Deep Learning Theory and Applications

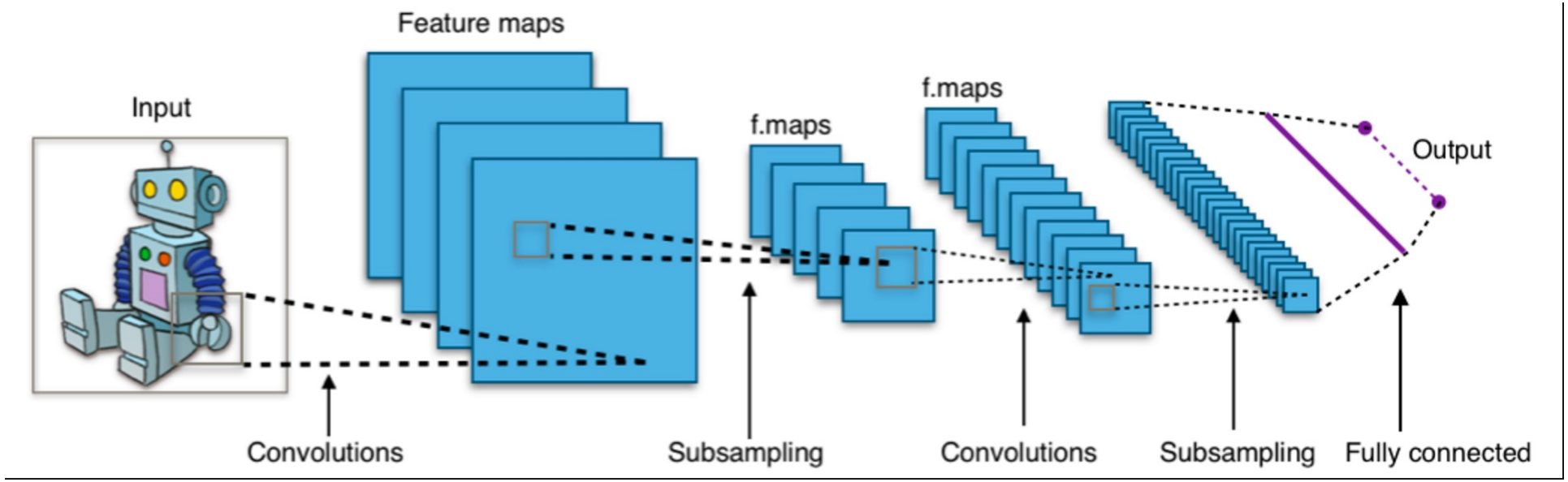
Graph Neural Networks

Yale

CPSC/AMTH 663

CPSC 452

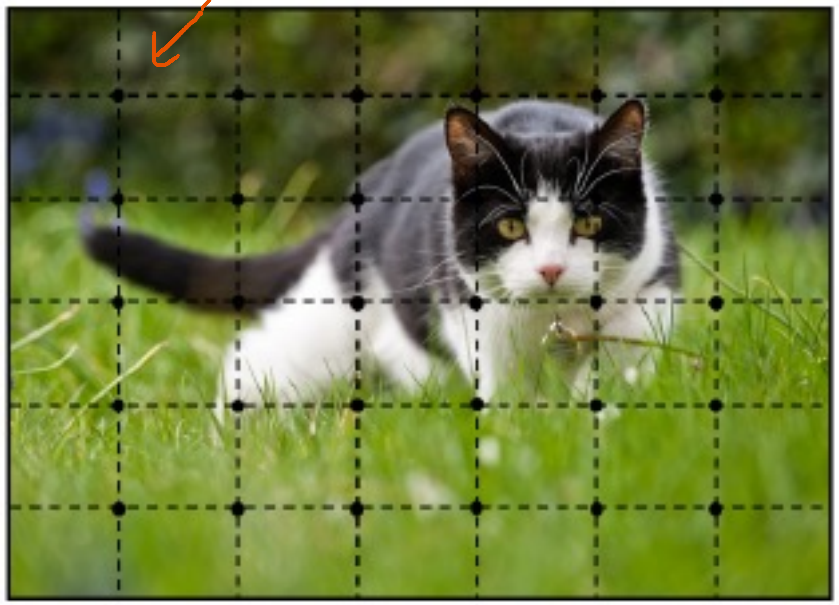
Recall CNNs

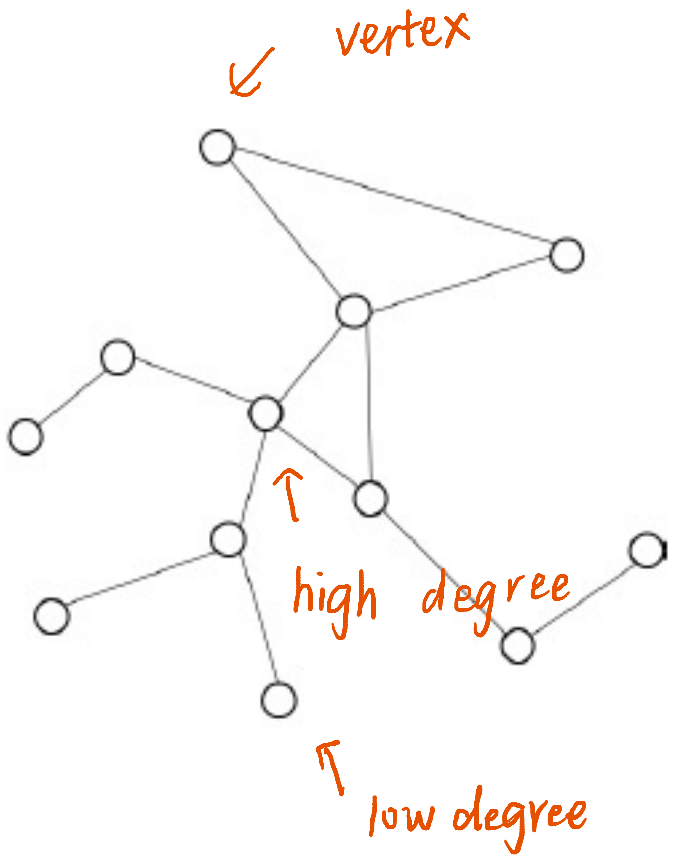


Design to account for image structure

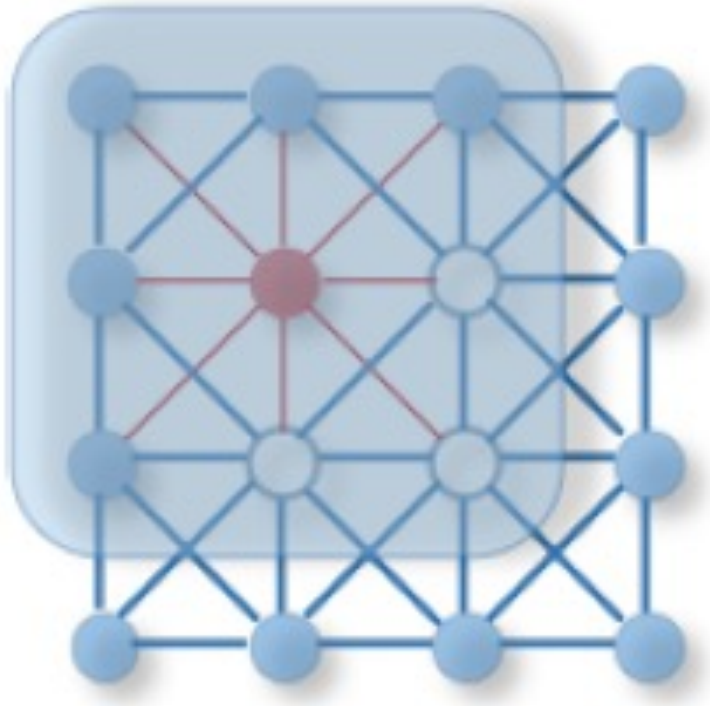
Images are a specific sort of graph—grid graph

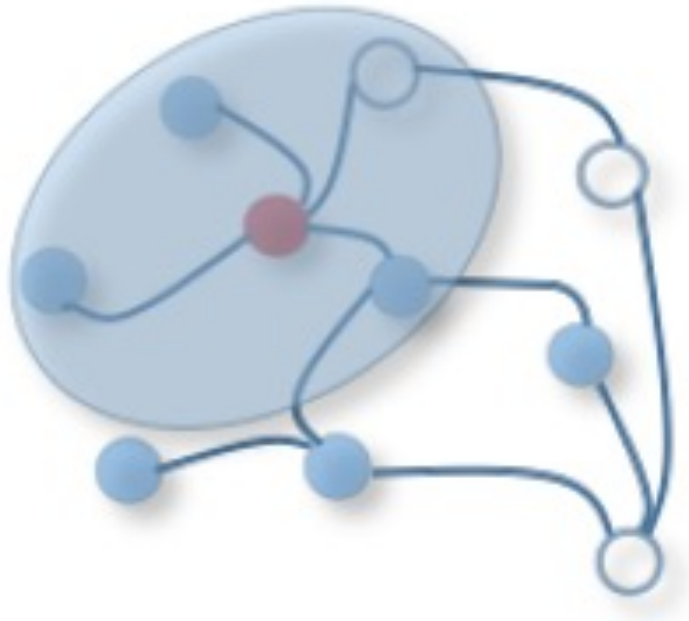
every pixel is a vertex





More general graph structures





Recall: CNNs use three basic ideas

all can be translated to graph

1. Local receptive fields
    1. Problem: There is no clearly defined direction or method of translation over a graph (cannot cleanly slide a convolution)
2. Shared weights

center neighbors on a vertex

3. Pooling

in a filter

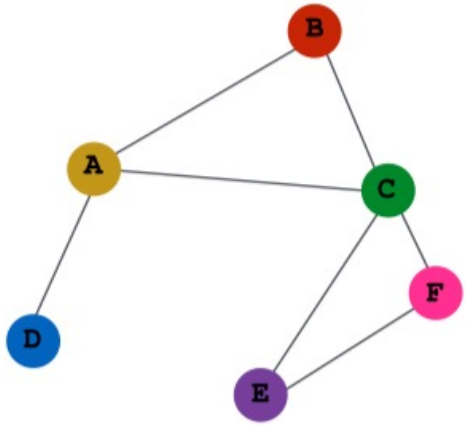
How can we approximate these on more general non-grid graphs

- Spatial construction (Bruna, Hamilton)
  - Apply the same operator and move over the graph space in some fashion
  - Notion of locality based on neighbors of a node (vertex domain)

- Spectral construction (Bruna, Defferard)
  - Define a convolutional operation in the graph spectral space (spectral domain)
  - Uses signal processing theory to define filters in the graph Fourier domain
  - Notion of locality based on global properties of the filter

Spatial Construction

Graph Notation



think of each vertex as a sample, has several features

$x_B = \{age, Job \cdots\}$

edge can also has feature
eg. edge type is_a for know graph

adjacency matrix

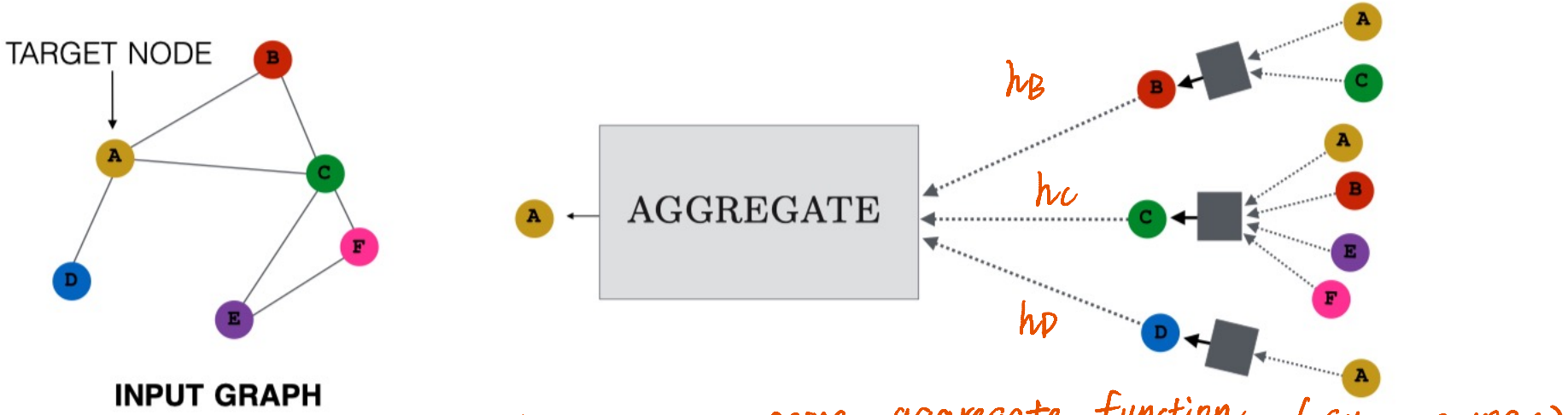
	A	B	C	D	E
A	0	1	1	1	0
B					
C					
D					
E					

- Graphs $G = (V, E)$
- $u \in V$ vertices
- $(u, v) \in E$ edges
- A is an adjacency matrix
  - $\mathrm{A}(u, v) = 1$ if $(u, v) \in E$
  - 0 otherwise
  - If A is weighted then $\mathrm{A}(u, v)$ can be a real value
- Vertex features $\mathbf{x}_u$

store connectivity info or similarity

Message passing

get message from neighbors to node



some aggregate function (sum or max)

k: iteration step

updated feature

$$\mathbf{h}_u^{(k+1)} = \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \text{AGGREGATE}^{(k)}(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\})\right)$$

$$= \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right),$$

v: neighbor nodes of node u

h_u: certain feature h of node u

A's neighbors B.C.D aggarated feature h of

message

Message passing iterations

- There can be several iterations of message passing through a GNN
- At each iteration, the message from the previous iteration is aggregated through neighboring vertices
- After k iterations you get a latent embedding of node u

$$\mathbf{z}_u = \mathbf{h}_u^{(K)}, \forall u \in \mathcal{V}.$$

latent embedding could then be used for tasks
- reconstruct neighbor nodes
- classify nodes

Aggregation, Updating

Vanilla graph NN

$$\mathbf{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \mathbf{h}_v,$$ typical aggregation is summing over features of neighbor nodes

can have LSTM aggregation

$$\text{UPDATE}(\mathbf{h}_u, \mathbf{m}_{\mathcal{N}(u)}) = \sigma\left(\mathbf{W}_{\text{self}}\mathbf{h}_u + \mathbf{W}_{\text{neigh}}\mathbf{m}_{\mathcal{N}(u)}\right)$$

↑ Sigmoid

W are trainable parameter matrices

$$\mathbf{h}_u^{(k)} = \sigma\left(\mathbf{W}_{\text{self}}^{(k)}\mathbf{h}_u^{(k-1)} + \mathbf{W}_{\text{neigh}}^{(k)} \sum_{v \in \mathcal{N}(u)} \mathbf{h}_v^{(k-1)} + \mathbf{b}^{(k)}\right)$$

↑ bias

Normalized Aggregation

- Normalize by number of vertices in each neighborhood $|\mathcal{N}(u)|$

$$\mathbf{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \frac{\mathbf{h}_v}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}}$$

reason:
avoid vertices have dense neighbors pass more message, let one of the vertices dominate the graph, especially scale-free graph in social network

- Normalized message passing in the GCN

$$\mathbf{h}_u^{(k)} = \sigma\left(\mathbf{W}^{(k)} \sum_{v \in \mathcal{N}(u) \cup \{u\}} \frac{\mathbf{h}_v}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}}\right)$$

graph conv network used this normalization

a famous celebrity has lots of followers
If you follow him, he will dominate your graph

Graph Attention

attention is a dynamic mechanism to reweight input
not prefix as trained weight matrix
we can choose to pay different attentions to different neighbors

there are also mask attention

attention score

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

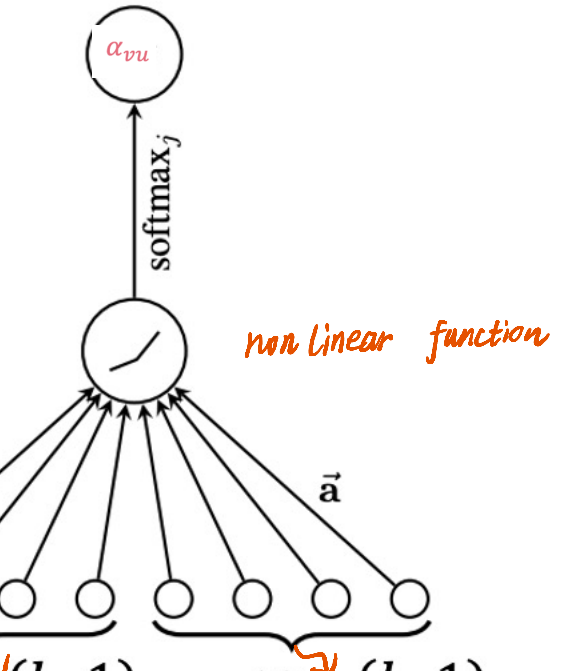
softmax

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

weight of ajacency matrix determines which nodes are v's neighbors

pretrained weight

there are 3 weights
- attention α
- weight matrix W
- ajacency matrix A



non linear function

attention func

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

features of node u

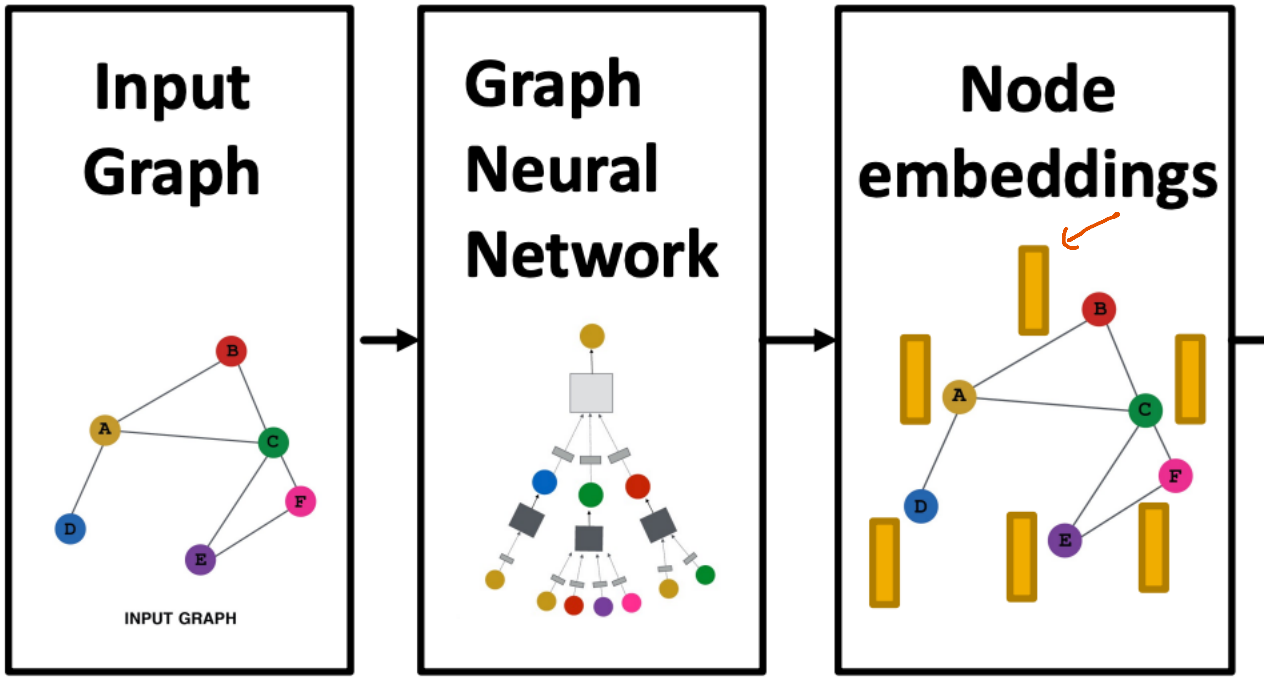
features of node v

attention score for node v to node u

softmax makes $\sum_{u \in N(v)} \alpha_{vu} = 1$

Training GNNs to do tasks

message passing



$$\{\mathbf{h}_v^{(L)}, \forall v \in G\}$$

L: number of iterations

3 types of task

1. Node level tasks
   - ① citation graph → classify article topic
   - ② predict features of node eg. sb loves some genere of movie
2. Edge level tasks
   - missing link prediction (predict edge)
   - Facebook: suggest who should be your friend
3. Graph level tasks
   - classify graph
   - benchmark dataset ENZYMES

Node level tasks

- Node classification/regression

computed features at node v
↓ after L itera

$$\hat{\boldsymbol{y}}_v = \text{Head}_{\text{node}}(\mathbf{h}_v^{(L)}) = \mathbf{W}^{(H)}\mathbf{h}_v^{(L)}$$

final layer = linear layer = regression

- Map node embeddings directly via node head to prediction

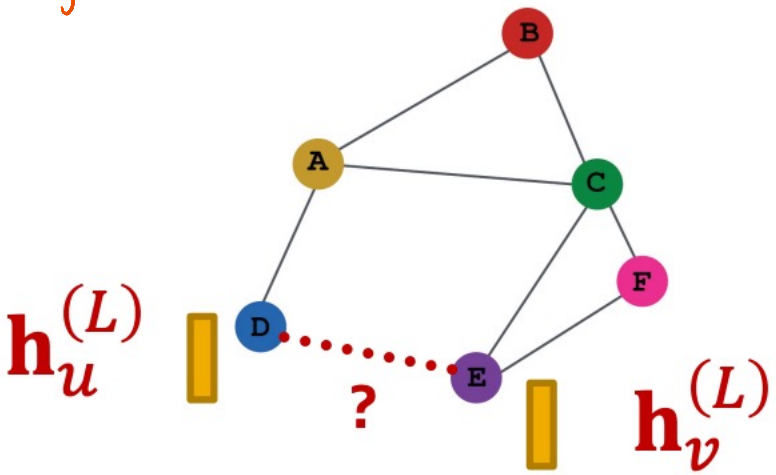
- Loss= $|| \hat{\boldsymbol{y}}_v - \boldsymbol{y}_v ||_2$ MSE

↑ label

Edge level tasks

- Examples: missing link prediction
- Edge weight/strength prediction

Edge classification : what type of edge it is ? eg. Facebook edge type is family, colleague, classmat



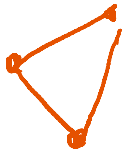
$$\hat{\boldsymbol{y}}_{uv} = \text{Head}_{\text{edge}}(\mathbf{h}_u^{(L)}, \mathbf{h}_v^{(L)})$$

↑

a func, may be a FC layer

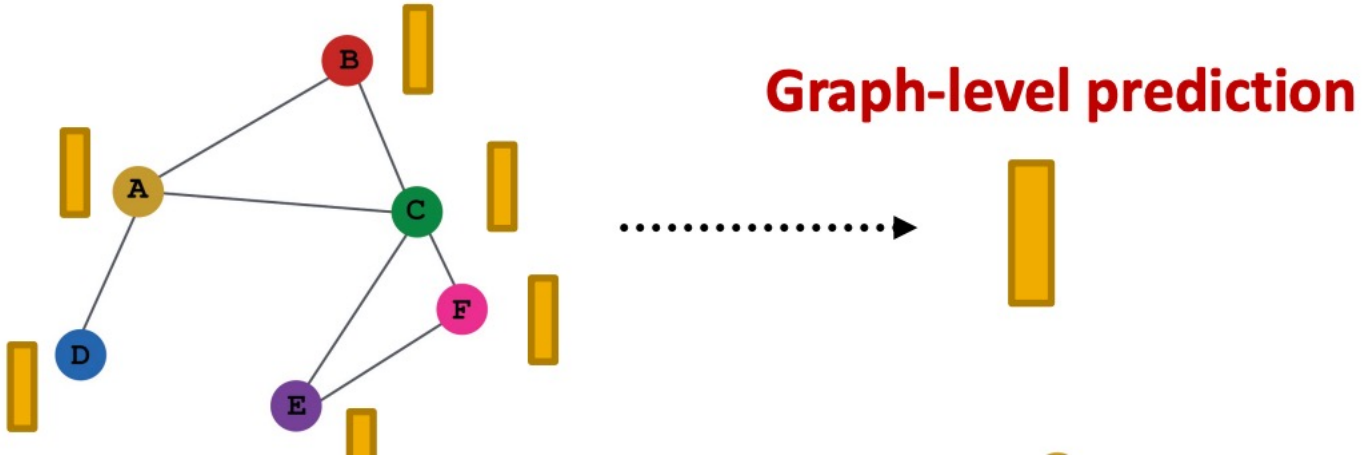
Graph Level Tasks

clique 团 concept in graph theory



a subset of vertices of an undirected graph such that every 2 distinct vertices in the clique are adjacent

- Graph classification (does this graph have a 5-clique?)
- Graph regression (does the molecule this graph encodes bind to a virus?)



Graph level pooling

- Mean pooling

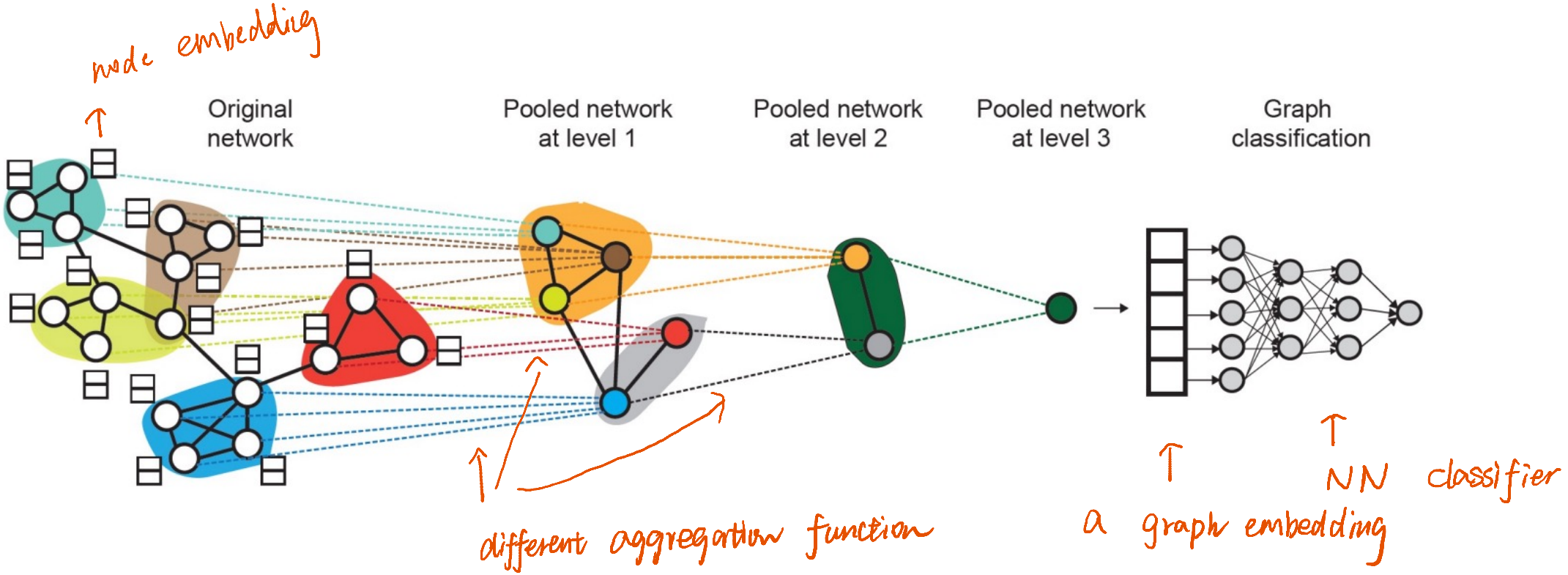
take mean of features h at all the vertices over the graph G

$$\hat{\mathbf{y}}_G = \mathbf{Mean}(\{\mathbf{h}_v^{(L)} \in \mathbb{R}^d, \forall v \in G\})$$

- Can also have max pooling, sum pooling
- But pooling can result in a loss of information

Eg of pooling function

DiffPool



Prop 1: Permutation equivariance

排列等变性

property

- Graph and node representations are the same regardless of how nodes are ordered in the adjacency matrix

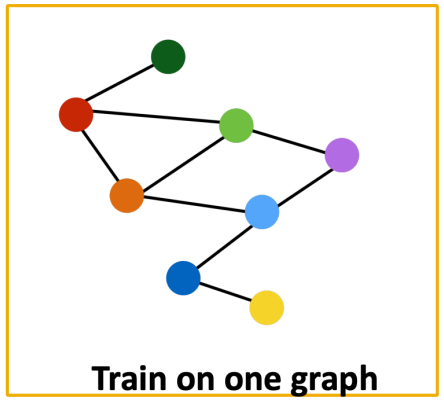
  computed message isn't depend on index of nodes in a graph

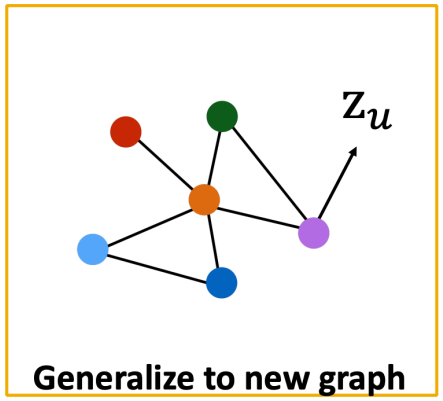
- This is achieved via permutation equivariant aggregation operations
  - Mean
  - Sum
  - Max

How to make message passing permutation inequivariance ? 排列会变性

Identity function on node eg. position of node in a graph

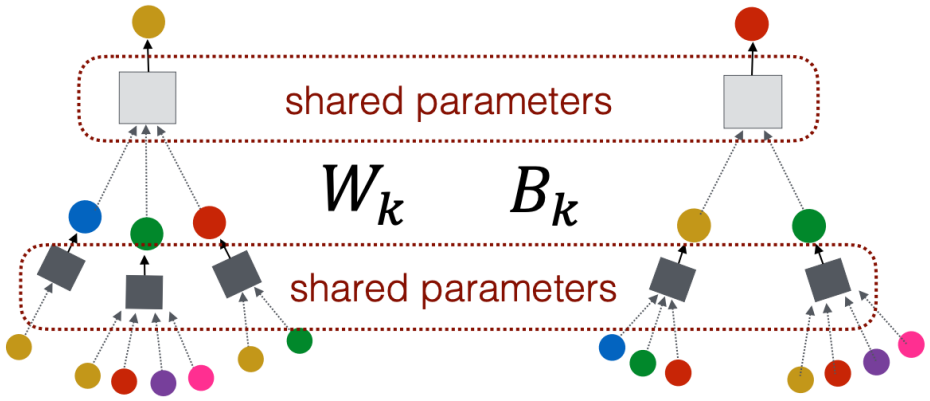
Prop 2: Inductive capability





If add a new node to a graph
can train new graph easily as old graph
reason: operations repeat at all nodes

- This is achieved by sharing weights across all nodes

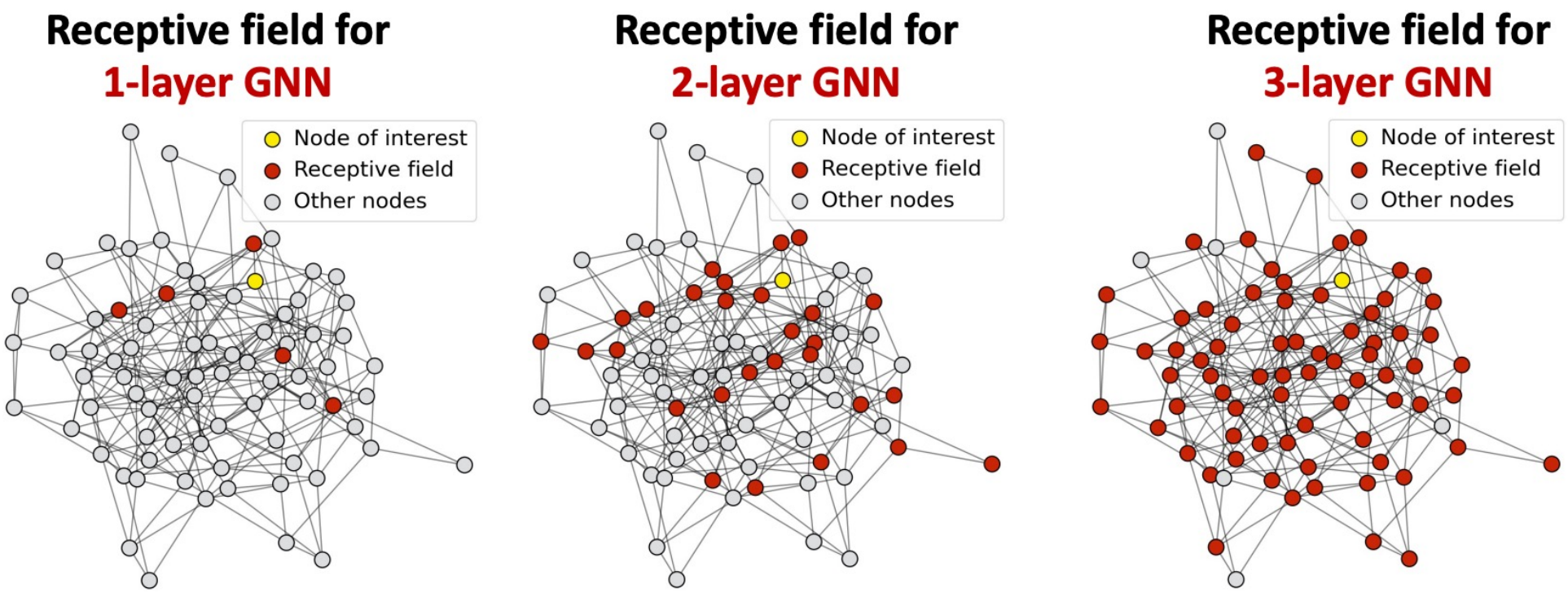


Limitations of GNNs

smooth: nodes on the graph has same message
make nodes indistinguishable from each other

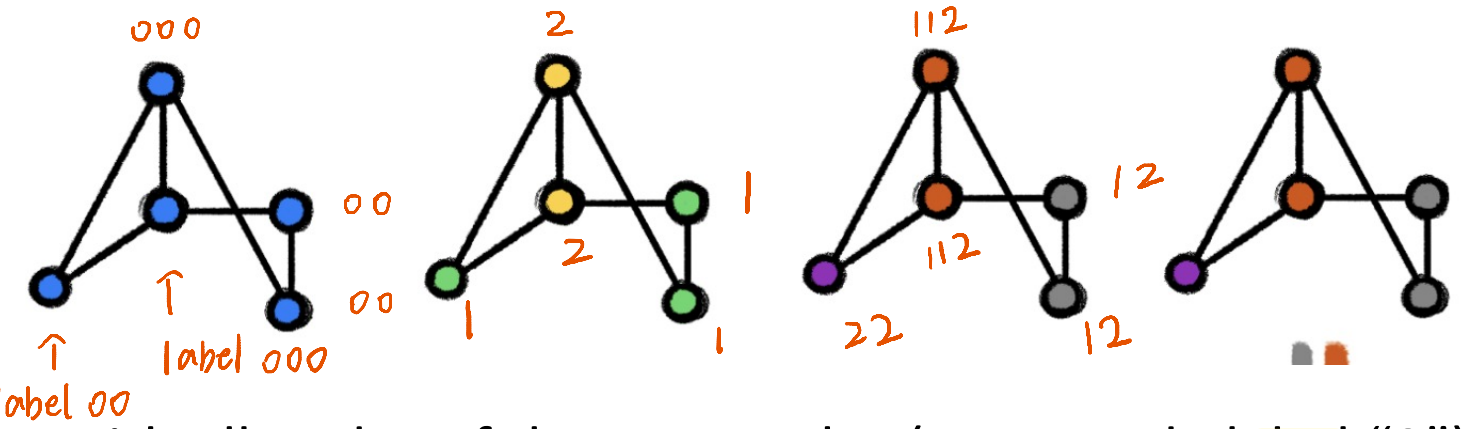
- Message passing can lead to oversmoothing!

caused by deeper depth



Message passing GNN using vanilla aggregation funcs

1950s Weisfeler-Lehman Test of Isomorphism 同构性



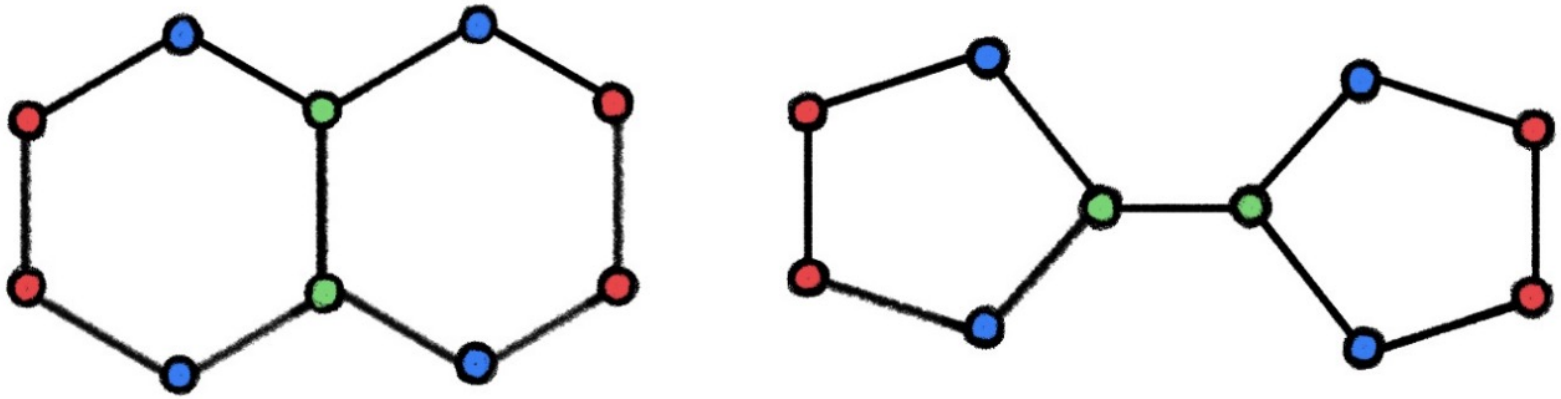
- Start with all nodes of the same color (same node label "0")
- Aggregates color of a node with its neighbors ("00" or "000")
- Hashes aggregated colors into new label ("00"=1 "000" = 2)
- Aggregates new labels ("12" or "21")
- Repeats hashing
- Stops when coloring is stable
- If at that point two graphs are differently colored they are non-isomorphic

iteratively label the node based on connectivity

非同构

Limitations of W-L test

W-L Test Test results in false positives



- These graphs are colored the same but are not isomorphic!

Power: GNN ≤ WL

GNNs are AT MOST as powerful as WL

- In a GNN that uses message passing, each node aggregates neighbor features
- Thus it is very similar to its neighbors
- But choice of aggregation function important in giving as much power as WL algorithm
- Most popular aggregators (mean, sum) are less powerful than WL!
- Xu et al. proposed an aggregation function that makes it equivalent to WL

$$h_v^{(k)} = \mathrm{MLP}^{(k)}\left(\left(1+\epsilon^{(k)}\right)\cdot h_v^{(k-1)} + \sum_{u\in\mathcal{N}(v)} h_u^{(k-1)}\right).$$

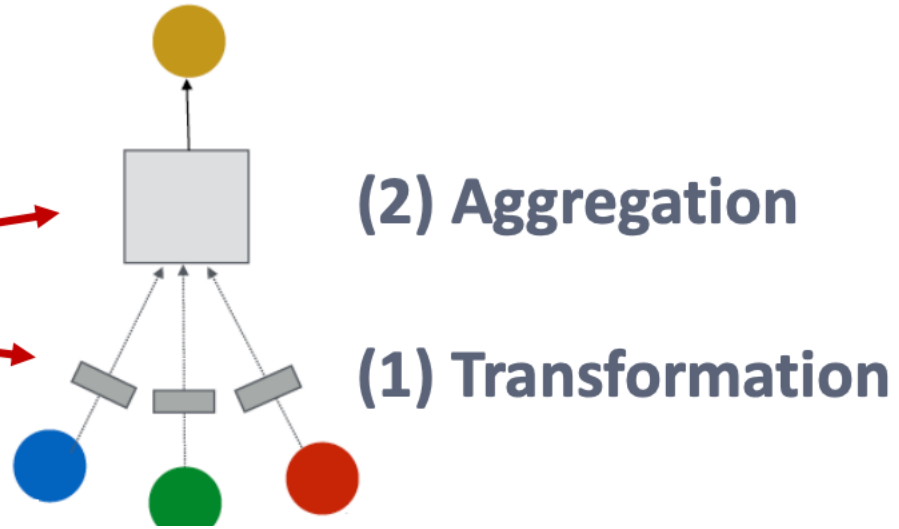
↑ use MLP as aggregation func

Solution 1

- Don't just add layers to GNNs (message passing iterations)
- Increase power within an aggregation

to avoid oversmooth labels

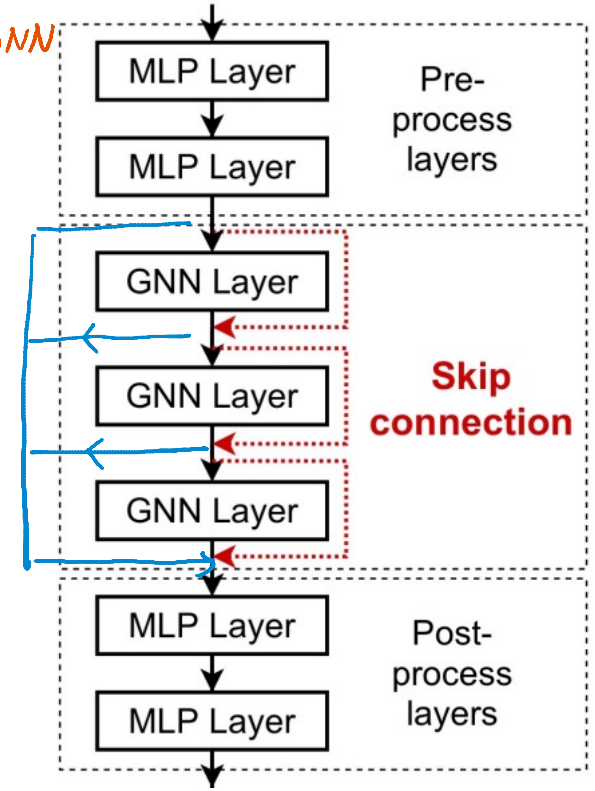
If needed, each box could include a 3-layer MLP



Solution 2

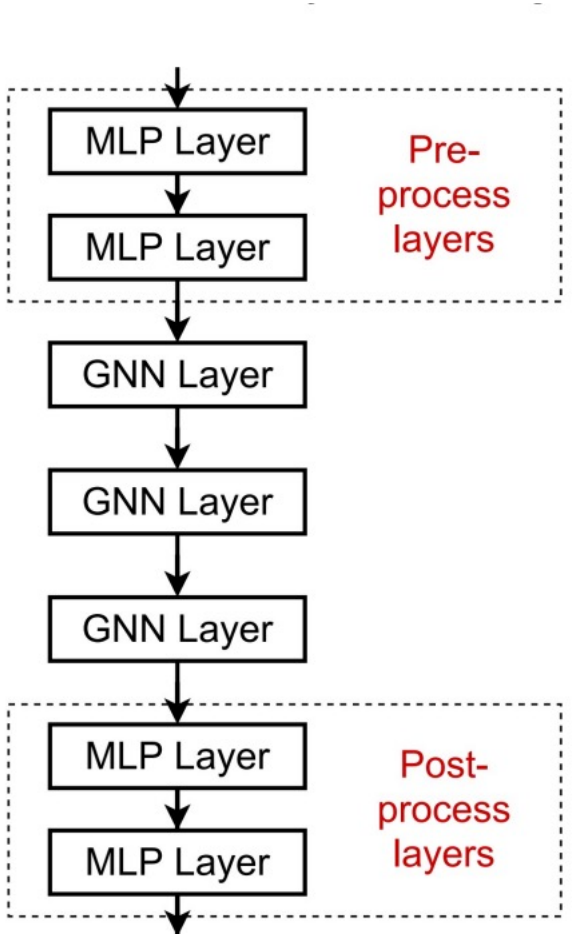
- Add Skip connections !

can forward all the outputs in GNN layers to last layer of GNN



Solution 3

- Add fully connected layers!

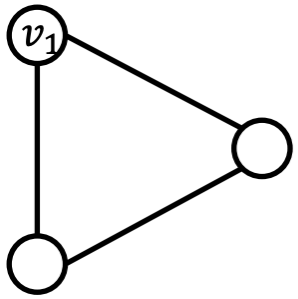


Feature augmentation/engineering

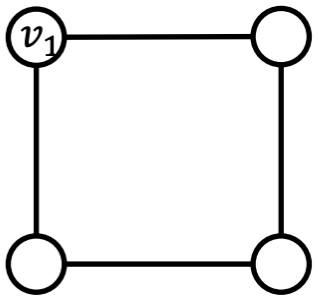
on node

- Can add features to make GNNs more expressive

v_1 resides in a cycle with length 3



v_1 resides in a cycle with length 4



- Local aggregation won't distinguish these but can add vertex features
  - Cycle count, Node degree, Edge degree, Node centrality, anything from graph theory!
- Unique identities to nodes

Graph augmentation/engineering

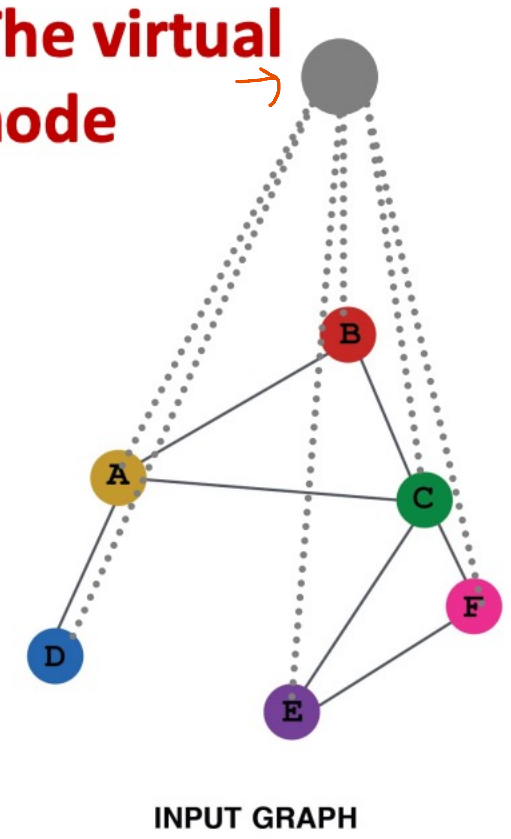
on graph

- Graph can be too sparse
- Too dense
- Not connected enough

} problems of graph depends on domain of research

- Can add virtual nodes and edges to the graph!
- Subsample or cluster nodes for the graph

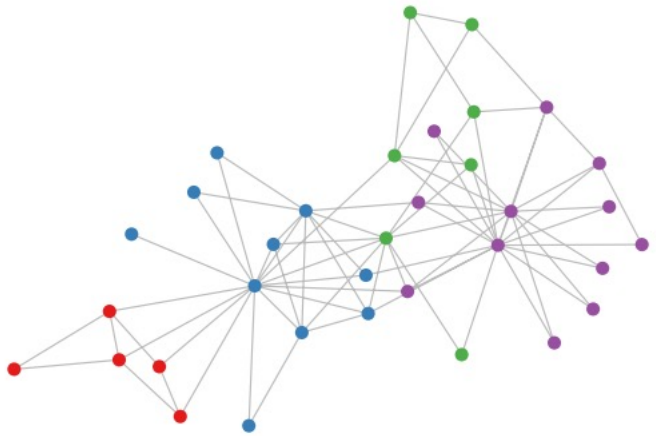
just don't care labels on virtual nodes



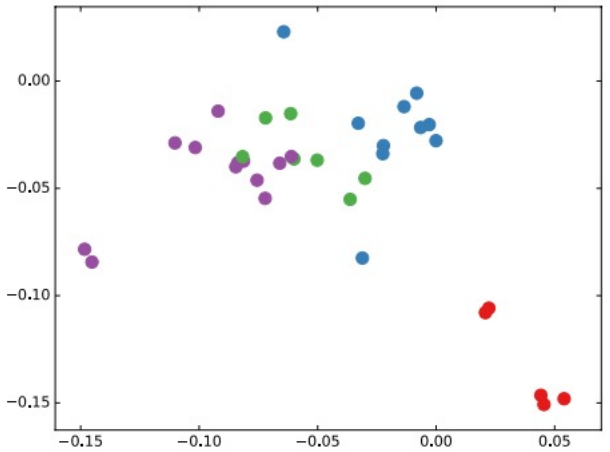
GNN Applications

Node Embeddings

2D embedding by dim reduction



(a) Karate club network



(b) Random weight embedding

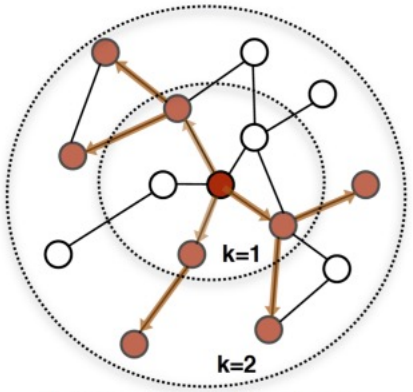
a benchmark graph dataset

Original Graph

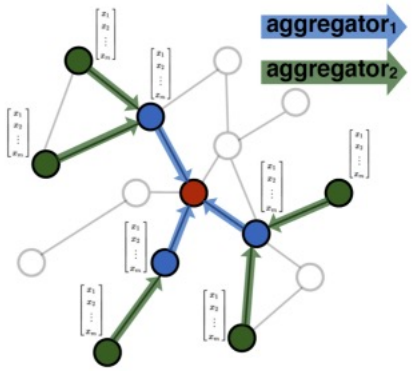
GCN Embedding based on message passing

Node classification

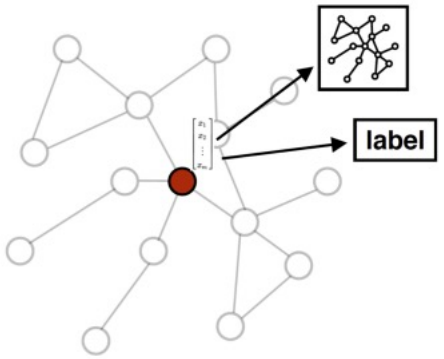
GraphSAGE



1. Sample neighborhood



2. Aggregate feature information from neighbors



3. Predict graph context and label using aggregated information

- Inductive, message passing network
- Mean aggregation –aggregates neighbors and concatenates self (skip)

more complex aggregation func

$$\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W} \cdot \text{MEAN}(\{\mathbf{h}_v^{k-1}\} \cup \{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$$

more powerful aggre func

- LSTM Aggregation – visit nodes in random order to perform aggregation

around neighborhood

use LSTM to compute node embeddings for node classification

Node classification results from Graph SAGE

Protein-protein interaction

	Citation		Reddit		PPI	
Name	Unsup. F1	Sup. F1	Unsup. F1	Sup. F1	Unsup. F1	Sup. F1
Random	0.206	0.206	0.043	0.042	0.396	0.396
Raw features	0.575	0.575	0.585	0.585	0.422	0.422
DeepWalk	0.565	0.565	0.324	0.324	—	—
DeepWalk + features	0.701	0.701	0.691	0.691	—	—
GraphSAGE-GCN	0.742	0.772	**0.908**	0.930	0.465	0.500
GraphSAGE-mean	0.778	0.820	0.897	0.950	0.486	0.598
GraphSAGE-LSTM	0.788	0.832	**0.907**	**0.954**	0.482	**0.612**
GraphSAGE-pool	**0.798**	**0.839**	0.892	0.948	**0.502**	0.600
% gain over feat.	39%	46%	55%	63%	19%	45%

non-NN methods

Reading list

- Hamilton et al. Inductive Representation Learning on Large Graphs Graph SAGE
- https://towardsdatascience.com/expressive-power-of-graph-neural-networks-and-the-weisefeiler-lehman-test-b883db3c7c49
- Hamilton Book chapter 5
- https://arxiv.org/pdf/1810.00826.pdf GCN
- https://arxiv.org/pdf/1609.02907.pdf WL-test