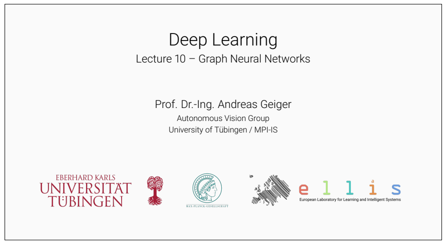
Introduction to Graph Neural Networks

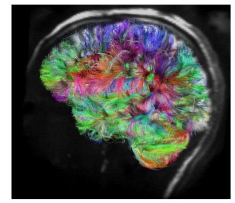
Minji Yoon
(CMU)

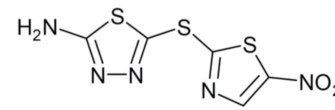
10707 Introduction to Deep Learning, Spring 2022



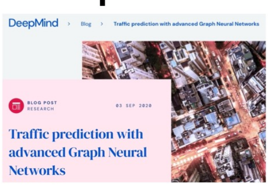
(slide: P. Veličković)

Graphs are everywhere



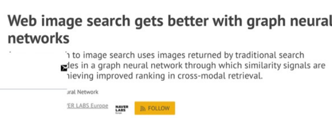


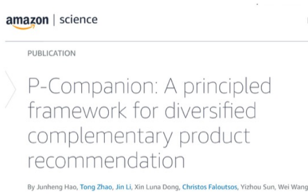
Graph Neural Networks have a large impact on…



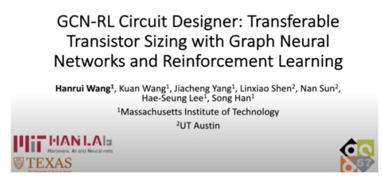
Food Discovery with Uber Eats: Using Graph Learning to Power Recommendations

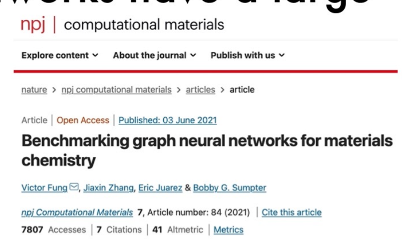
Ankit Jain, Isaac Liu, Ankur Sarda, and Piero Molino December 4, 2019

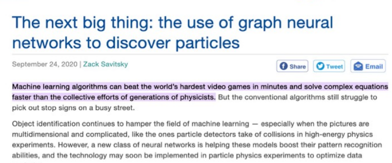




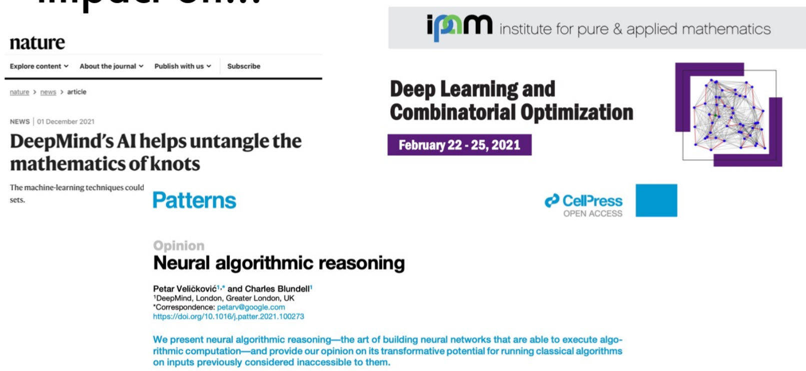
Graph Neural Networks have a large impact on…



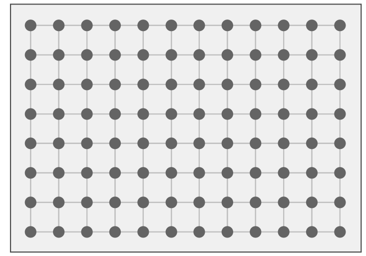


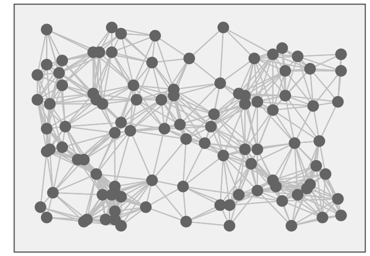


Graph Neural Networks have a large impact on…



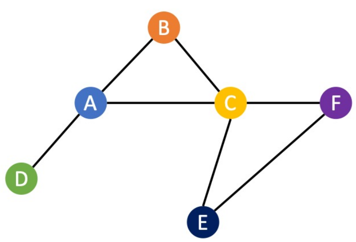
Motivation





- We know how to build neural nets over larger **grid** structured domains (CNNs)
- How can we build neural nets for general **graph** structured inputs? (GNNs)
- Goal: We want to exploit the **local structure** of the graph

What is a graph?

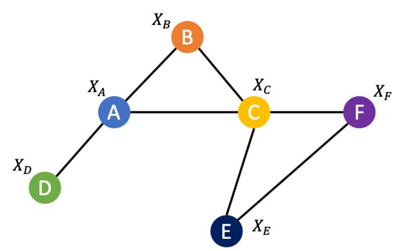


A graph is composed of
- **Nodes** (also called vertices)
- **Edges** connecting a pair of nodes

presented in an **adjacency matrix**

	A	B	C	D	E	F
A		1	1	1		
B	1		1			
C	1	1			1	1
D	1					
E			1			1
F			1		1	

What is a graph?



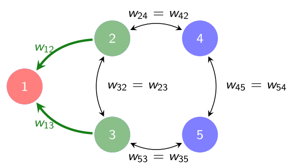
A graph is composed of
- **Nodes** (also called vertices)
- **Edges** connecting a pair of nodes

presented in an **adjacency matrix**

Nodes can have **feature vectors**

A	X_A
B	X_B
C	X_C
D	X_D
E	X_E
F	X_F

Neighborhoods and Degree



Node 1 neighborhood $\Rightarrow$ $\mathcal{N}(1) = \{2, 3\}$

Node 1 degree $\Rightarrow$ $d_1 = w_{12} + w_{13}$

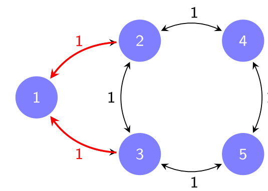
- The **neighborhood** $\mathcal{N}(i)$ of node i is the set of nodes that influence node i:

$$\mathcal{N}(i) = \{j | (i, j) \in \mathcal{E}\}$$

- The **degree** d_i of node i is the sum of weights of its incident edges:

$$d_i = \sum_{j \in \mathcal{N}(i)} w_{ij}$$

Degree Matrix



$$\mathbf{D} = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

- The **degree matrix** $\mathbf{D}$ is a diagonal matrix of degrees: $D_{ii} = d_i$
- We can write $\mathbf{D}$ in terms of the adjacency matrix: $\mathbf{D} = \text{diag}(\mathbf{A1})$

Laplacian Matrix

$$\mathbf{L} = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{pmatrix} - \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 \\ -1 & -1 & 3 & 0 & -1 \\ 0 & -1 & 0 & 2 & -1 \\ 0 & 0 & -1 & -1 & 2 \end{pmatrix}$$

- The **Laplacian** matrix $\mathbf{L}$ of a graph with adjacency matrix $\mathbf{A}$ is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$

- It can also be written explicitly in terms of graph weights
  - Off diagonal entries: $L_{ij} = -A_{ij} = -w_{ij}$
  - Diagonal entries: $L_{ii} = d_i = \sum_{j \in \mathcal{N}(i)} w_{ij}$

Normalized Adjacency and Laplacian Matrix

- The **normalized adjacency matrix** expresses weights relative to node degrees

$$\bar{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}} \quad \Rightarrow \quad \bar{A}_{ij} = \frac{w_{ij}}{\sqrt{d_i d_j}}$$

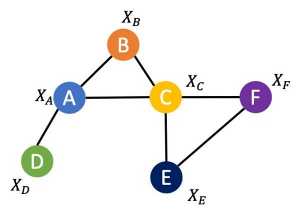
- The **normalized Laplacian matrix** is similarly defined as

$$\bar{\mathbf{L}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{L} \mathbf{D}^{-\frac{1}{2}} = \mathbf{D}^{-\frac{1}{2}} (\mathbf{D} - \mathbf{A}) \mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \bar{\mathbf{A}}$$

- Normalized operators are more **homogeneous**

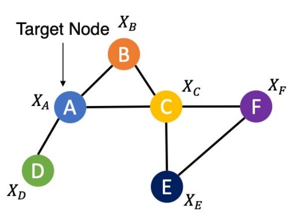
What is Graph Neural Network?

Problem definition



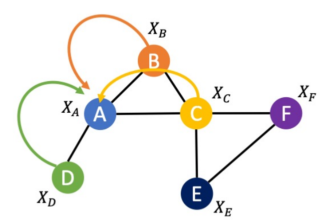
- **Given**
  - A graph
  - Node attributes
  - (part of nodes are labeled)
- **Find**
  - Node embeddings
- **Predict**
  - Labels for the remaining nodes

Graph Neural Networks



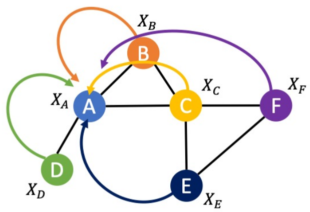
"Homophily: connected nodes are related/informative/similar"

Graph Neural Networks



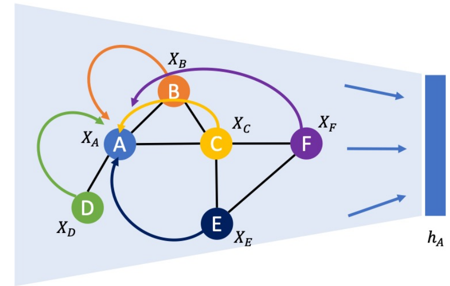
"Homophily: connected nodes are related/informative/similar"

Graph Neural Networks

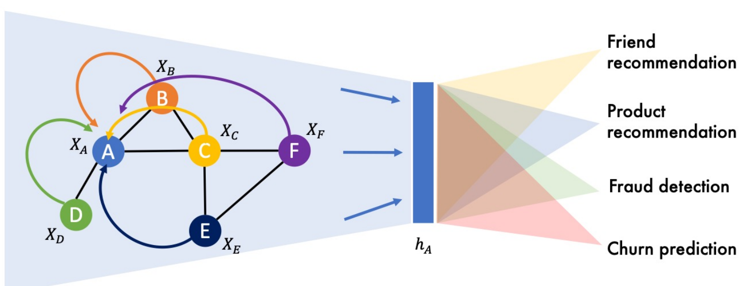


“Homophily: connected nodes are related/informative/similar”

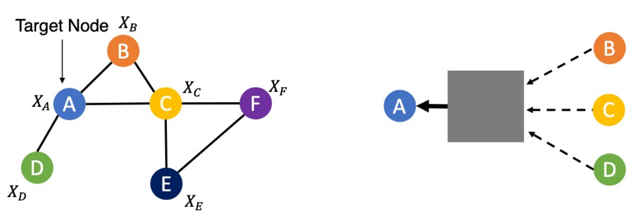
Graph Neural Networks

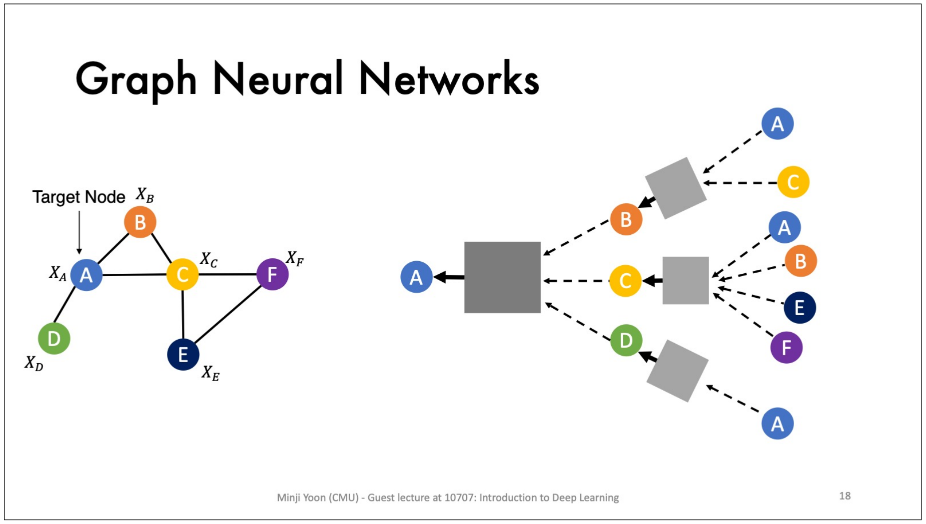


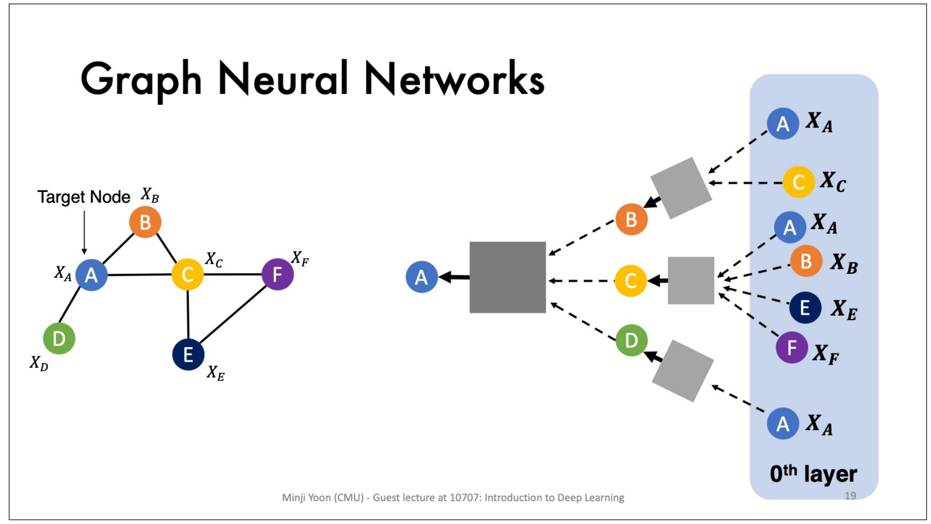
Graph Neural Networks

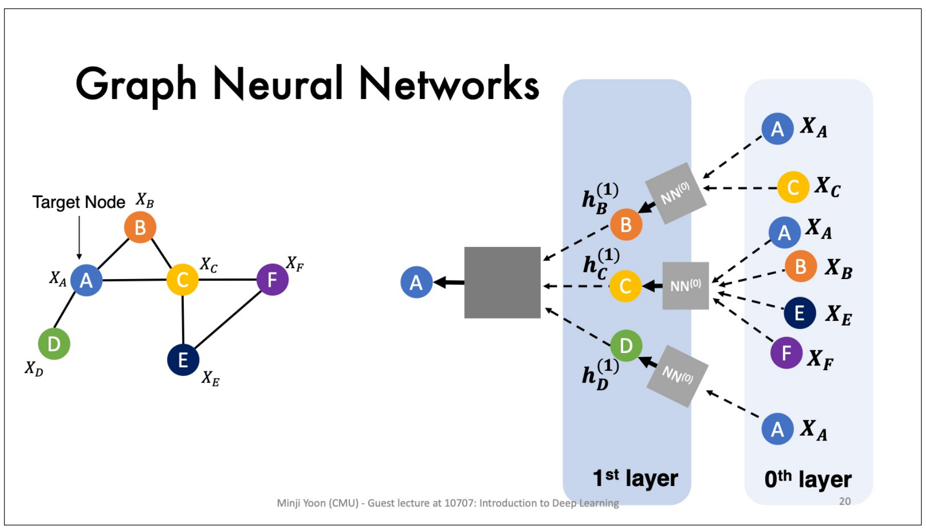


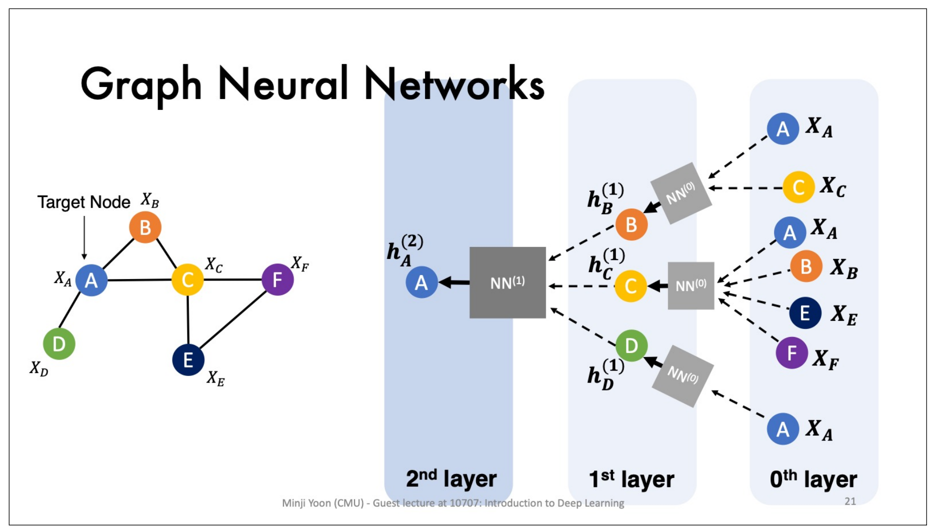
Graph Neural Networks




Graph Neural Networks
Target Node
X_A X_B X_C X_D X_E X_F
A B C D E F
Minji Yoon (CMU) - Guest lecture at 10707: Introduction to Deep Learning
18


Graph Neural Networks
Target Node
X_A X_B X_C X_D X_E X_F
A B C D E F
0th layer
Minji Yoon (CMU) - Guest lecture at 10707: Introduction to Deep Learning
19

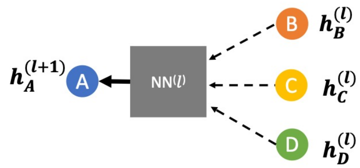

Graph Neural Networks
Target Node
X_A X_B X_C X_D X_E X_F
$h_B^{(1)}$ $h_C^{(1)}$ $h_D^{(1)}$
NN(0)
1st layer
0th layer
Minji Yoon (CMU) - Guest lecture at 10707: Introduction to Deep Learning
20


Graph Neural Networks
Target Node
X_A X_B X_C X_D X_E X_F
$h_A^{(2)}$ $h_B^{(1)}$ $h_C^{(1)}$ $h_D^{(1)}$
NN(1)
NN(0)
2nd layer
1st layer
0th layer
Minji Yoon (CMU) - Guest lecture at 10707: Introduction to Deep Learning
21

Graph Neural Networks

1. **Aggregate messages from neighbors**
   - $h_v^{(l)}$: node embedding of v at l-th layer
   - $\mathcal{N}(v)$: neighboring nodes of v
   - $f^{(l)}$: aggregation function at l-th layer
   - $m_v^{(l)}$: message vector of v at l-th layer

$$m_A^{(l)} = f^{(l)}\left(h_A^{(l)}, \left\{h_u^{(l)} : u \in \mathcal{N}(A)\right\}\right)$$
$$= f^{(l)}\left(h_A^{(l)}, h_B^{(l)} h_C^{(l)} h_D^{(l)}\right)$$



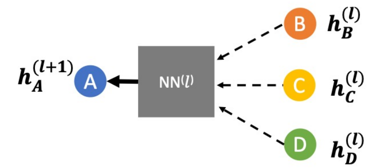
Neighbors of node A
$\mathcal{N}(A) = \{B, C, D\}$

Graph Neural Networks

1. **Aggregate messages from neighbors**

$$m_A^{(l)} = f^{(l)}\left(h_A^{(l)}, \left\{h_u^{(l)} : u \in \mathcal{N}(A)\right\}\right)$$
$$= f^{(l)}\left(h_A^{(l)}, h_B^{(l)} h_C^{(l)} h_D^{(l)}\right)$$

2. **Transform** messages
   - $g^{(l)}$: transformation function at l-th layer
   - $h_A^{(l+1)} = g^{(l)}(m_A^{(l)})$



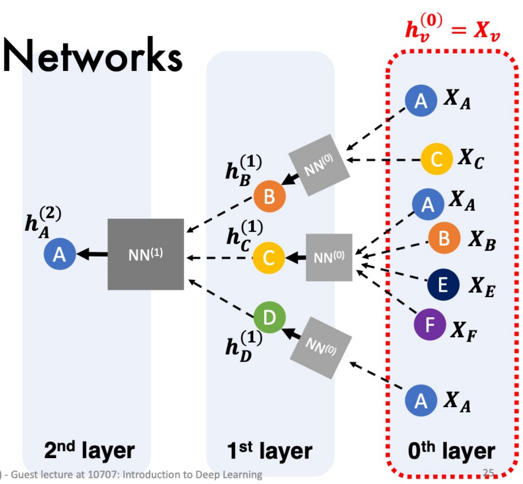
Neighbors of node A
$\mathcal{N}(A) = \{B, C, D\}$

Graph Neural Networks

In each layer l ,
for each target node v :

1. Aggregate messages
$m_v^{(l)} = f^{(l)}\left(h_v^{(l)}, \left\{h_u^{(l)} : u \in \mathcal{N}(v)\right\}\right)$

2. Transform messages
$h_v^{(l+1)} = g^{(l)}(m_v^{(l)})$

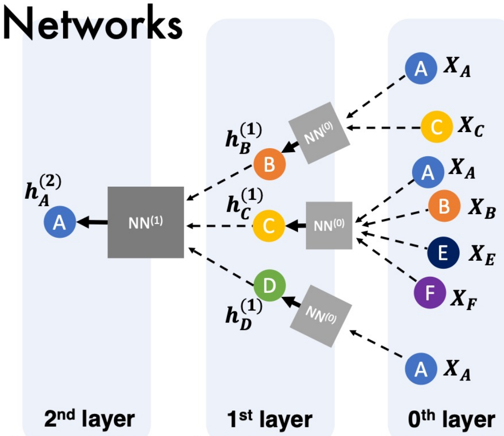


Graph Neural Networks

In each layer l ,
for each target node v :

1. Aggregate messages
$m_v^{(l)} = \boxed{f^{(l)}}\left(h_v^{(l)}, \left\{h_u^{(l)} : u \in \mathcal{N}(v)\right\}\right)$

2. Transform messages
$h_v^{(l+1)} = \boxed{g^{(l)}}(m_v^{(l)})$



Graph Neural Networks

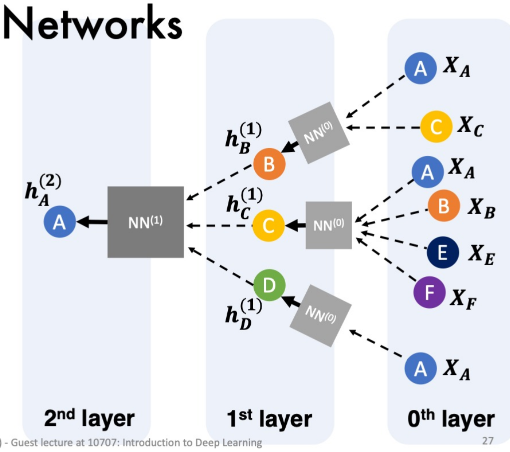
Graph Convolutional Networks[1]

1. Aggregate messages

$$m_v^{(l)} = \frac{1}{|\mathcal{N}(v)+1|} \sum_{u \in \mathcal{N}(v) \cup \{v\}} h_u^{(l)}$$

2. Transform messages

$$h_v^{(l+1)} = \sigma(\boldsymbol{W}^{(l)} \circ m_v^{(l)})$$



[1] Kipf, Thomas N., et al. "Semi-supervised classification with graph convolutional networks."

Graph Neural Networks

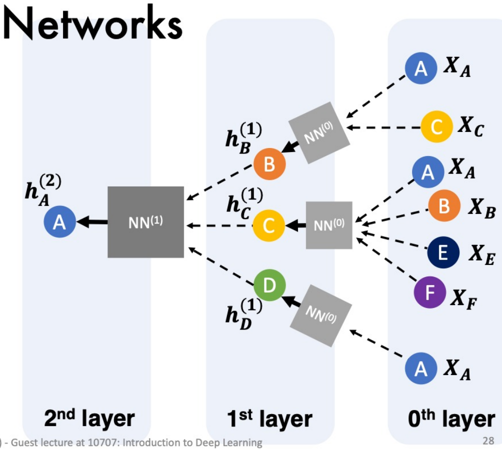
Graph Isomorphism Networks[2]

1. Aggregate messages

$$m_v^{(l)} = \sum_{u \in \mathcal{N}(v) \cup \{v\}} h_u^{(l)}$$

2. Transform messages

$$h_v^{(l+1)} = \sigma(\boldsymbol{W}^{(l)} \circ m_v^{(l)})$$



[2] Xu, Keyulu, et al. "How powerful are graph neural networks?."

Graph Neural Networks

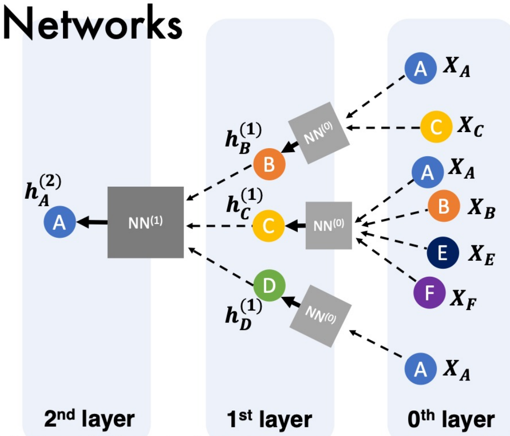
Simplified GCN[3]

1. Aggregate messages

$$m_v^{(l)} = \frac{1}{|\mathcal{N}(v)+1|} \sum_{u \in \mathcal{N}(v) \cup \{v\}} h_u^{(l)}$$

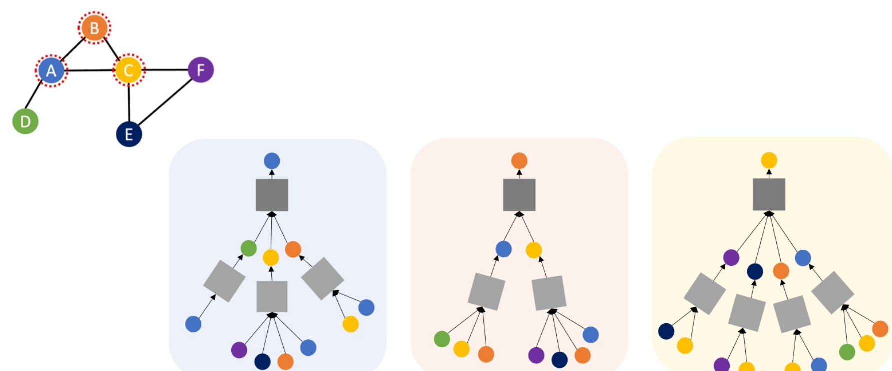
2. Transform messages

$$h_v^{(l+1)} = \boldsymbol{W}^{(l)} \circ m_v^{(l)}$$

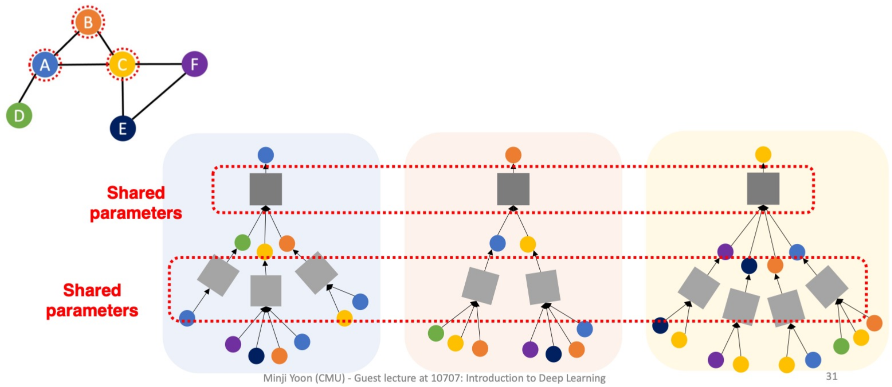


[3] Wu, Felix, et al. "Simplifying graph convolutional networks."

Computation graphs

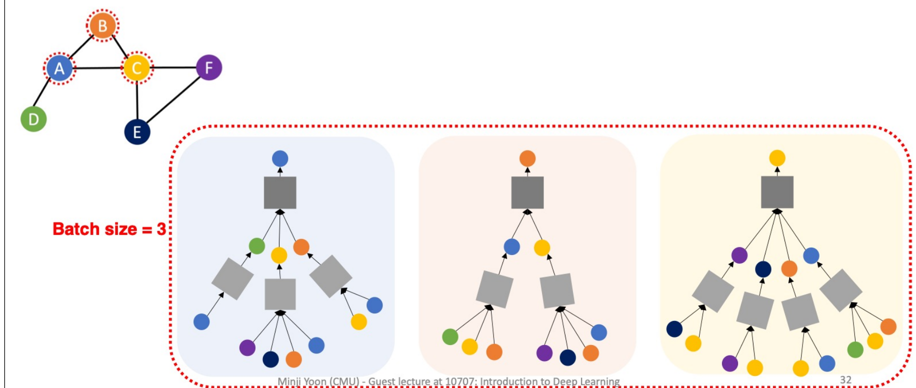


Computation graphs



Batch execution

$h_v^{(l)} = \sigma(\boldsymbol{W}^{(l)} \circ (\frac{1}{|\mathcal{N}(v)+1|}\sum_{u \in \mathcal{N}(v)\cup\{v\}} h_u^{(l-1)}))$



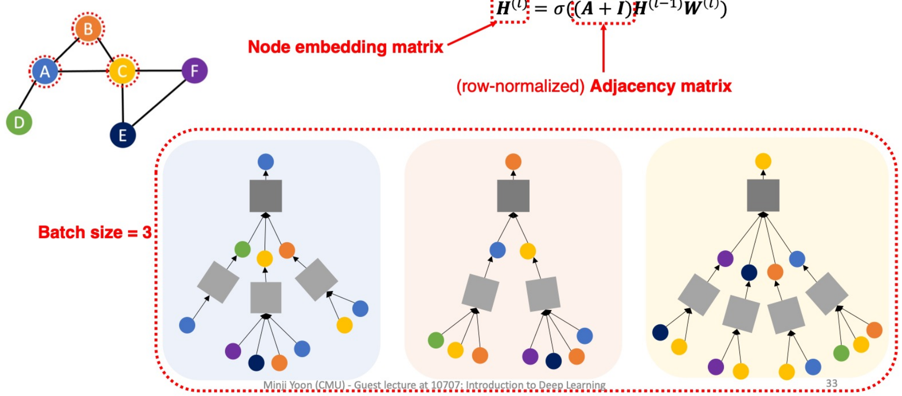
Batch execution

$h_v^{(l)} = \sigma(\boldsymbol{W}^{(l)} \circ (\frac{1}{|\mathcal{N}(v)+1|}\sum_{u \in \mathcal{N}(v)\cup\{v\}} h_u^{(l-1)}))$

$\boxed{\boldsymbol{H}^{(l)}} = \sigma(\boxed{\widetilde{(\boldsymbol{A}+\boldsymbol{I})}}\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)})$

Node embedding matrix

(row-normalized) Adjacency matrix

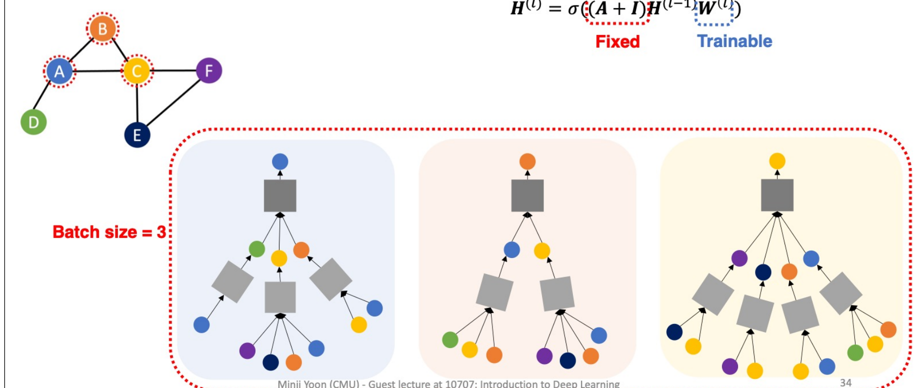


Batch execution

$h_v^{(l)} = \sigma(\boldsymbol{W}^{(l)} \circ (\frac{1}{|\mathcal{N}(v)+1|}\sum_{u \in \mathcal{N}(v)\cup\{v\}} h_u^{(l-1)}))$

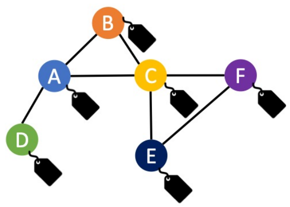
$\boldsymbol{H}^{(l)} = \sigma(\boxed{\widetilde{(\boldsymbol{A}+\boldsymbol{I})}}\boldsymbol{H}^{(l-1)}\boxed{\boldsymbol{W}^{(l)}})$

Fixed Trainable



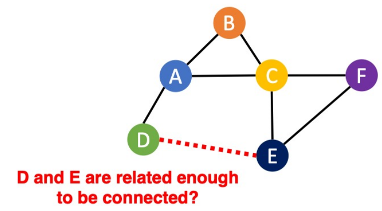
Downstream tasks

• Node-level prediction



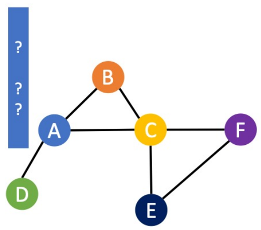
Downstream tasks

• Node-level prediction
• Edge-level prediction



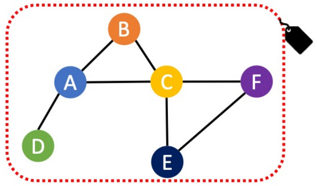
Downstream tasks

• Node-level prediction
• Edge-level prediction
• Attribute-level prediction



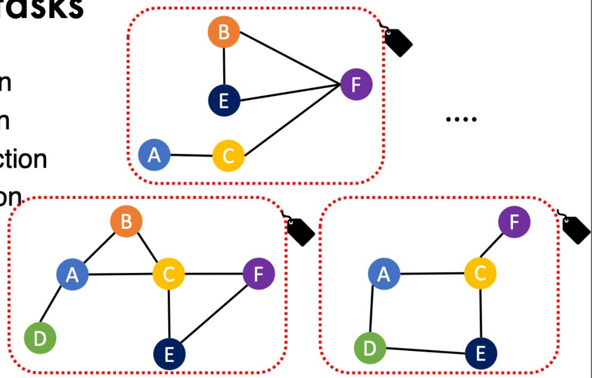
Downstream tasks

• Node-level prediction
• Edge-level prediction
• Attribute-level prediction
• Graph-level prediction

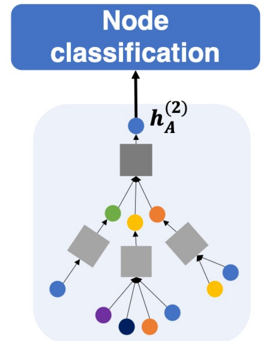


Downstream tasks

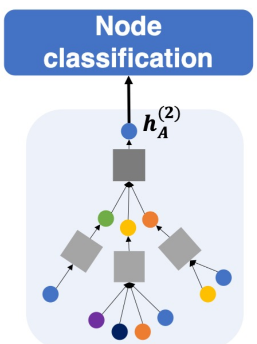
- Node-level prediction
- Edge-level prediction
- Attribute-level prediction
- Graph-level prediction

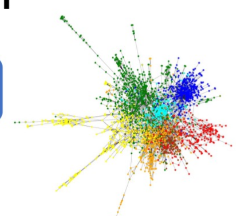


Node-level prediction tasks



Node-level prediction tasks



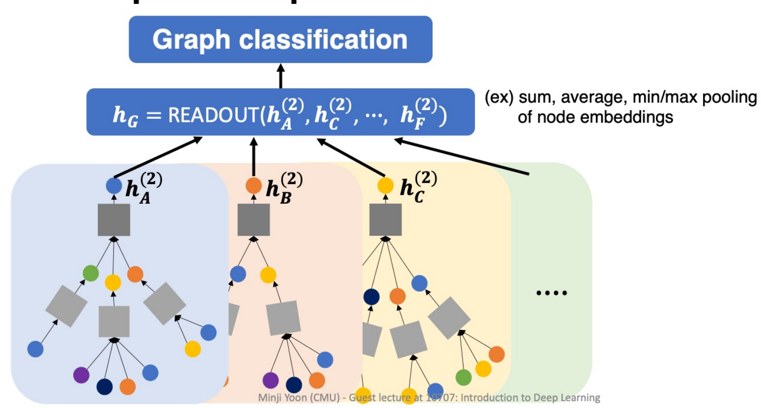


reddit

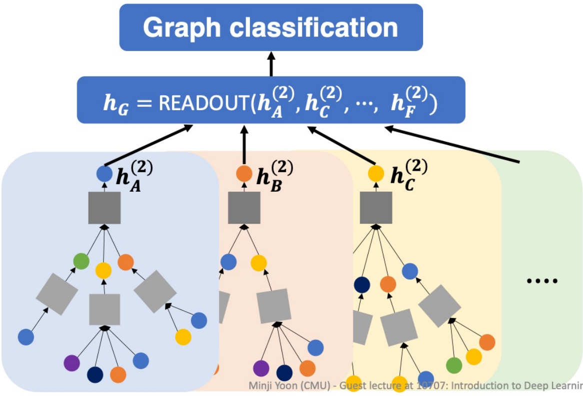
amazon

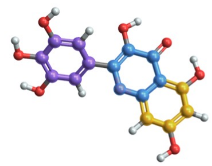
- Classify **papers** into topics on **citation networks**
- Cluster **posts** into subgroups on **Reddit networks**
- Classify **products** into categories on **Amazon co-purchase graphs**

Graph-level prediction tasks



Graph-level prediction tasks



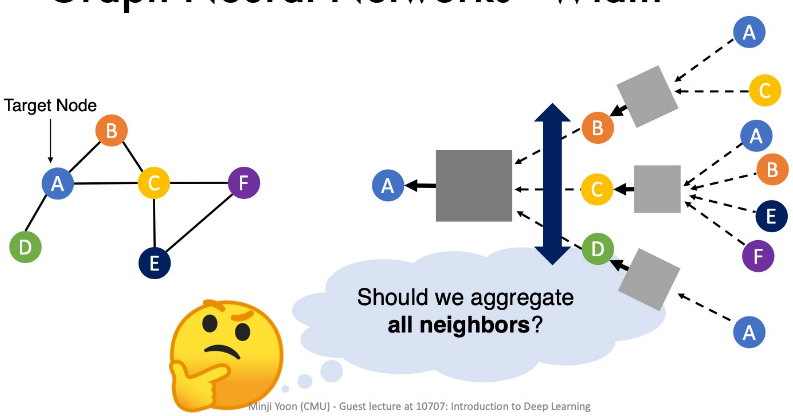


- Predict **properties of a molecule (graph)** where nodes are atoms and edges are chemical bonds

So far, we have talked about..

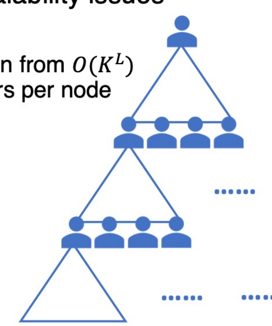
1. **Graph Neural Network**
   - Problem definition
   - Skeleton
     - Aggregation operation
     - Transformation operation
2. **Implementation**
   - Computation graph
   - Batch execution
3. **Downstream tasks**
   - Node-level prediction
   - Graph-level prediction

Graph Neural Networks - Width



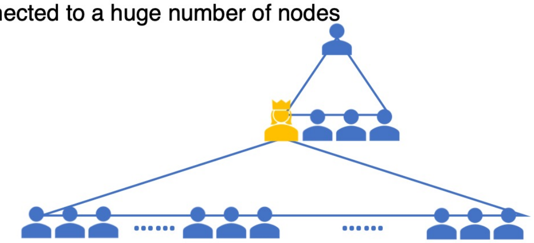
Aggregation Width in GNNs

- If we aggregate all neighbors, GNNs have scalability issues
- Neighbor explosion
  - In L -layer GNNs, one node aggregates information from $O(K^L)$ nodes where K is the average number of neighbors per node



Aggregation Width in GNNs

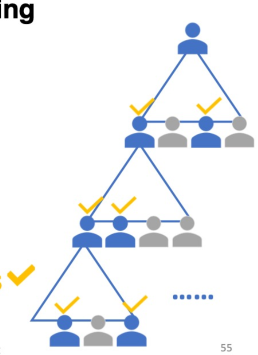
- If we aggregate all neighbors, GNNs have scalability issues
- Neighbor explosion
  - Hub nodes who are connected to a huge number of nodes



Aggregation Width in GNNs

- Limit the neighborhood expansion by **sampling** a fixed number of neighbors

Sample the neighbors



Aggregation Width in GNNs

- Random sampling
  - Assign **same** sampling probabilities to all neighbors
  - *GraphSage*[4]
- Importance sampling
  - Assign **different** sampling probabilities to all neighbors
  - *FastGCN*[5], *LADIES*[6], *AS-GCN*[7], *GCN-BS*[8], *PASS*[9]

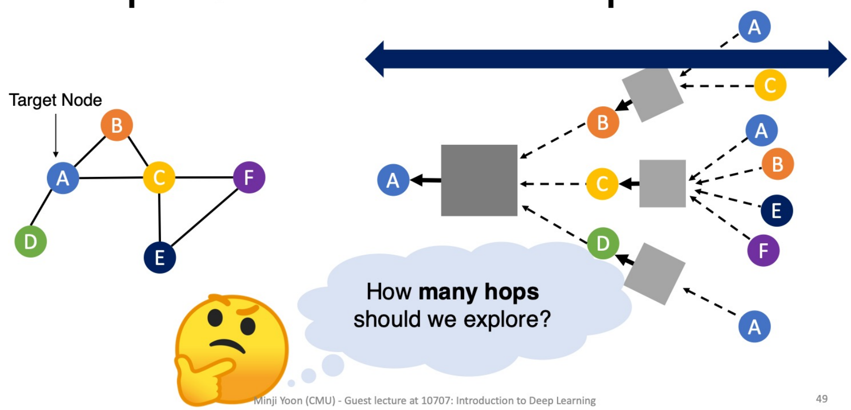
[4] Will Hamilton, et al. "Inductive representation learning on large graphs"
[5] Jie Chen, et al. "Fastgcn: fast learning with graph convolutional networks via importance sampling"
[6] Difan Zou, et al. "Layer-Dependent Importance Sampling for Training Deep and Large Graph Convolutional Networks"
[7] Wenbing Huang, et al. "Adaptive sampling towards fast graph representation learning"
[8] Ziqi Liu, et al. "Bandit Samplers for Training Graph Neural Networks"
[9] Minji Yoon, et al. "Performance-Adaptive Sampling Strategy Towards Fast and Accurate Graph Neural Networks"

Aggregation Width in GNNs

Method	Cora	Citeseer	Pubmed	AmazonC	AmazonP	MsCS	MsPhysics
FastGCN	0.582	0.496	0.569	0.480	0.542	0.520	0.638
AS-GCN	0.462	0.387	0.502	0.419	0.480	0.403	0.516
GraphSage	0.788	0.698	0.792	0.707	0.787	0.766	0.875
GCN-BS	0.788	0.693	0.809	0.736	0.800	0.780	0.887
PASS	**0.821**	**0.715**	**0.858**	**0.757**	**0.855**	**0.884**	**0.934**

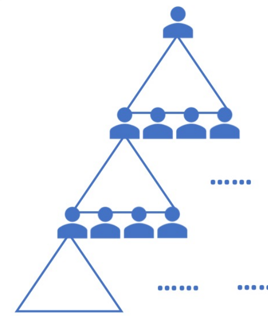
- **Node classification task on 7 different real-world graphs**
- **PASS outperforms all variance-minimizing methods by up to 10.4%**

Graph Neural Networks - Depth



Aggregation Depth in GNNs

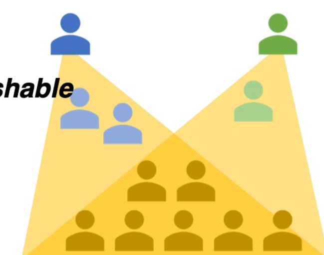
- When we increase the depth L more than this, GNNs face neighbor explosion $O(K^L)$
  - **Over-smoothing**
  - **Over-squashing**



Aggregation Depth in GNNs

Over-smoothing[10]

- When GNNs become deep, nodes share many neighbors
- Node embeddings become ***indistinguishable***

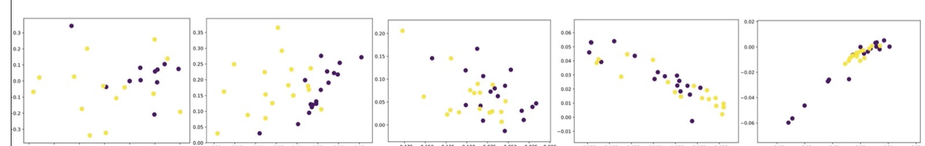


[10] Qimai Li, et al. "Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning"

Aggregation Depth in GNNs

Over-smoothing[10]

- Node embeddings of Zachary's karate club network with GNNs



(a) 1-layer (b) 2-layer (c) 3-layer (d) 4-layer (e) 5-layer

[10] Qimai Li, et al. "Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning"

Aggregation Depth in GNNs

Mitigate over-smoothing

PairNorm[11]

- Keep total pairwise squared distance (TPSD) ***constant*** across layers
- Push away pairs that are not connected

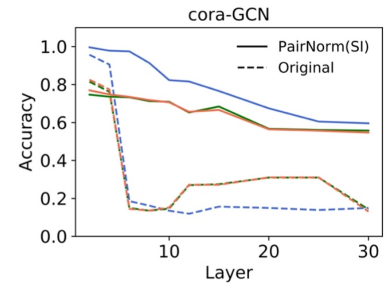
$$\mathrm{TPSD}(\dot{X}) = \overbrace{\boxed{\sum_{(i,j)\in \mathcal{E}} ||\dot{x}_i - \dot{x}_j||_2^2}}^{\text{Connected pairs}} + \overbrace{\boxed{\sum_{(i,j)\notin \mathcal{E}} ||\dot{x}_i - \dot{x}_j||_2^2}}^{\text{Disconnected pairs}} = \boldsymbol{C}$$

[11] Lingxiao Zhao, et al. "PAIRNORM: TACKLING OVERSMOOTHING IN GNNS"

Aggregation Depth in GNNs

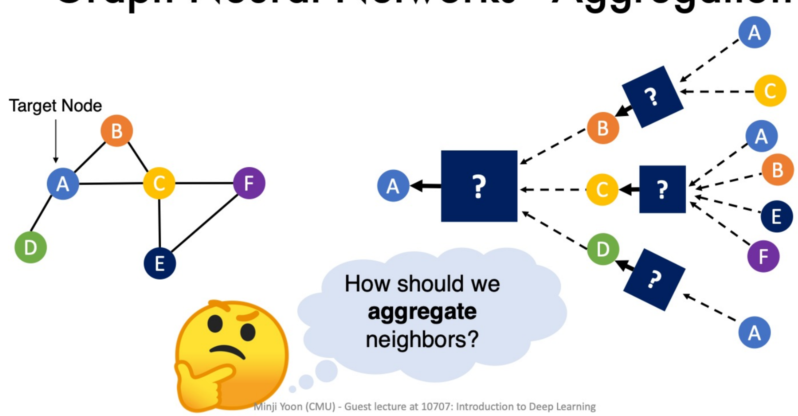
Mitigate over-smoothing

PairNorm[11]



[11] Lingxiao Zhao, et al. "PAIRNORM: TACKLING OVERSMOOTHING IN GNNS"

Graph Neural Networks - Aggregation



Aggregation strategy in GNNs

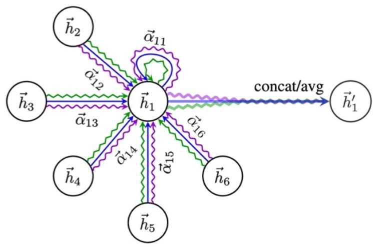
- GCN[1]
  - Average embeddings of neighboring nodes

[1] Kipf, Thomas N., et al. "Semi-supervised classification with graph convolutional networks."

Aggregation strategy in GNNs

- GAT[14]
  - Different weights to different nodes in a neighborhood
  - Multi-head attention

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$



[14] Petar Veličković., et al. "GRAPH ATTENTION NETWORKS."

Aggregation strategy in GNNs

In each layer l :

Aggregate over neighbors

$$m_v^{(l-1)} = \boxed{f^{(l)}}\left(h_v^{(l-1)}, \left\{h_u^{(l-1)} : u \in \mathcal{N}(v)\right\}\right)$$

Core part of GNNs

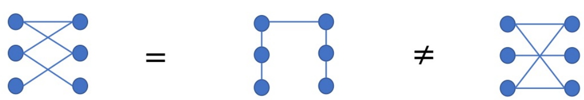
Transform messages

$$h_v^{(l)} = \boxed{g^{(l)}}(m_v^{(l-1)})$$

Any neural network module can fit in
1-layer MLP is commonly used

Aggregation strategy in GNNs

- How powerful are Graph Neural Networks?[2]
- Metric
  - Graph-level prediction task
  - Can a GNN model distinguish two non-isomorphic graphs?



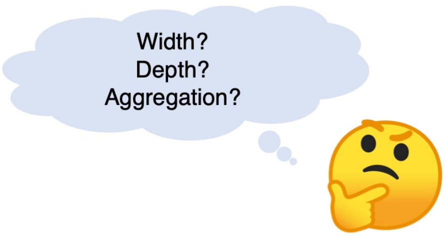
[2] Keyulu Xu., et al. "HOW POWERFUL ARE GRAPH NEURAL NETWORKS?"

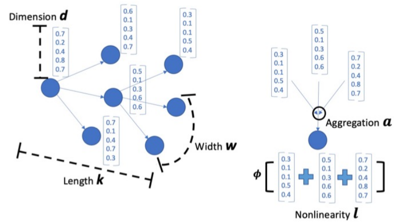
Aggregation strategy in GNNs

- Can we make more powerful GNNs?
  - Very active area, with many open problems

Neural Architecture Search for GNNs

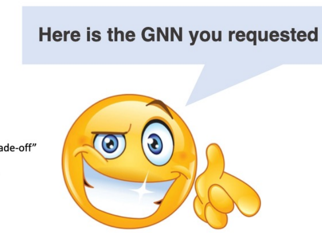
- Which *width, depth, and aggregation strategy* are proper for a given graph and task?





Neural Architecture Search for GNNs

- Finding proper *width, depth, and aggregation strategy* for a given graph and task **automatically**[1,2,3]



[23] Minji Yoon., et al. "Autonomous Graph Mining Algorithm Search with Best Speed/Accuracy Trade-off"
[24] Kaixiong Zhou, et al. "Auto-GNN: Neural Architecture Search of Graph Neural Networks"
[25] Yang Gao, et al. "GraphNAS: Graph Neural Architecture Search with Reinforcement Learning"