

# Structure-preserving Graph Neural Networks

Prof. O-Joun Lee

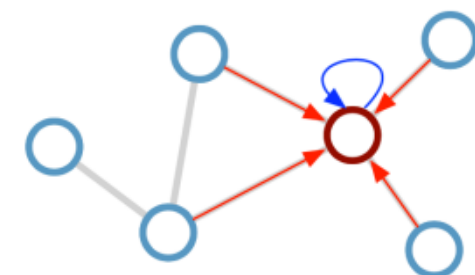
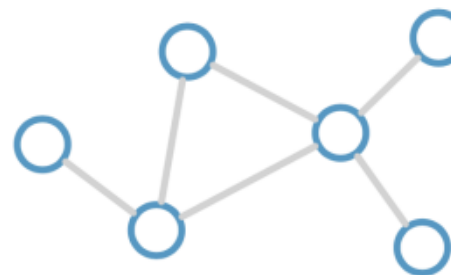
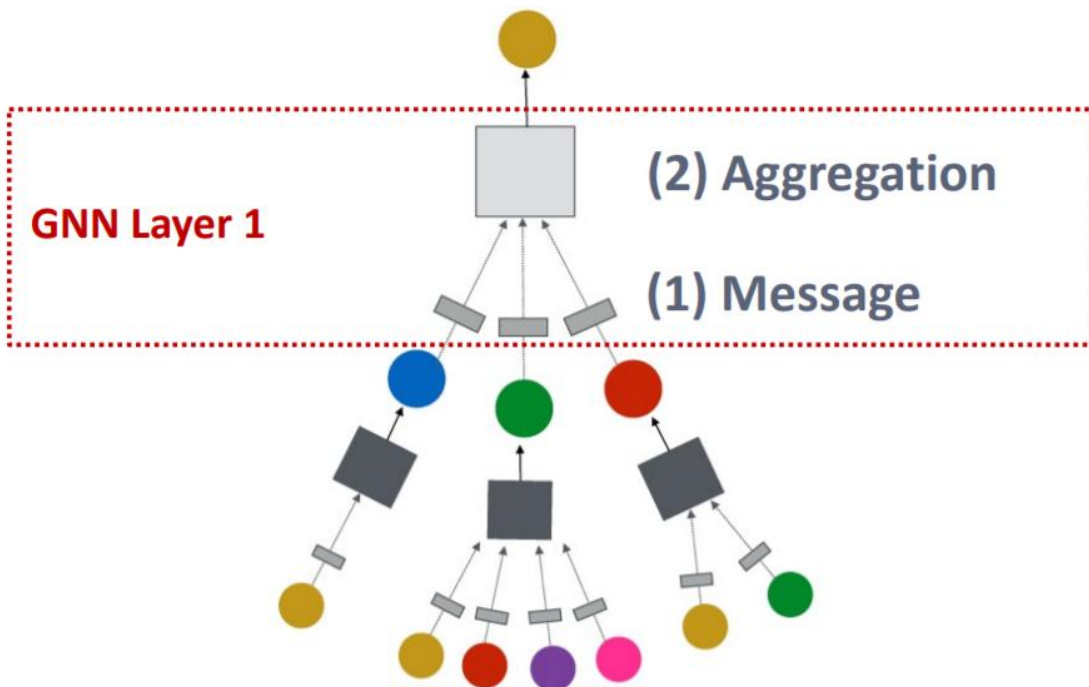
Dept. of Artificial Intelligence,  
The Catholic University of Korea  
*ojlee@catholic.ac.kr*

# Contents



- Computational graph
- Graph Isomorphism and GNN problems
- Expressive power of GNNs
- How to build the most powerful GNNs?

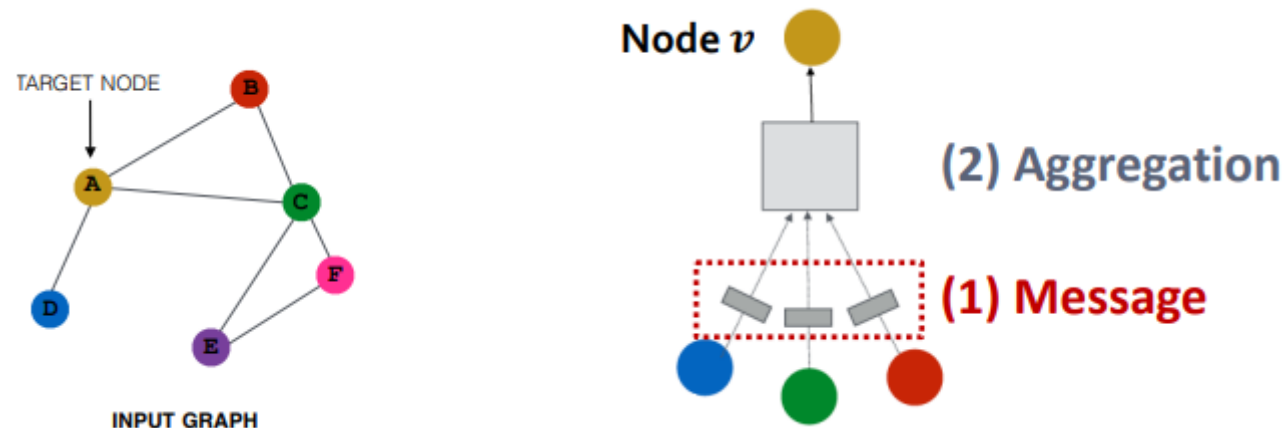
- GNN Layer = **Message** + **Aggregation**
  - Message COMPUTATION
    - how to make each neighborhood node as embedding?
  - Message AGGERGATION
    - how to combine those embeddings?



Update rule: 
$$\mathbf{h}_i^{(l+1)} = \sigma \left( \mathbf{h}_i^{(l)} \mathbf{W}_0^{(l)} + \sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} \mathbf{h}_j^{(l)} \mathbf{W}_1^{(l)} \right)$$

- Intuition: Each node will create a message, which will be sent to other nodes later
- Example: A Linear layer  $\mathbf{m}_u^{(l)} = \mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)}$ 
  - Multiply node features with weight matrix  $\mathbf{W}^{(l)}$

**Message function:**  $\mathbf{m}_u^{(l)} = \text{MSG}^{(l)} \left( \mathbf{h}_u^{(l-1)} \right)$



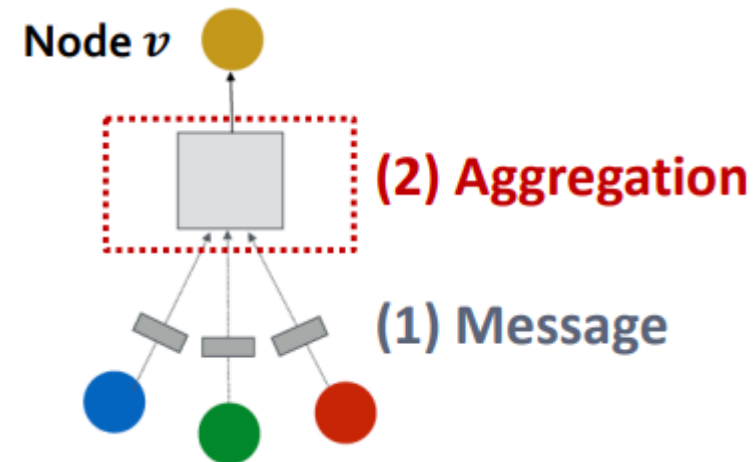
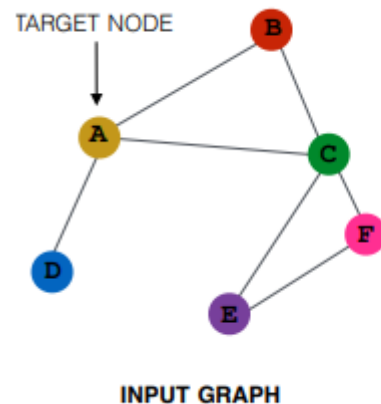


- Intuition: Each node will aggregate the messages from node  $v$ 's neighbors

$$\mathbf{h}_v^{(l)} = \text{AGG}^{(l)} \left( \left\{ \mathbf{m}_u^{(l)}, u \in N(v) \right\} \right)$$

- Example: Sum( $\cdot$ ), Mean( $\cdot$ ) or Max( $\cdot$ ) aggregator

$$\mathbf{h}_v^{(l)} = \text{Sum}(\{\mathbf{m}_u^{(l)}, u \in N(v)\})$$



- **Issue:** Information from node  $v$  itself could get lost
  - Computation of  $\mathbf{h}_v^{(l)}$  does not directly depend on  $\mathbf{h}_v^{(l-1)}$
- **Solution:** Include  $\mathbf{h}_v^{(l-1)}$  when computing  $\mathbf{h}_v^{(l)}$

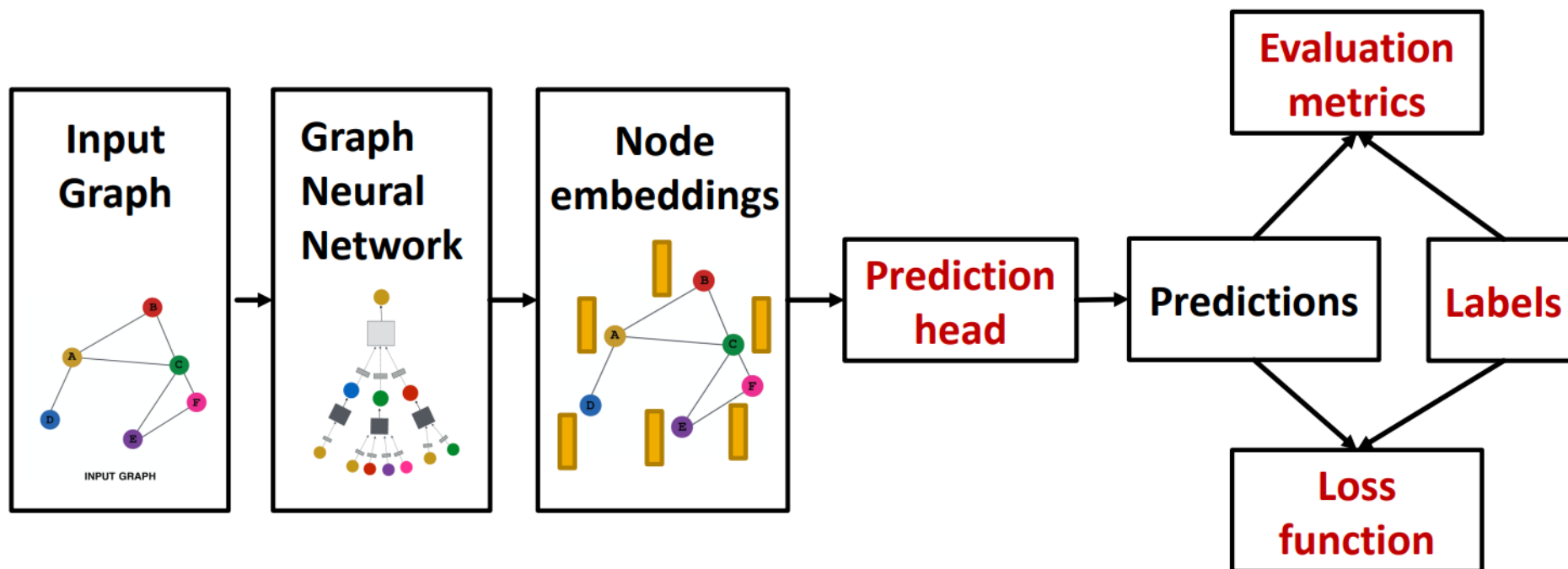
- (1) Message: compute message from node  $v$  itself

$$\text{●} \text{●} \text{●} \quad \mathbf{m}_u^{(l)} = \mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)} \qquad \text{●} \quad \mathbf{m}_v^{(l)} = \mathbf{B}^{(l)} \mathbf{h}_v^{(l-1)}$$

- (2) Aggregation: After aggregating from neighbors, we can aggregate the message from node  $v$  itself
  - Via concatenation or summation

$$\mathbf{h}_v^{(l)} = \text{CONCAT} \left( \underbrace{\text{AGG} \left( \left\{ \mathbf{m}_u^{(l)}, u \in N(v) \right\} \right)}_{\text{First aggregate from neighbors}}, \underbrace{\mathbf{m}_v^{(l)}}_{\text{Then aggregate from node itself}} \right)$$

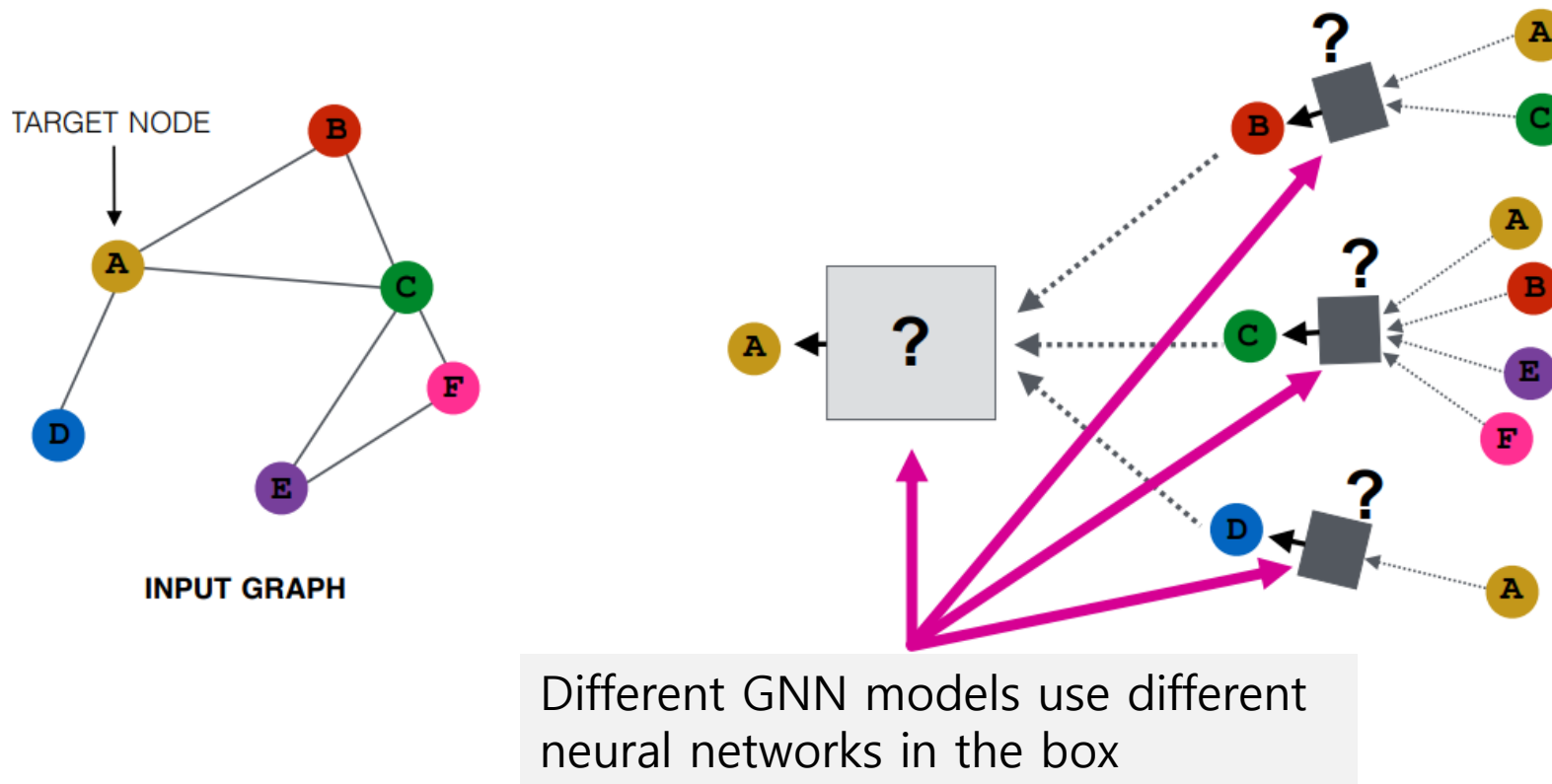
- Inputs: a graph dataset
- Outputs: node embeddings



- How powerful are GNNs?
  - Many GNN models have been proposed (e.g., GCN, GAT, GraphSAGE, design space).
  - What is the expressive power (ability to distinguish different graph structures) of these GNN models?
  - How to design a maximally expressive GNN model?

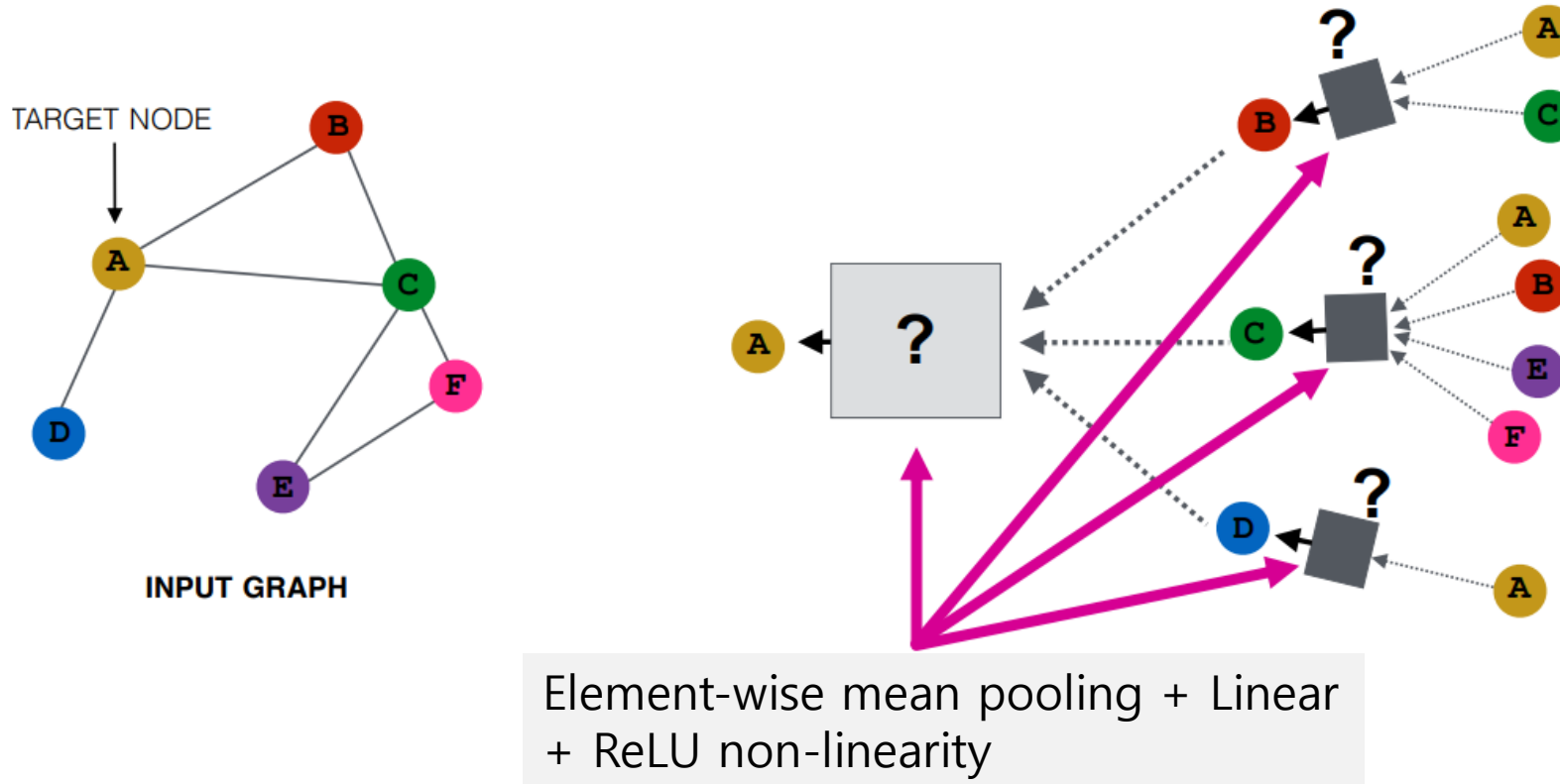


- Many GNN models have been proposed:
  - GCN, GraphSAGE, GAT, Design Space etc.

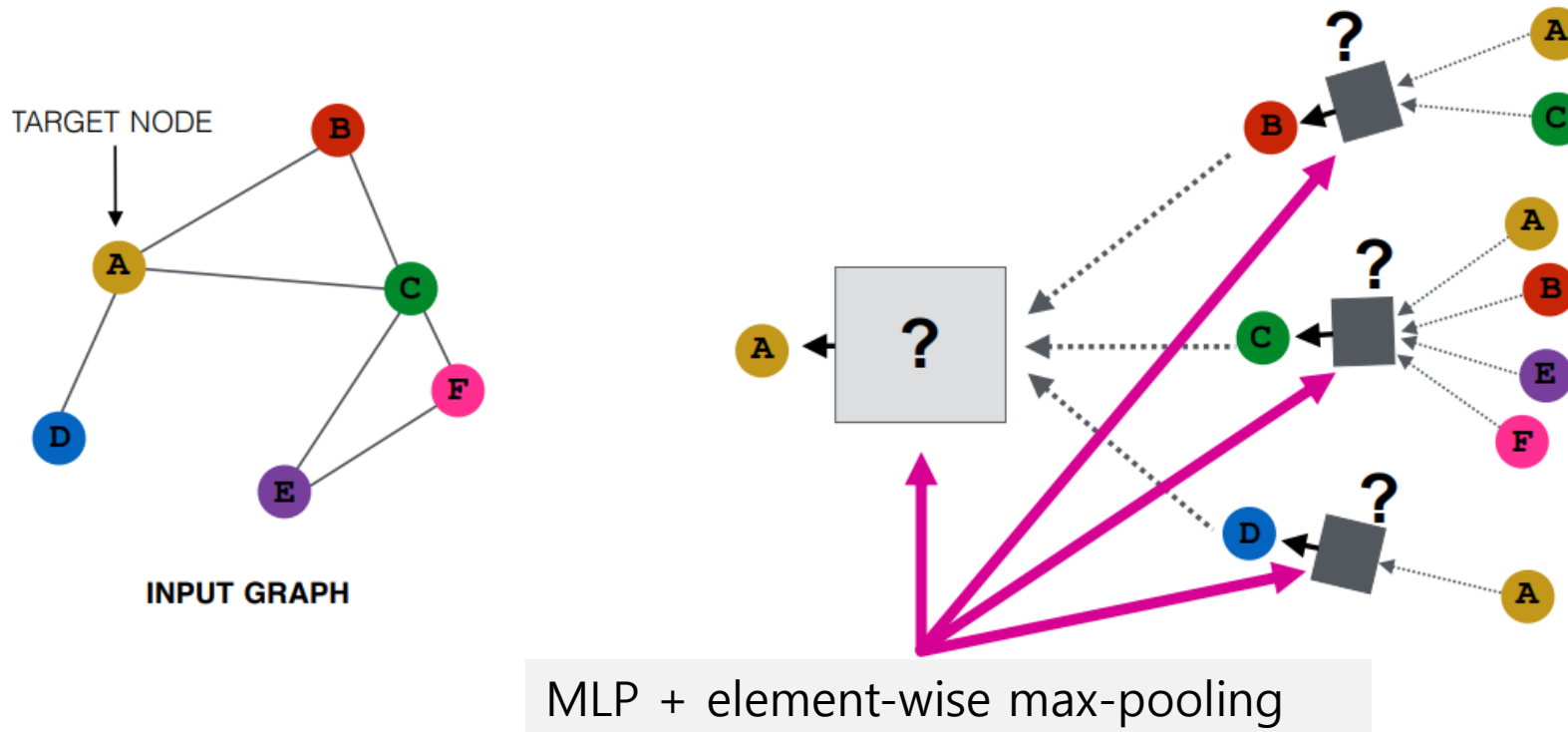


- GCN (mean-pool) [Kipf and Welling ICLR 2017]

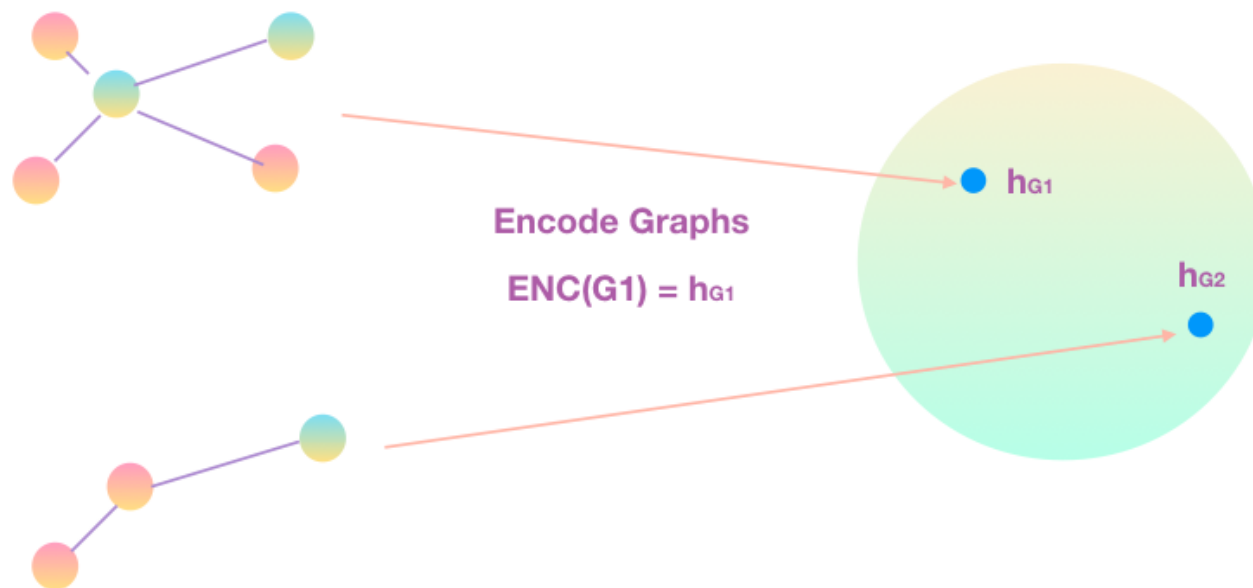
$$H^{(l+1)} = \sigma \left( \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} H^{(l)} W^{(l)} \right)$$



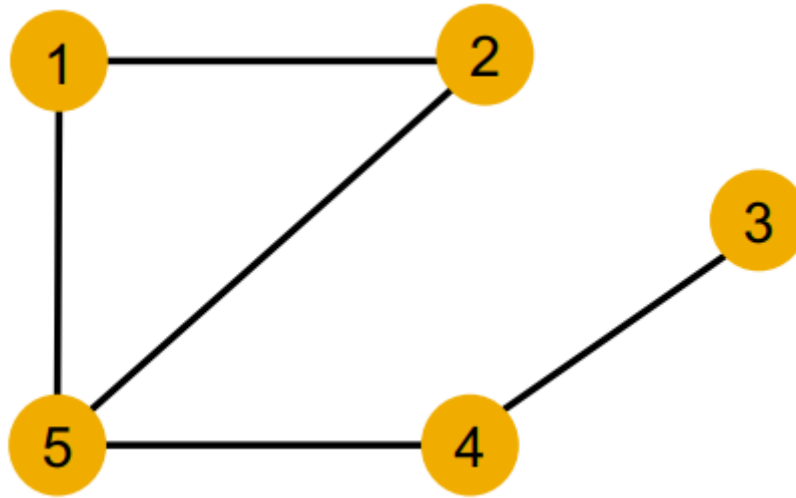
- GraphSAGE (max-pool) [Hamilton et al. NeurIPS 2017]: node features are sampled from neighborhoods instead of using the full adjacency matrix.



- **Do similar graphs really have the same label?**
  - The Graph Classification problem and the Graph Isomorphism problem are not the same.
  - Graph Isomorphism can be judged as isomorphic if the positions on the vector are close, but Graph Classification adds an element called "Label" to the graph, so that it is classified by the same Label, not by position.



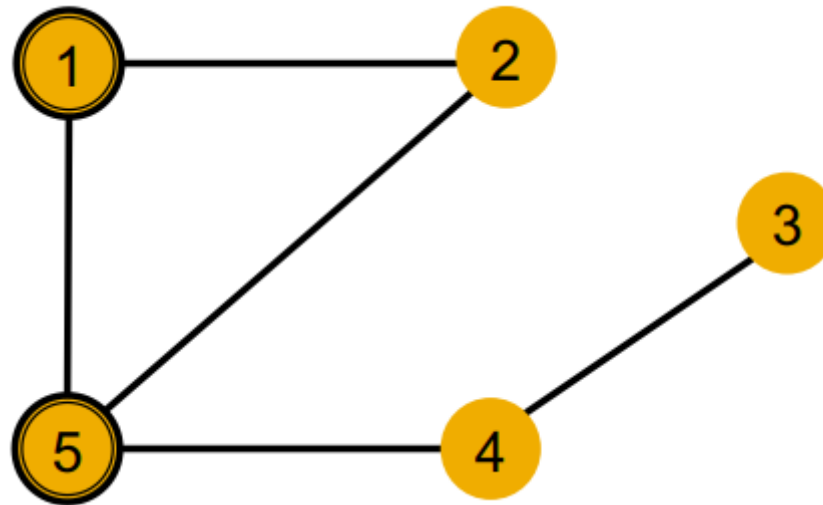
- We use node same/different colors to represent nodes with same/different features.
  - For example, the graph below assumes all the nodes share the same feature



**Key question:** How well can a GNN distinguish different graph structures?

➤ **Local Neighbourhood structures:**

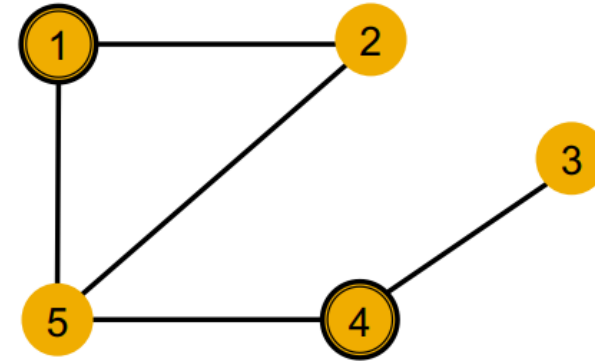
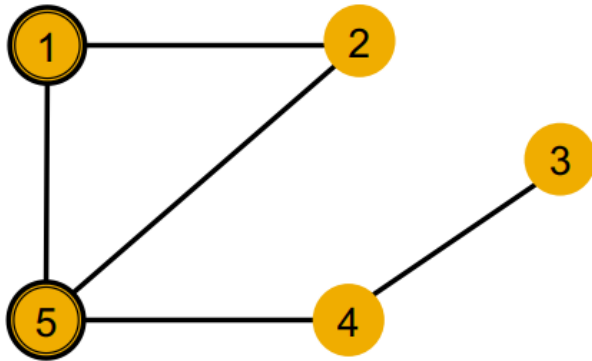
- We specifically consider **local neighborhood structures** around each node in a graph.
- Example: Nodes 1 and 5 have different neighborhood structures because they have different node degrees.





➤ **Local Neighbourhood structures:**

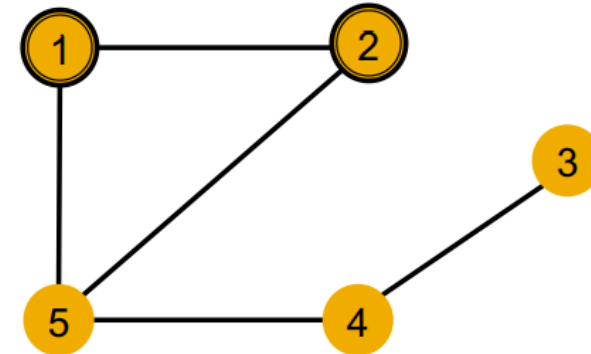
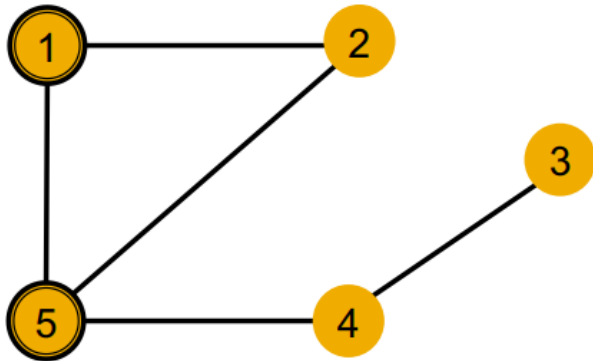
- We specifically consider **local neighborhood structures** around each node in a graph.
- Example: Nodes 1 and 5 have different neighborhood structures because they have different node degrees.
- Nodes 1 and 4 both have the same node degree of 2. However, they still have different neighborhood structures because their neighbors have different node degrees.



Node 1 has neighbors of degrees 2 and 3.  
Node 4 has neighbors of degrees 1 and 3.

## ➤ Local Neighbourhood structures:

- We specifically consider **local neighborhood structures** around each node in a graph.
- Example: Nodes 1 and 5 have different neighborhood structures because they have different node degrees.
- Nodes 1 and 4 both have the same node degree of 2. However, they still have different neighborhood structures because their neighbors have different node degrees.

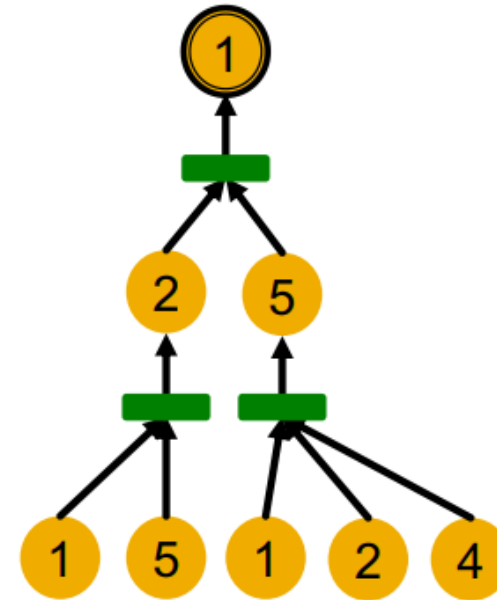
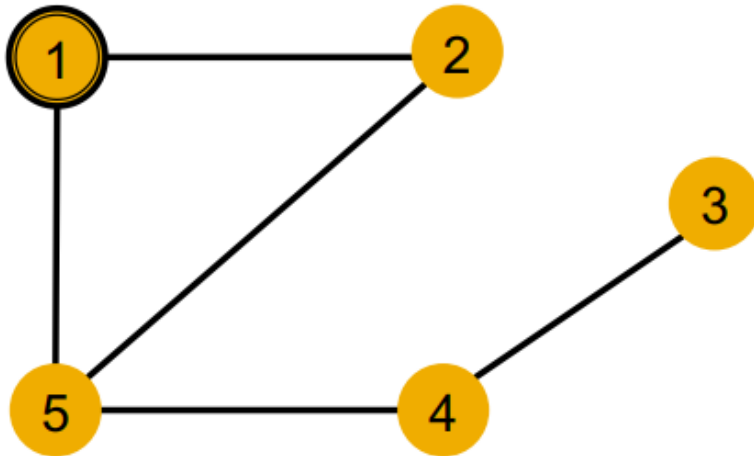


Node 1 has neighbors of degrees 2 and 3.  
Node 2 has neighbors of degrees 2 and 3.  
And even if we go a step deeper to 2nd hop neighbors,  
both nodes have the same degrees (Node 4 of degree 2)

- **Key question:** Can GNNs distinguish different node's local neighborhood structures?
  - If so, when? If not, when will a GNN fail?
- We need to understand how a GNN captures local neighborhood structures.
  - **Key concept:** Computational graph

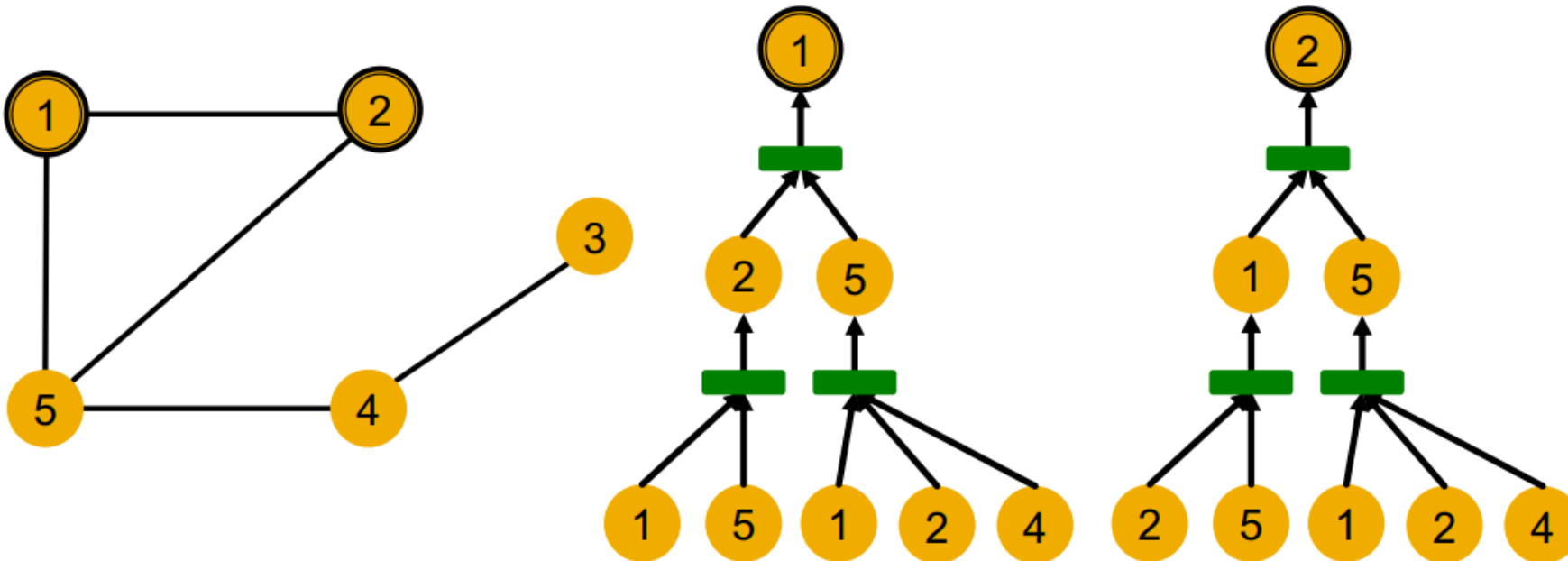
- In each layer, a GNN aggregates neighboring node embeddings.
- A GNN generates node embeddings through a computational graph defined by the neighborhood.

For example: Node 1's computational graph (2-layer GNN)



- In each layer, a GNN aggregates neighboring node embeddings.
- A GNN generates node embeddings through a computational graph defined by the neighborhood.

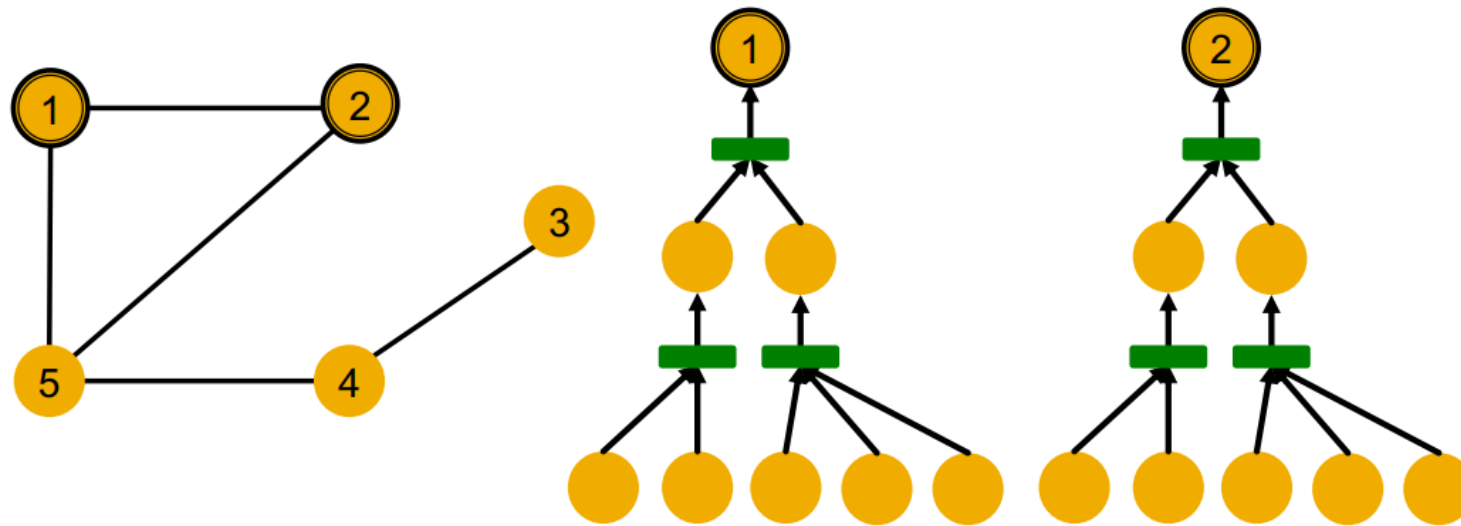
For example: Nodes 1 and 2's computational graphs.



But GNN only sees node features (not IDs)

- A GNN will generate the same embedding for nodes 1 and 2 because:
  - Computational graphs are the same.
  - Node features (colors) are identical.

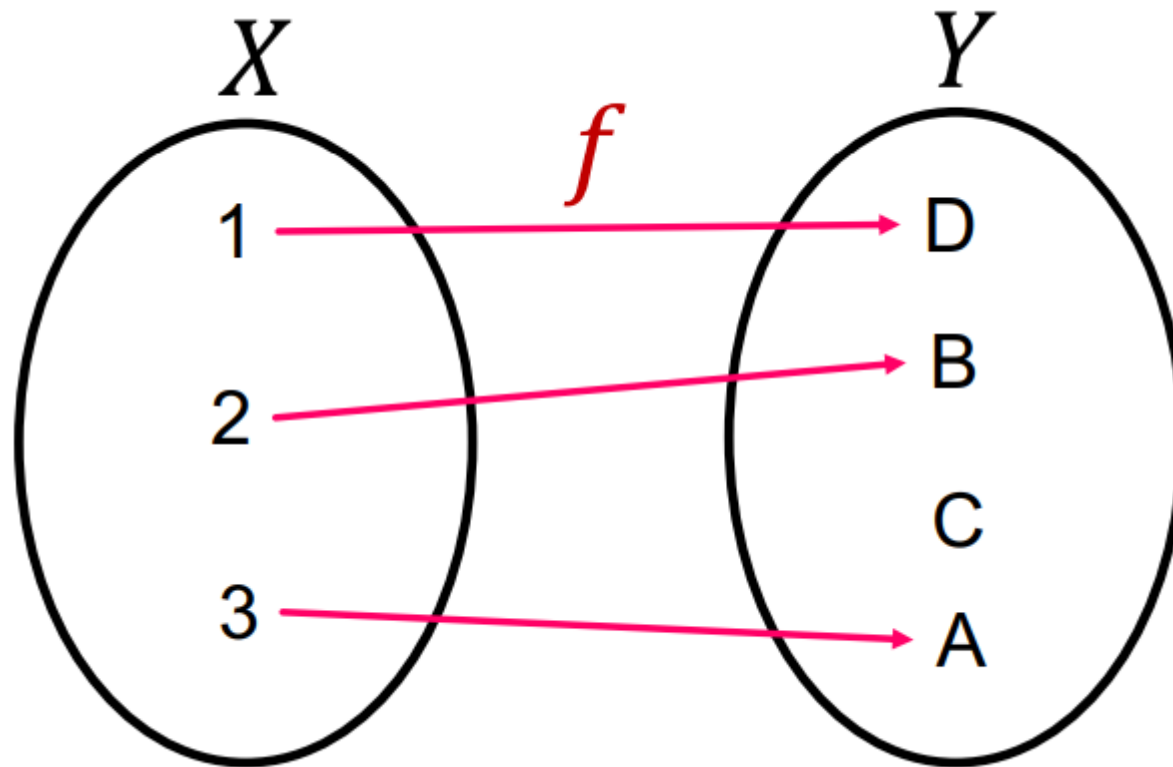
**Note:** GNN does not care about node ids, it just aggregates features vectors of different nodes.



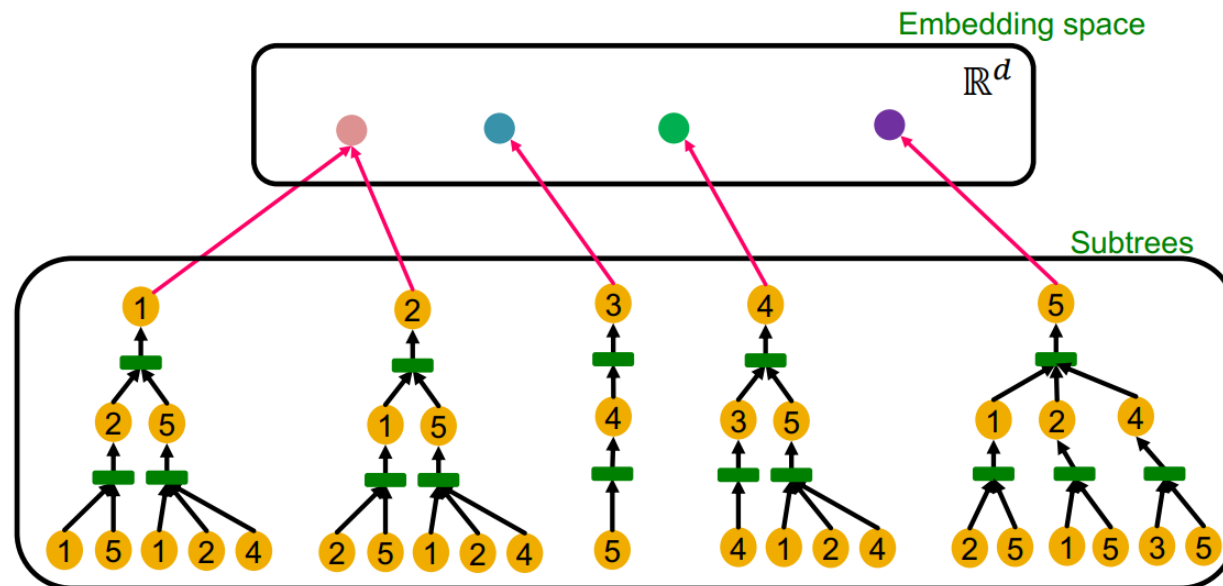
GNN won't be able to distinguish nodes 1 and 2



- Function  $f: X \rightarrow Y$  is injective if it maps different elements into different outputs.
- Intuition:  $f$  retains all the information about input.

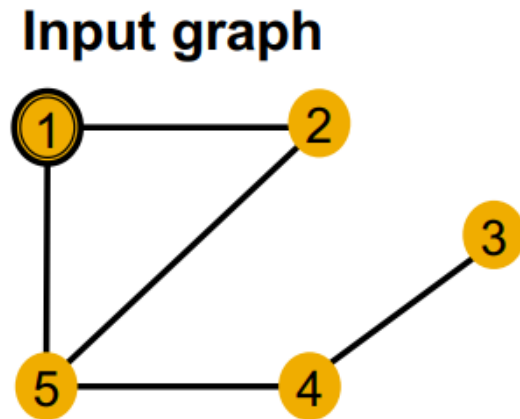


- Most expressive GNN should map subtrees to the node embeddings injectively.
  - **Subtrees represent local structure:** The neighborhood of a node, often viewed as a rooted subtree, contains structural and feature-based information. Two non-isomorphic graphs can have different subtrees around their nodes.
  - **Injective mapping of subtrees:** If a GNN is able to map these subtrees injectively, meaning that different subtrees get different embeddings, the GNN can distinguish more complex graph structures and capture richer relationships between nodes.

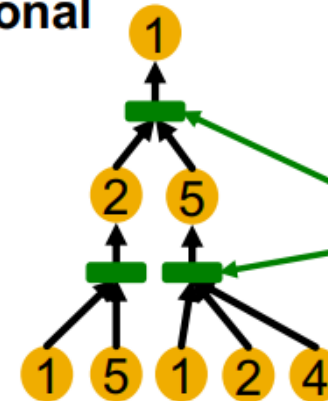


## Summary:

- To generate a node embedding, GNNs use a computational graph corresponding to a subtree rooted around each node.



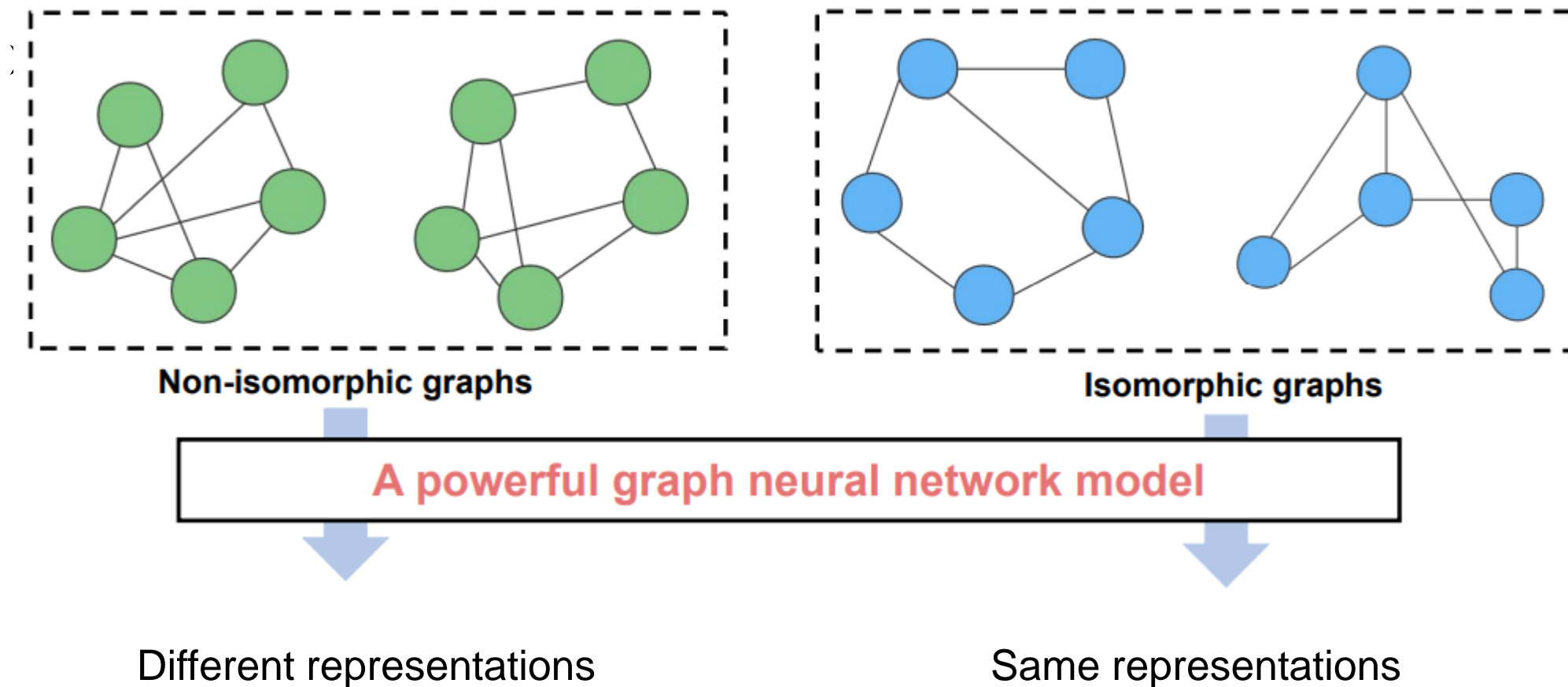
**Computational graph**  
= **Rooted subtree**



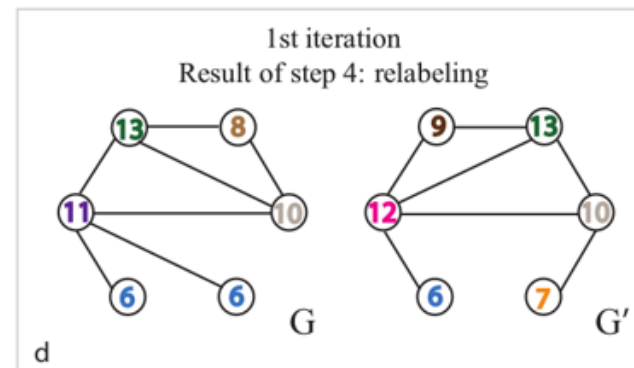
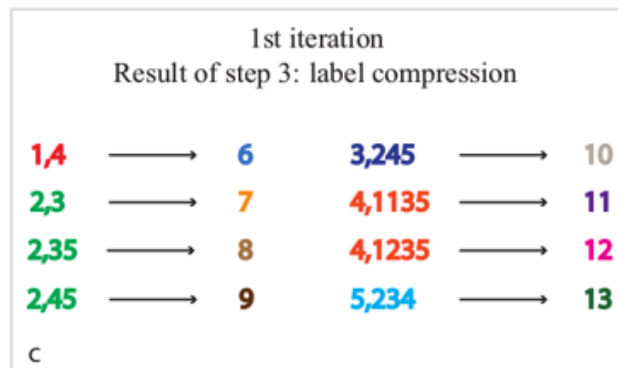
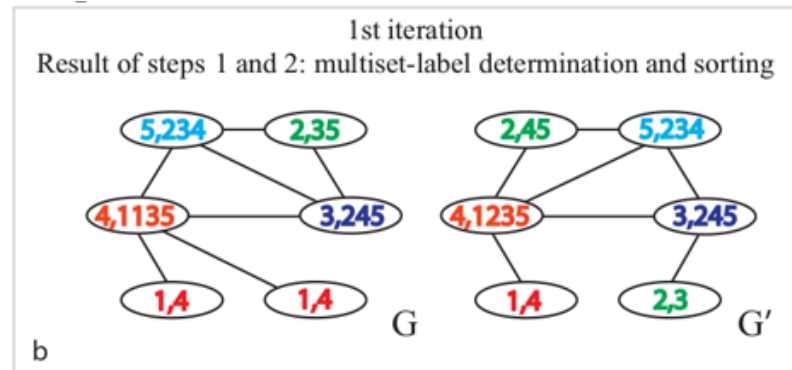
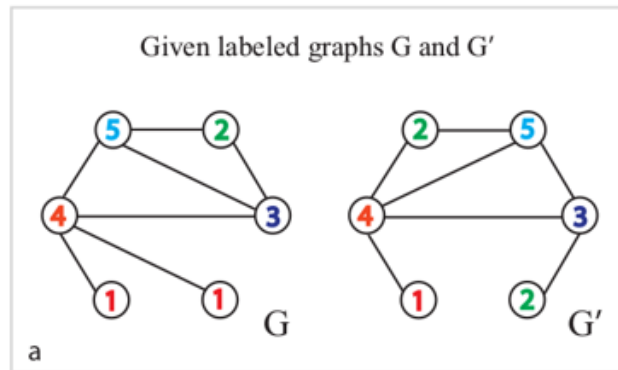
Using injective  
neighbor  
aggregation  
→ distinguish  
different  
subtrees

- GNN can fully distinguish different subtree structures if every step of its neighbor aggregation is injective.

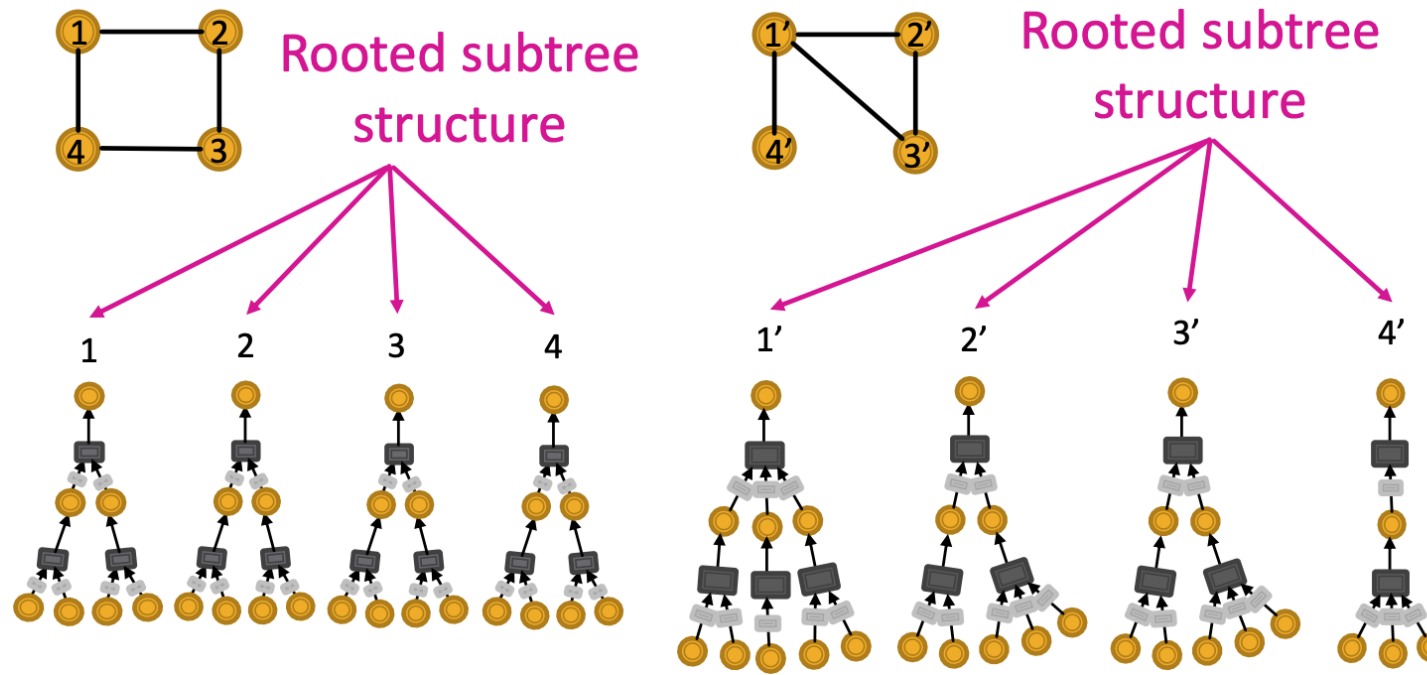
- A GNN model can distinguish the difference between two graphs



- **Looking back to Isomorphism test:** The algorithm determines that two graphs are non-isomorphic if the labels of nodes between the two graphs differ in some iterations.
- Asks whether two graphs are topologically identical

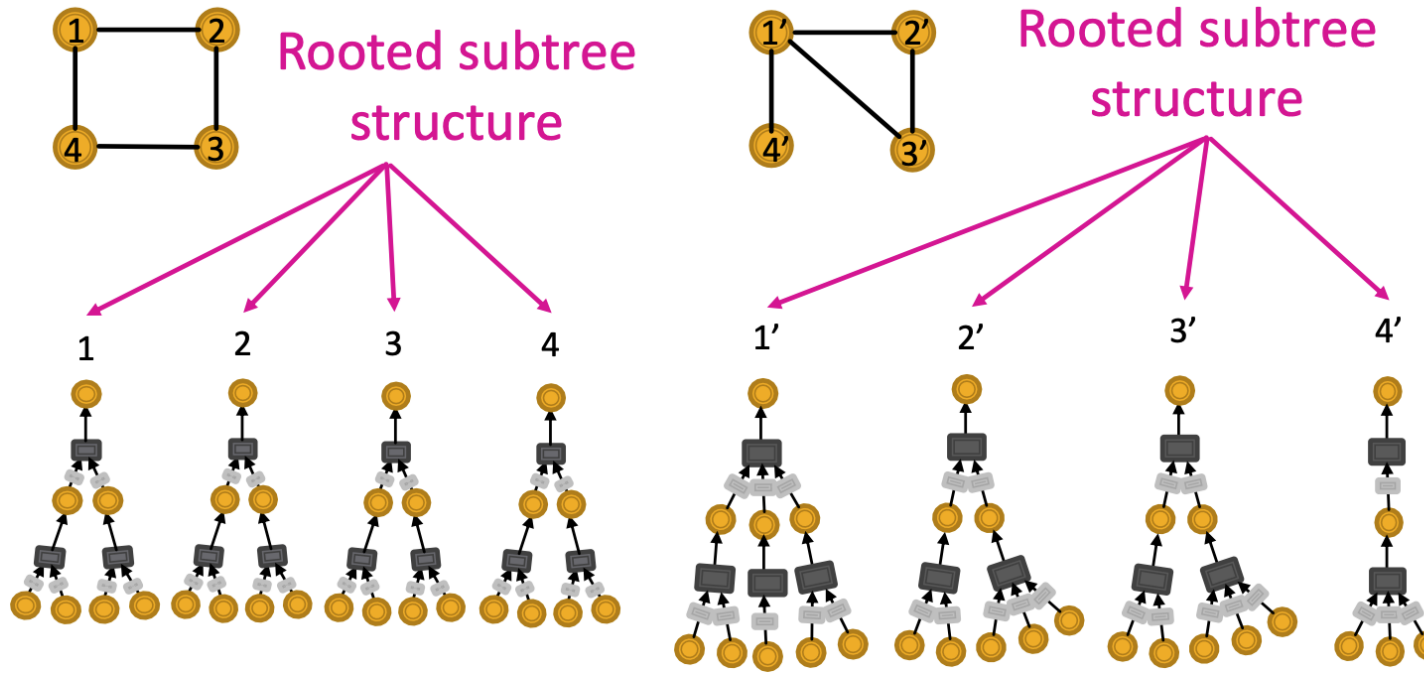


- A subtree pattern of height 2 rooted at the node 1.
- Note the repetitions of nodes in the unfolded subtree pattern on the right
  - The representation ability of GNN is studied by studying when GNN maps two nodes to the same feature representation.
  - The theoretical upper limit of the representation ability is that when the subtrees corresponding to the two nodes are the same, they are mapped to the same feature representation.

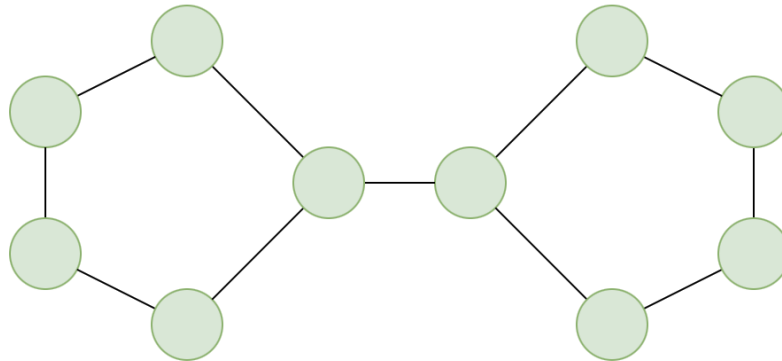
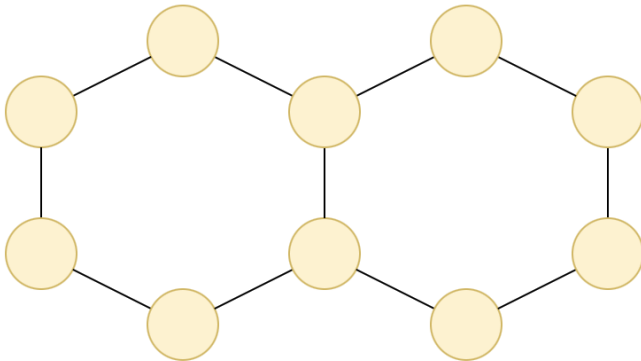




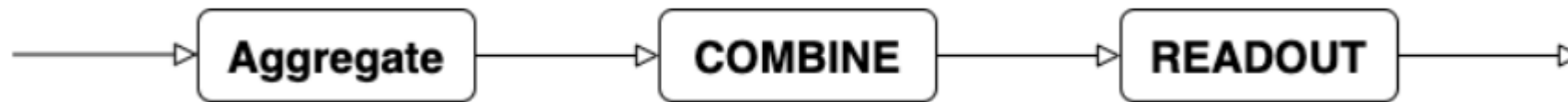
- Represent the set of feature vectors of a given node's neighbors as a multiset
- A maximally powerful GNN would never map two different neighborhoods i.e., multisets of feature vectors, to the same representation.
  - Injective aggregation scheme
  - Isomorphic graphs have to be mapped to the same representation



- Most GNNs fail to distinguish the difference between two graphs
- Examples of two graphs indistinguishable by the WL test.
  - They produce similar distributions through the iterations of colour refinement.
  - It's catastrophic if datasets have graphs that exhibit similar properties and can't be told apart.



- When every function in between is injective, the output function is injective as well:



- How can we build such model?

- The Graph Isomorphism Network (GIN) is considered one of the most powerful GNN architectures because it is designed to be as expressive as the Weisfeiler-Lehman (WL) test
- Consider a multiset  $\mathcal{X}$ , and  $\epsilon$  to be any number
- Assume a function  $g(c, X)$ , where  $c, X \in \mathcal{X}$ , which maps each pair of inputs to a unique number
- There exists an  $f$ , such that for some function  $\varphi$ ,  $g$  can be decomposed as follows:

$$g(c, X) = \varphi((1 + \epsilon)f(c) + \sum_{x \in X} f(x))$$

- Model the  $f^{(k+1)} \circ \phi^{(k)}$  with one MLP:

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

$\epsilon$  can be a learnable parameter or a scalar

- **READOUT:**
  - More iterations gives better representational power
  - But less generalization
  - So GIN concatenates the embedding (information) from all layers

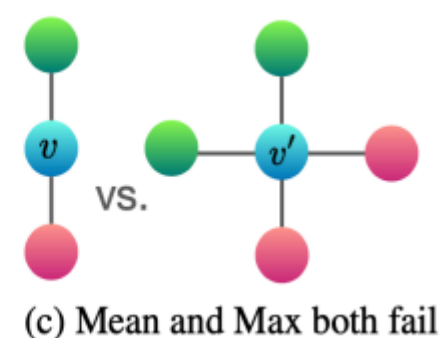
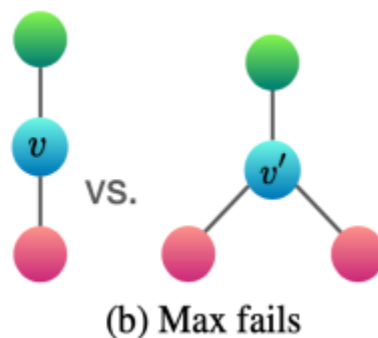
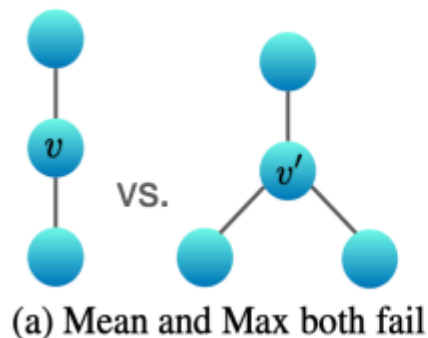
- 1-layer perceptron instead of MLP
- Linear mapping can map two multisets to the same representations
- Unlike models using MLPs, the 1-layer perceptron (even with the bias term) is not a universal approximator of multiset functions.

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

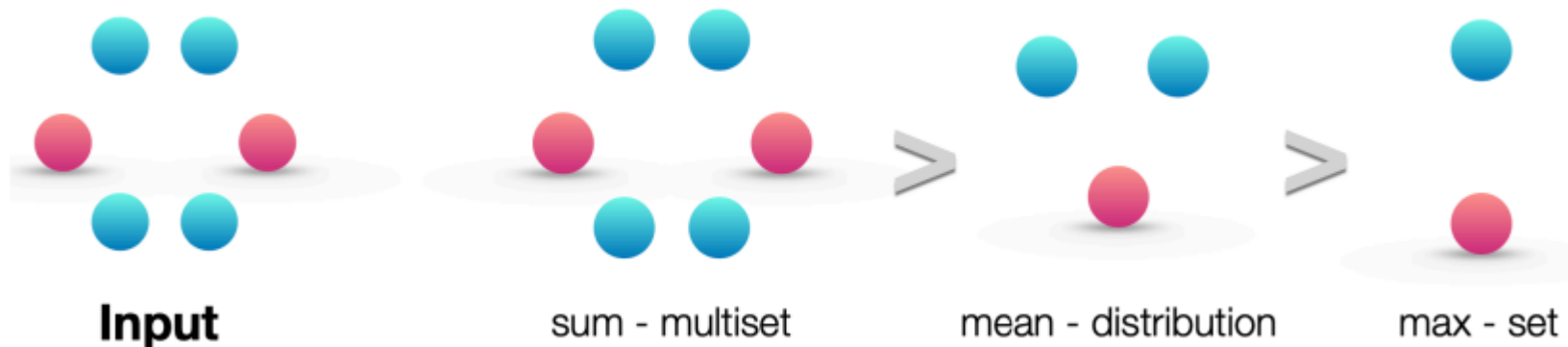
Use an MLP with more than 1 layer



- The mean captures the distribution (proportions) of elements in a multiset, but not the exact multiset
- Max can capture the skeleton



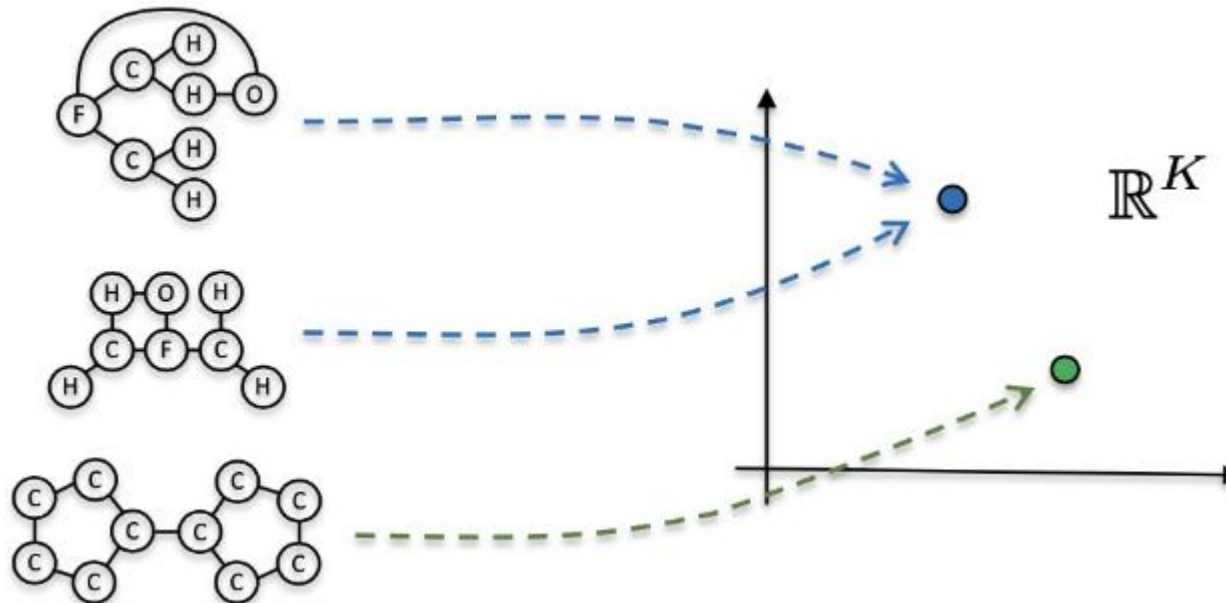
- Instead of sum in  $h(X) = \sum_{x \in X} f(x)$ , what if we use mean or max?



- GNNs are at most as powerful as the WL test in distinguishing graph structures
- Conditions on aggregation and the readout function to be as powerful as WL test

$$h_G = \text{MLP}\left(\sum_{i \in V} h_i^L\right) \in \mathbb{R}^K$$

Sum function



[🏠](#) / [torch\\_geometric.nn](#) / [conv.GINConv](#)

## conv.GINConv

```
class GINConv ( nn: Callable, eps: float = 0.0, train_eps: bool = False, **kwargs ) \[source\]
```

Bases: `MessagePassing`

The graph isomorphism operator from the “How Powerful are Graph Neural Networks?” paper

$$\mathbf{x}'_i = h_{\Theta} \left( (1 + \epsilon) \cdot \mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} \mathbf{x}_j \right)$$

or

$$\mathbf{X}' = h_{\Theta} ((\mathbf{A} + (1 + \epsilon) \cdot \mathbf{I}) \cdot \mathbf{X}),$$

```
61  class GIN(torch.nn.Module):
62      def __init__(self, num_features, num_classes, dim=16, drop=0.5):
63          super(GIN, self).__init__()
64
65          nn1 = Sequential(Linear(num_features, dim), ReLU(), Linear(dim, dim))
66          self.conv1 = GINConv(nn1)
67          self.bn1 = torch.nn.BatchNorm1d(dim)
68
69          nn2 = Sequential(Linear(dim, dim), ReLU(), Linear(dim, num_classes))
70          self.conv2 = GINConv(nn2)
71          self.bn2 = torch.nn.BatchNorm1d(num_classes)
72
73          self.drop = torch.nn.Dropout(p=drop)
74
75      def forward(self, x, edge_index):
76          x = F.relu(self.conv1(x, edge_index))
77          x = self.bn1(x)
78          x = self.drop(x)
79          x = F.relu(self.conv2(x, edge_index))
80          x = self.bn2(x)
81          return F.log_softmax(x, dim=1), x
82
```

➤ Let's try some simple GIN code in the sample code file



네트워크 과학연구실  
NETWORK SCIENCE LAB



가톨릭대학교  
THE CATHOLIC UNIVERSITY OF KOREA

