

Graph Neural Networks

“I checked it very thoroughly, said the computer, and that quite definitely is the answer. I think the problem, to be quite honest with you, is that you’ve never actually known what the question is.”

- Douglas Adams, The Hitchhiker’s Guide to the Galaxy (1979)

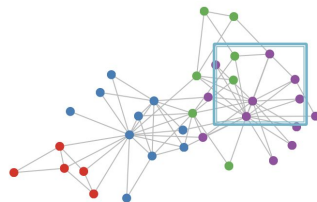
Jay Urbain, Ph.D.

Electrical Engineering and Computer Science Department

Milwaukee School of Engineering

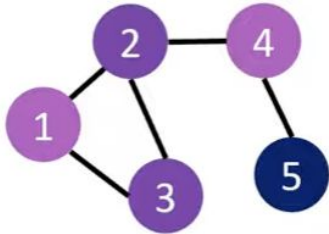
Credits

- **Graph Neural Networks: A review of Methods and Applications, Zhou, 2018.**
- **Graph Representation Learning, Hamilton, 2020.**
- Graph Convolutional Networks, Kipf and Welling, 2016.
- Multi-Layer Perceptron as Aggregator, Zaheer, 2017.
- Graph Attention Networks, Velickovic, 2017.
- Gated Graph Neural Networks, Li, 2015.
- DeepFindr
- Michael Bronstein, Oxford, Geometric Deep Learning Course
- Jure Lesvovec, Stanford



Simple Graph

Represent graph using an adjacency matrix (shown) or adjacency list.



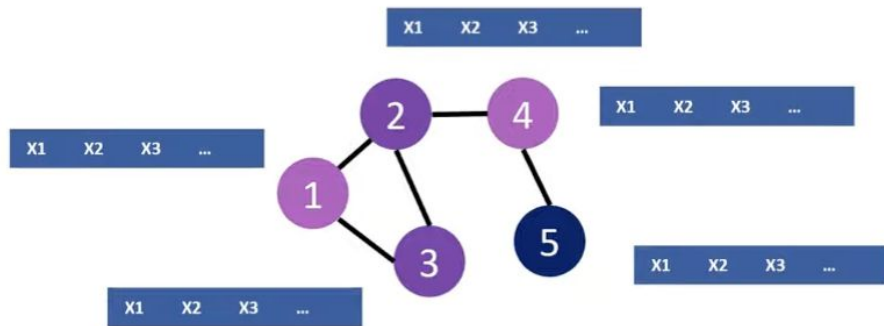
$G = (V, E)$

	v1	v2	...
v1	0	1	...
v2	1	0	...
v3	1	1	...
...

Adjacency matrix
 $V \times V$

Simple Graph

Node attributes: Modeling people, people have features. Modeling protein molecules with atoms, atoms have features.



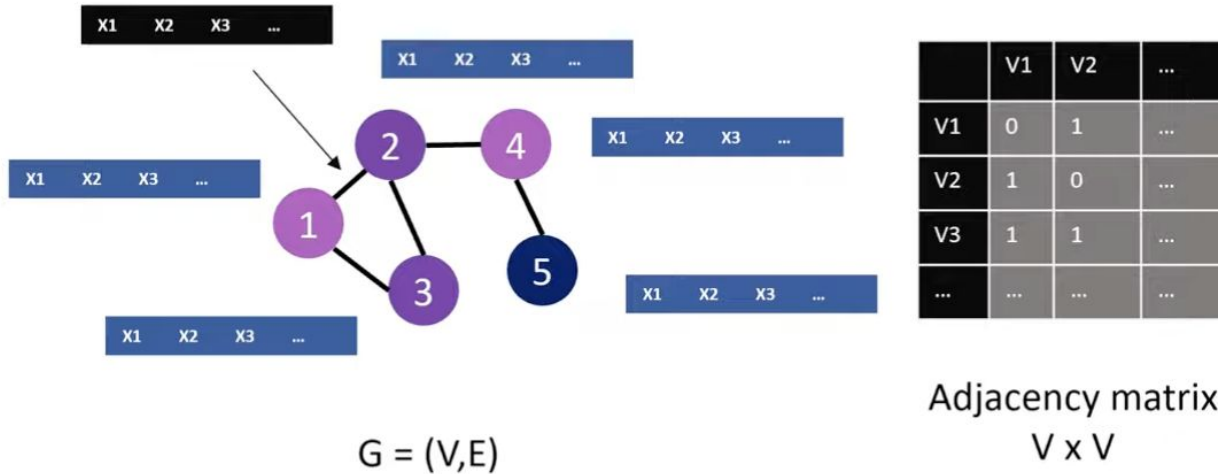
$G = (V, E)$

	V1	V2	...
V1	0	1	...
V2	1	0	...
V3	1	1	...
...

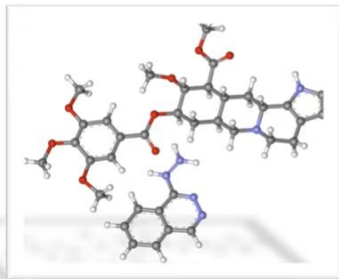
Adjacency matrix
 $V \times V$

Simple Graph

Edges - People have relationships, atoms have bonds.



Graph Data and Applications



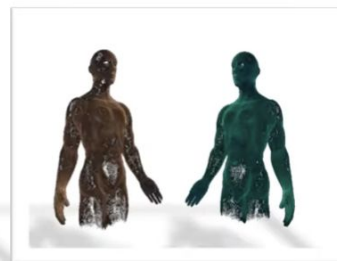
Medicine / Pharmacy



Recommender Systems



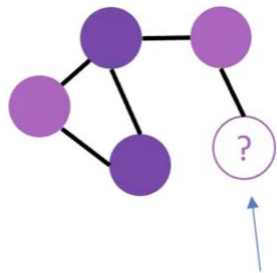
Social Networks



3D Games / Meshes

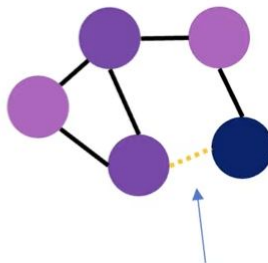
Examples of Graph ML Problems

Node-level predictions



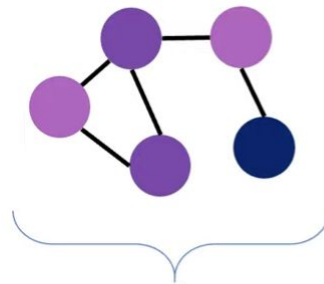
Does this person smoke?
(unlabeled node)

Edge-level predictions (Link prediction)



Next Netflix video?

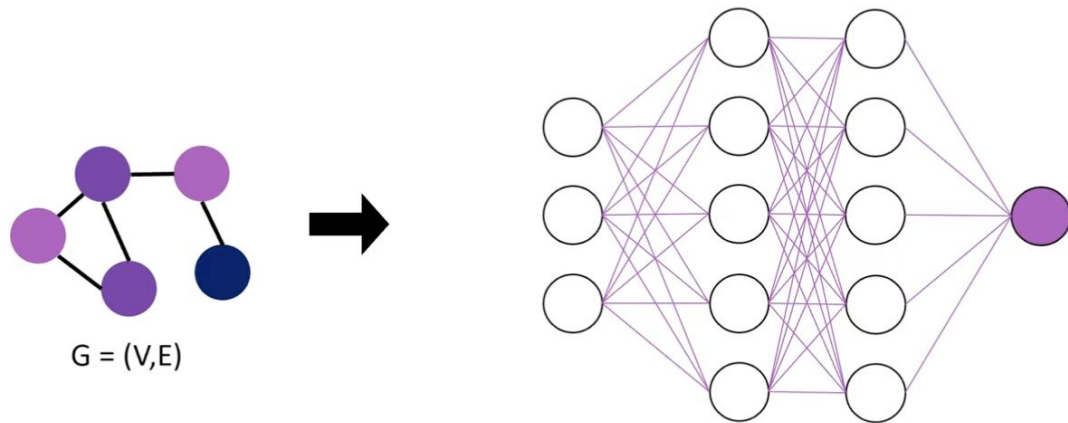
Graph-level predictions



Is this molecule a suitable drug?

Neural Networks

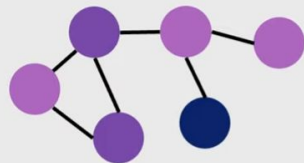
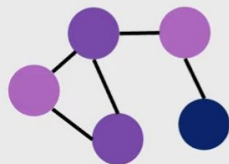
- Graph machine learning (GML) extend classic neural networks.
- Graphs have interesting features that make them difficult to work with.
- But how is graph data is different?



Graph Data *is* Different

- The size and shape of graph data might change within a dataset.
- With images you can crop, pad or resize an image.
- Can't resize graphs like images.
- Can't simply remove extra nodes in a graph.
- Graph methods need to be size independent.
- Need methods to handle arbitrary input shapes.

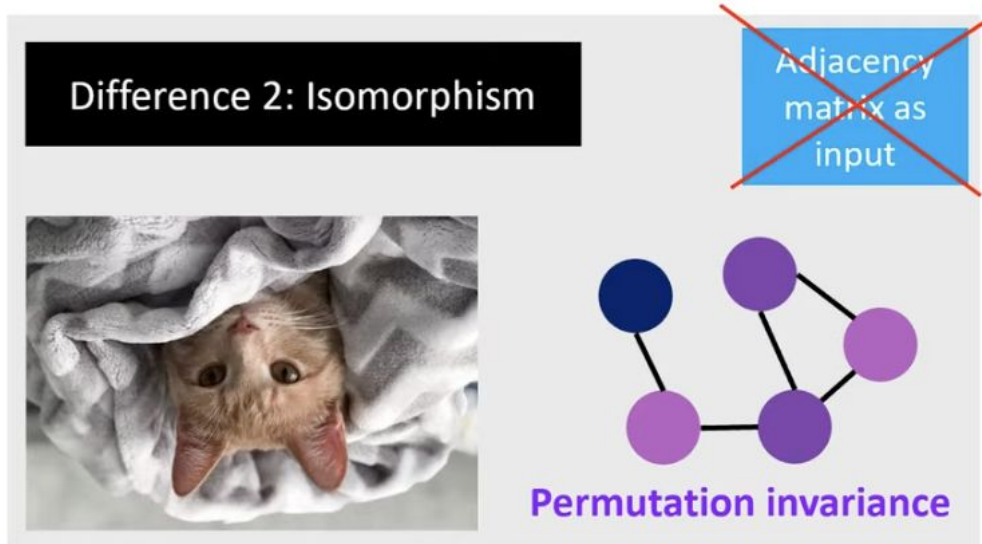
Difference 1: Size and Shape



Size independent

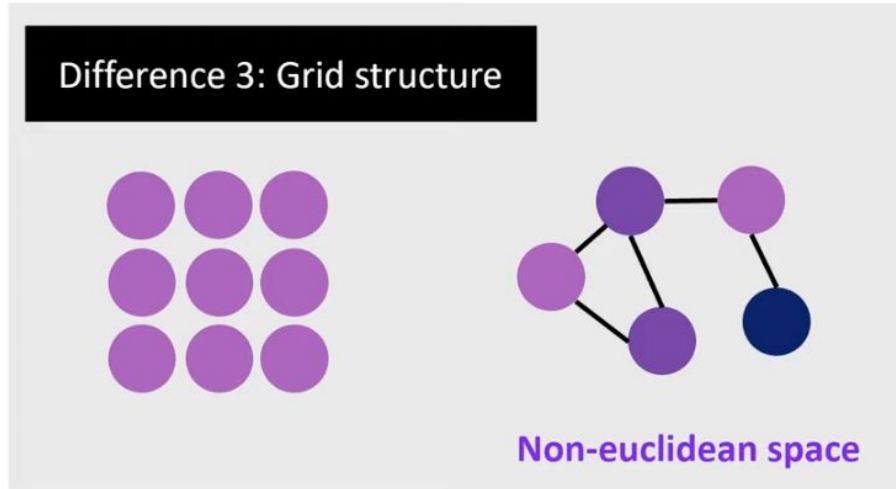
Graph Data is Different

- Two graphs that look different can still be structurally identical.
- Algorithm needs to be permutation invariant.
- Can't use adjacency list directly since it is subject to node order.



Graph Data is Different

- Euclidean distance is not clearly defined.
- Can't use simple distance metrics.
- Why machine learning around graph is often called geometric deep learning.
 - Why you can't just feed an adjacency list into a graph network!

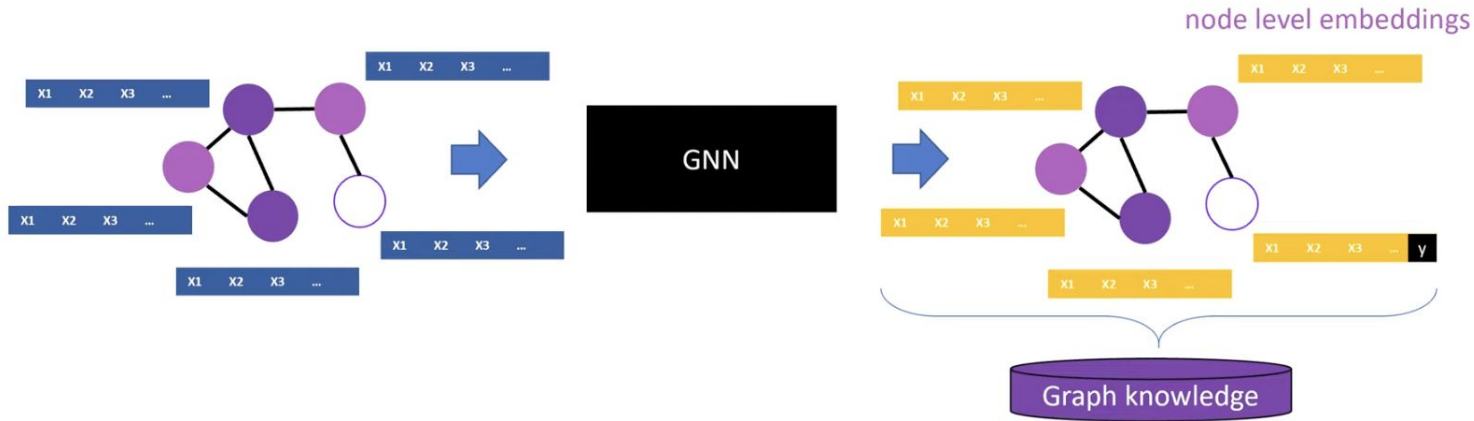


Fundamental Idea of GNNs

Big idea - representation learning!

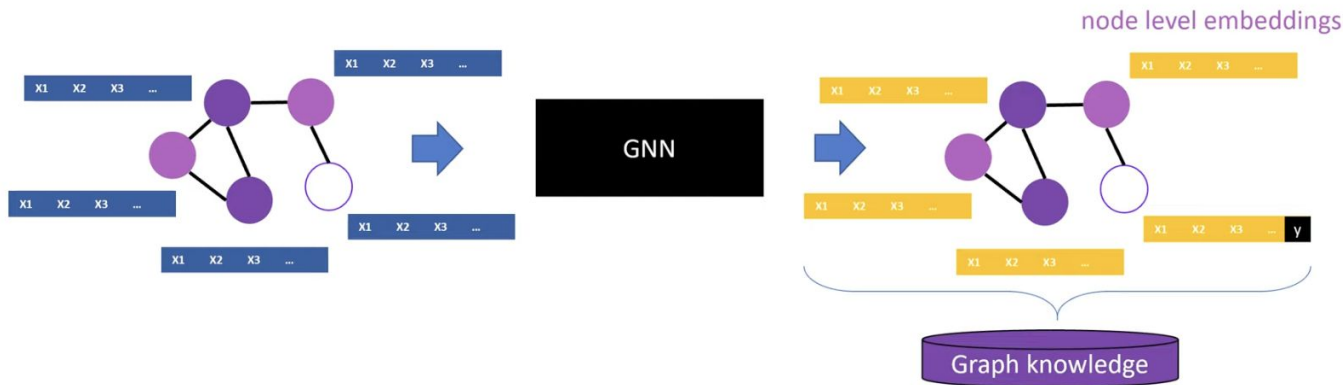
- Learning for a neural network a suitable *representation* of graph data.
- Using all information about the graph, the node features, and the connections from the adjacency matrix.
- The GNN outputs neural representations (embeddings) for each of the nodes.

Fundamental Idea of GNNs



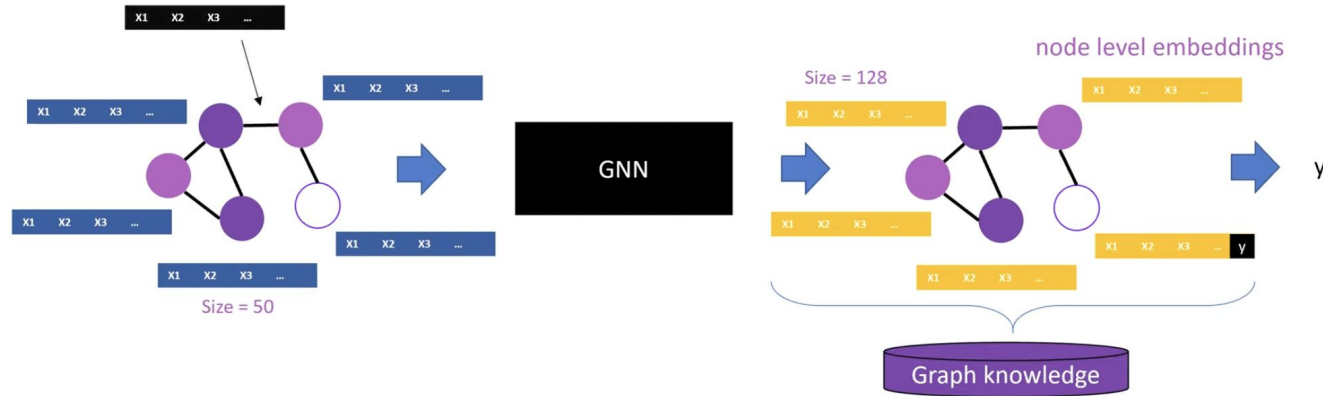
- Each node embedding contains *structural* and *feature information* of the other nodes in the graph.
- So each node knows something about the other nodes, the connections to these nodes, and its context in the graph.
- The embeddings can be used to perform predictions.
- How you use the embeddings is heavily dependent on the application.

Fundamental Idea of GNNs



- Similar nodes lead to similar node embeddings.
- For node level predictions, use the learned embedding of a specific unlabeled node to obtain a prediction.
- To perform graph level prediction, use all of the node embeddings, combine them some say to get a representation of the whole graph.
 - Pooling operations can be used to create a single graph representation.

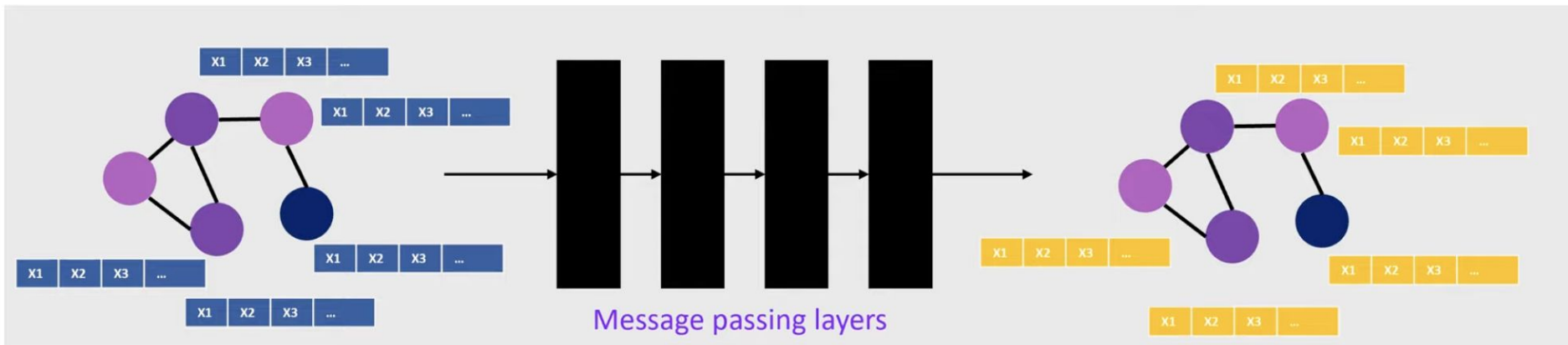
Fundamental Idea of GNNs



- Edge features can also be explicitly included.
- For edge level prediction use learned edge embeddings.
- Size of embedding is a hyperparameter.
- *Note: Embeddings can't be directly interpreted since they're an artificial construct of node and edge information in the graph.*

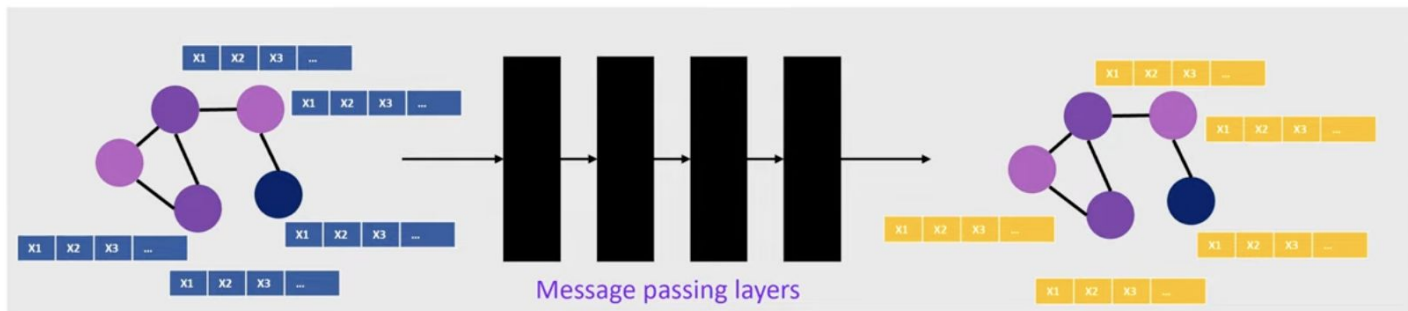
How to Graph Neural Networks work?

- Within the GNN there are several message passing layers.
- They're responsible for combining the node and edge information into the embeddings.



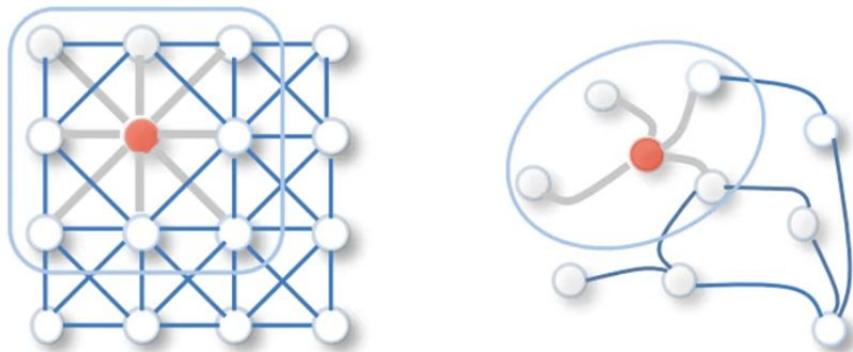
How to GNNs work - Summary

- Graph information includes node features and structural information passed through message passing layers.
- These layers construct the node embeddings that contain the knowledge about other nodes and edges in a compressed format.
- This is done by gathering the current known information of neighbor nodes, combining it in certain ways and updating the node features and states with these embeddings
- This is called **graph convolution** and can be seen as an extension of applying convolutions to graphs



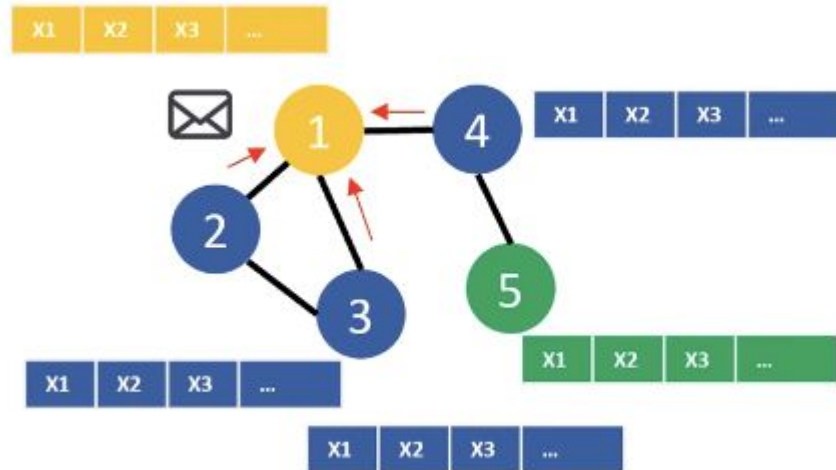
Convolution

- For images (left) slide learnable kernels over the grid structure of the pixels.
- Combines the local information for each pixel neighborhood.
- For non-euclidean graphs (right), this idea is extended via connected nodes.
- This sharing mechanism is also called *message passing*.



What is happening in the Message Passing Layers?

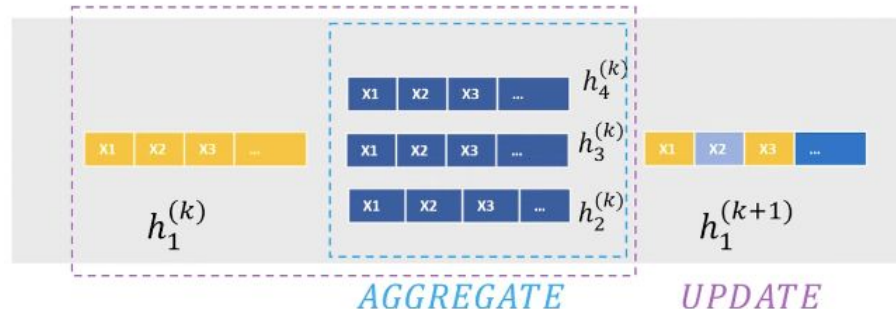
- To update the node state, we collect information from direct neighbors - message passing.
- Get information about our state and our neighbor's state.



What is happening in the Message Passing Layers?

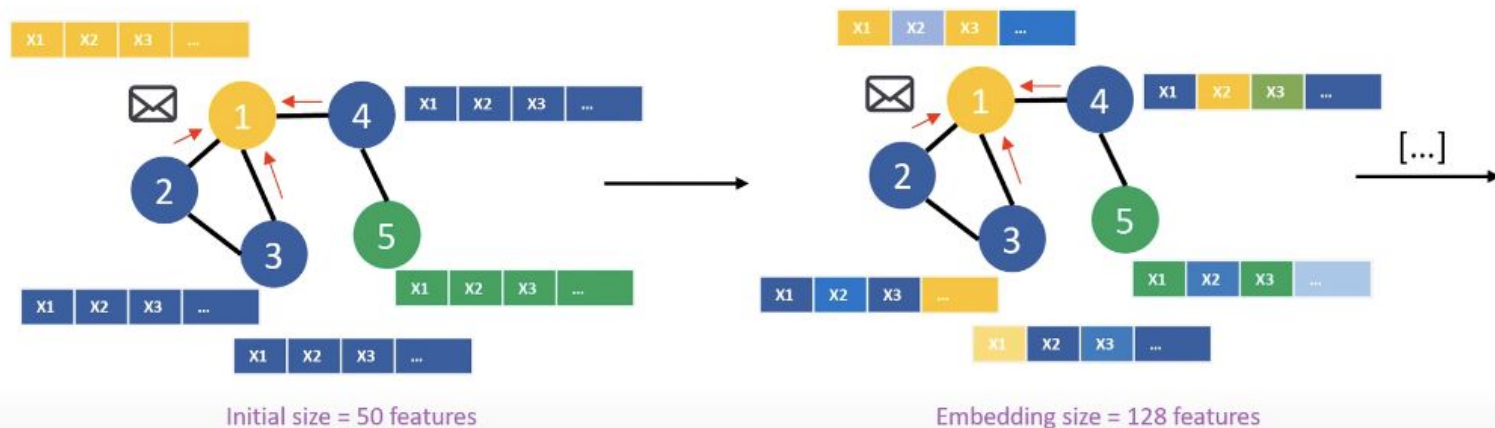
To update a node's state:

- Collect state information from neighbor nodes using message passing at time step k .
- AGGREGATE the state information from neighbor nodes.
- UPDATE the node's state h^{k+1} at time step $k+1$ by combining node's previous state h^k with the aggregated neighboring nodes state.
- *Note how the node's updated state includes information from its neighboring nodes.*



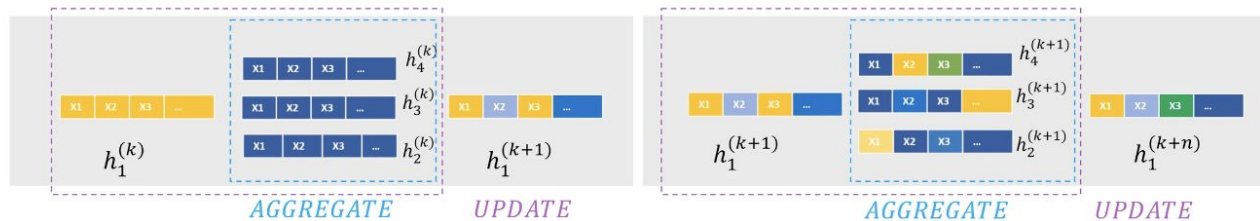
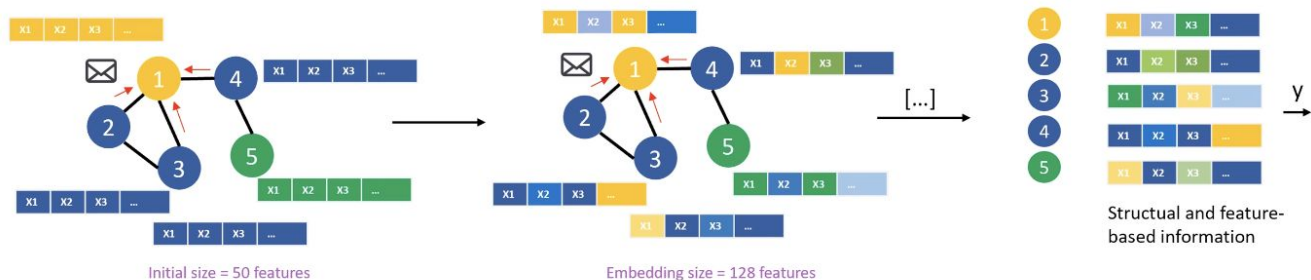
What is happening in the Message Passing Layers?

- Message passing is done for all nodes.
- The size of the embedding is a hyper-parameter.
- The number of message passing steps is defined by the number of message passing layers.



What is happening in the Message Passing Layers?

- Over time steps, each node in the graph learns something about all of the other nodes in the graph.
- Question: If a graph is much deeper than the number of layers in the GNN how does each node learn about all of the nodes in the graph?



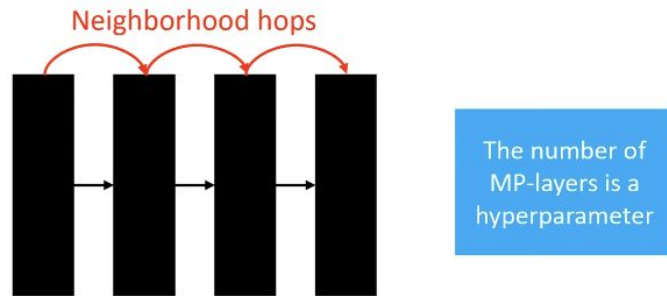
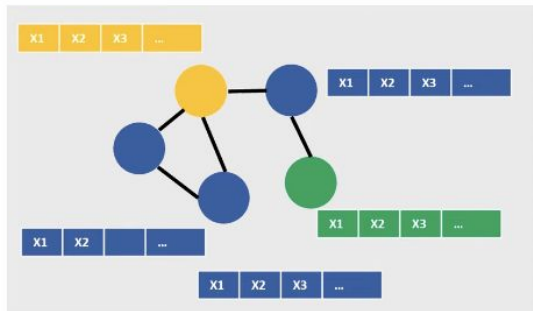
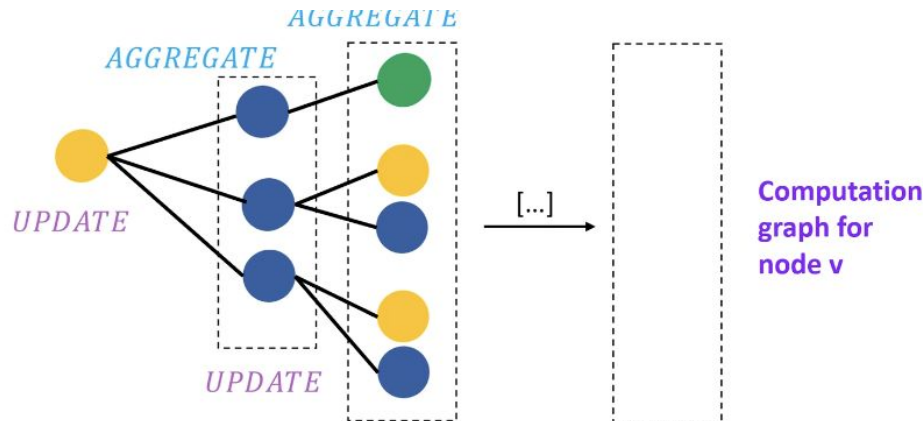
What is happening in the Message Passing Layers?

- Iteratively learning information from neighbors, and then learning information from their neighbor's neighbors, etc!
- Repeat for multiple epochs to learn characteristics of entire graph at each node.

The local feature aggregation can be compared to learnable CNN kernels

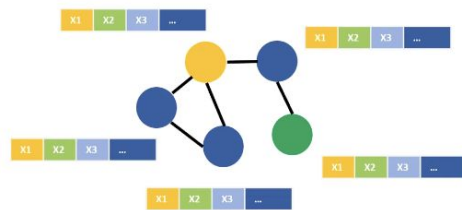
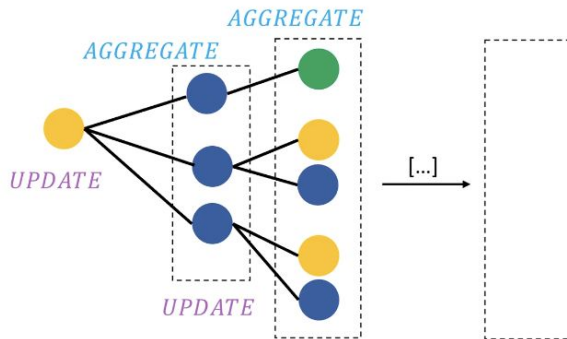
Computation Graph Representation

Number of layers is largely dependent on application



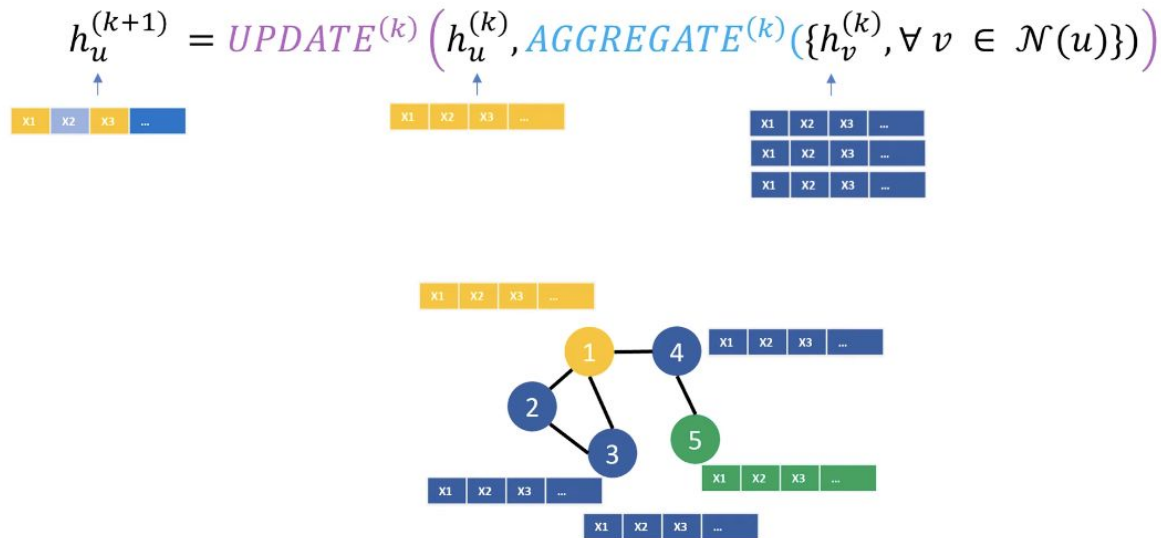
Over-smoothing in GNN's

- Too many message passing layers can lead to a condition called **over-smoothing**.
- Embeddings of nodes converge to similar values.
- 2 or 3 layers is often sufficient.
- *Pairnorm* smoothing using a normalization layer to prevent nodes from becoming too similar. <https://arxiv.org/abs/1909.12223>



Message Passing Update and Aggregation Functions

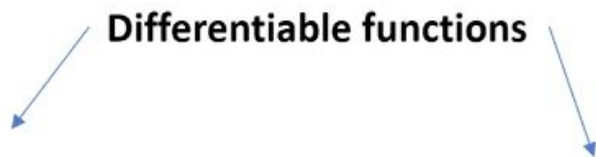
The message passing process is essentially the same for all graph machine learning methods.



Message Passing Update and Aggregation Functions

Primary difference in models is how **AGGREGATE** and **UPDATE** are performed.

Differentiable functions



$$h_u^{(k+1)} = \text{UPDATE}^{(k)} \left(h_u^{(k)}, \text{AGGREGATE}^{(k)}(\{h_v^{(k)}, \forall v \in \mathcal{N}(u)\}) \right)$$

- Mean
- Max
- Neural Network
- Recurrent NN

- Mean
- Max
- Normalized Sum
- Neural Network



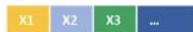
GNN Variants

Graph Representation Learning, Ch. 4 Graph Neural Network Model

https://cs.mcgill.ca/~wlh/comp766/files/chapter4_draft_mar29.pdf



UPDATE



**Graph Convolutional Networks,
Kipf and Welling [2016]**

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

Sum of normalized neighbor embeddings

**Multi-Layer-Perceptron as
Aggregator, Zaheer et al. [2017]**

Aggregated message

$$\mathbf{m}_{\mathcal{N}(u)} = \boxed{\text{MLP}_{\theta}} \left(\sum_{v \in \mathcal{N}(u)} \text{MLP}_{\phi}(\mathbf{h}_v) \right)$$

trainable!

Send states through a MLP

**Graph Attention Networks,
Veličković et al. [2017]**

$$\mathbf{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \alpha_{u,v} \mathbf{h}_v$$

Attention weights

$$\alpha_{u,v} = \frac{\exp(\mathbf{a}^{\top} [\mathbf{W}\mathbf{h}_u \oplus \mathbf{W}\mathbf{h}_v])}{\sum_{v' \in \mathcal{N}(u)} \exp(\mathbf{a}^{\top} [\mathbf{W}\mathbf{h}_u \oplus \mathbf{W}\mathbf{h}_{v'}])}$$

**Gated Graph Neural Networks,
Li et al. [2015]**

$$\mathbf{h}_u^{(k)} = \text{GRU}(\mathbf{h}_u^{(k-1)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)})$$

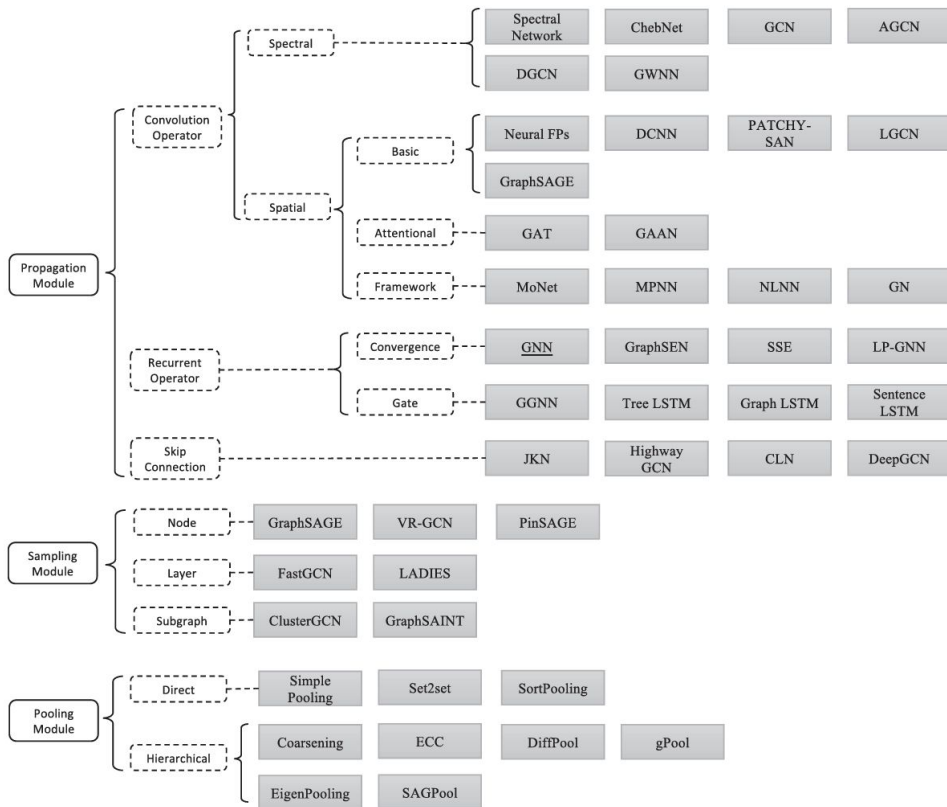
Recurrent update of the state

GNN Variants

Graph Neural Networks: A Review of Methods and Applications, Zhou, 2022. <https://arxiv.org/pdf/1812.08434.pdf>

J. Zhou et al.

AI Open 1 (2020) 57–81



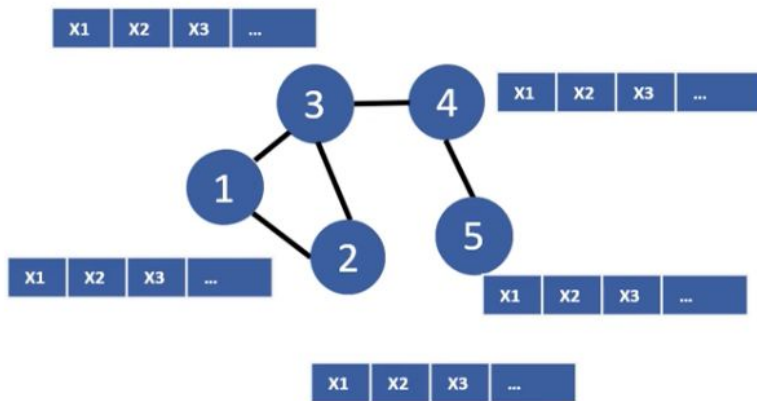
Different variants of recurrent operators.

Variant	Aggregator	Updater
GGNN	$\mathbf{h}_{i,f_v}^t = \sum_{k \in \mathcal{N}_v} \mathbf{h}_k^{t-1} + \mathbf{b}$	$\mathbf{z}_v^t = \sigma(\mathbf{W}^z \mathbf{h}_{i,f_v}^t + \mathbf{U}^z \mathbf{h}_v^{t-1})$ $\mathbf{r}_v^t = \sigma(\mathbf{W}^r \mathbf{h}_{i,f_v}^t + \mathbf{U}^r \mathbf{h}_v^{t-1})$ $\tilde{\mathbf{h}}_v^t = \tanh(\mathbf{W} \mathbf{h}_{i,f_v}^t + \mathbf{U}(\mathbf{r}_v^t \odot \mathbf{h}_v^{t-1}))$ $\mathbf{h}_v^t = (1 - \mathbf{z}_v^t) \odot \mathbf{h}_v^{t-1} + \mathbf{z}_v^t \odot \tilde{\mathbf{h}}_v^t$
Tree LSTM (Child sum)	$\mathbf{h}_{i,f_v}^d = \sum_{k \in \mathcal{N}_v} \mathbf{U}^d \mathbf{h}_k^{t-1}$ $\mathbf{h}_{i,f_v,k}^f = \mathbf{U}^f \mathbf{h}_k^{t-1}$ $\mathbf{h}_{i,f_v}^{to} = \sum_{k \in \mathcal{N}_v} \mathbf{U}^o \mathbf{h}_k^{t-1}$ $\mathbf{h}_{i,f_v}^{tu} = \sum_{k \in \mathcal{N}_v} \mathbf{U}^u \mathbf{h}_k^{t-1}$	$\mathbf{i}_v^t = \sigma(\mathbf{W}^i \mathbf{x}_v^t + \mathbf{h}_{i,f_v}^d + \mathbf{b}^i)$ $\mathbf{f}_{vk}^t = \sigma(\mathbf{W}^f \mathbf{x}_v^t + \mathbf{h}_{i,f_v,k}^f + \mathbf{b}^f)$ $\mathbf{o}_v^t = \sigma(\mathbf{W}^o \mathbf{x}_v^t + \mathbf{h}_{i,f_v}^{to} + \mathbf{b}^o)$ $\mathbf{u}_v^t = \tanh(\mathbf{W}^u \mathbf{x}_v^t + \mathbf{h}_{i,f_v}^{tu} + \mathbf{b}^u)$ $\mathbf{c}_v^t = \mathbf{i}_v^t \odot \mathbf{u}_v^t + \sum_{k \in \mathcal{N}_v} \mathbf{f}_{vk}^t \odot \mathbf{c}_k^{t-1}$ $\mathbf{h}_v^t = \mathbf{o}_v^t \odot \tanh(\mathbf{c}_v^t)$
Tree LSTM (N-ary)	$\mathbf{h}_{i,f_v}^d = \sum_{l=1}^K \mathbf{U}_l^d \mathbf{h}_{v_l}^{t-1}$ $\mathbf{h}_{i,f_v,k}^f = \sum_{l=1}^K \mathbf{U}_{kl}^f \mathbf{h}_{v_l}^{t-1}$ $\mathbf{h}_{i,f_v}^{to} = \sum_{l=1}^K \mathbf{U}_l^o \mathbf{h}_{v_l}^{t-1}$ $\mathbf{h}_{i,f_v}^{tu} = \sum_{l=1}^K \mathbf{U}_l^u \mathbf{h}_{v_l}^{t-1}$	
Graph LSTM in (Peng et al., 2017)	$\mathbf{h}_{i,f_v}^d = \sum_{k \in \mathcal{N}_v} \mathbf{U}_{m(v,k)}^d \mathbf{h}_k^{t-1}$ $\mathbf{h}_{i,f_v,k}^f = \mathbf{U}_{m(v,k)}^f \mathbf{h}_k^{t-1}$ $\mathbf{h}_{i,f_v}^{to} = \sum_{k \in \mathcal{N}_v} \mathbf{U}_{m(v,k)}^o \mathbf{h}_k^{t-1}$ $\mathbf{h}_{i,f_v}^{tu} = \sum_{k \in \mathcal{N}_v} \mathbf{U}_{m(v,k)}^u \mathbf{h}_k^{t-1}$	

Graph Matching: Global Graph Pooling in GNNs

Want to combine graph level representations into a single graph representation.

How?

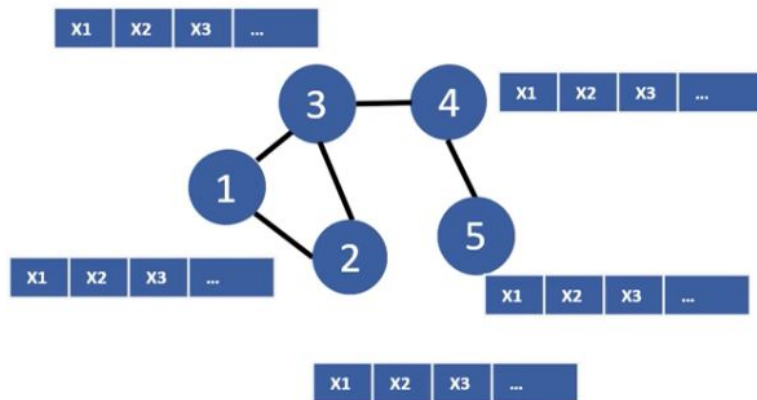


Graph Matching: Global Graph Pooling in GNNs

Want to combine graph level representations into a single graph representation.

How - typical measures: max, min

Problems?

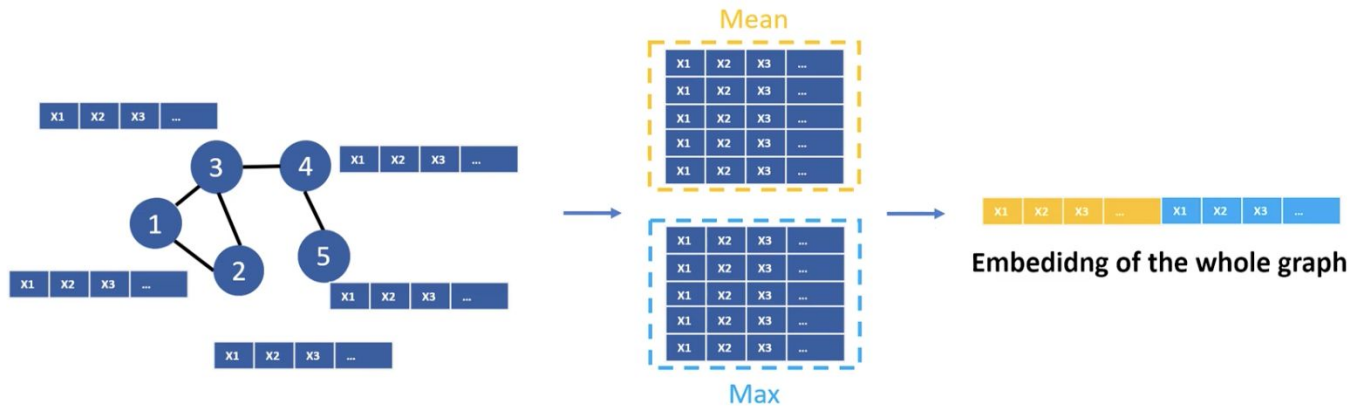


Graph Matching: Global Graph Pooling in GNNs

Problems: graphs may have different numbers of nodes. Graphs may be dynamic.

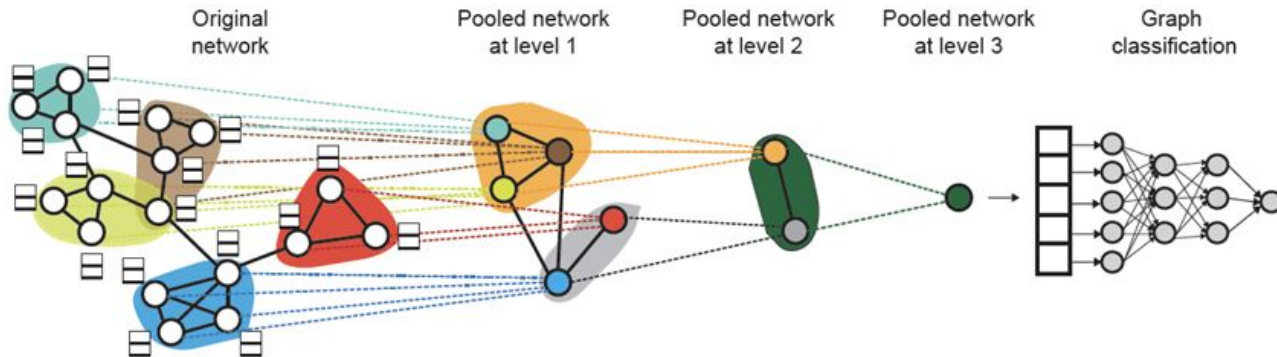
Want global pooling mechanism that can “squeeze” any number of nodes into a single representation.

Alternative: graph coursening. Size of graph is reduced over time.



Graph Coarsening Approach in GNNs

- Size of graph is reduced over time.
- At each hierarchical level, the nodes in the graph are assigned to different clusters based on their learned embeddings.
- A new graph is then formed based on these clusters, where the nodes in the graph at layer l correspond to clusters in layer $l-1$.
- GNNs are used to process the graphs at the different hierarchical layers.



Binary Mask for Node-level or Edge-level Prediction

For training and prediction, you don't have all node labels available.

Masks tell you what node (or edge) to use for each task.

