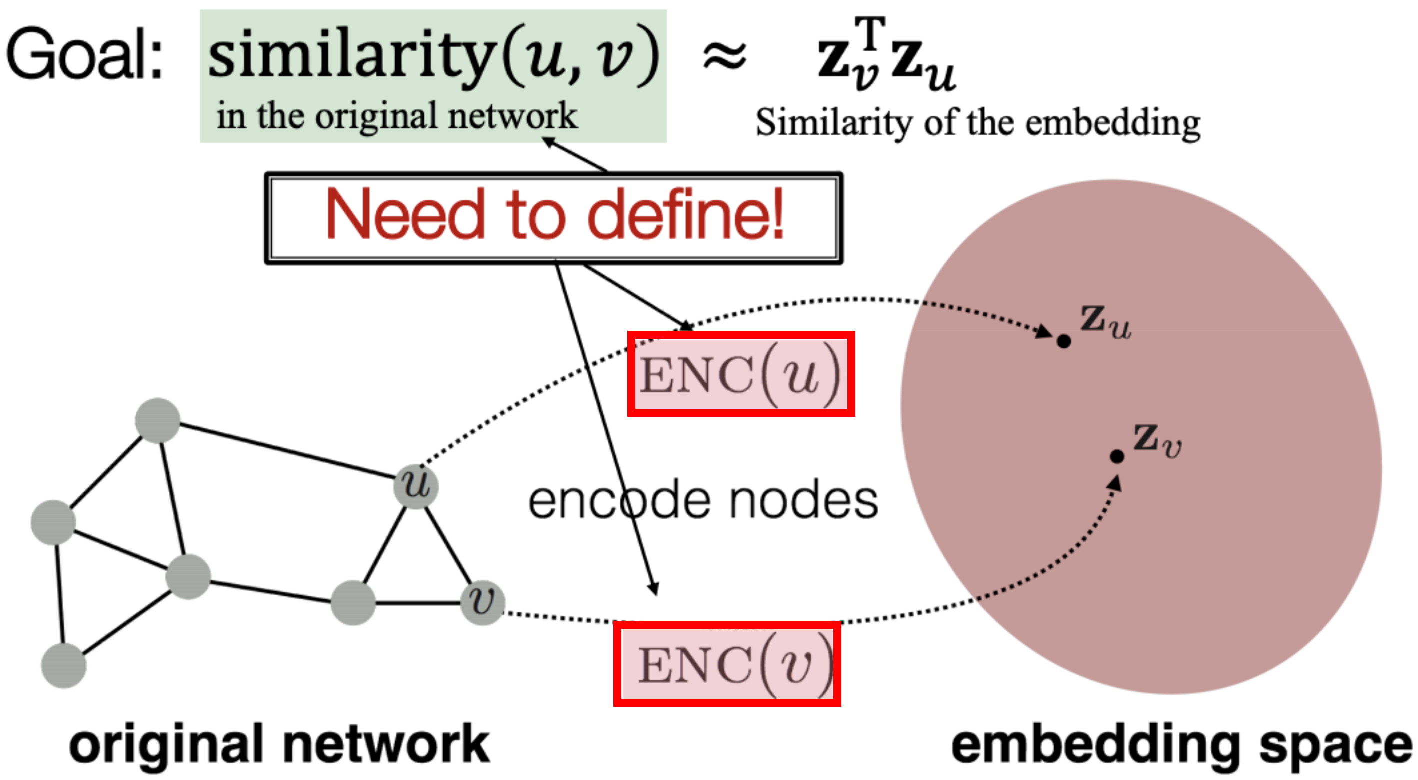
Two hops away

Graph networks: aggregate neighbors





Encoder-decoder for graph data



Q: how to learn ENC?
Linear transformation

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

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

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

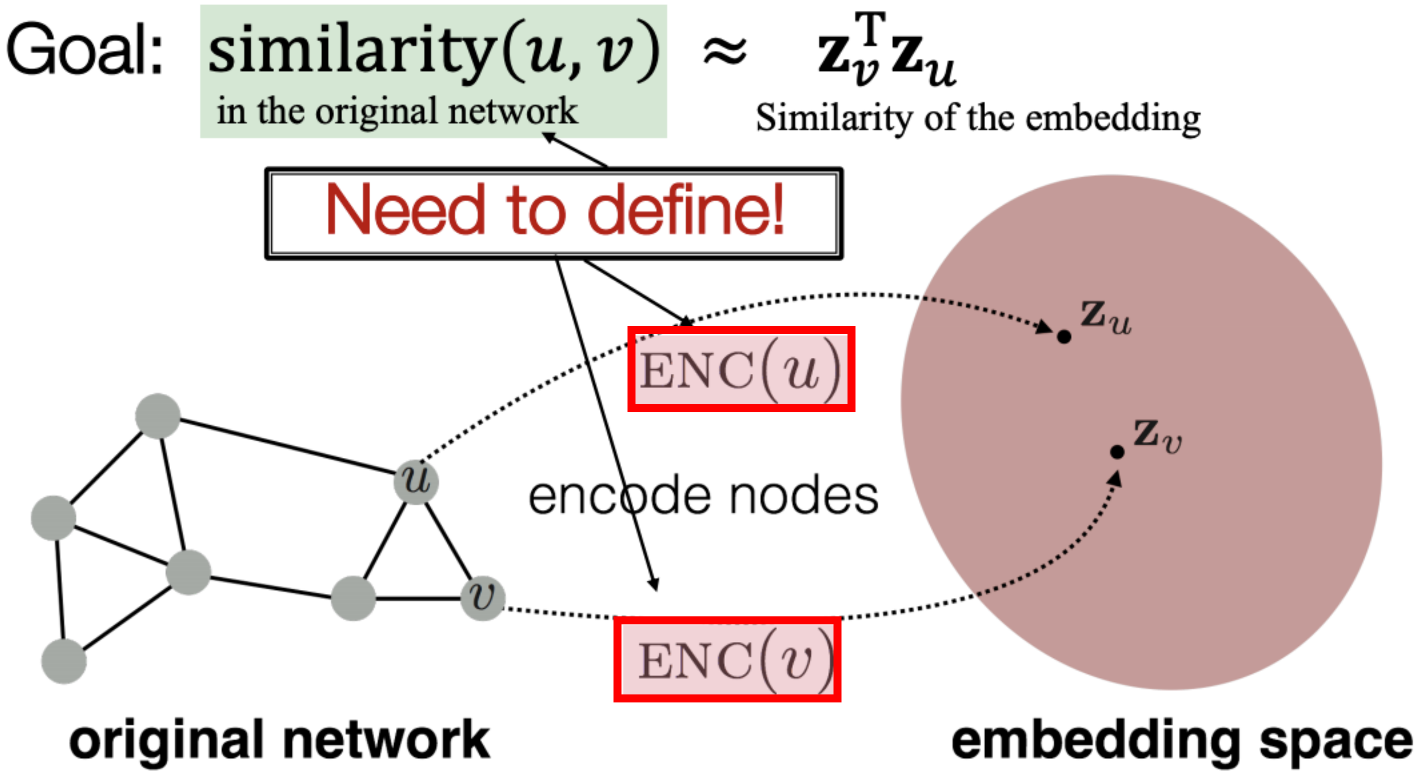
Each node

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

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

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

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

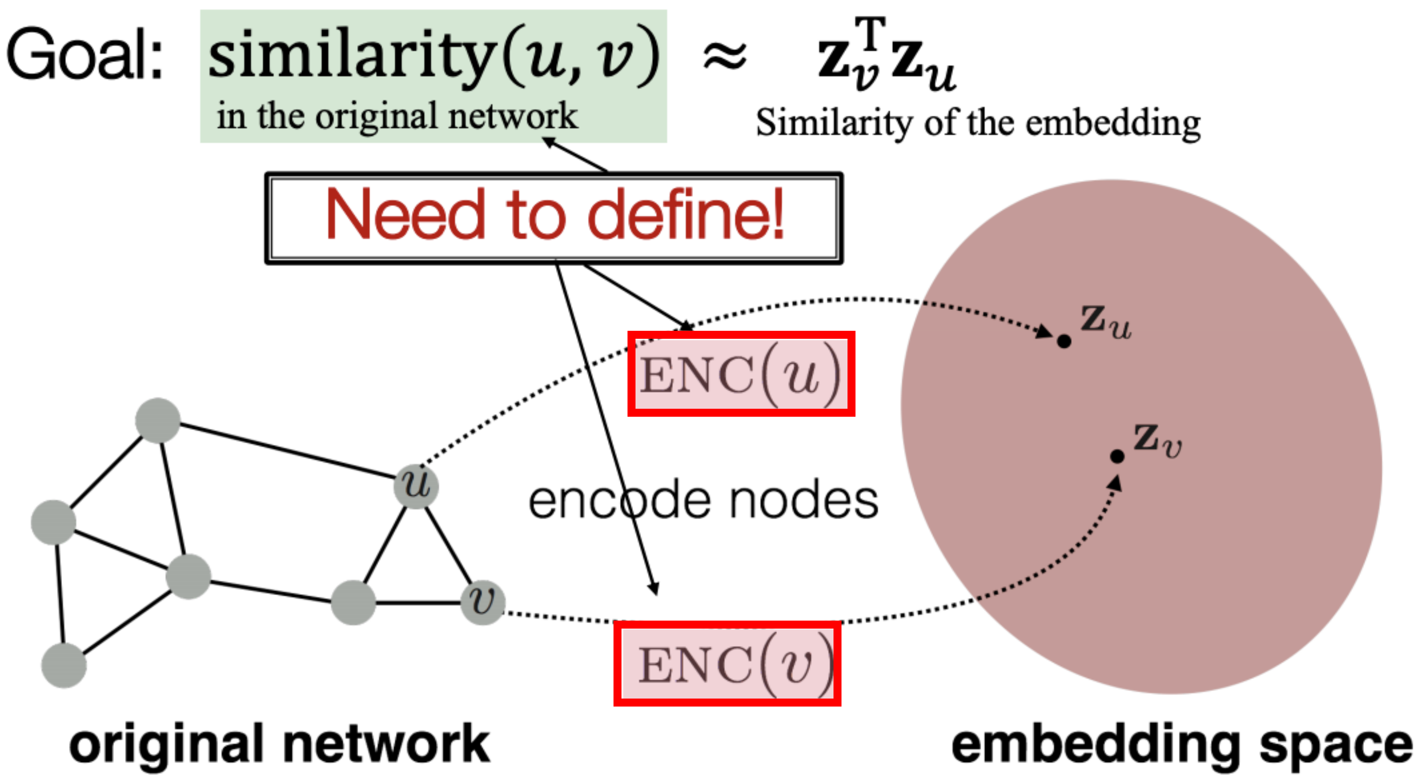
Each node

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

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

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

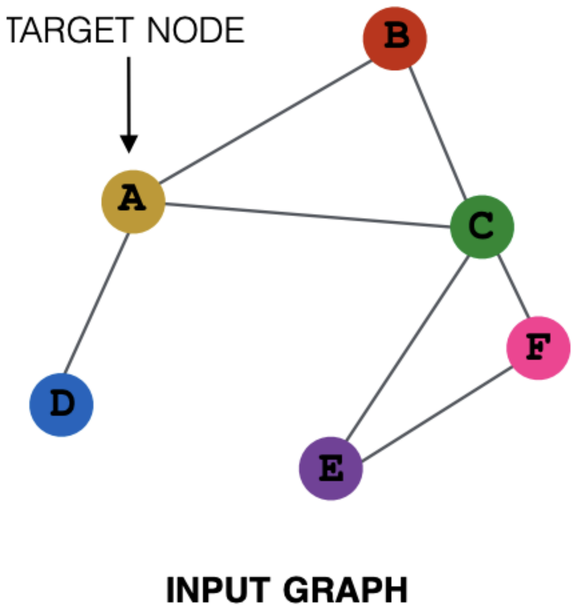
$v \in \mathbb{I}^{|\mathcal{V}|}$

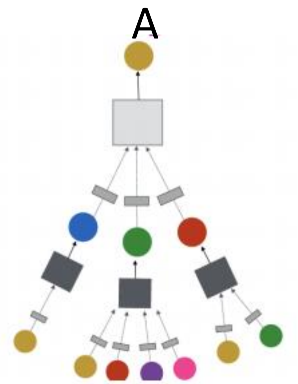
Each node

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

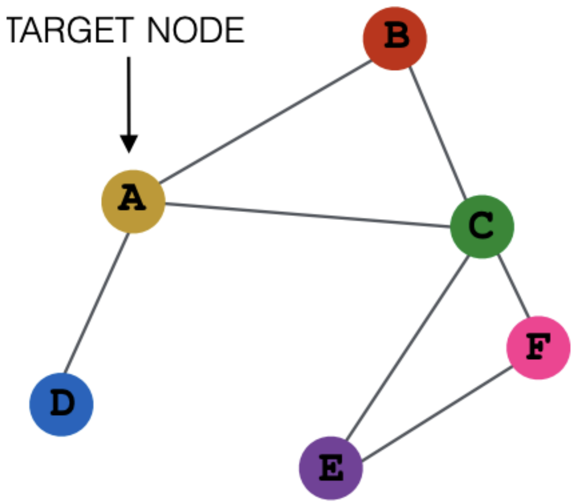
Image credit http://web.stanford.edu/class/cs224w/slides/03-nodeemb.pdf

Graph networks: aggregate neighbors

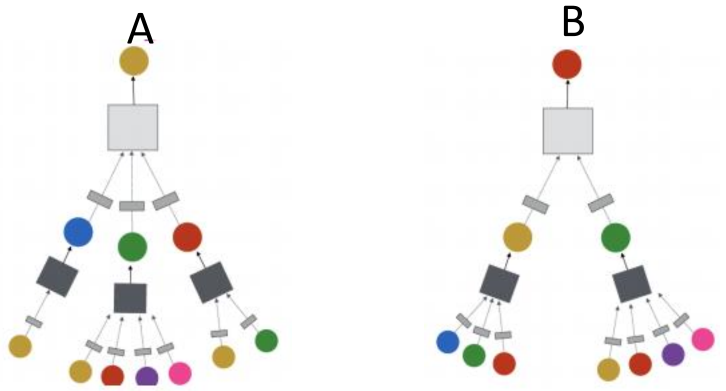




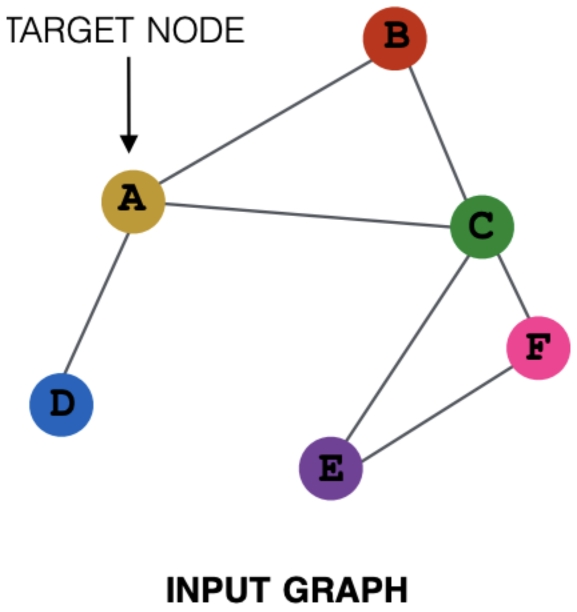
Graph networks: aggregate neighbors

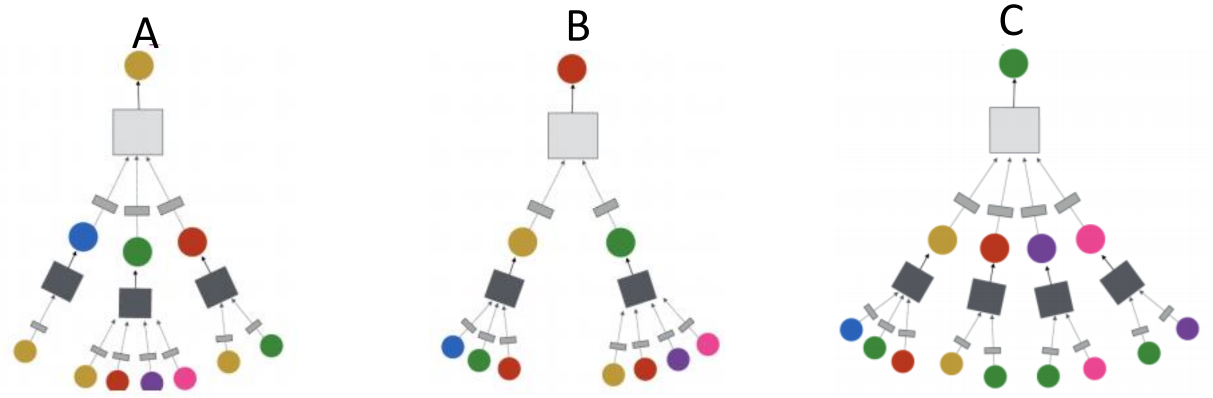


INPUT GRAPH

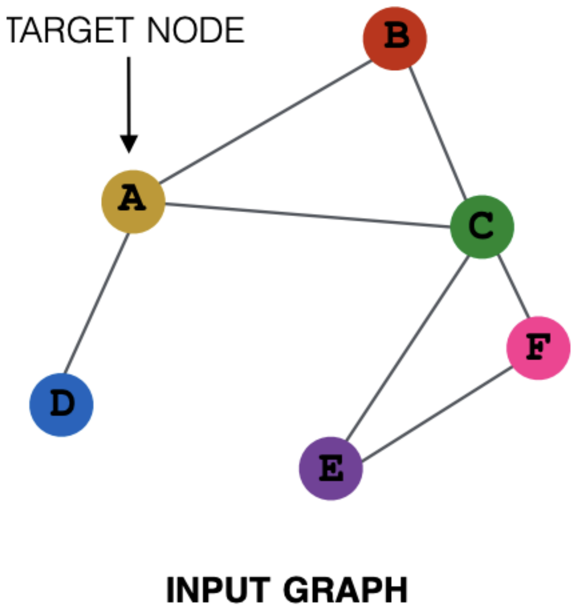


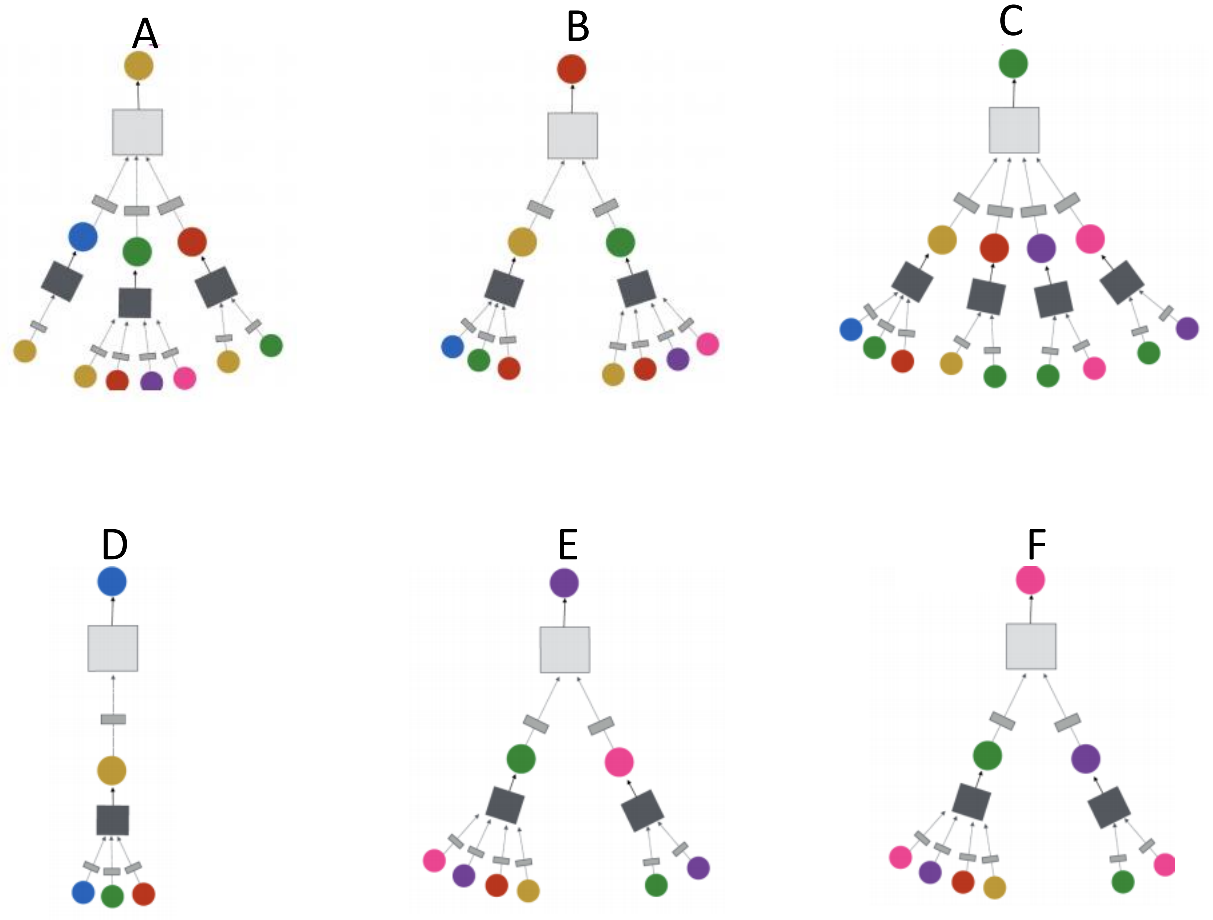
Graph networks: aggregate neighbors



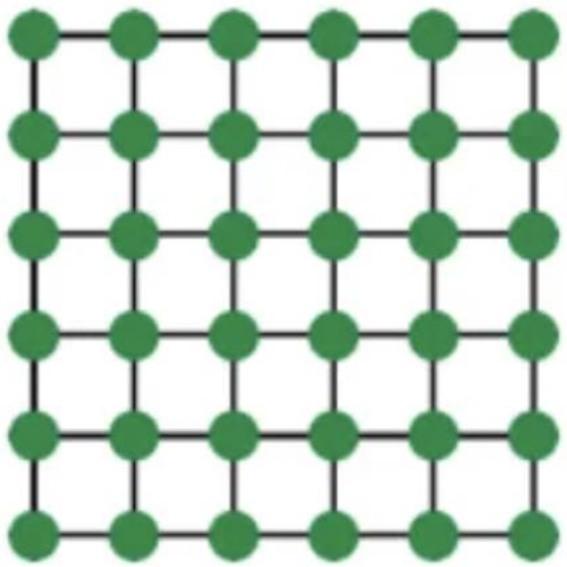


Graph networks: aggregate neighbors



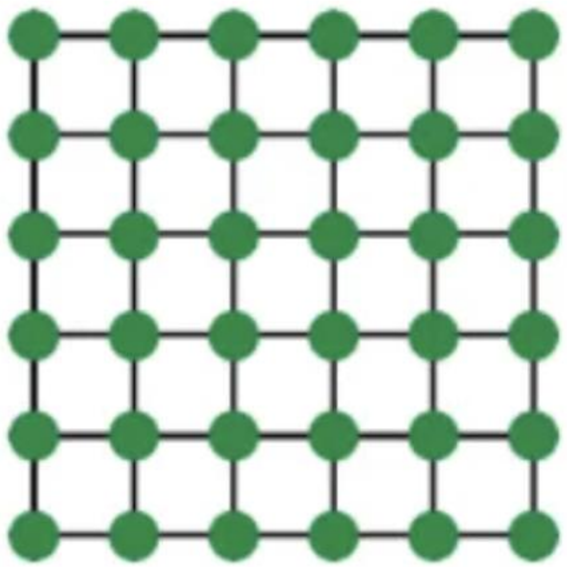


Graph networks: aggregate neighbors

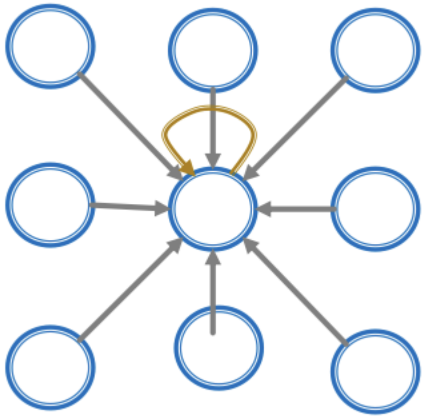


Grid graph

Graph networks: aggregate neighbors

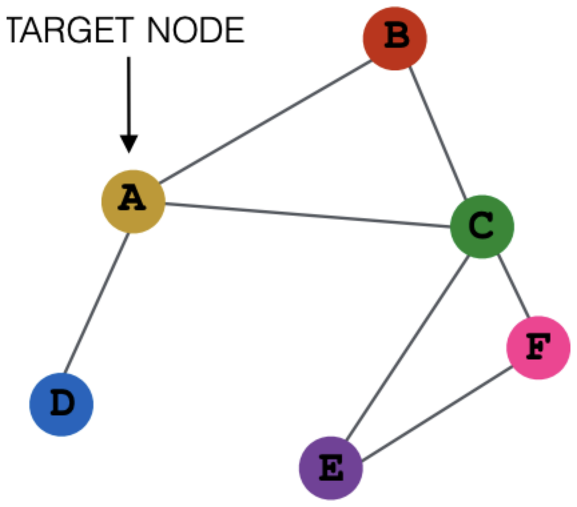


Grid graph

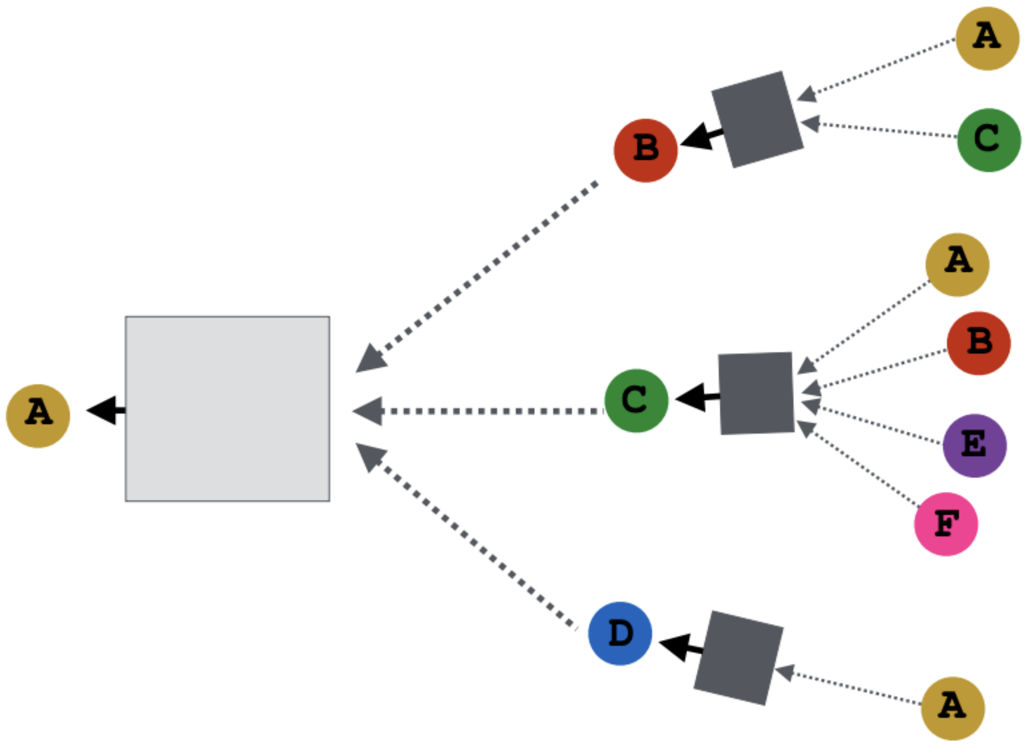


General graph

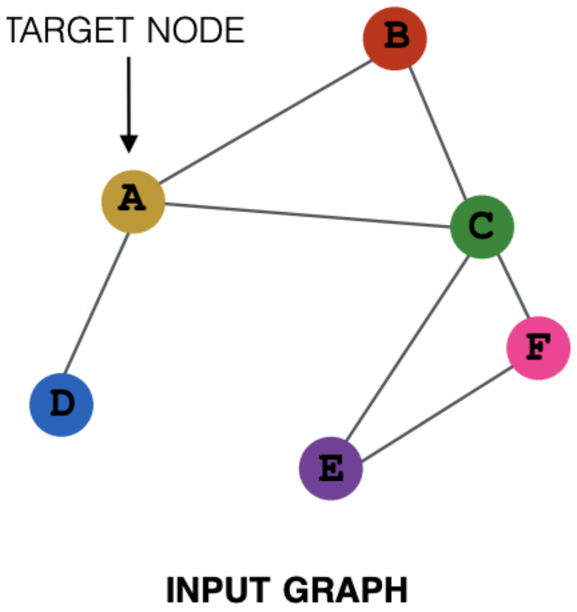
Graph networks: aggregate neighbors

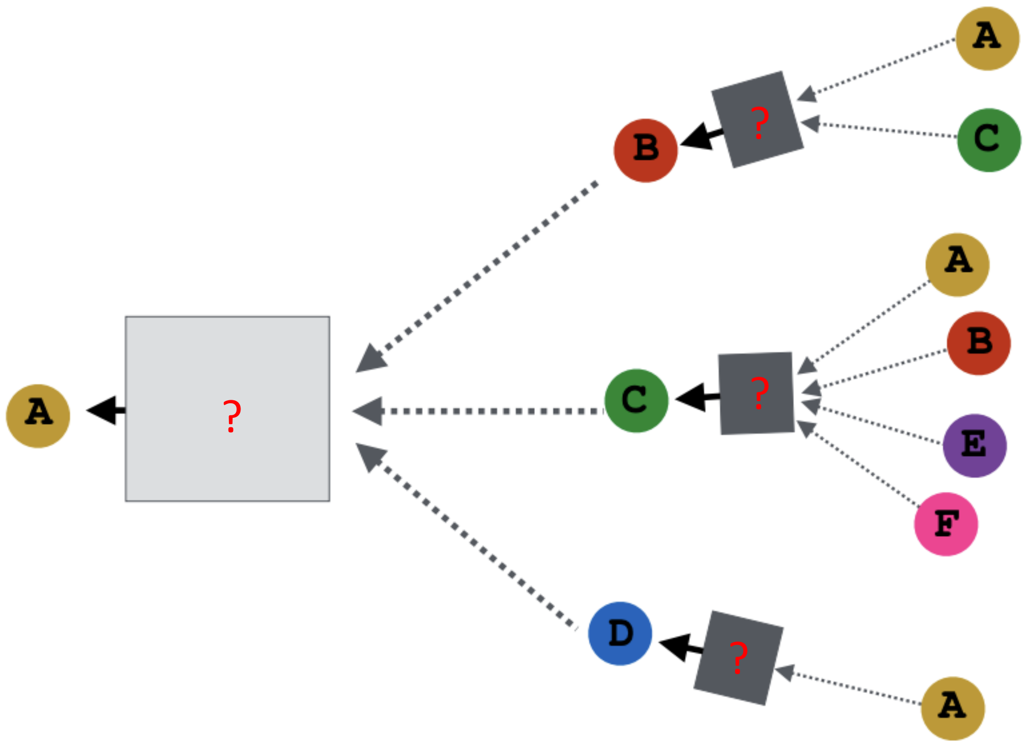


INPUT GRAPH

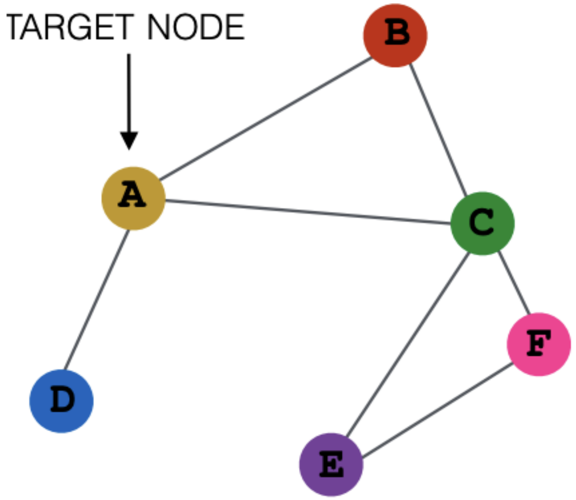


Graph networks: aggregate neighbors

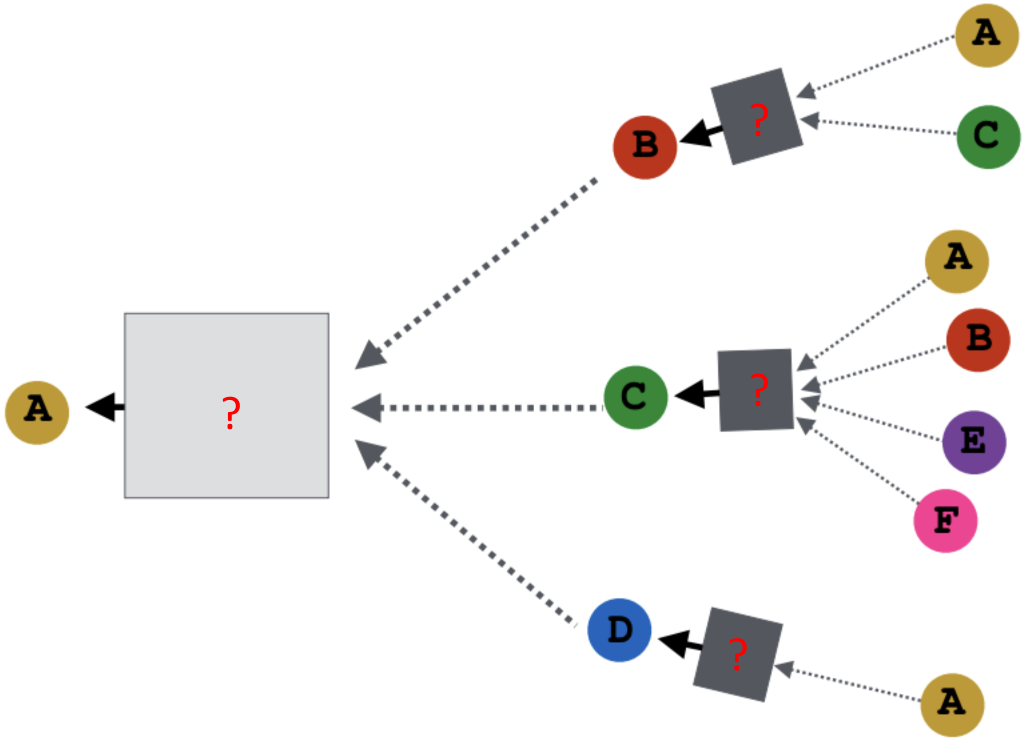




Graph networks: aggregate neighbors

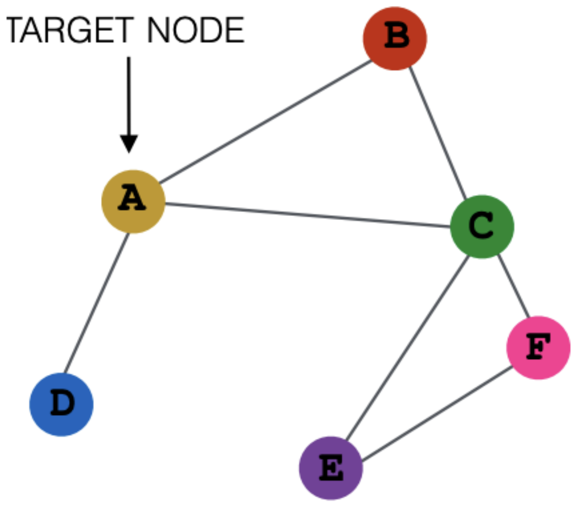


INPUT GRAPH

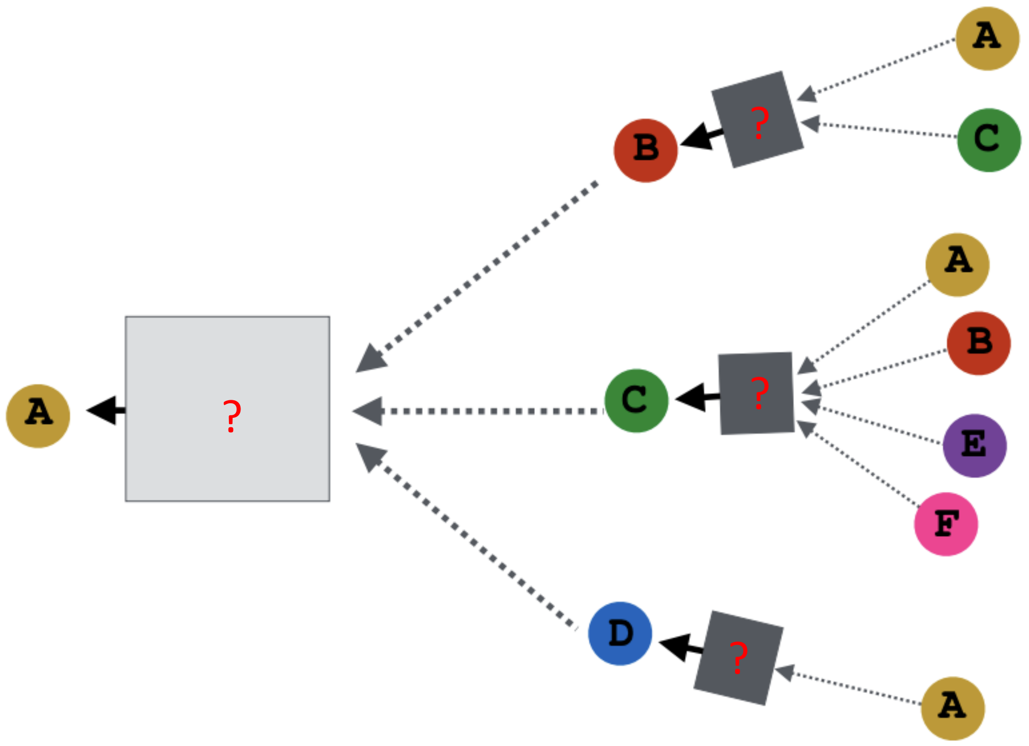


Q: what can we do in the box to aggregate information?

Graph networks: aggregate neighbors

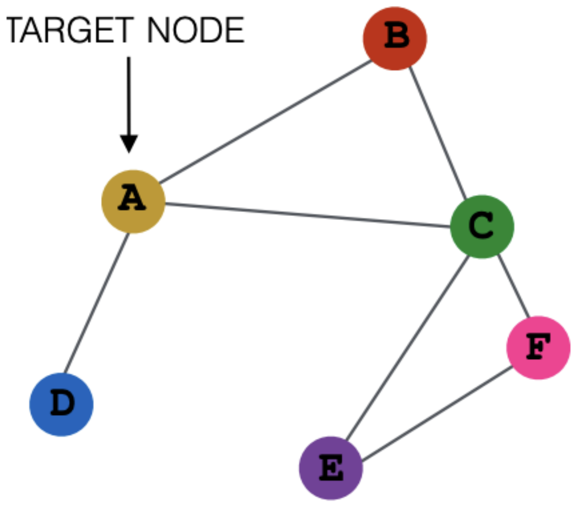


INPUT GRAPH

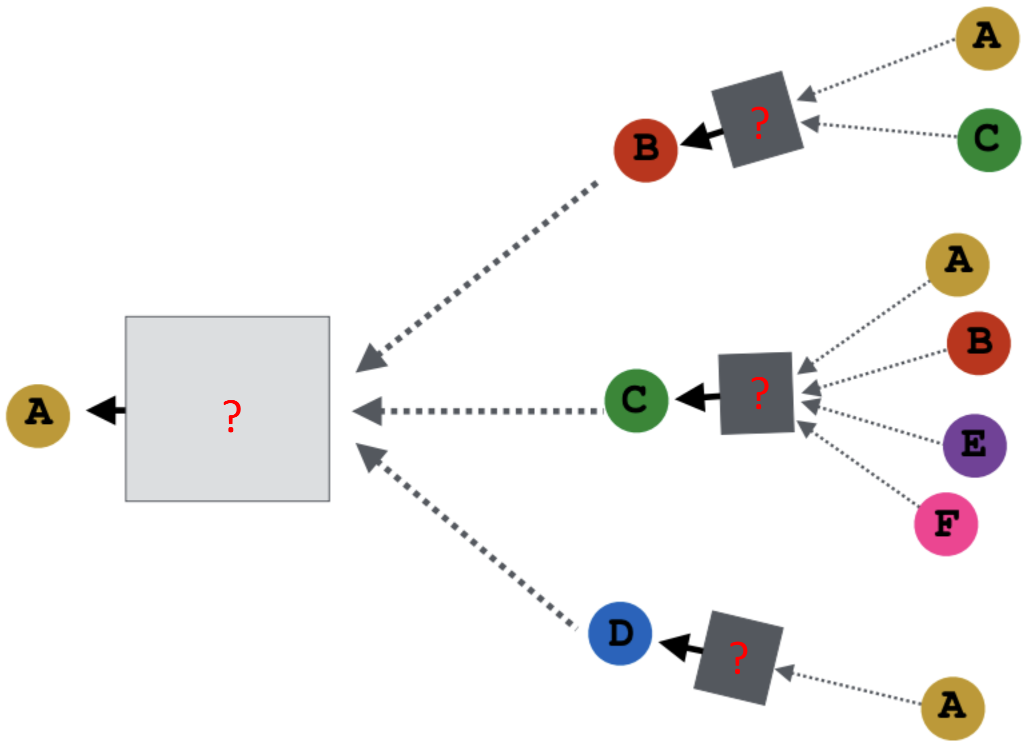


Q: what can we do in the box to aggregate information?
Average/summation?

Graph networks: aggregate neighbors

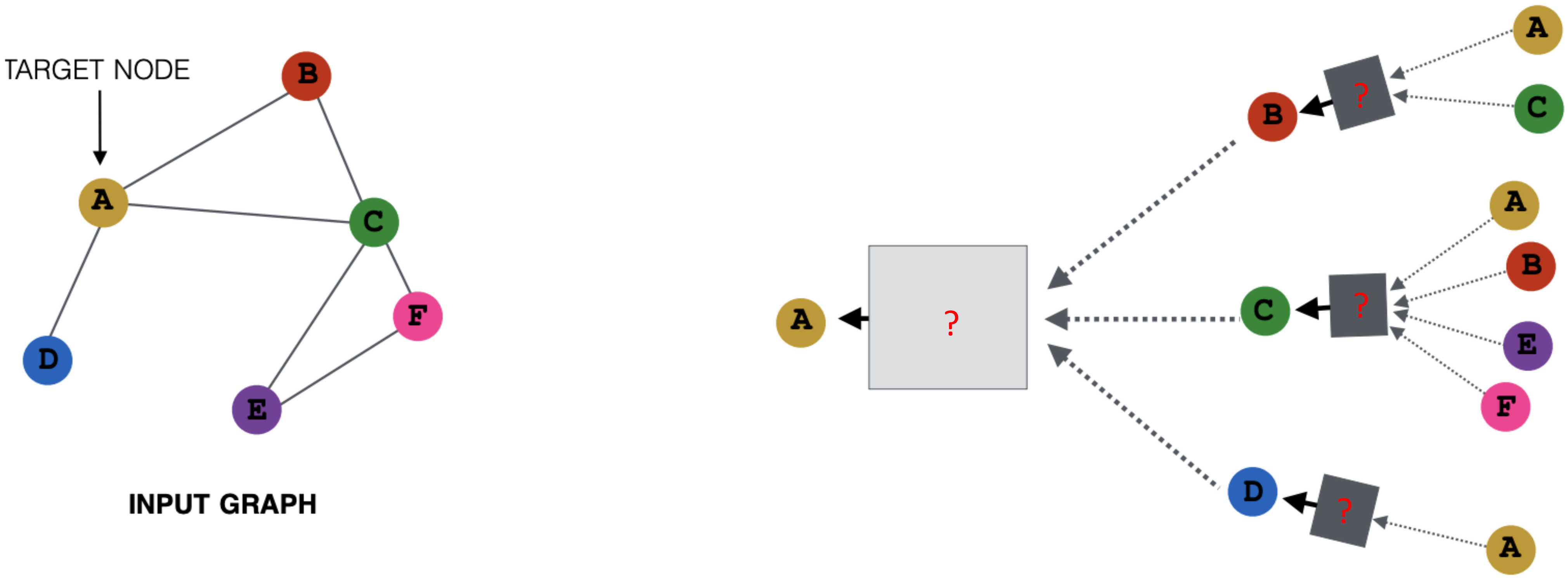


INPUT GRAPH



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

$$h_v^0 = x_v$$

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

Average of neighbor's previous layer embeddings

Total number of layers

Non-linearity (e.g., ReLU)

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

Embedding after L layers of neighborhood aggregation

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

$$h_v^0 = x_v$$

Initial 0-th layer embeddings are equal to node features

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

embedding of v at layer l

Average of neighbor's previous layer embeddings

Total number of layers

Non-linearity (e.g., ReLU)

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

Embedding after L layers of neighborhood aggregation

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

embedding of v at layer l

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

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

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

$$h_v^0 = x_v$$

Initial 0-th layer embeddings are equal to node features

embedding of v at layer l

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

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

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

$$h_v^0 = x_v$$

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

Some layer l

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

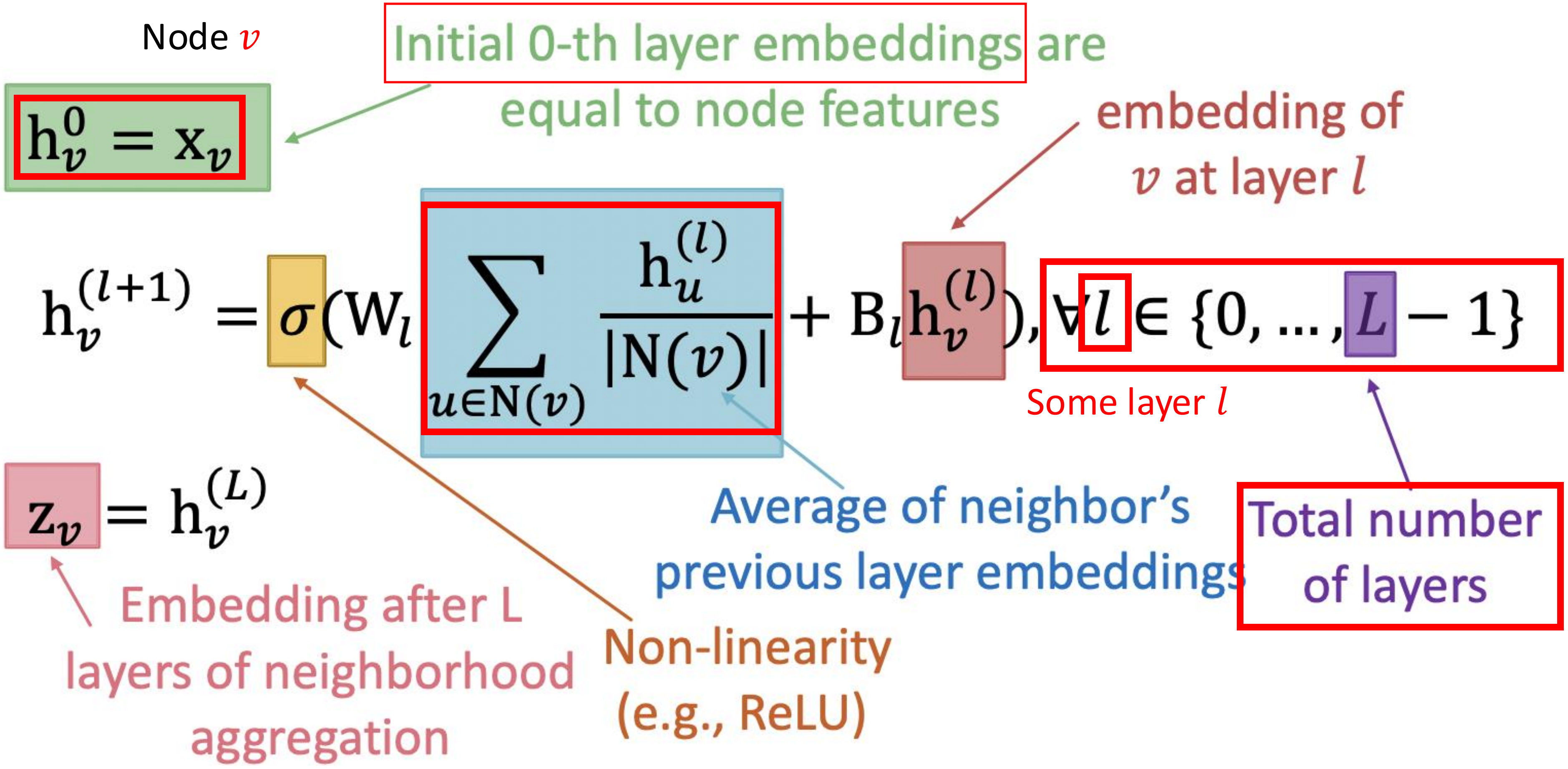
Embedding after L layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

Average of neighbor's previous layer embeddings

Total number of layers

Graph networks: aggregate neighbors



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

Some layer l

u: nearest neighbors

Average of neighbor's previous layer embeddings

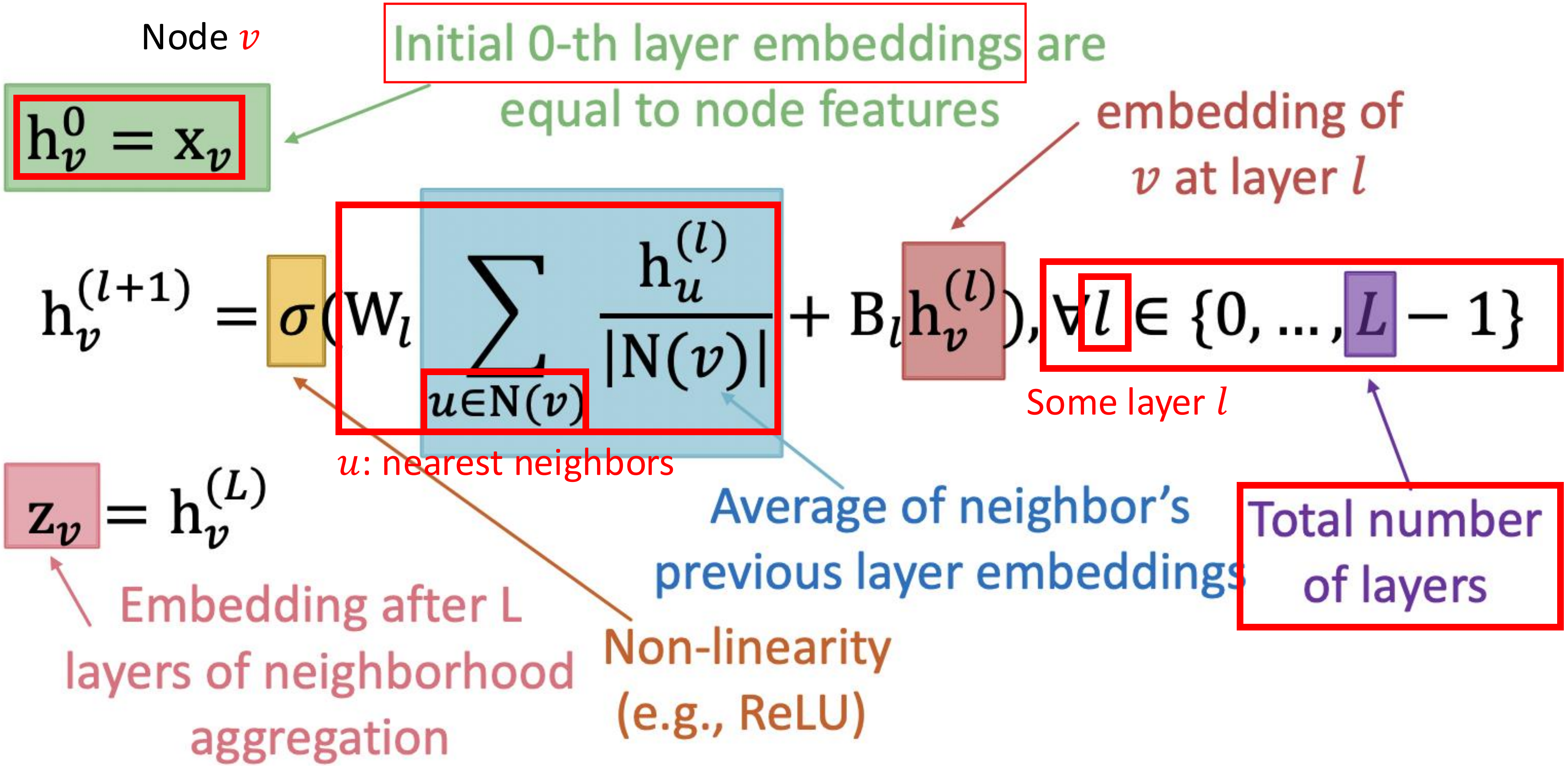
Total number of layers

Non-linearity (e.g., ReLU)

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

Embedding after L layers of neighborhood aggregation

Graph networks: aggregate neighbors



Graph networks: aggregate neighbors

Node v

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

Initial 0-th layer embeddings are equal to node features

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

embedding of v at layer l

Some layer l

u: nearest neighbors

Average of neighbor's previous layer embeddings

Total number of layers

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

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

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

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

Output of nonlinear function

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

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

Initial 0-th layer embeddings are equal to node features

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

Output of nonlinear function

embedding of v at layer l

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

Some layer l

u: nearest neighbors

Average of neighbor's previous layer embeddings

Total number of layers

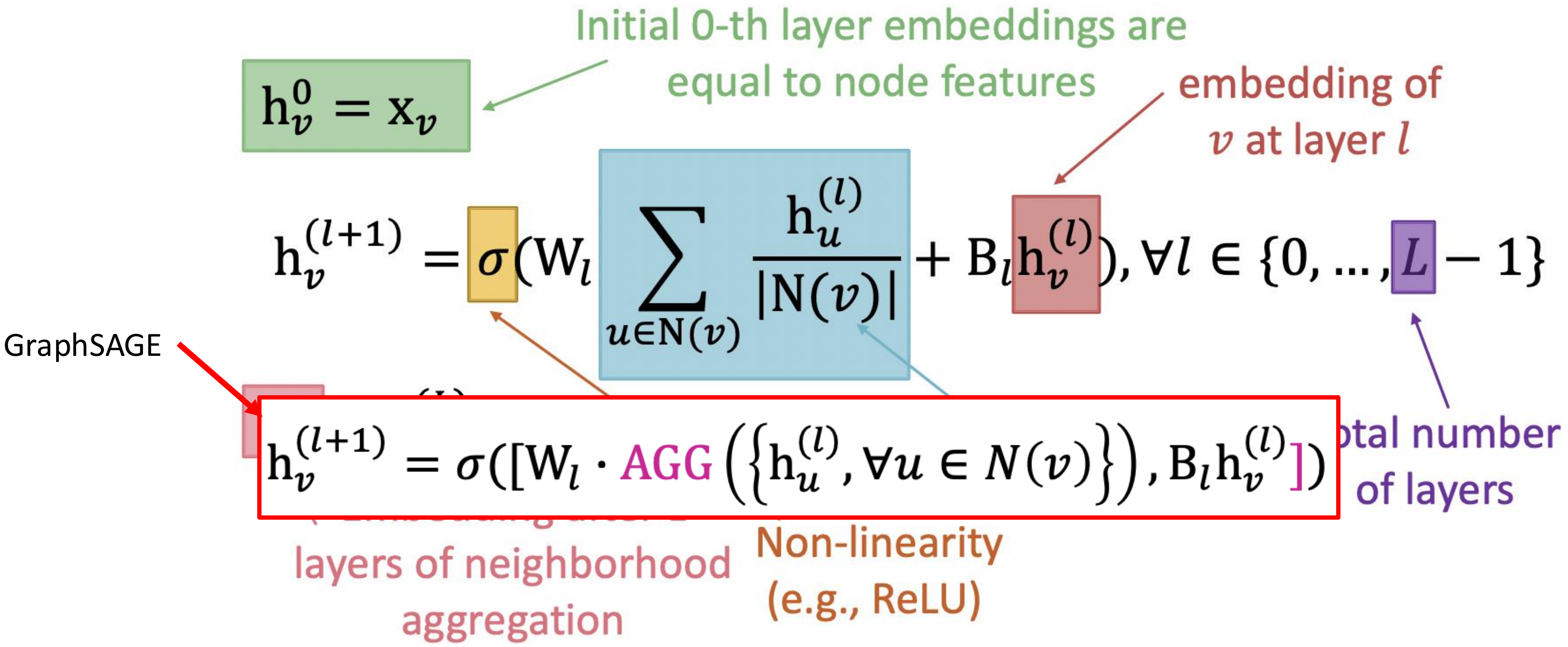
Final layer

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

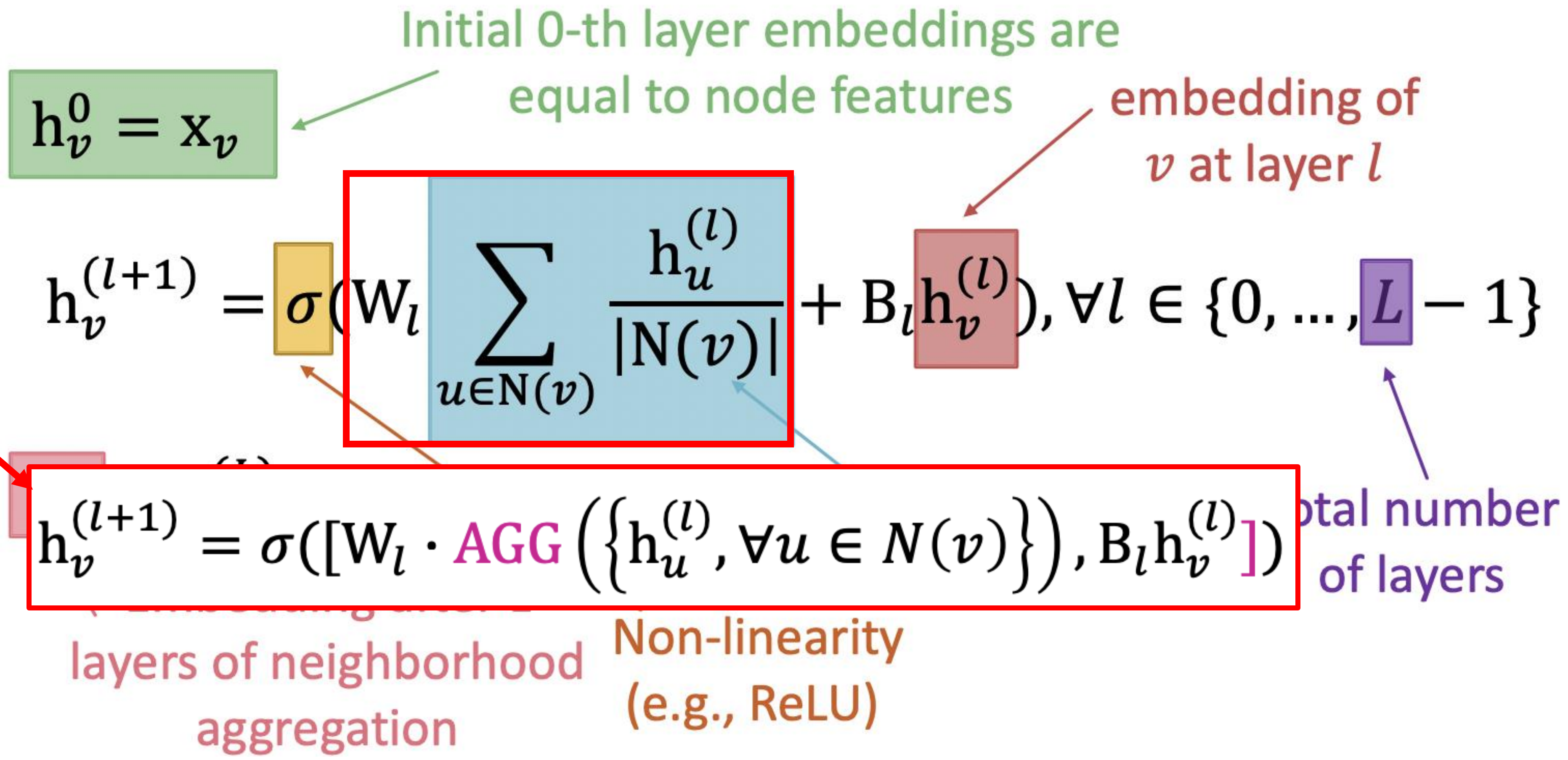
Embedding after L layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

Graph networks: aggregate neighbors



Graph networks: aggregate neighbors

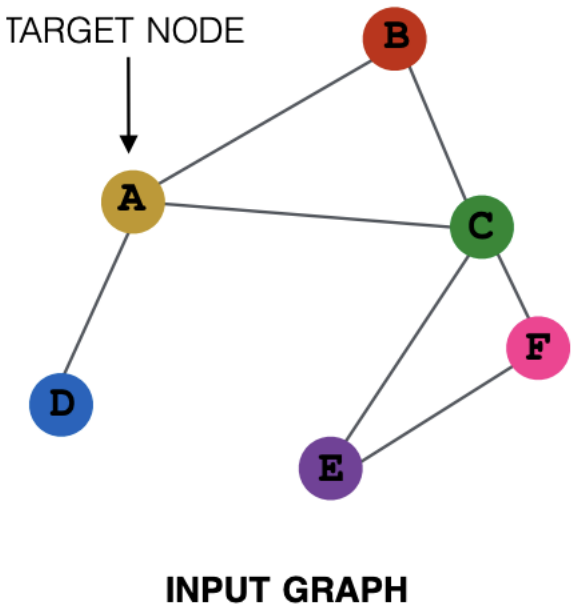


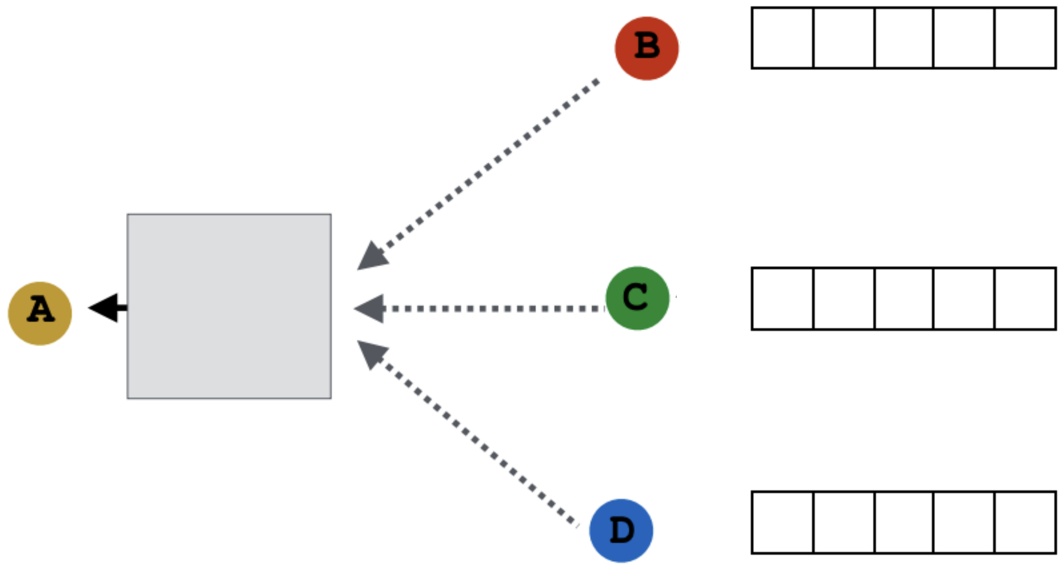
GraphSAGE

$$h_v^{(l+1)} = \sigma([W_l \cdot \mathrm{AGG}\left(\left\{h_u^{(l)}, \forall u \in N(v)\right\}\right), B_l h_v^{(l)}])$$

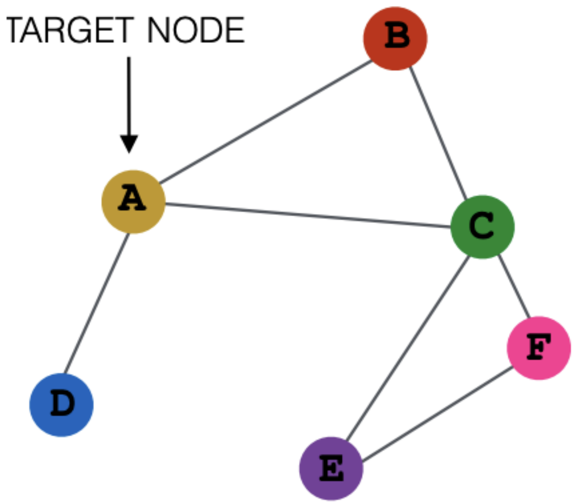
Q: why GraphSAGE can be more general?

Graph networks: aggregate neighbors



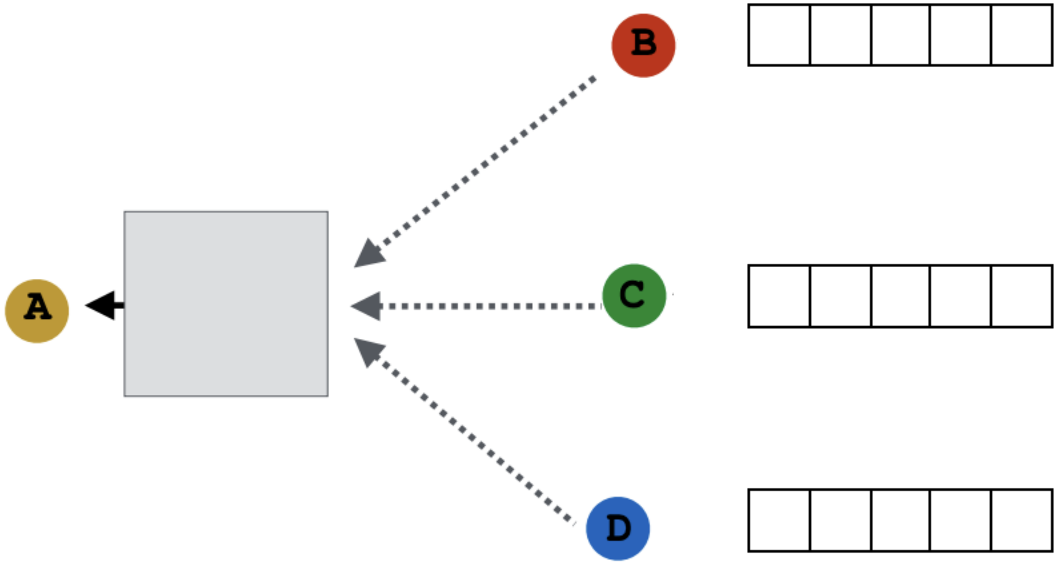


Graph networks: aggregate neighbors

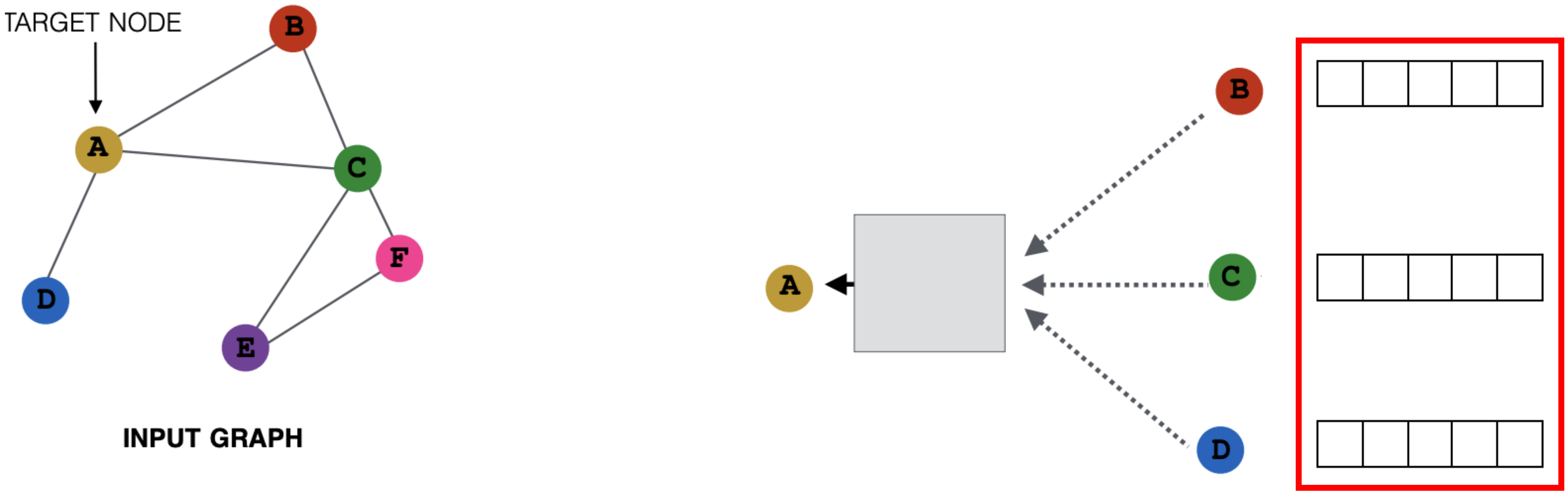


INPUT GRAPH

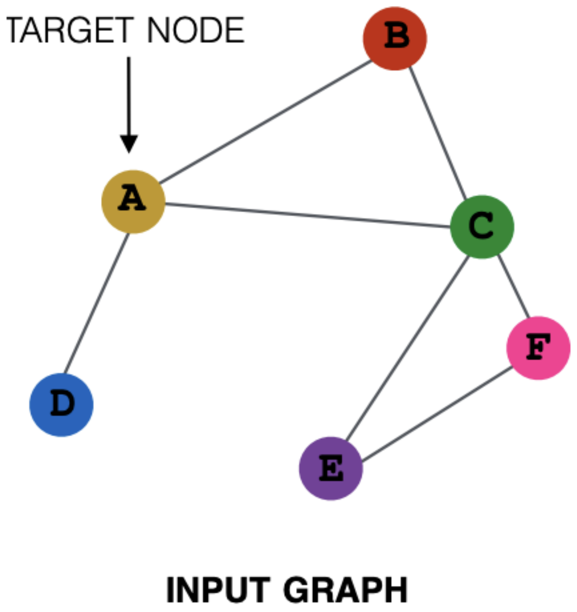
Suppose: we have 5d node features

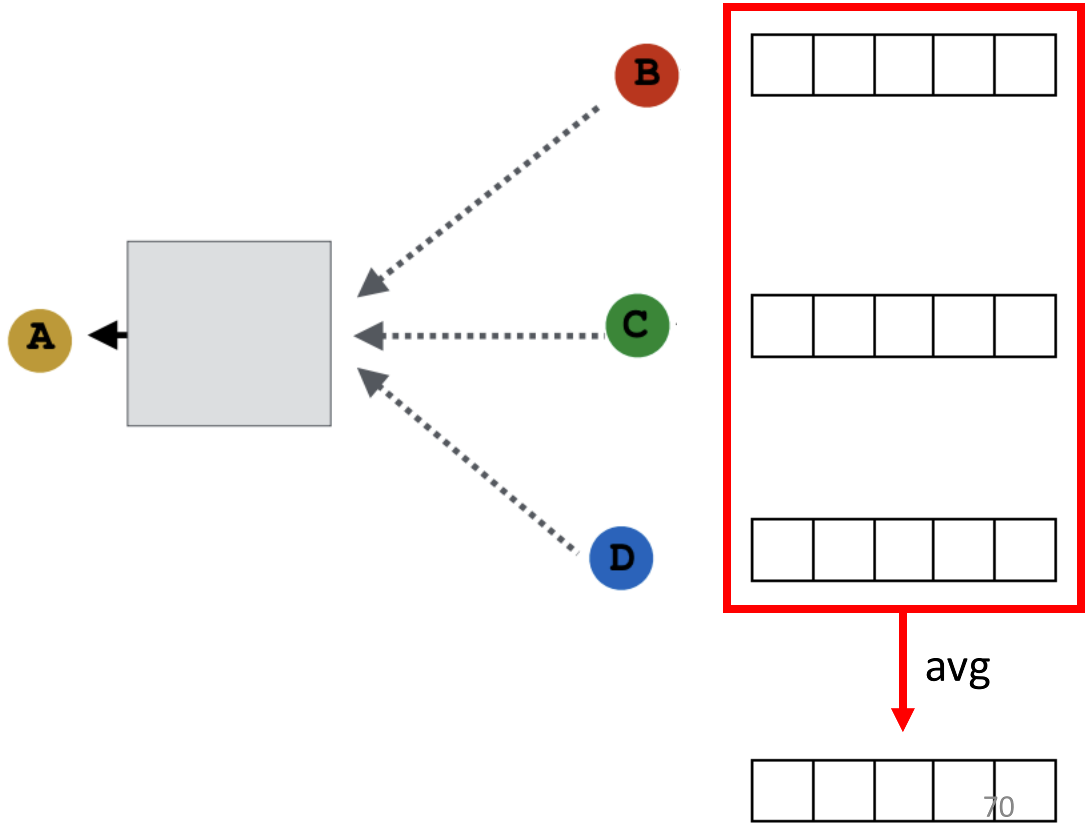


Graph networks: aggregate neighbors

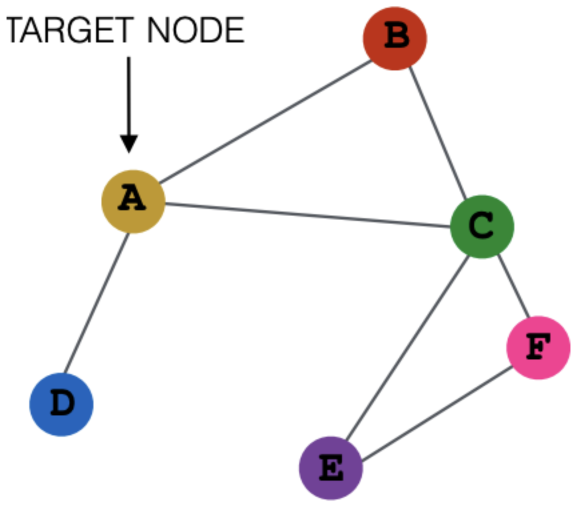


Graph networks: aggregate neighbors

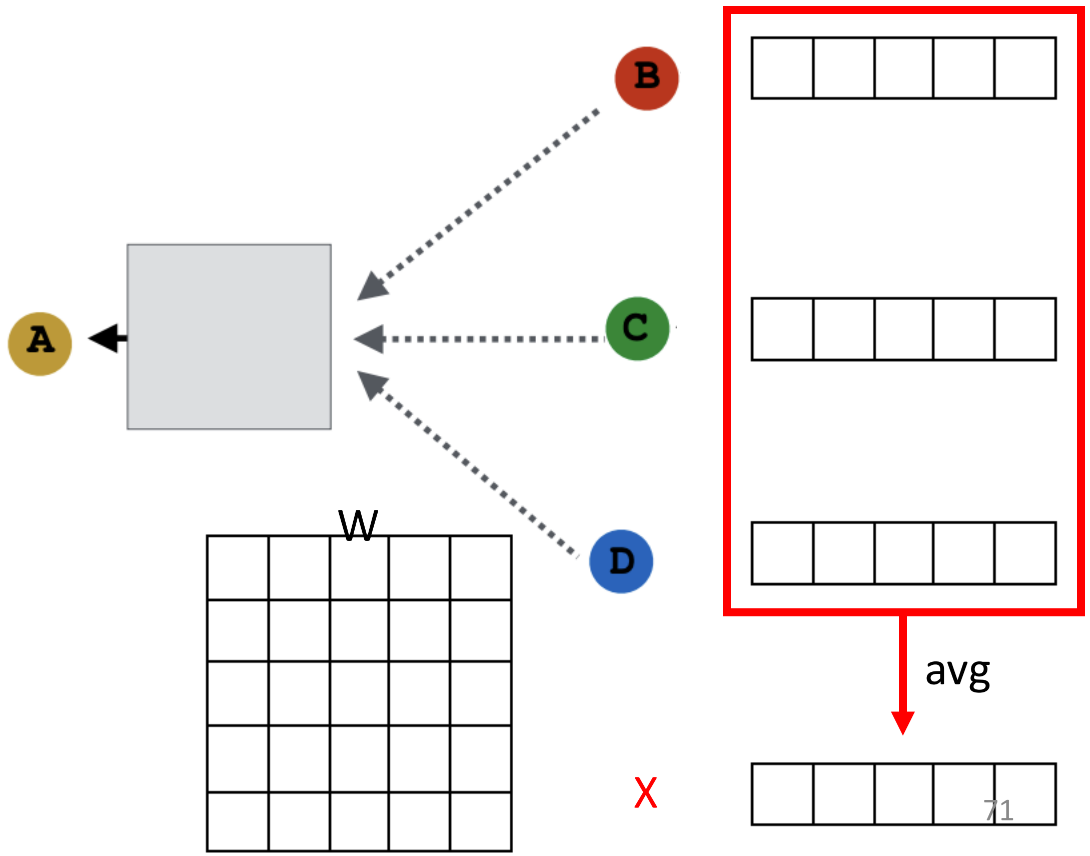




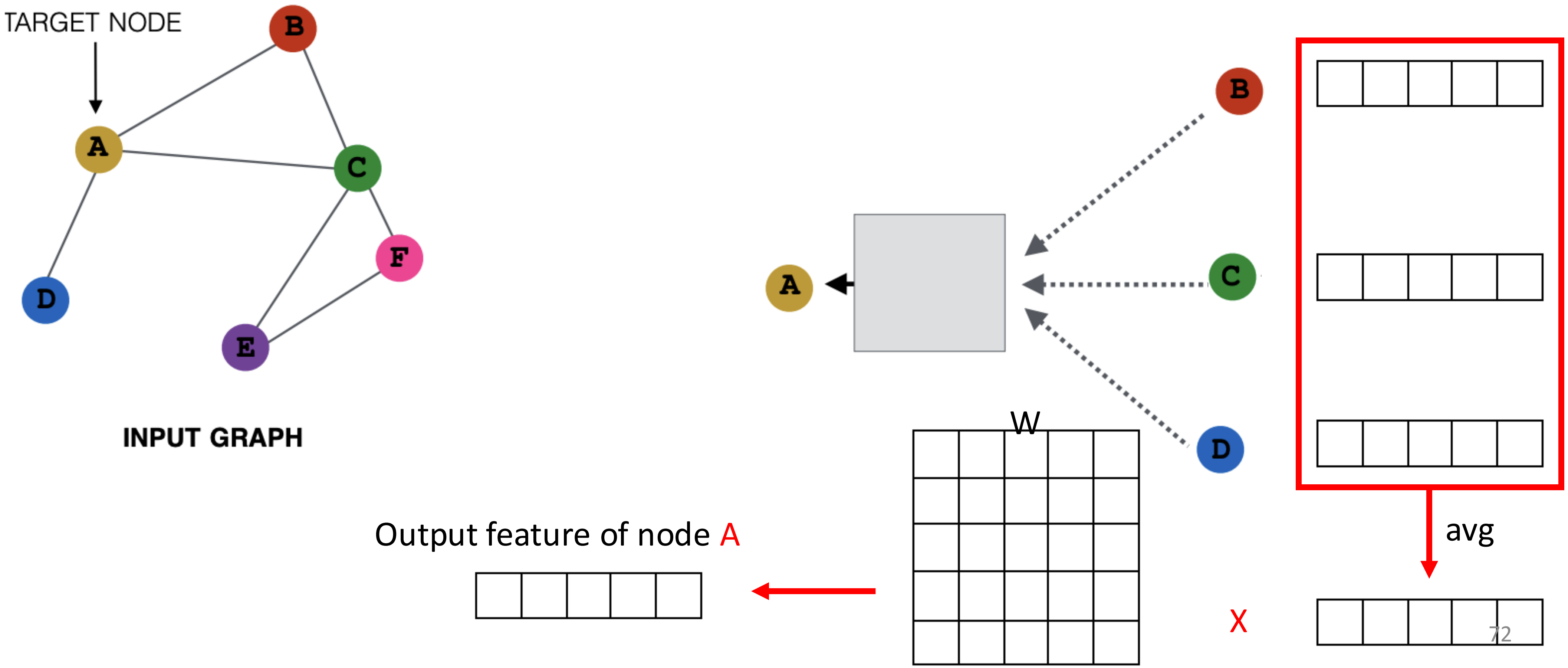
Graph networks: aggregate neighbors



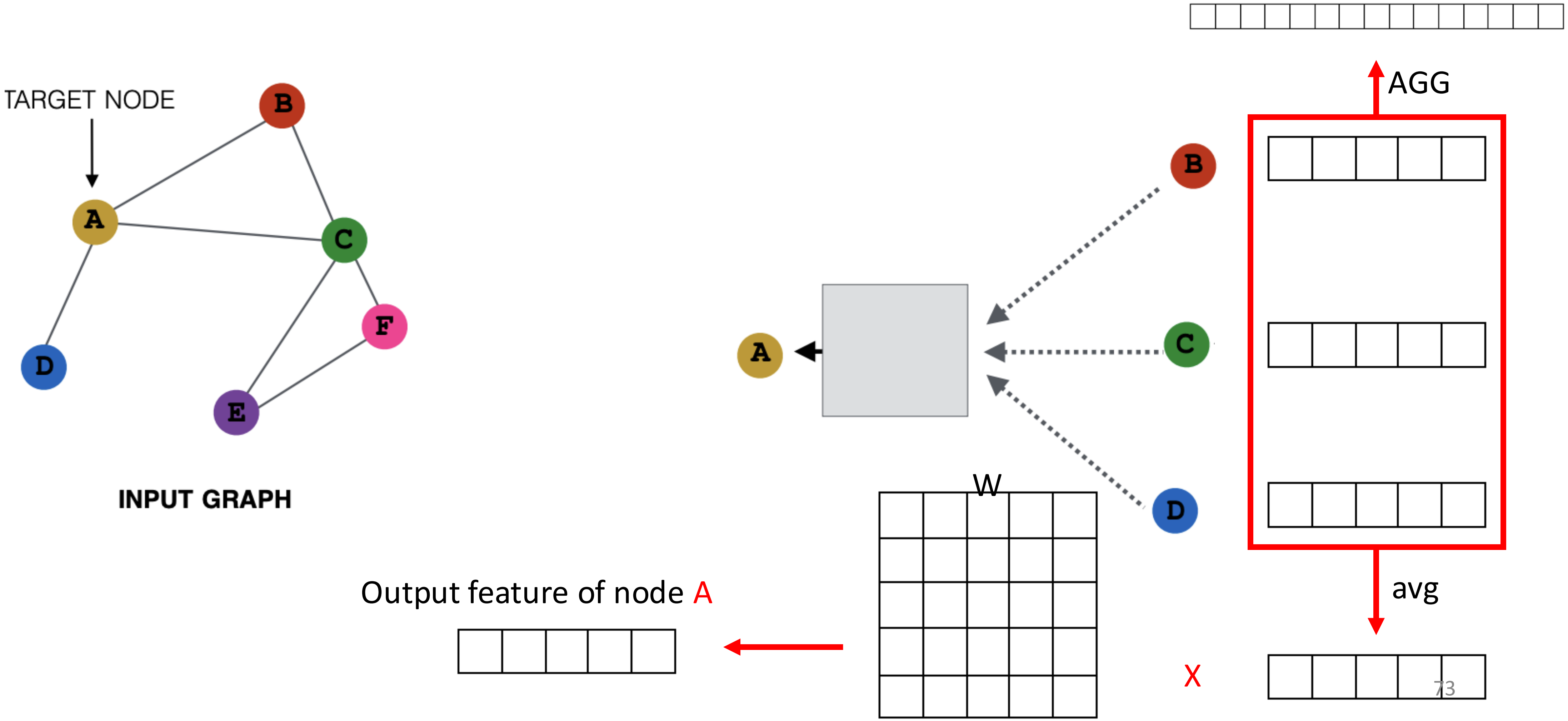
INPUT GRAPH



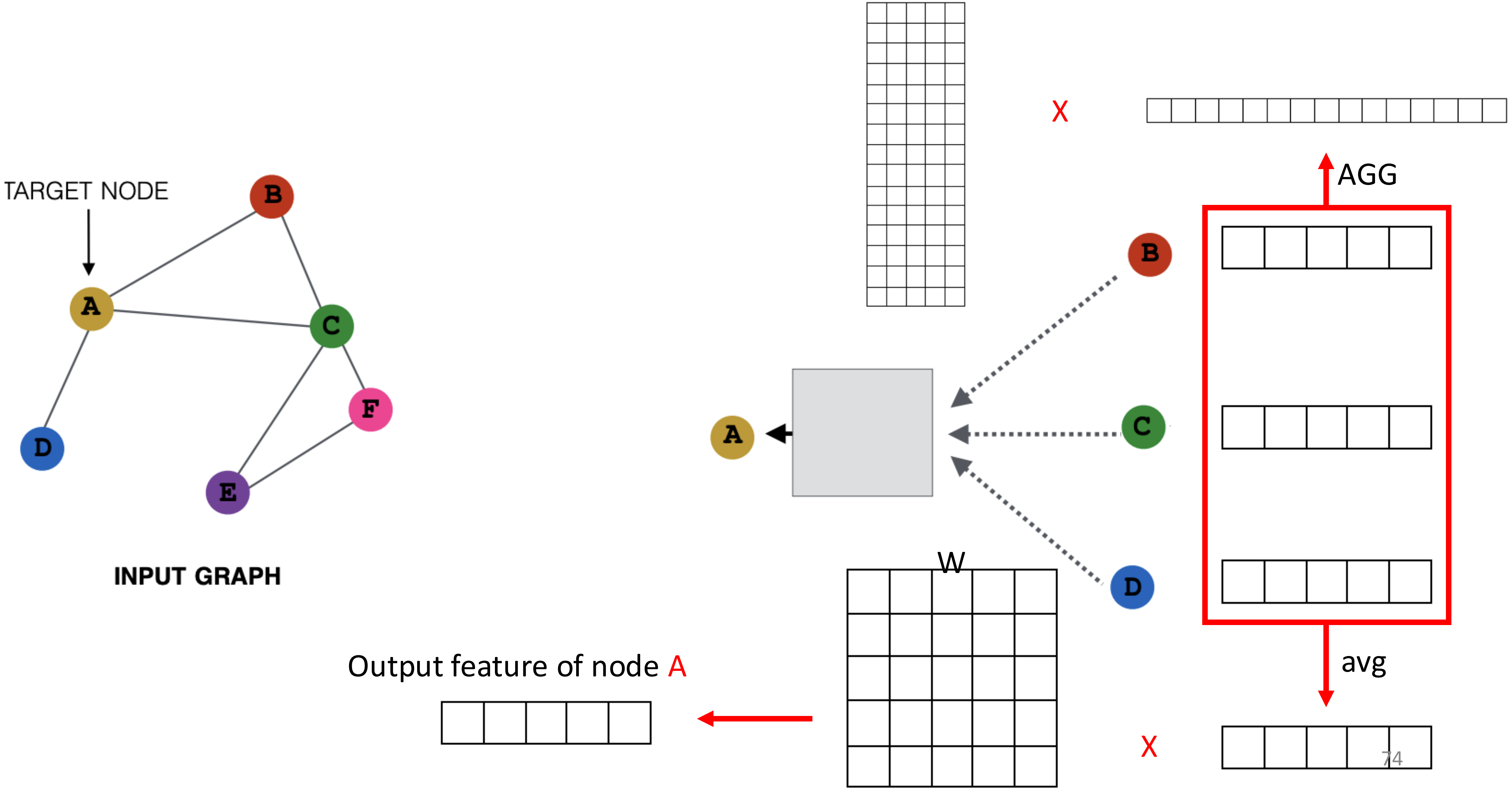
Graph networks: aggregate neighbors



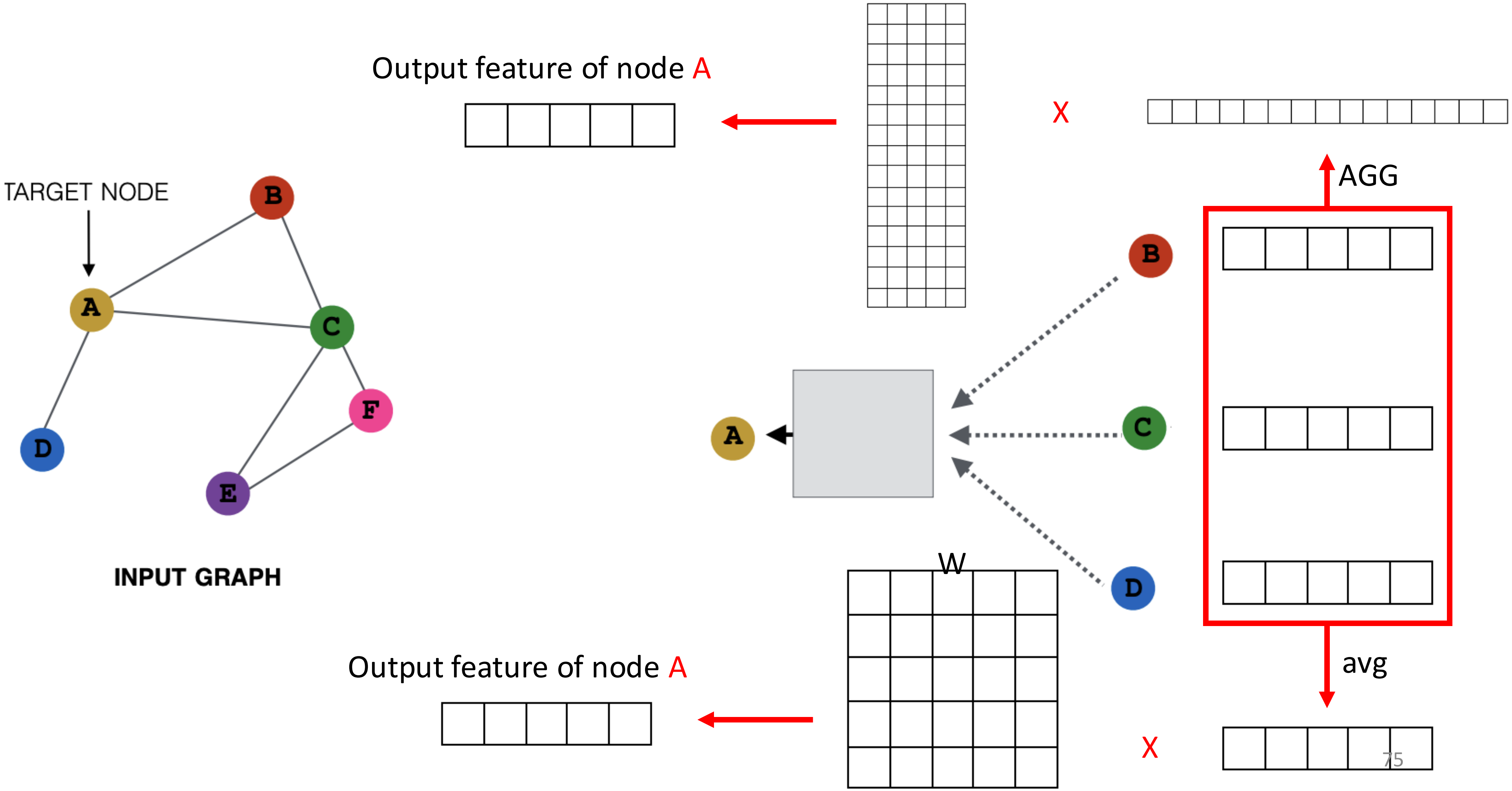
Graph networks: aggregate neighbors



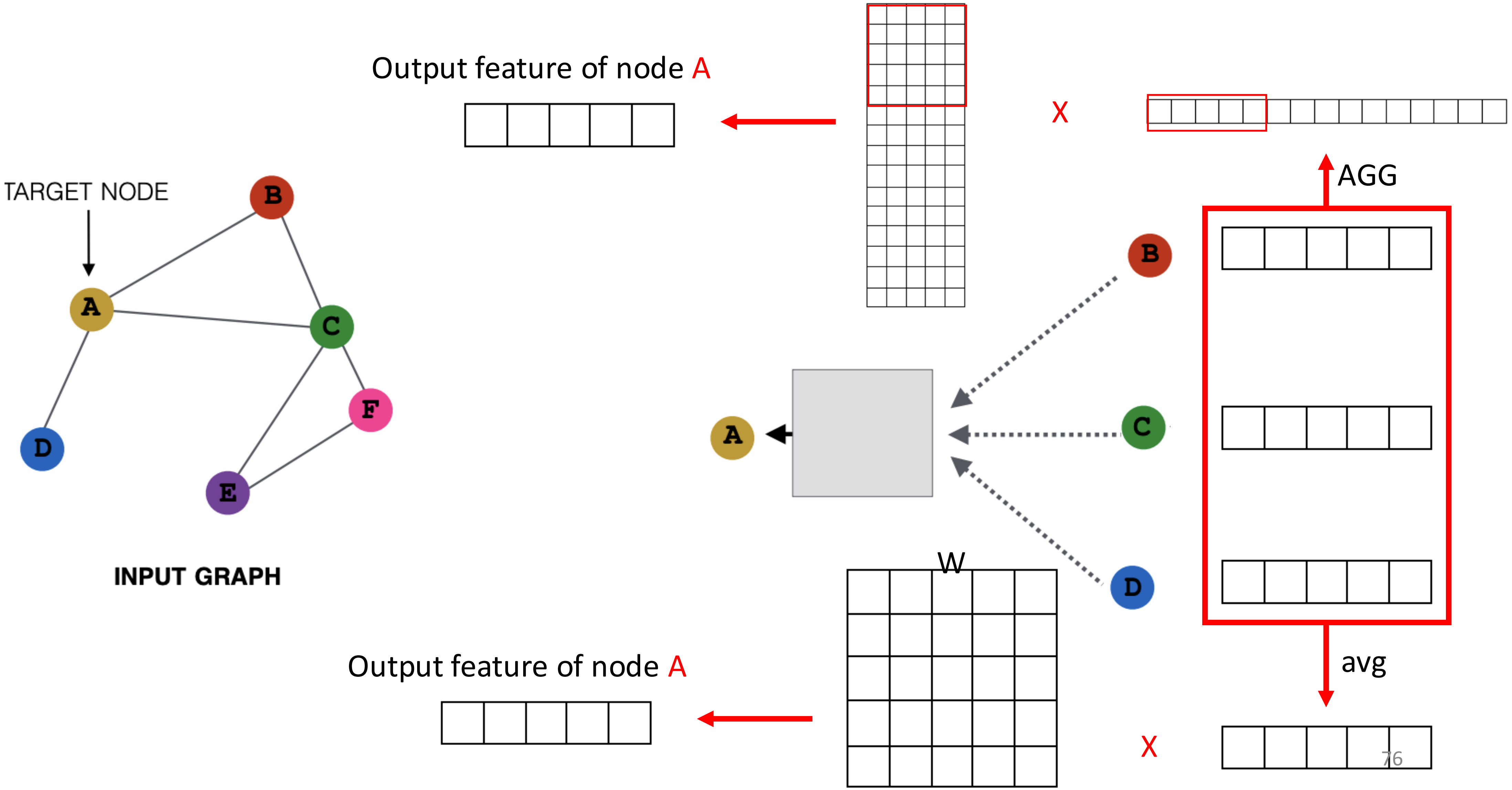
Graph networks: aggregate neighbors



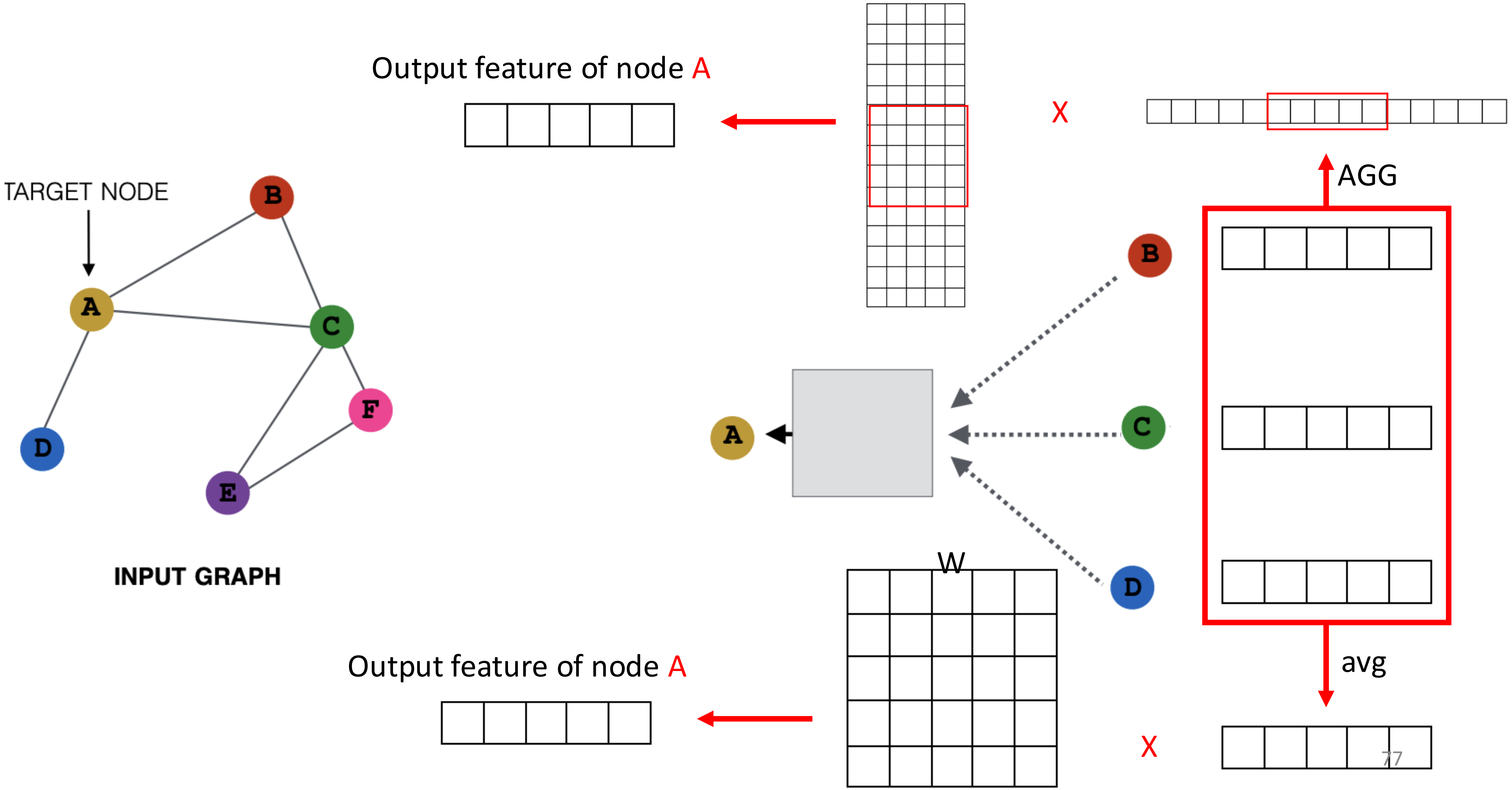
Graph networks: aggregate neighbors



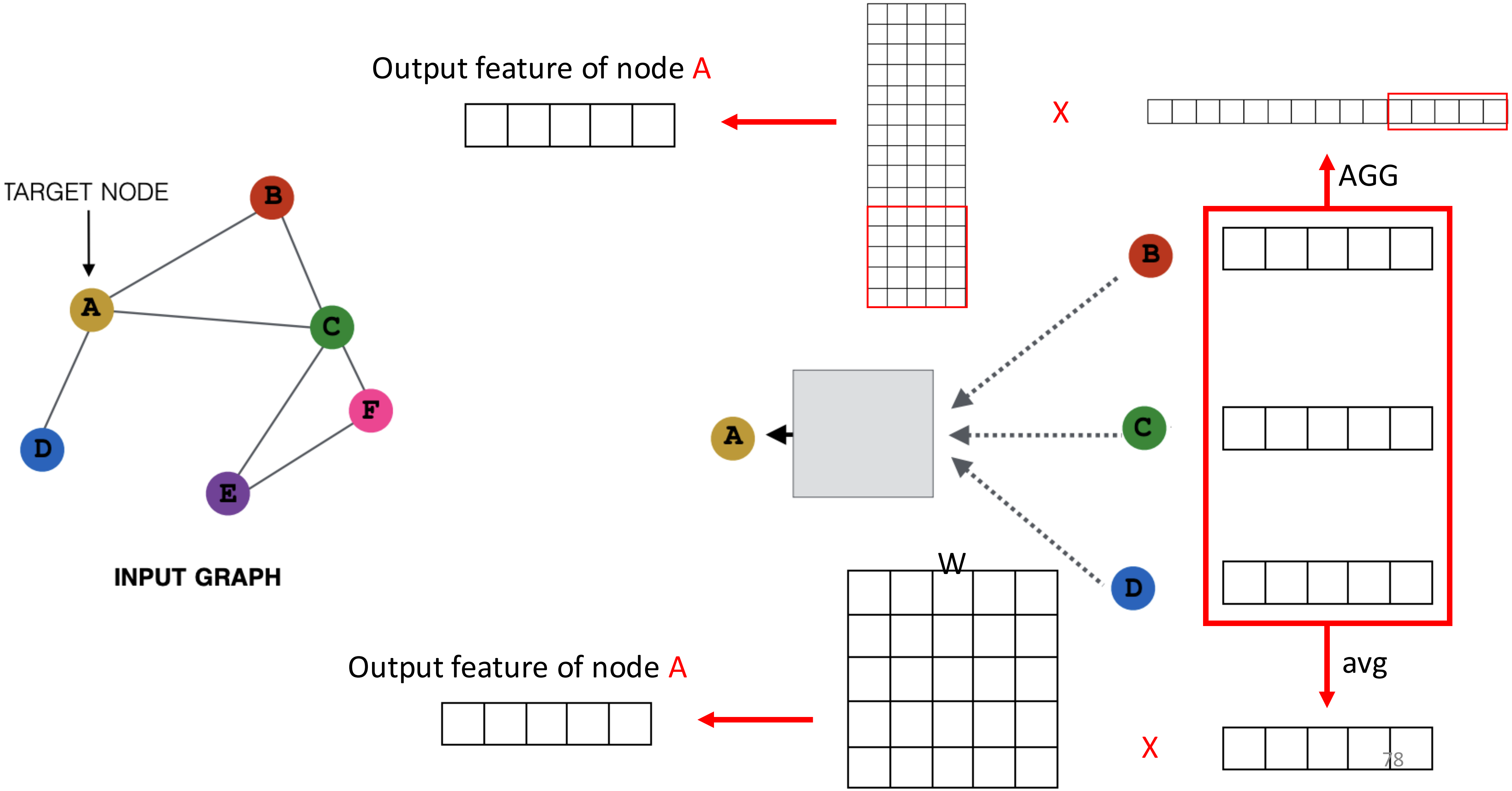
Graph networks: aggregate neighbors



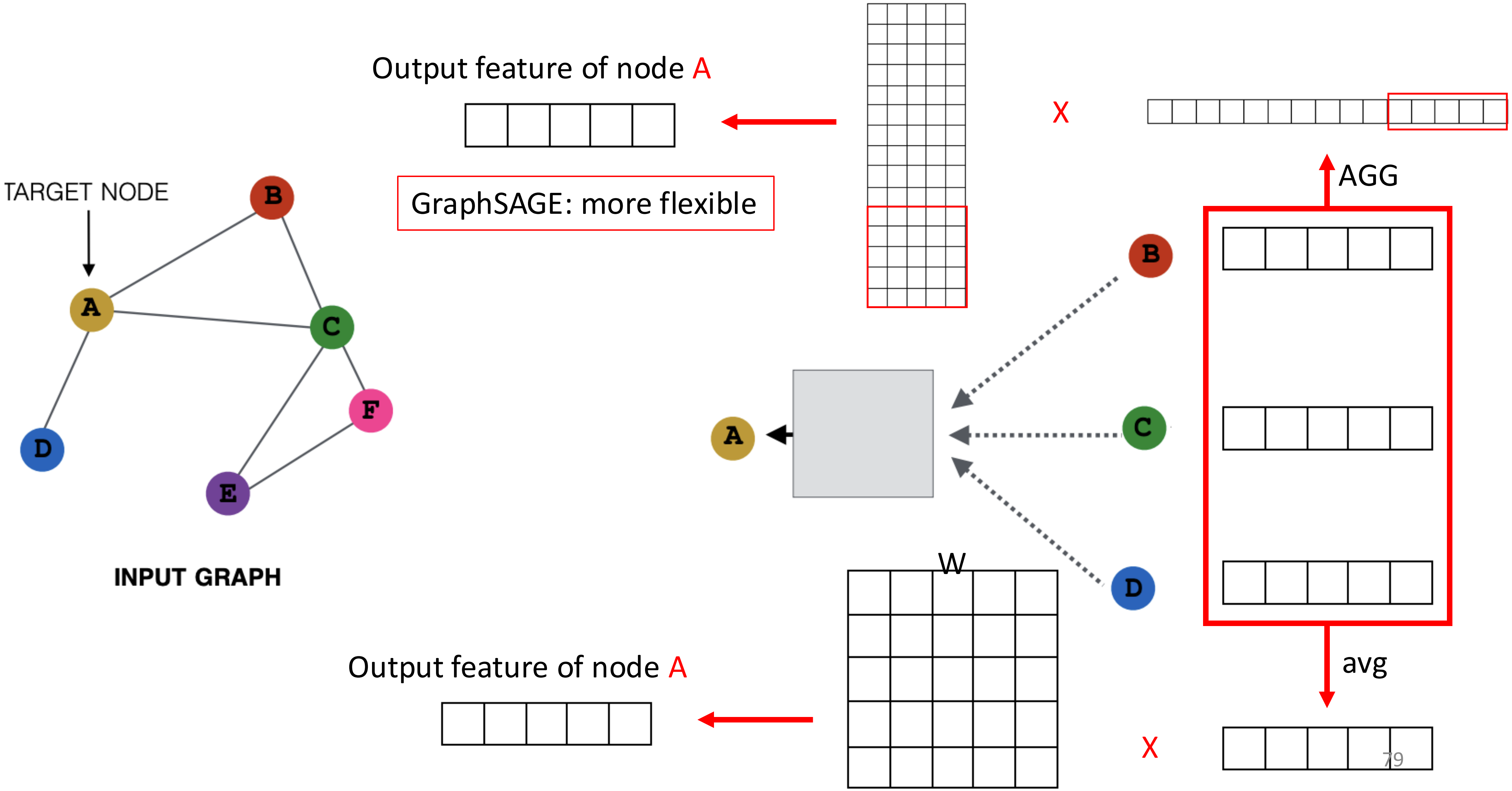
Graph networks: aggregate neighbors



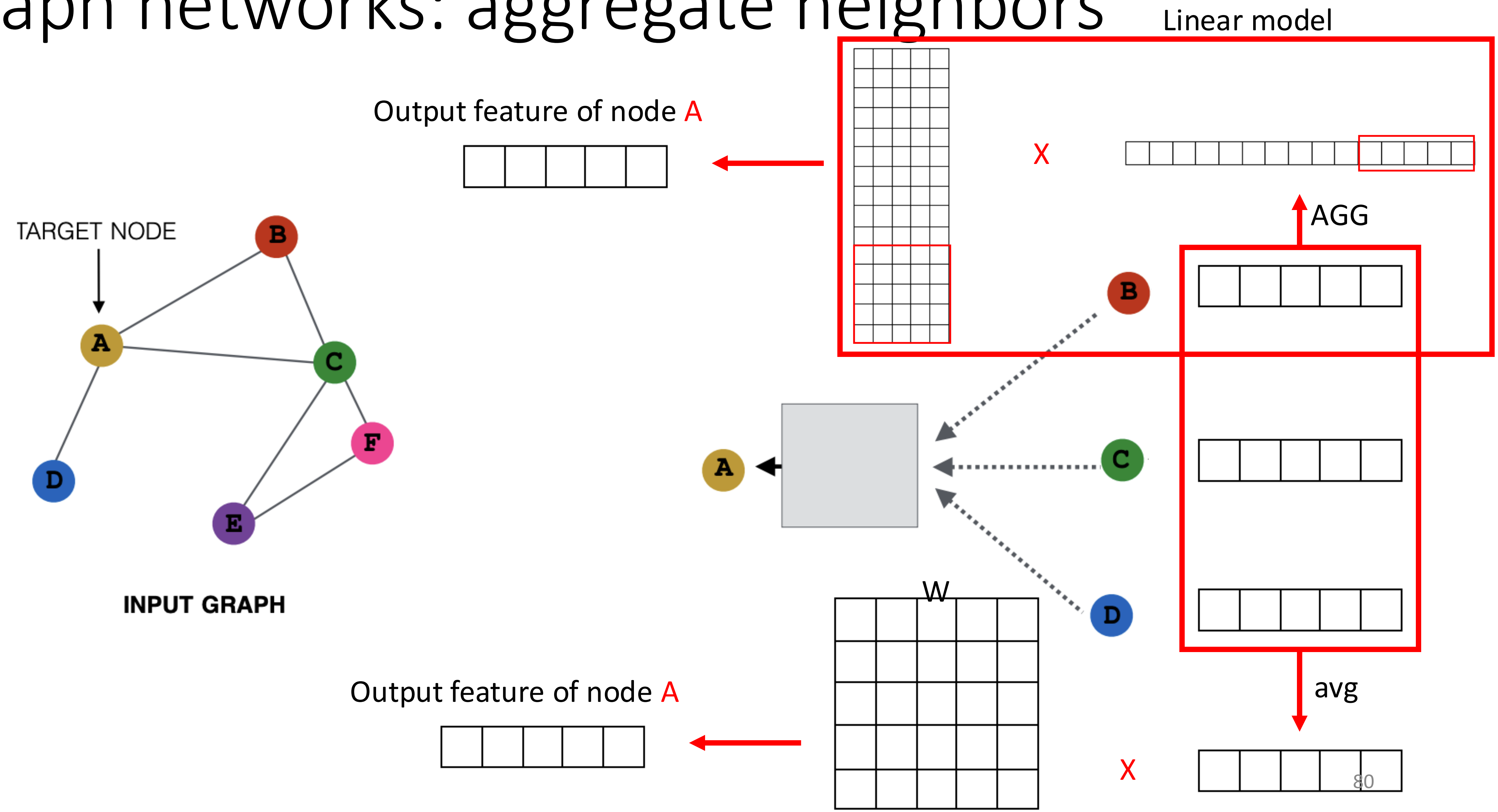
Graph networks: aggregate neighbors



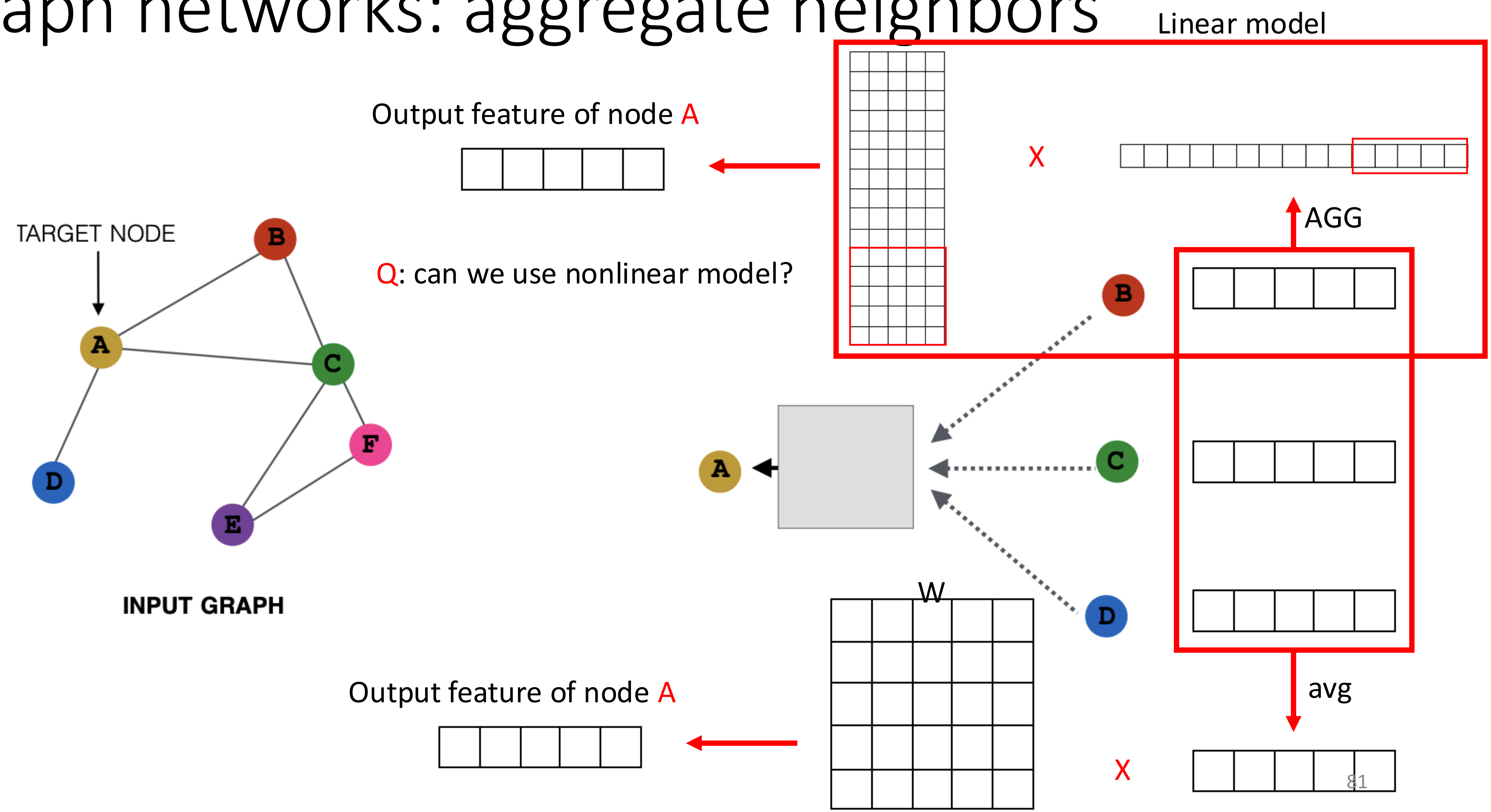
Graph networks: aggregate neighbors



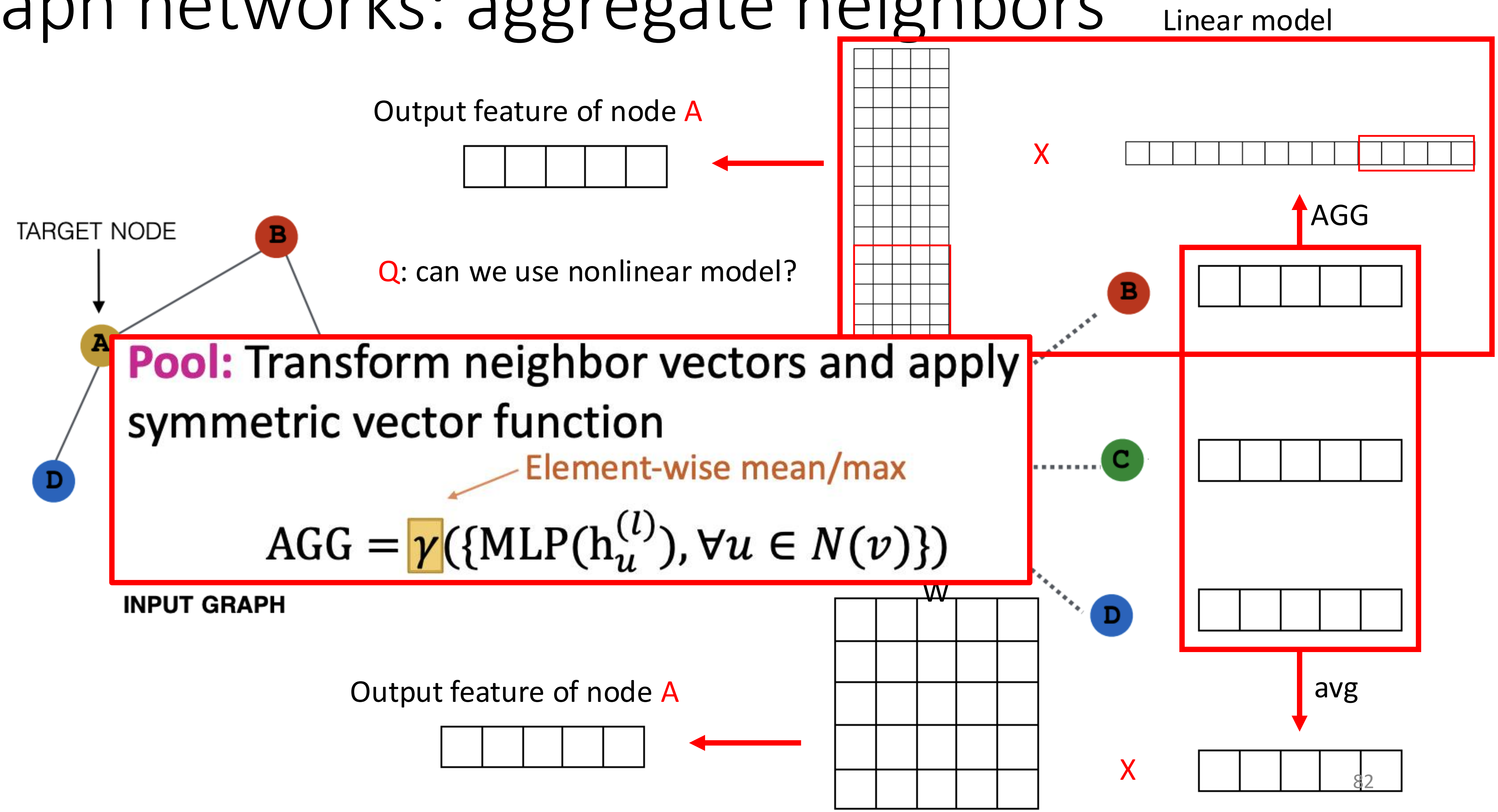
Graph networks: aggregate neighbors



Graph networks: aggregate neighbors



Graph networks: aggregate neighbors

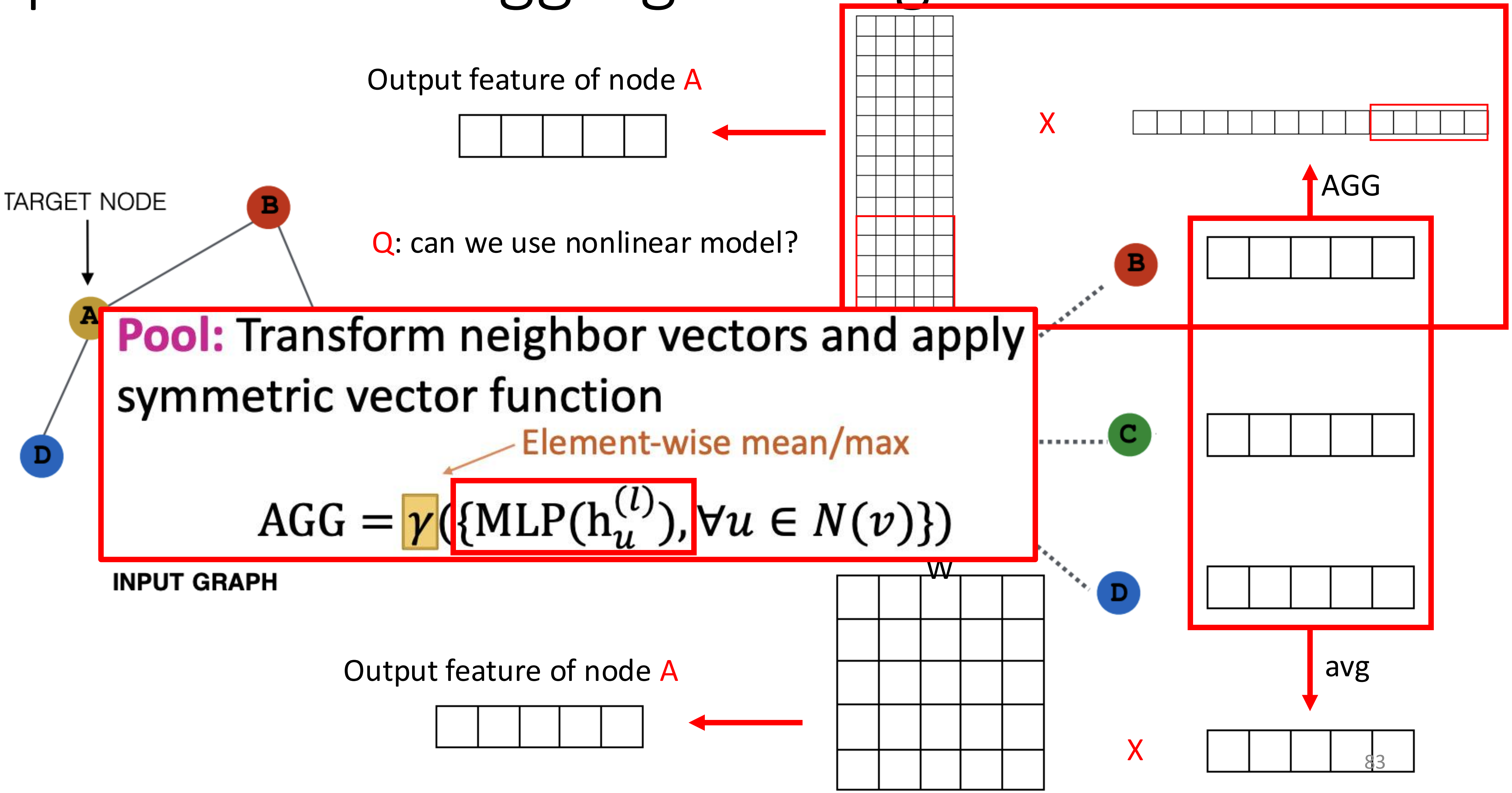


Pool: Transform neighbor vectors and apply symmetric vector function

Element-wise mean/max

$$\mathrm{AGG} = \gamma(\{\mathrm{MLP}(\mathrm{h}_u^{(l)}), \forall u \in N(v)\})$$

Graph networks: aggregate neighbors

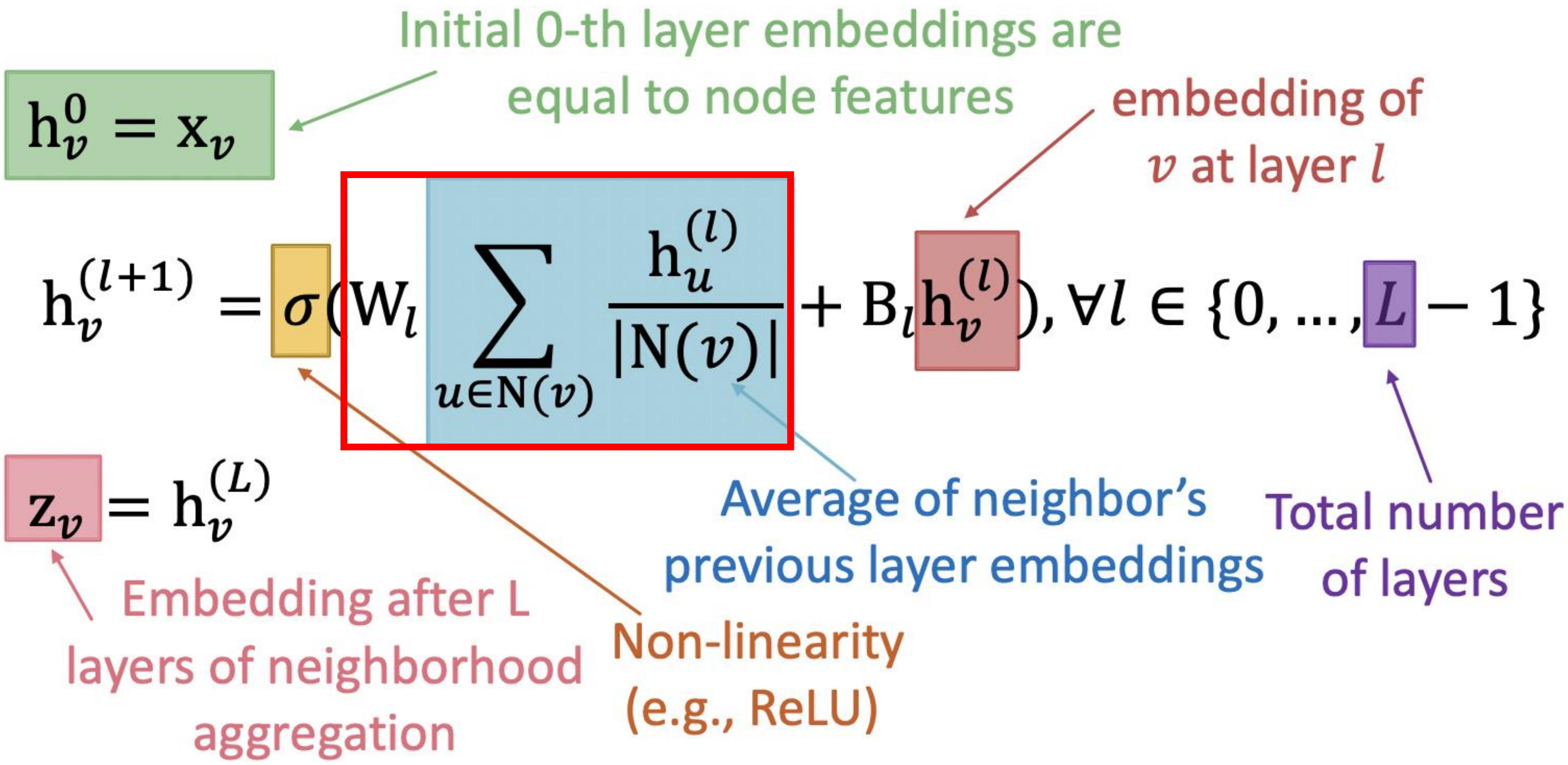


Pool: Transform neighbor vectors and apply symmetric vector function

Element-wise mean/max

$$\text{AGG} = \gamma\left(\{\text{MLP}(\mathrm{h}_u^{(l)}), \forall u \in N(v)\}\right)$$

Graph networks: aggregate neighbors



Graph networks: aggregate neighbors

Initial 0-th layer embeddings are equal to node features

$$h_v^0 = x_v$$

embedding of v at layer l

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

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

Embedding after L layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

Average of neighbor's previous layer embeddings

Total number of layers

Q: can we use attention mechanism?

Graph networks: aggregate neighbors

Initial 0-th layer embeddings are equal to node features

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

embedding of v at layer l

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

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

Embedding after L layers of neighborhood aggregation

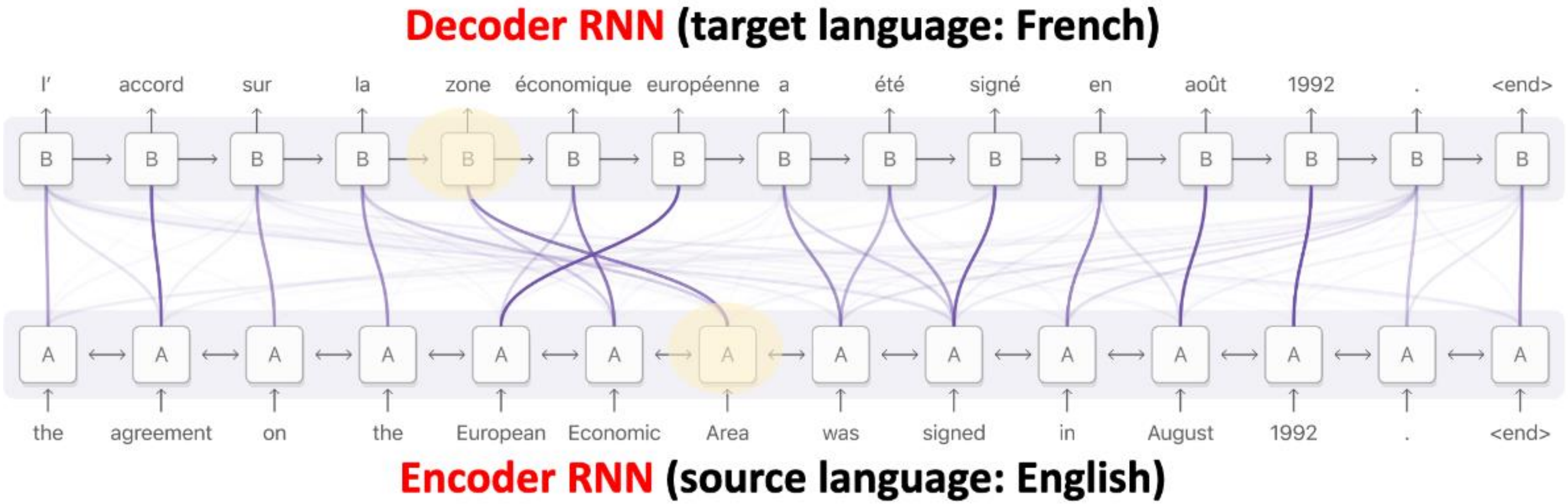
Non-linearity (e.g., ReLU)

Average of neighbor's previous layer embeddings

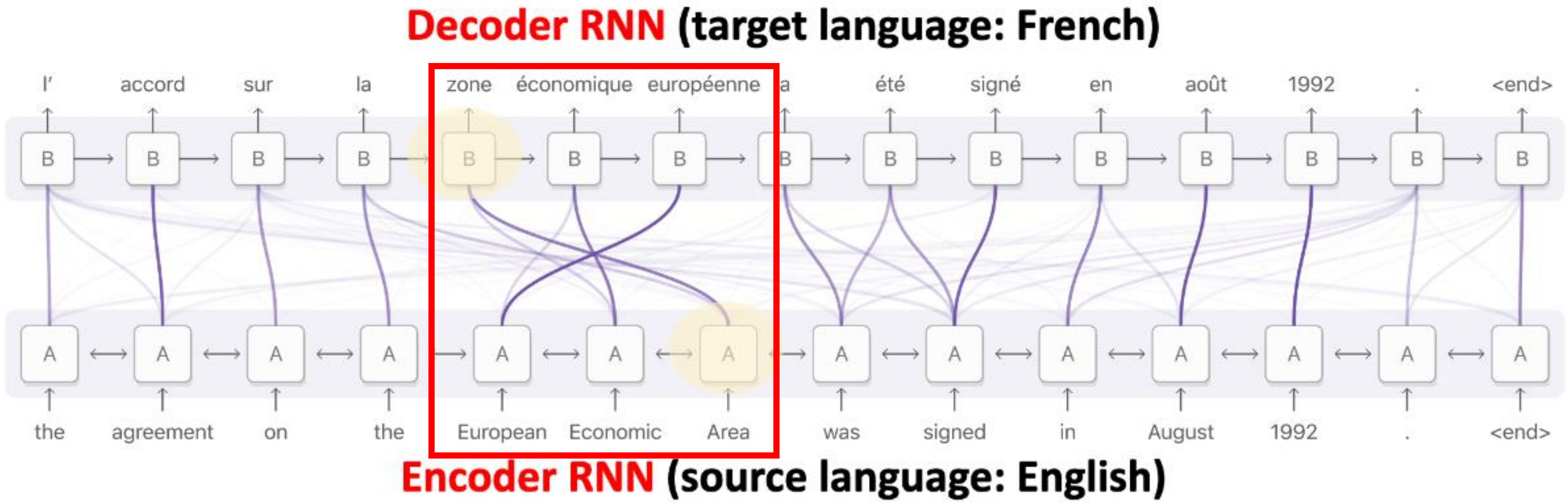
Total number of layers

Q: can we use attention mechanism? ← Not all node's neighbors are equally important

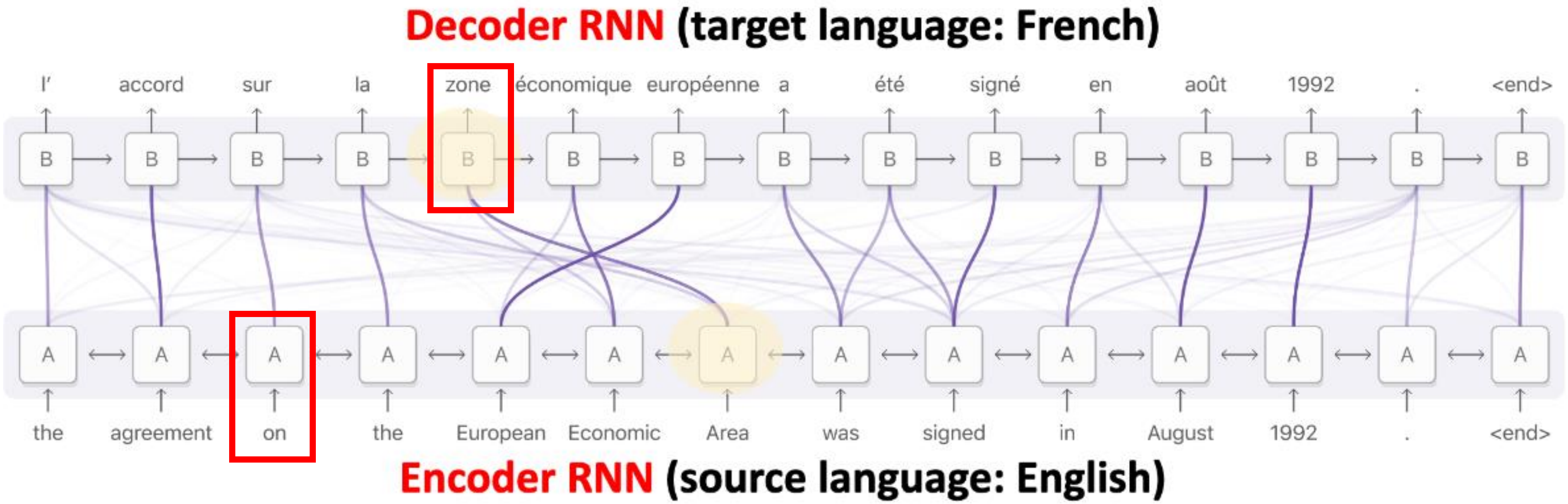
Input-output correlation



Input-output correlation



Input-output correlation



Graph attention networks

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

Attention weights

Graph attention networks

Q: what stand for importance of a node?

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

Attention weights

Graph attention networks

Node degree?

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

Attention weights

Graph attention networks

Learnable parameters?

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

Attention weights

Graph attention networks

Learnable parameters?

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

Attention weights
Between v and u

Graph attention networks

Learnable parameters?

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

Attention weights
Between v and u

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

Graph attention networks

Learnable parameters?

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

Attention weights

Between v and u

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

Graph attention networks

Learnable parameters?

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

Attention weights

Between v and u

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

Make use of last layer's output

Graph attention networks

Learnable parameters?

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

Attention weights

Between v and u

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

Make use of last layer's output

Graph attention networks

Learnable parameters?

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

Attention weights
Between v and u

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

Make use of last layer's output

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

Graph attention networks

Learnable parameters?

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

Attention weights
Between v and u

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

Make use of last layer's output

Softmax function

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

Graph attention networks

Learnable parameters?

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

Attention weights
Between v and u

Softmax function→normalized weights

$$e_{vu} = a\left(\mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)}, \mathbf{W}^{(l)} \boldsymbol{h}_v^{(l-1)}\right) \Longrightarrow \alpha_{vu} = \frac{\exp(e_{vu})}{\sum_{k \in N(v)} \exp(e_{vk})}$$

Make use of last layer's output

Graph attention networks

Learnable parameters?

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

Attention weights

Between v and u

Softmax function → normalized weights

$$e_{vu} = a\left(\mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)}, \mathbf{W}^{(l)} \mathbf{h}_v^{(l-1)}\right) \Longrightarrow \alpha_{vu} = \frac{\exp(e_{vu})}{\sum_{k \in N(v)} \exp(e_{vk})}$$

Make use of last layer's output

Graph attention networks

Learnable parameters?

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

Attention weights

Between v and u

Q: what is a's structure?

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

Make use of last layer's output

Softmax function→normalized weights

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

Graph attention networks

Learnable parameters?

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

Attention weights

Between v and u

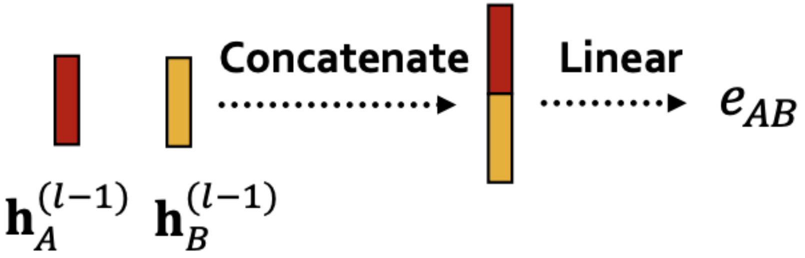
Q: what is a's structure?

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

Make use of last layer's output

Softmax function→normalized weights

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



http://web.stanford.edu/class/cs224w/slides/07-GNN2.pdf

Graph attention networks

Learnable parameters?

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

Attention weights

Between v and u

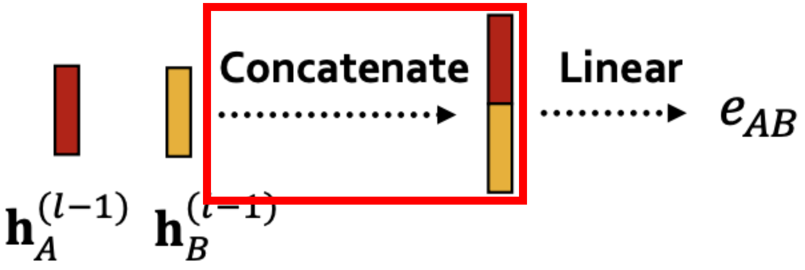
Q: what is a's structure?

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

Make use of last layer's output

Softmax function→normalized weights

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



http://web.stanford.edu/class/cs224w/slides/07-GNN2.pdf

Graph attention networks

Learnable parameters?

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

Attention weights

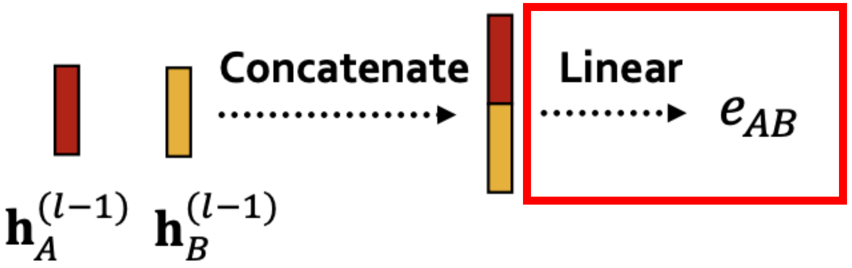
Between v and u

Q: what is a's structure?

Softmax function→normalized weights

$$e_{vu} = a\left(\mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)}, \mathbf{W}^{(l)} \mathbf{h}_v^{(l-1)}\right) \Longrightarrow \alpha_{vu} = \frac{\exp(e_{vu})}{\sum_{k \in N(v)} \exp(e_{vk})}$$

Make use of last layer's output



http://web.stanford.edu/class/cs224w/slides/07-GNN2.pdf

Graph attention networks

Learnable parameters?

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

Attention weights

Between v and u

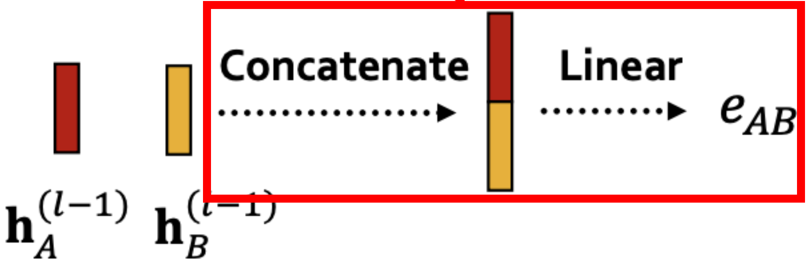
Q: what is a's structure?

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

Make use of last layer's output

Softmax function→normalized weights

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



http://web.stanford.edu/class/cs224w/slides/07-GNN2.pdf

Graph attention networks

Learnable parameters?

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

Attention weights

Between v and u

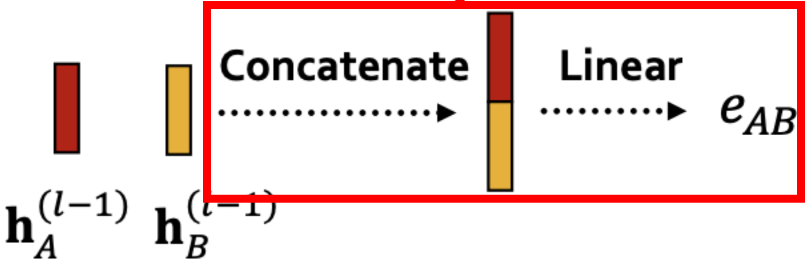
Q: what is a's structure?

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

Make use of last layer's output

Softmax function→normalized weights

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



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

http://web.stanford.edu/class/cs224w/slides/07-GNN2.pdf

Graph attention networks

Learnable parameters?

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

Attention weights

Between v and u

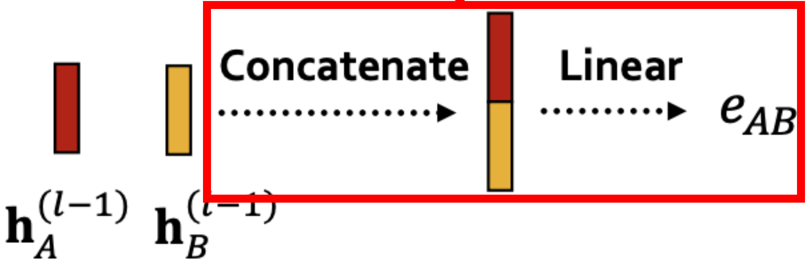
Q: what is a's structure?

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

Make use of last layer's output

Softmax function→normalized weights

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



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

http://web.stanford.edu/class/cs224w/slides/07-GNN2.pdf

Graph attention networks

Learnable parameters?

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

Attention weights
Between v and u

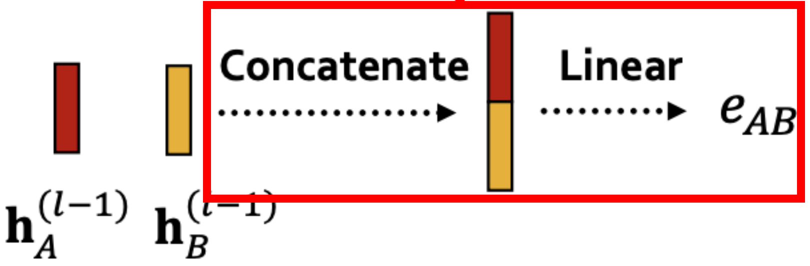
Q: what is a's structure?

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

Make use of last layer's output

Softmax function → normalized weights

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



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

http://web.stanford.edu/class/cs224w/slides/07-GNN2.pdf

Graph attention networks

Learnable parameters?

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

Attention weights

Between v and u

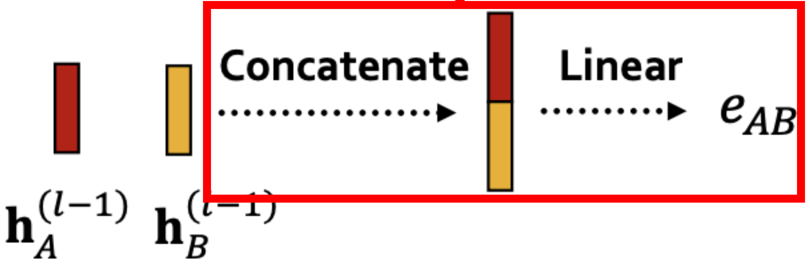
Softmax function→normalized weights

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

Q: what is a's structure?

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

Make use of last layer's output



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

$$= \text{Linear}\left(\text{Concat}\left(\mathbf{W}^{(l)} \mathbf{h}_A^{(l-1)}, \mathbf{W}^{(l)} \mathbf{h}_B^{(l-1)}\right)\right)$$

http://web.stanford.edu/class/cs224w/slides/07-GNN2.pdf