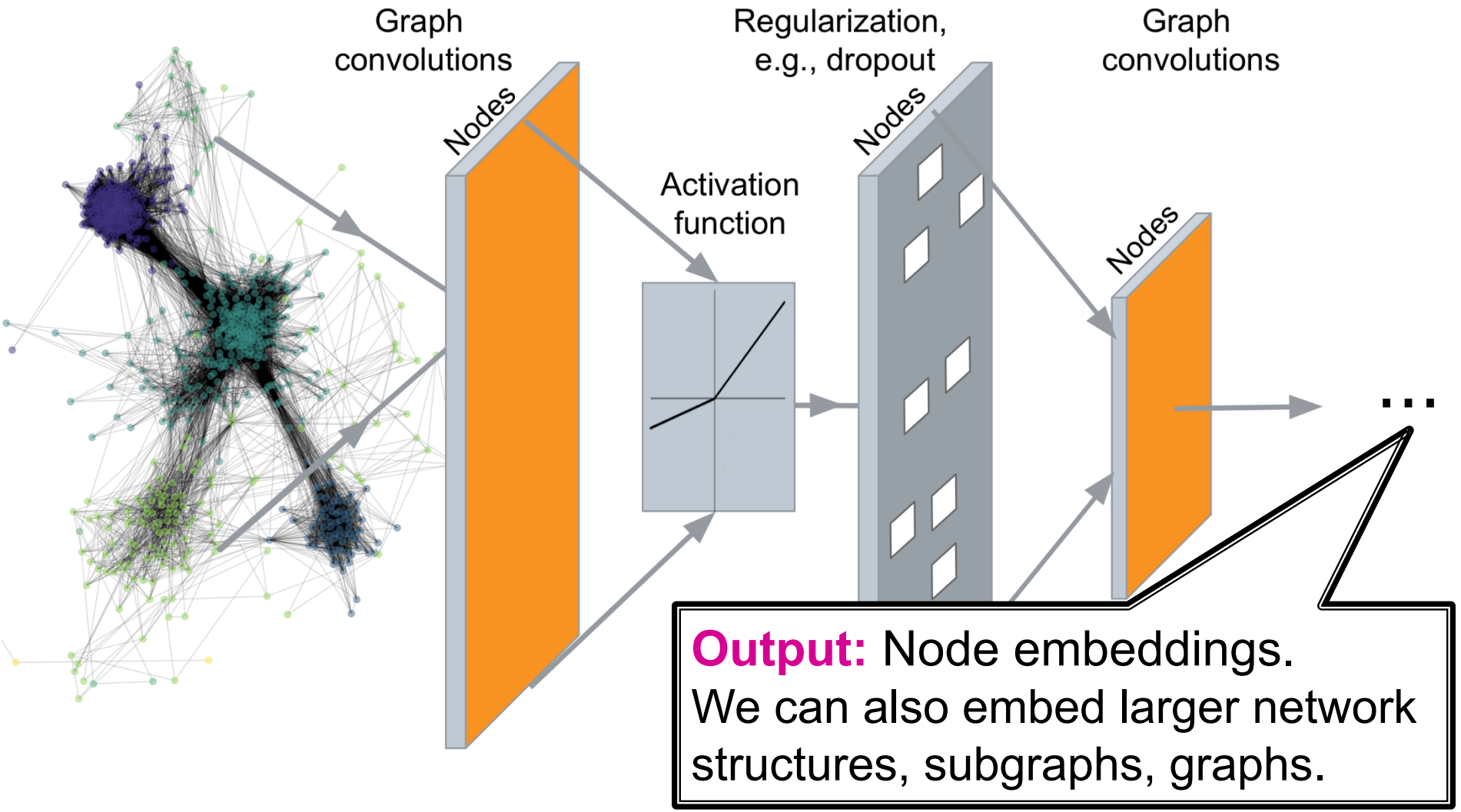
Limitations of Graph Neural Networks

CS224W: Machine Learning with Graphs

Jure Leskovec, Weihua Hu, Stanford University

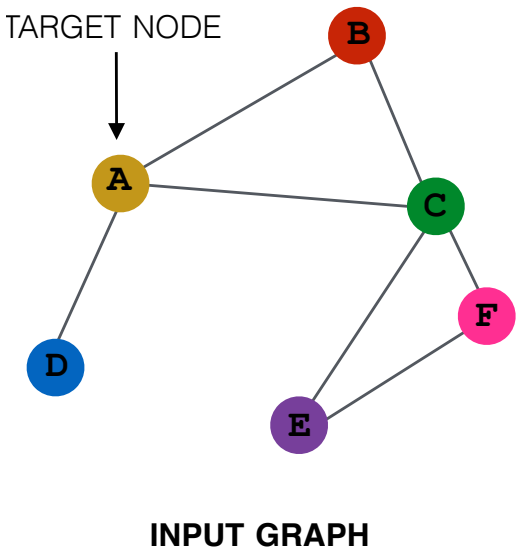
http://cs224w.stanford.edu

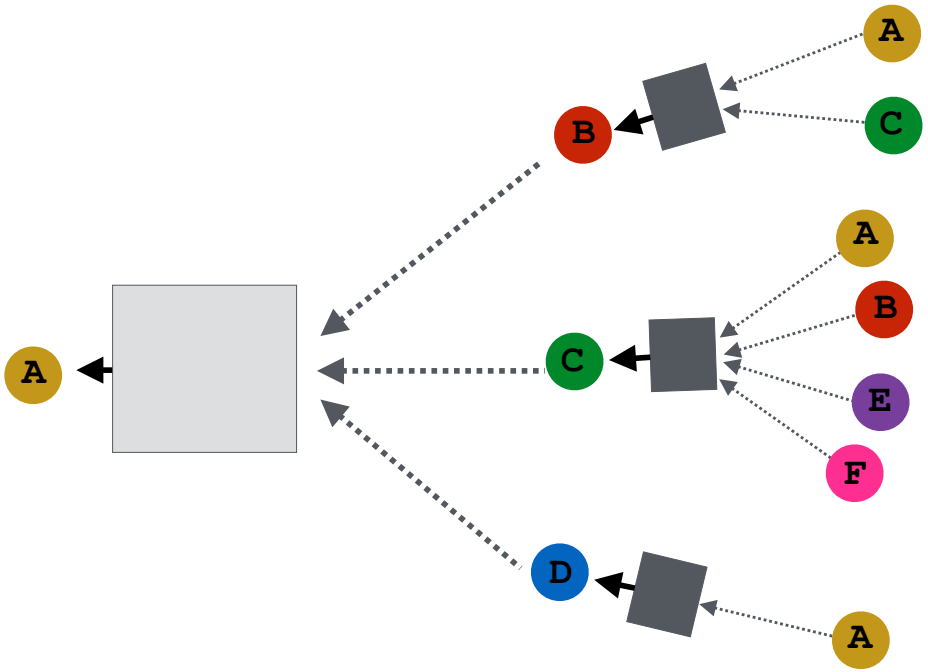
Recap: Graph Neural Networks



Idea: Aggregate Neighbors

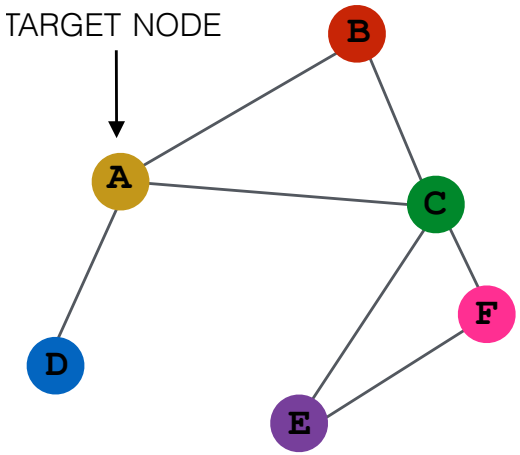
- **Key idea:** Generate node embeddings based on **local network neighborhoods**



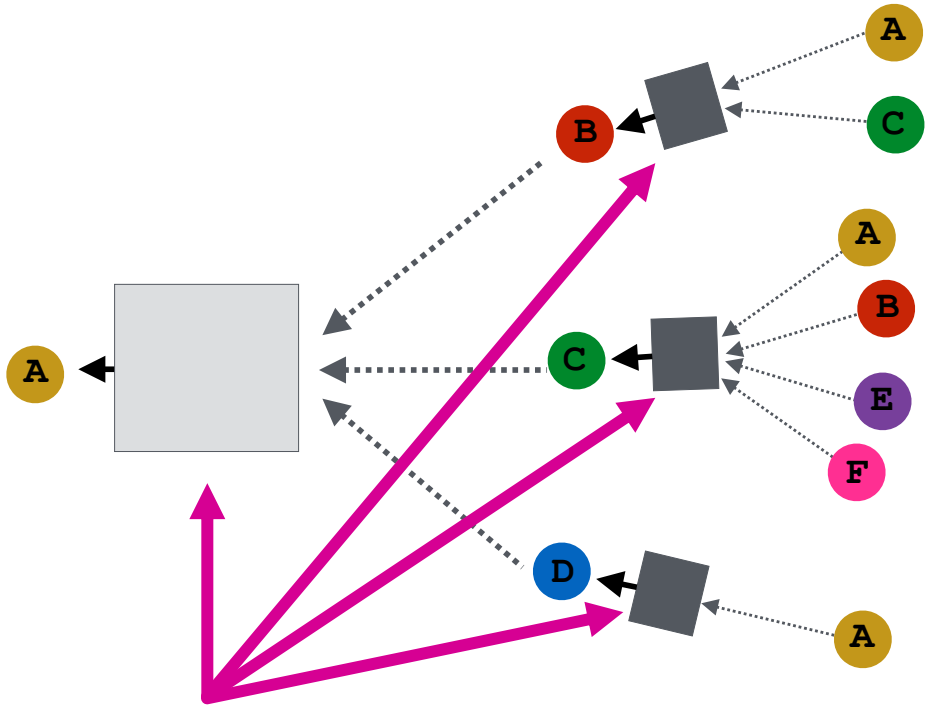


Idea: Aggregate Neighbors

- **Intuition:** Nodes aggregate information from their neighbors using neural networks



INPUT GRAPH

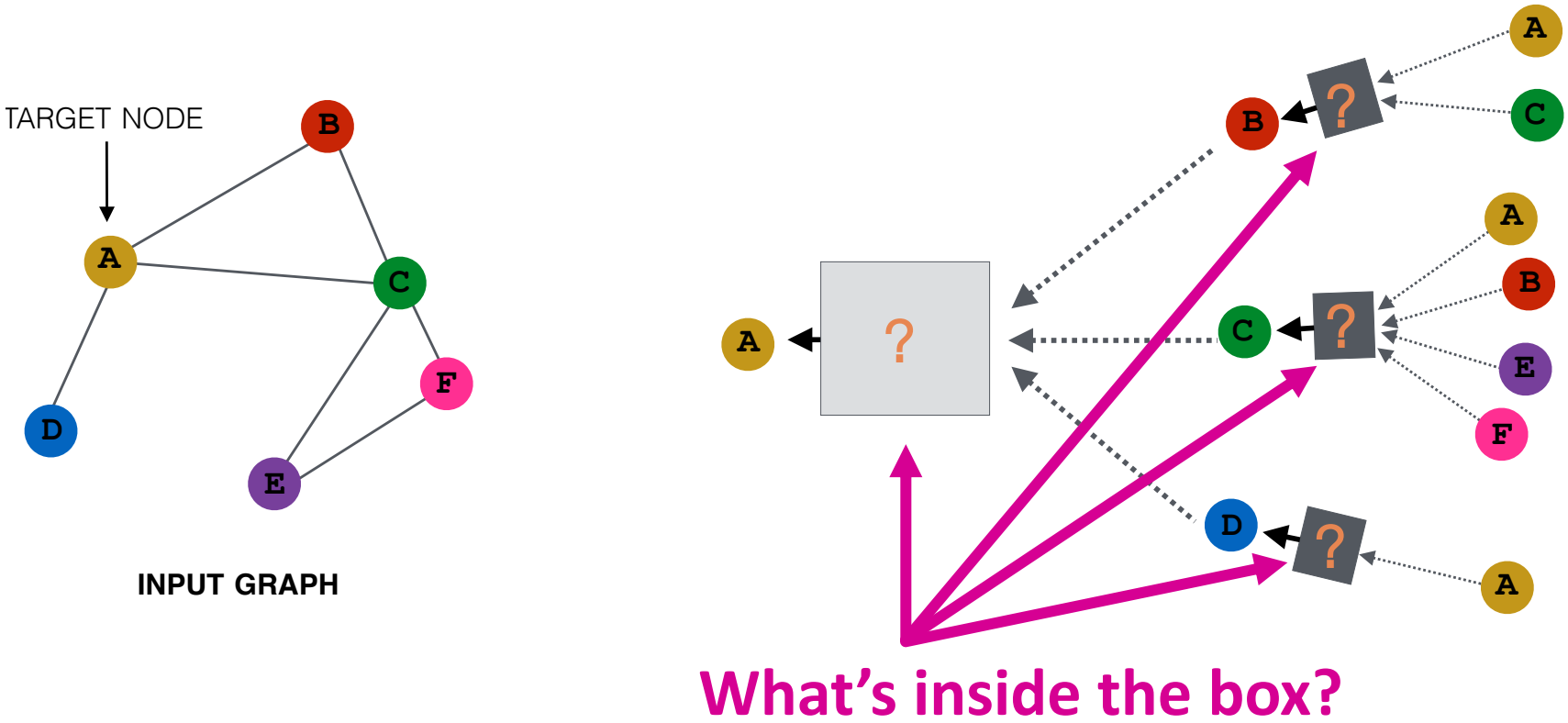


Neural networks

Many GNN Models

- Many model variants have been proposed with difference choice of neural networks.

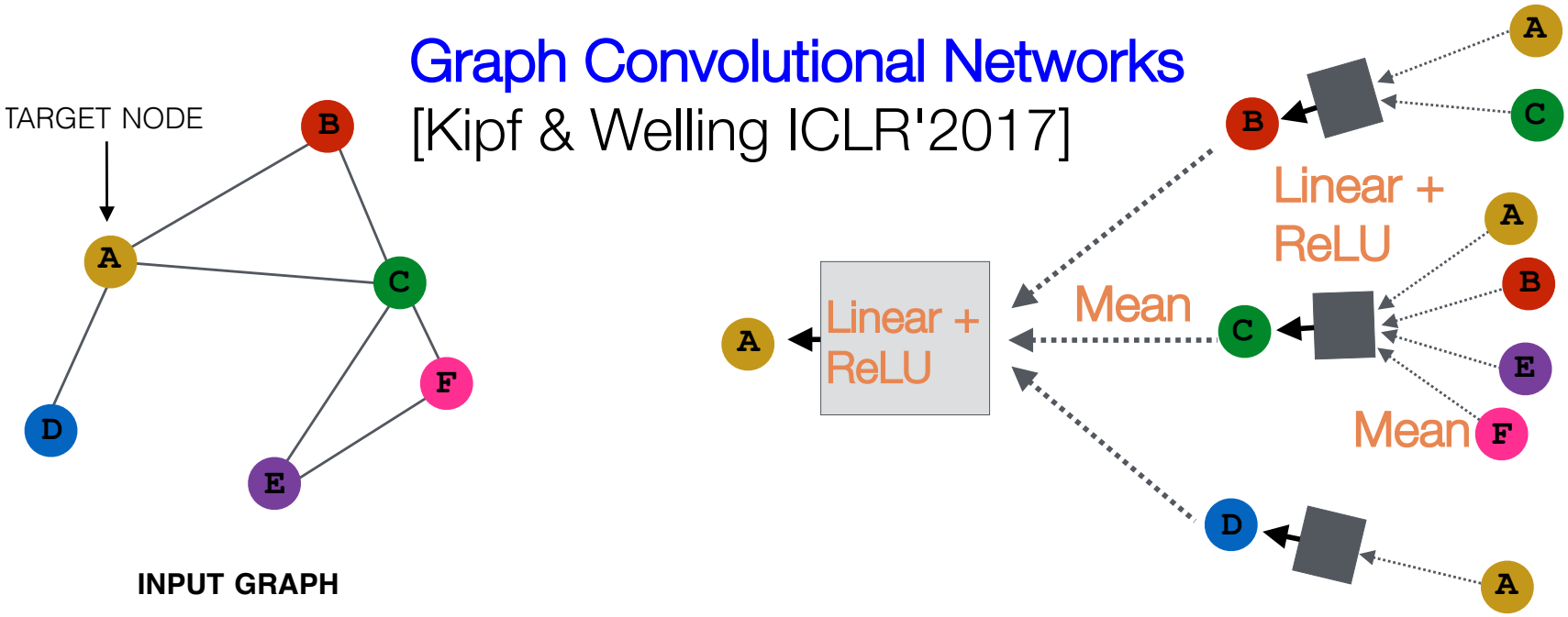
Scarselli et al., 2009b; Battaglia et al., 2016; Defferrard et al., 2016; Duvenaud et al., 2015; Hamilton et al., 2017a; Kearnes et al., 2016; Kipf & Welling, 2017; Lei et al., 2017; Li et al., 2016; Velickovic et al., 2018; Verma & Zhang, 2018; Ying et al., 2018; Zhang et al., 2018



Many GNN Models

- Many model variants have been proposed with difference choice of neural networks.

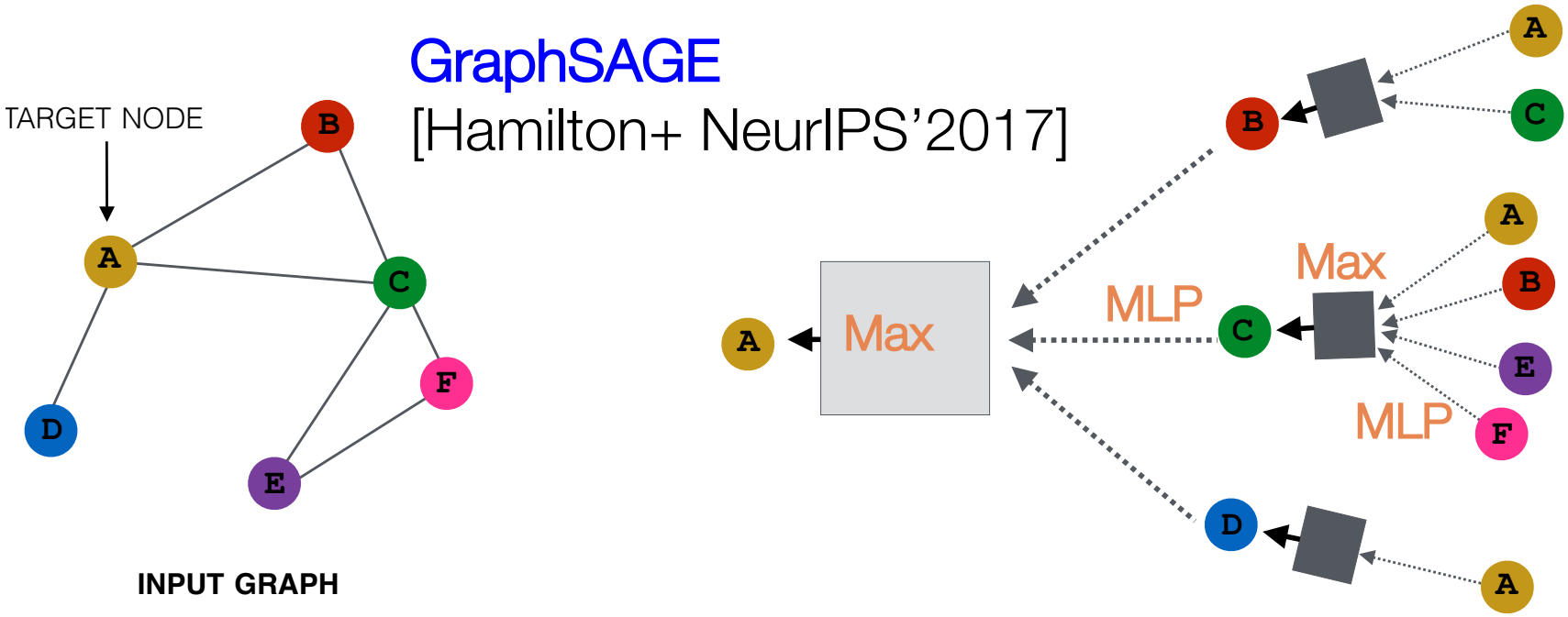
Scarselli et al., 2009b; Battaglia et al., 2016; Defferrard et al., 2016; Duvenaud et al., 2015; Hamilton et al., 2017a; Kearnes et al., 2016; Kipf & Welling, 2017; Lei et al., 2017; Li et al., 2016; Velickovic et al., 2018; Verma & Zhang, 2018; Ying et al., 2018; Zhang et al., 2018



Many GNN Models

- Many model variants have been proposed with difference choice of neural networks.

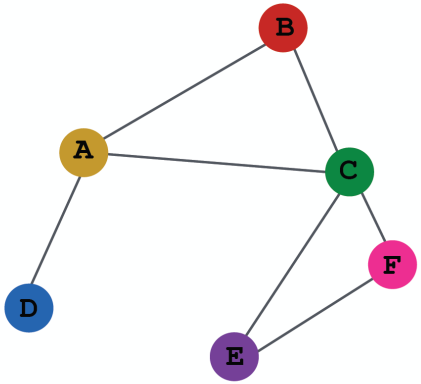
Scarselli et al., 2009b; Battaglia et al., 2016; Defferrard et al., 2016; Duvenaud et al., 2015; Hamilton et al., 2017a; Kearnes et al., 2016; Kipf & Welling, 2017; Lei et al., 2017; Li et al., 2016; Velickovic et al., 2018; Verma & Zhang, 2018; Ying et al., 2018; Zhang et al., 2018



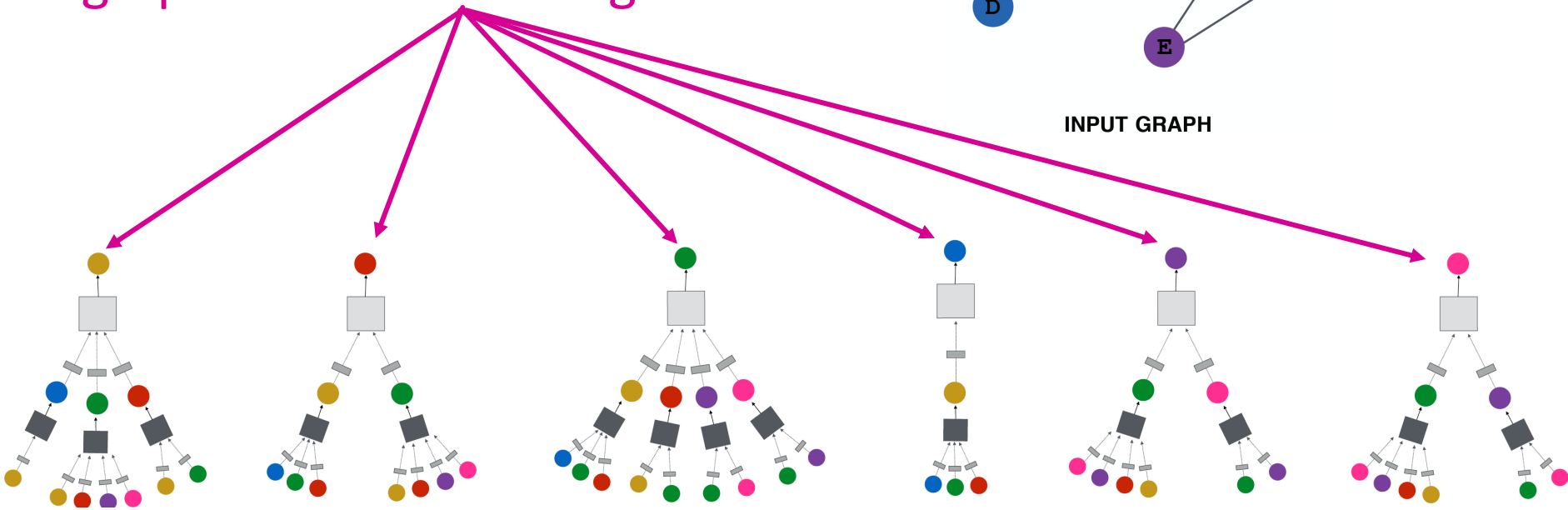
Idea: Computation Graph

- **Intuition:** Network neighborhood defines a computation graph

Every node defines a computation graph based on its neighborhood!

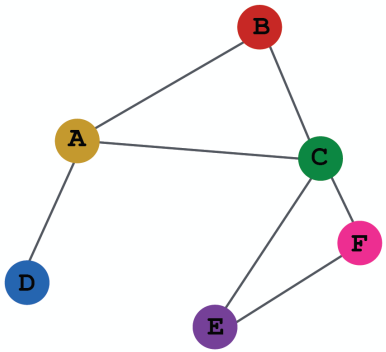


INPUT GRAPH

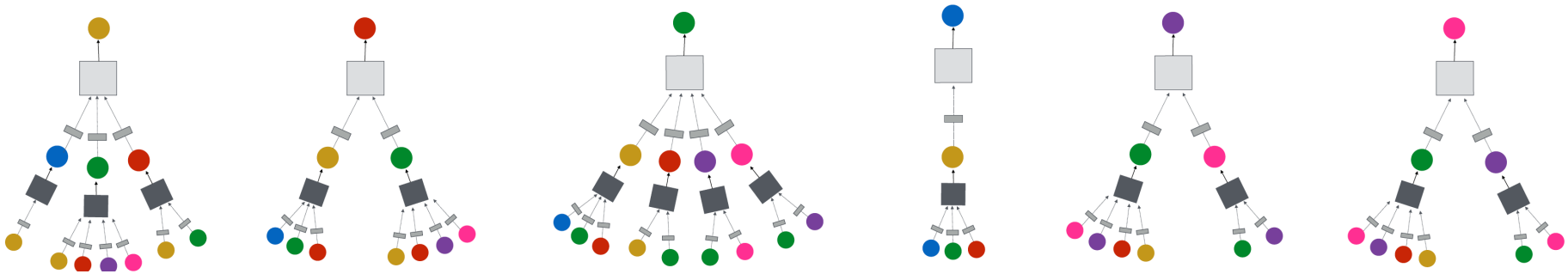


Idea: Aggregate Neighbors

- Obtain **node representation** by neighbor aggregation

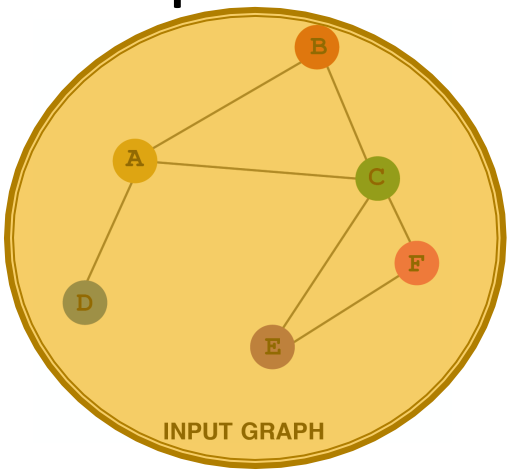


INPUT GRAPH

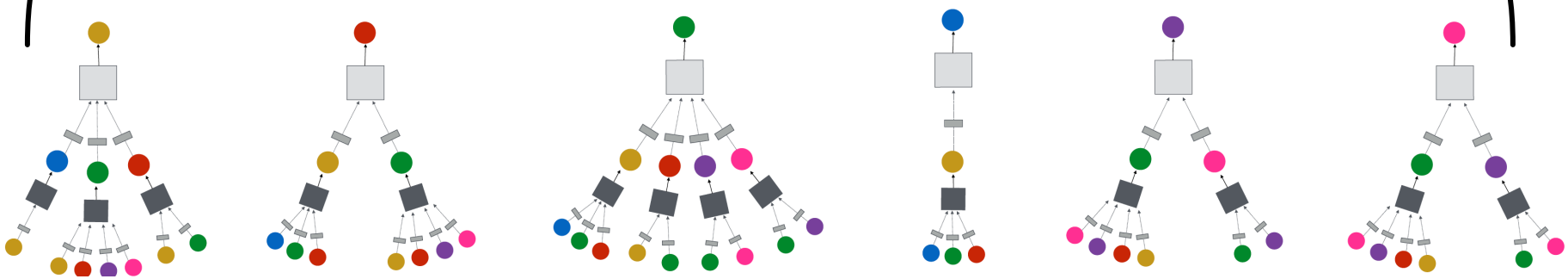


Idea: Graph Pooling

- Obtain **graph representation** by pooling node representation



Pool (e.g., Sum, Average)



Impressive performance

Graph Neural Networks have **achieved state-of-the-art performance** on:

- Node classification [Kipf+ ICLR'2017]
- Graph Classification [Ying+ NeurIPS'2018]
- Link Prediction [Zhang+ NeurIPS'2018]

Impressive performance

Graph Neural Networks have achieved state-of-the-art performance on

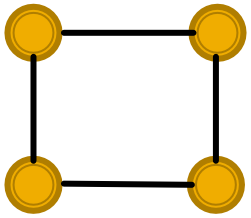
- Node classification [Kipf+ ICLR'2017]
- Graph Classification [Ying+ NeurIPS'2018]
- Link prediction [Zhang+ NeurIPS'2018]

Are GNNs perfect?

What are the limitations of GNNs?

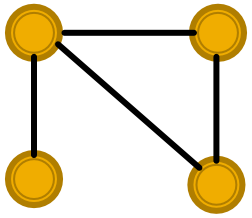
Today: Limitations of GNNs

- Some simple graph structures **cannot be distinguished** by conventional GNNs.



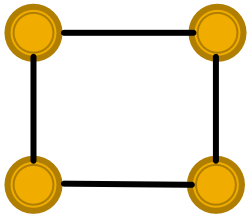
Assume: Input node features are uniform (denoted by the same node color)

GCN and GraphSAGE fail to distinguish the two graphs.



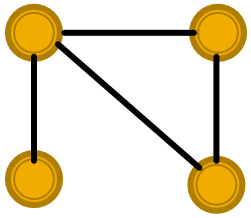
Today: Limitations of GNNs

- Some simple graph structures **cannot be distinguished** by conventional GNNs.



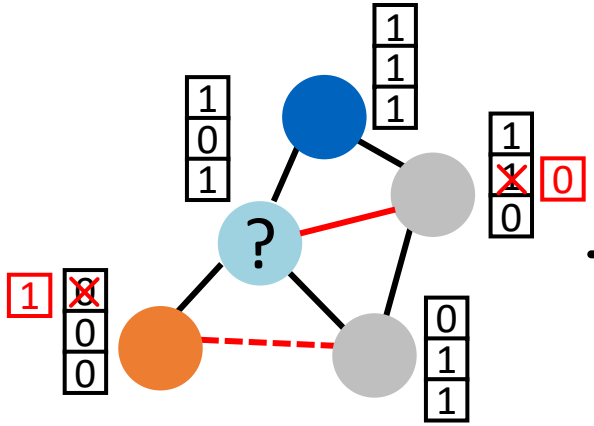
Assume: Input node features are uniform (denoted by the same node color)

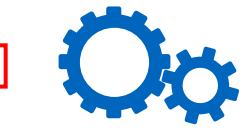
GCN and GraphSAGE fail to distinguish the two graphs.



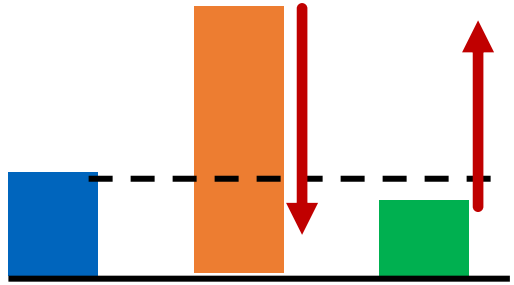
- GNNs are **not robust to noise** in graph data.

Noise in graph
1. Node feature perturbation
2. Edge addition/deletion





GNNs



Outline of Today's Lecture

1. **Limitations of conventional GNNs in capturing graph structure**

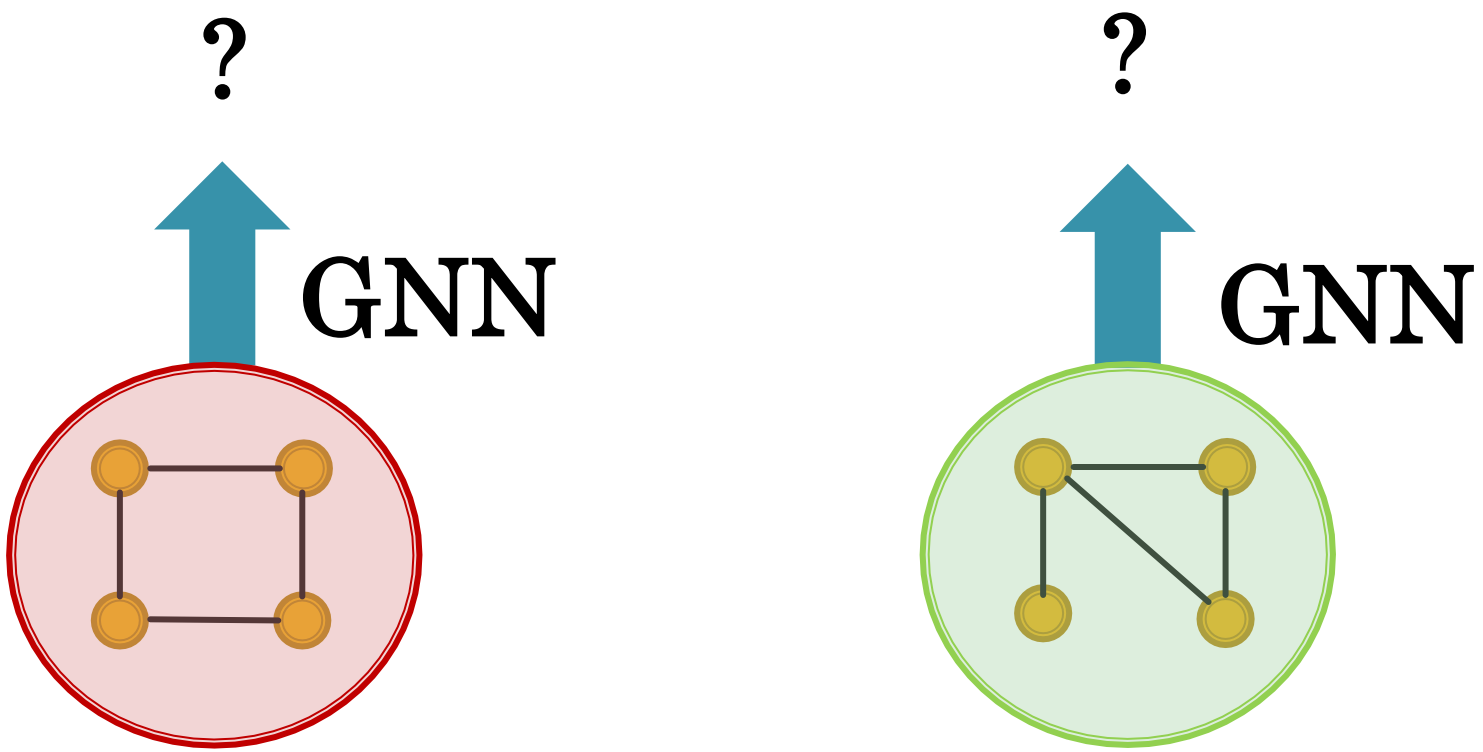
2. **Vulnerability of GNNs to noise in graph data**

3. **Open questions & Future direction**

1. Limitations of conventional GNNs in capturing graph structure

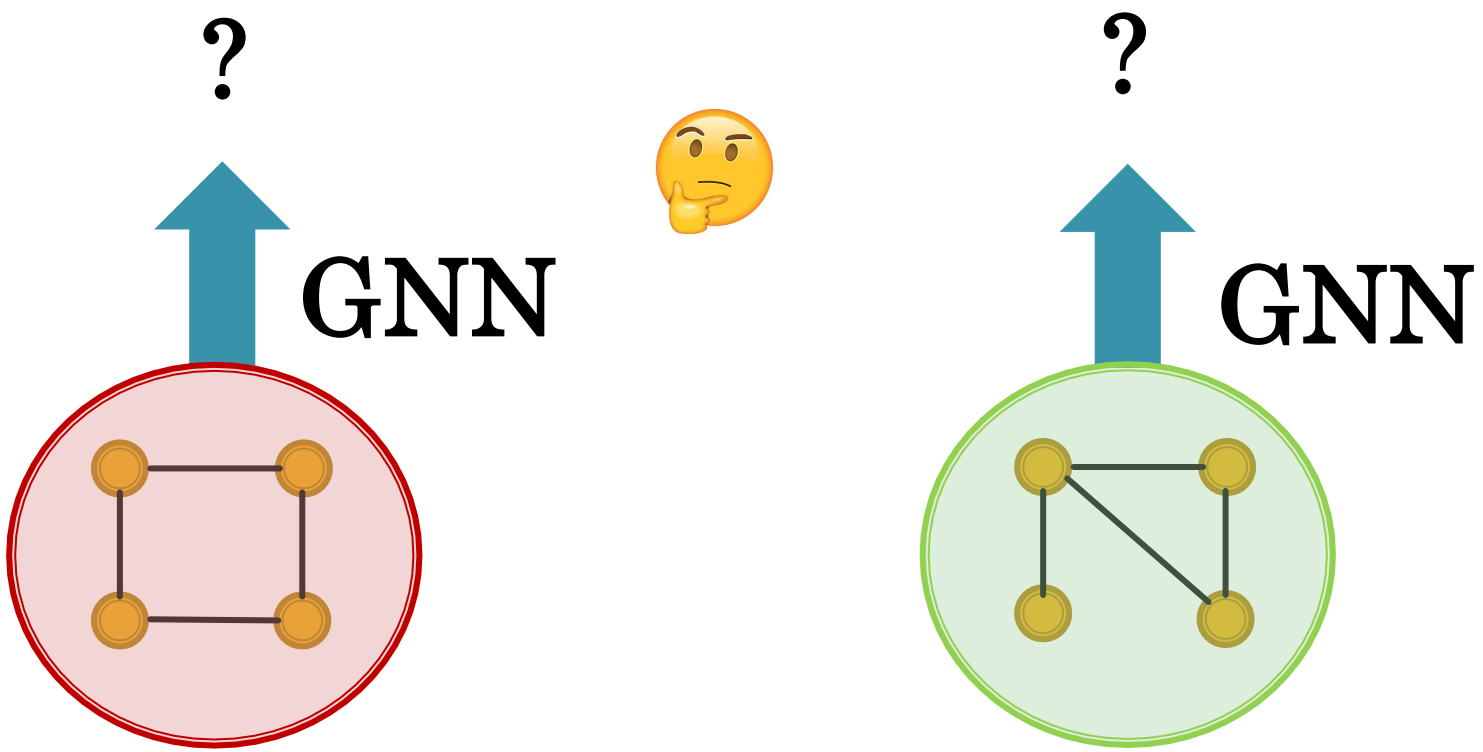
Fundamental question

- Given two different graphs, can GNNs map them into **different graph representations**?
- Important condition for classification scenario.



Graph Isomorphism

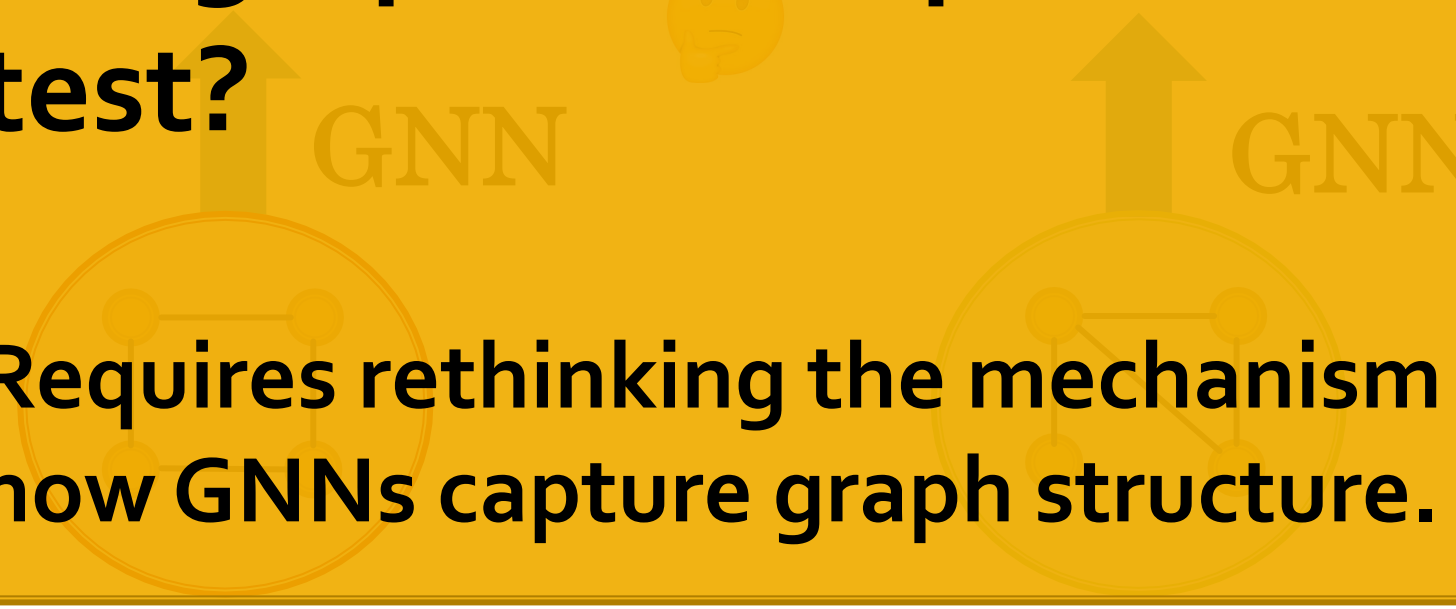
- Essentially, graph isomorphism test problem.
- **No polynomial algorithms** exist for general case.
- GNNs may not perfectly distinguish any graphs!



Graph Isomorphism

- Essentially graph isomorphism test problem.
- **No polynomial algorithms** exist for general case.
- GNNs may not perfectly distinguish any graphs.

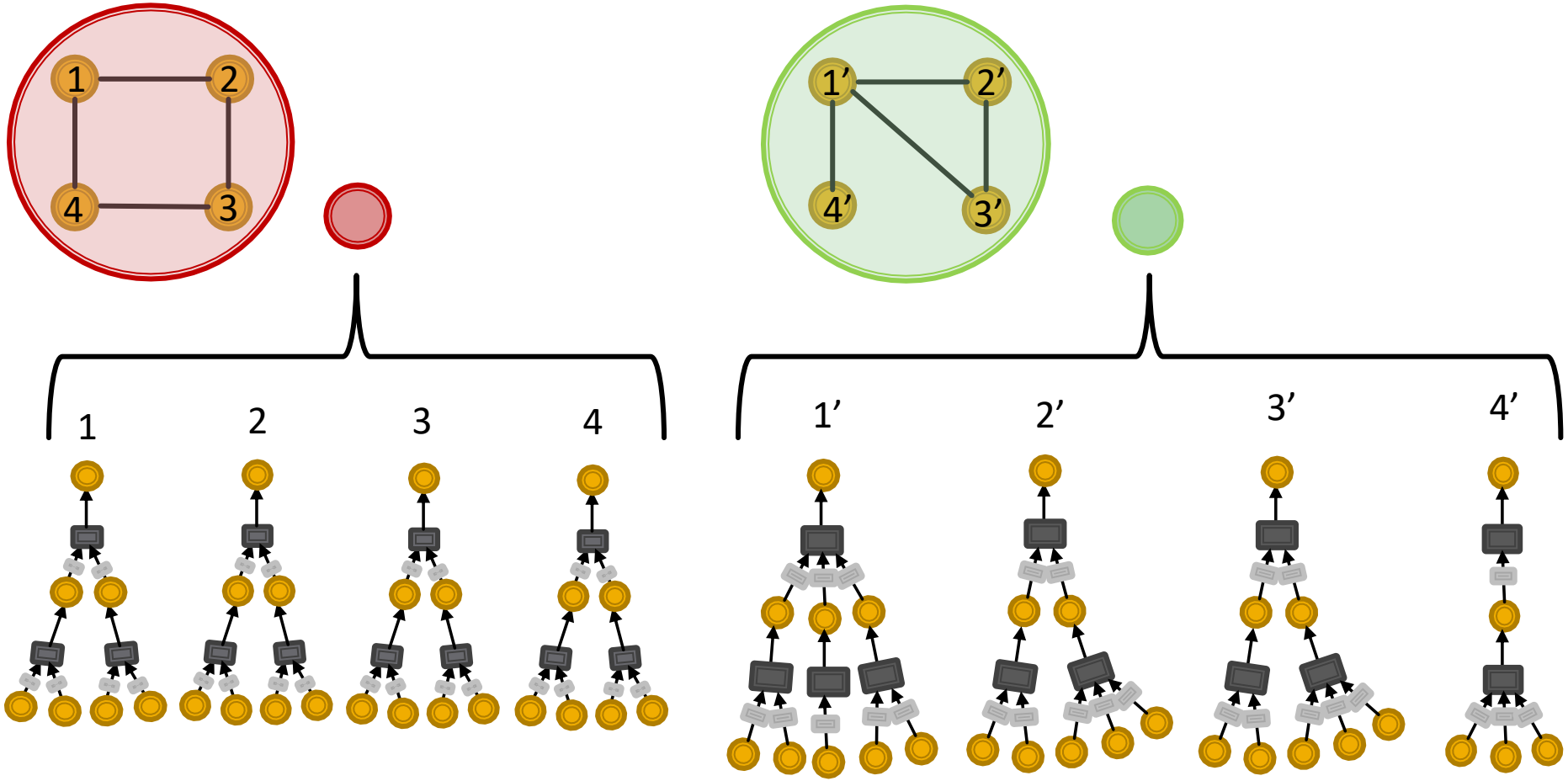
How well can GNNs perform the graph isomorphism test?



Requires rethinking the mechanism of how GNNs capture graph structure.

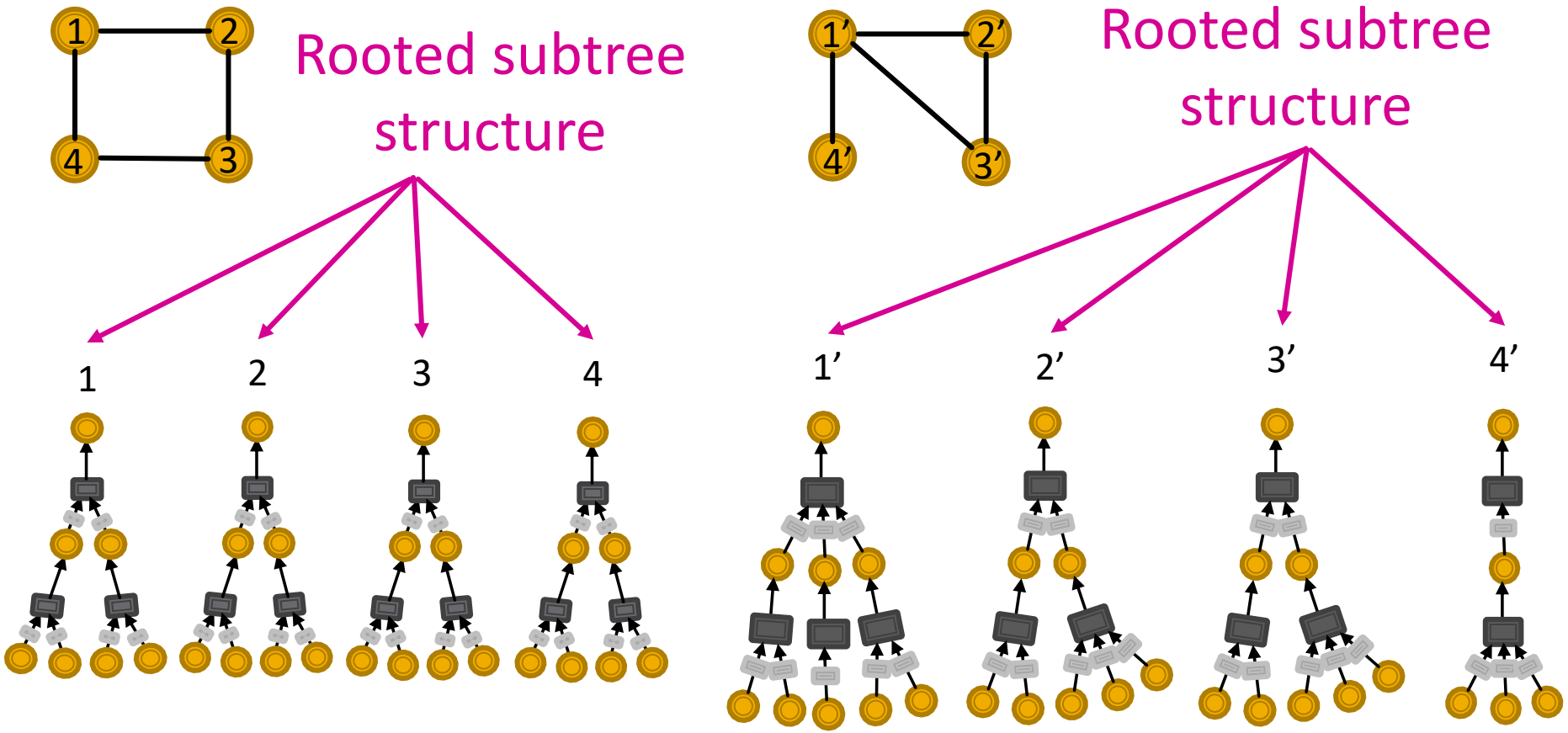
Rethinking GNNs

- GNNs use different computational graphs to distinguish different graphs.



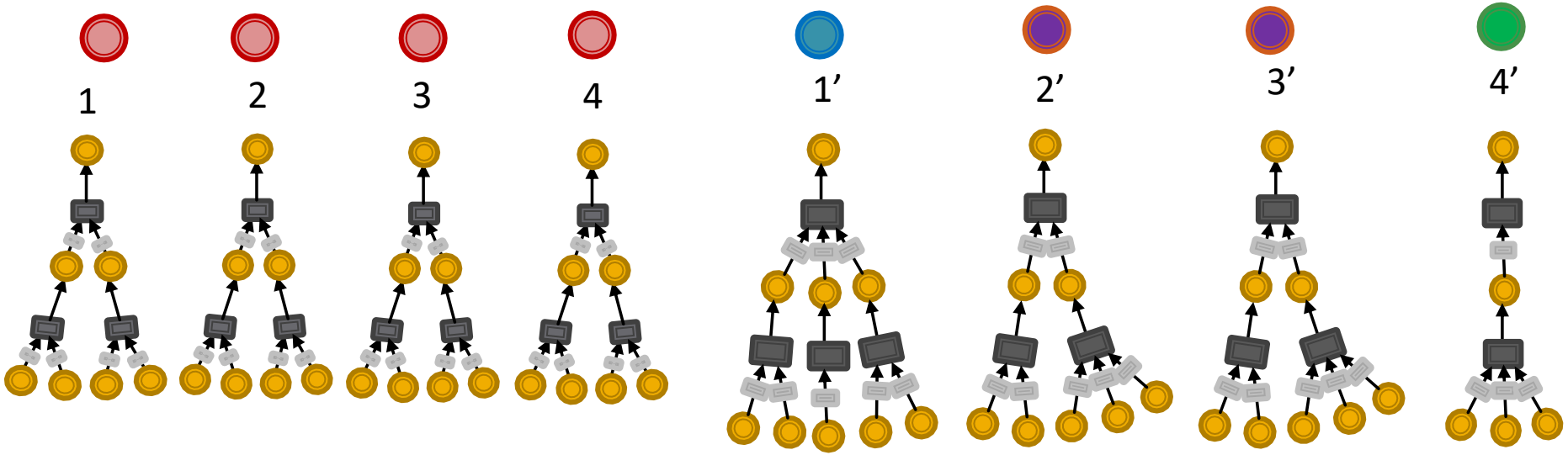
Rethinking GNNs

- Node representation captures rooted subtree structure.



Rethinking GNNs

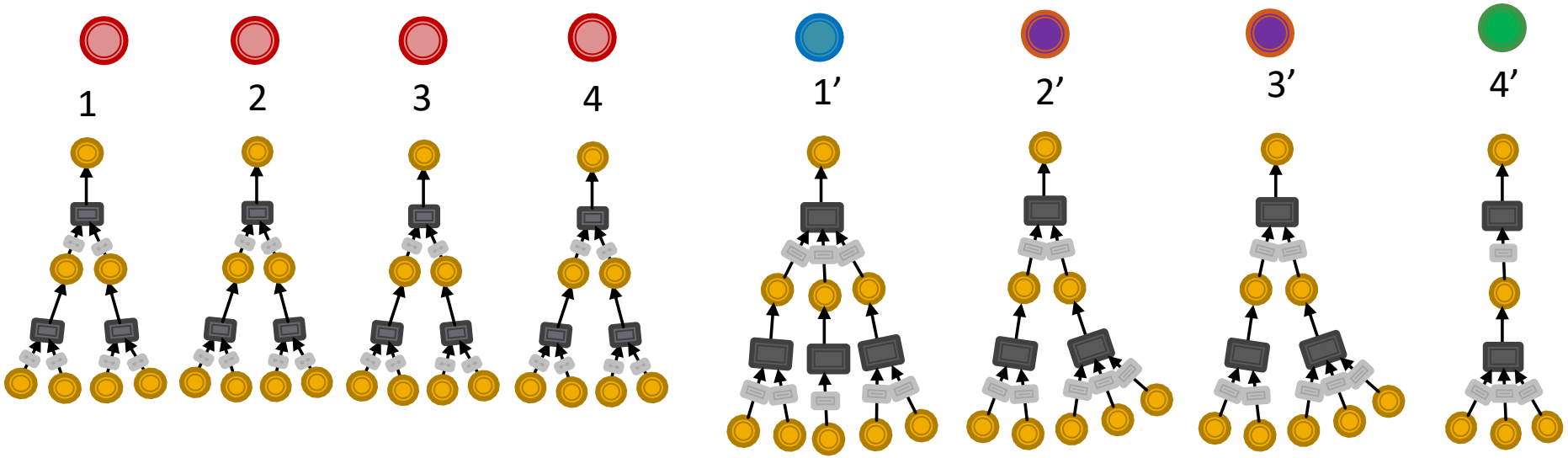
- Most discriminative GNNs map different subtrees into different node representations (denoted by different colors).



Rethinking GNNs

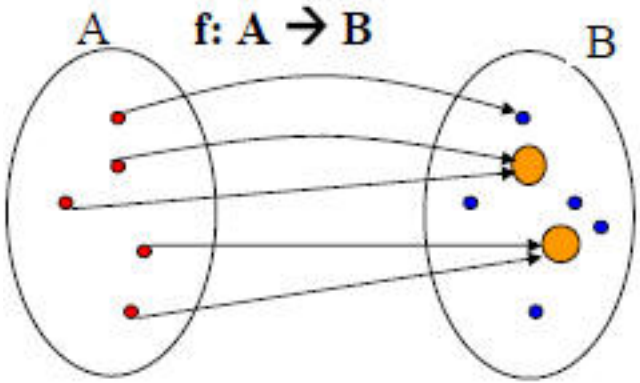
- Most discriminative GNNs map different subtrees into different node representation (denoted by different colors).

Injectivity

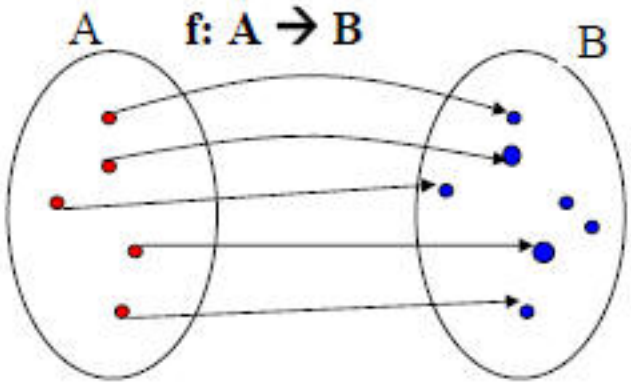


Recall: Injectivity

- Function is **injective** if it maps different elements into different outputs.



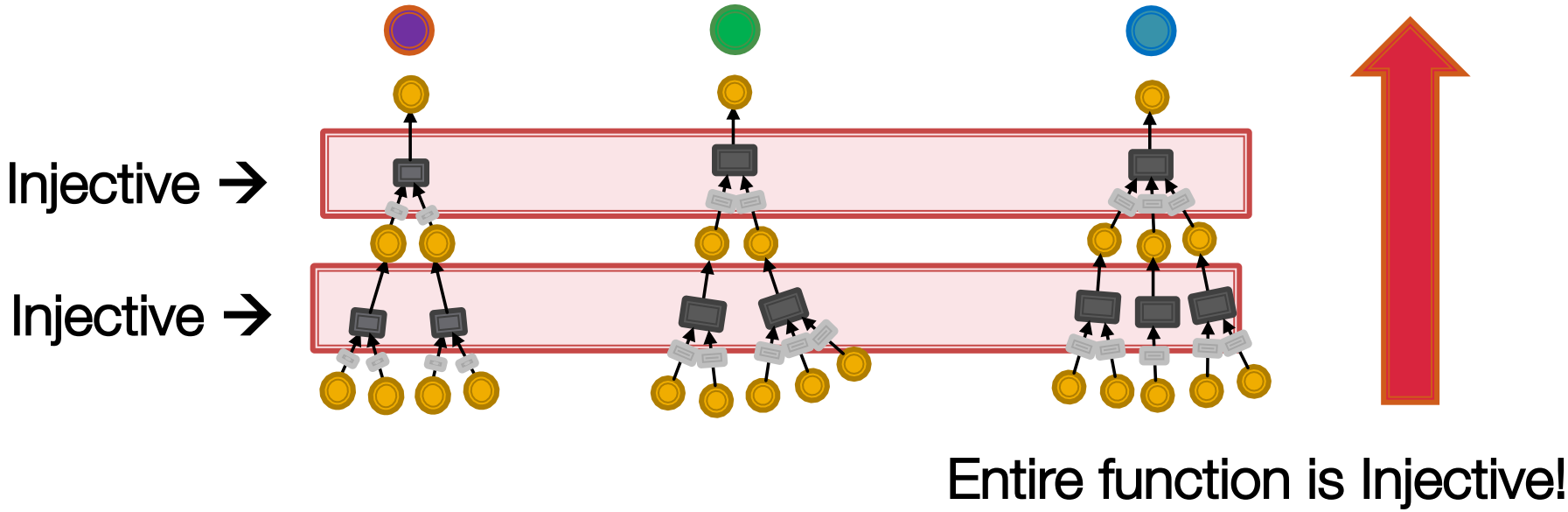
Not injective function



Injective function

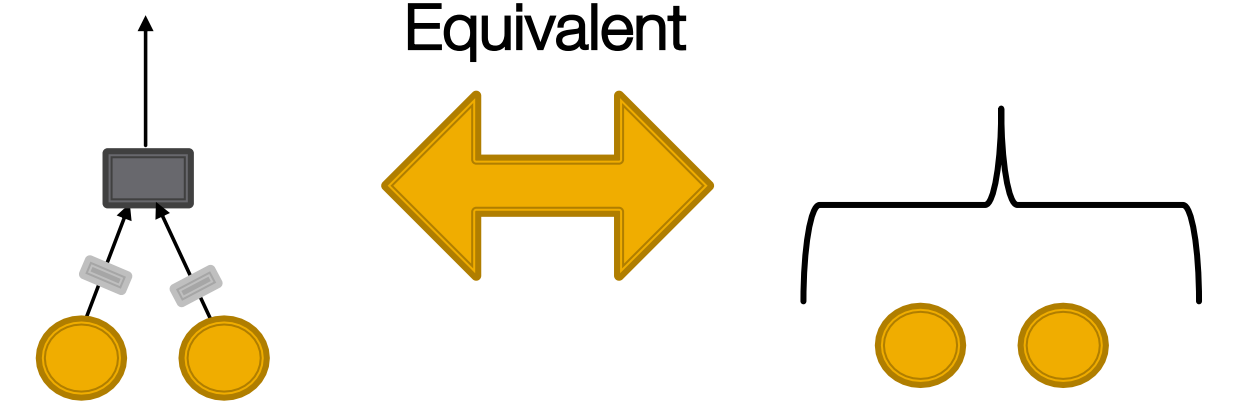
Injective Neighbor Aggregation

- Entire neighbor aggregation is injective if **every step** of neighbor aggregation is injective.

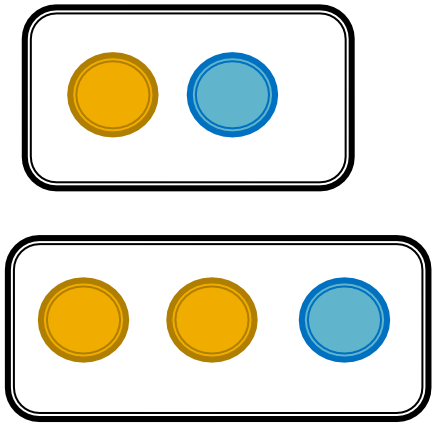


Neighbor aggregation

- Neighbor aggregation is essentially a function over multi-set (set with repeating elements).



Examples of multi-set



Same color indicates the same node features

Neighbor aggregation

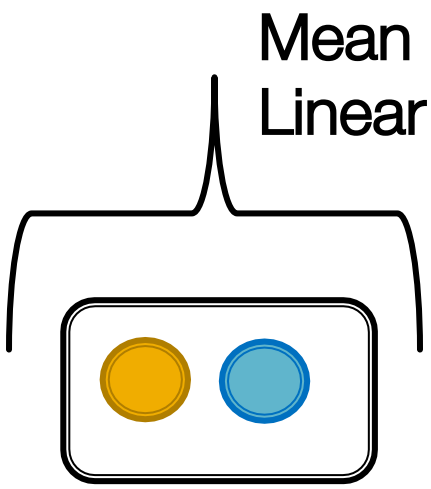
- Neighbor aggregation is essentially function over multi-set (set with repeating elements).

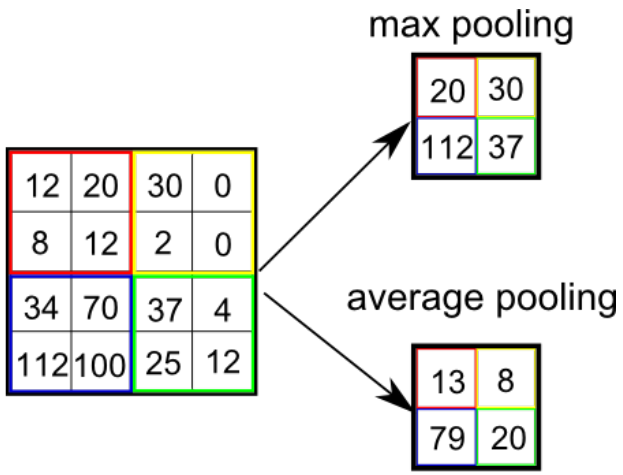
Discriminative Power of GNNs can be characterized by that of multi-set functions

Next: Analyzing GCN, GraphSAGE

Case Study 1: GCN

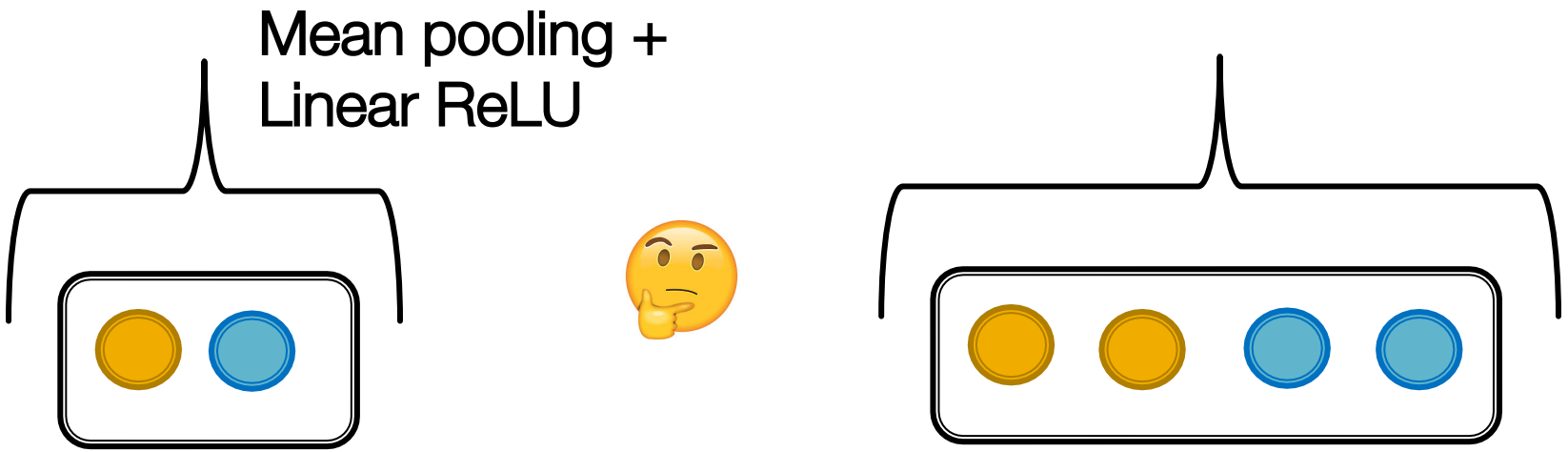
Recall: GCN uses mean pooling.





Case Study 1: GCN

Recall: GCN uses mean pooling.

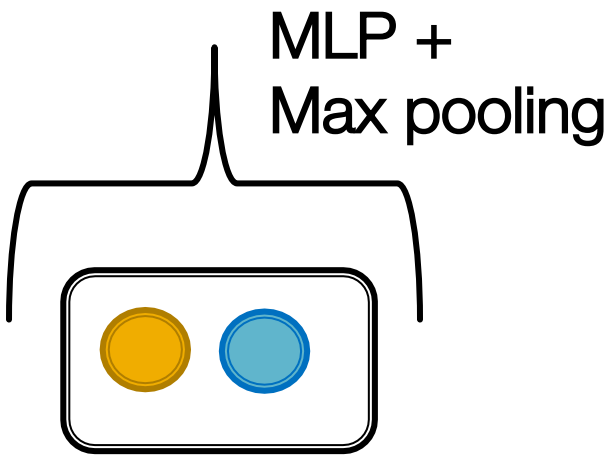


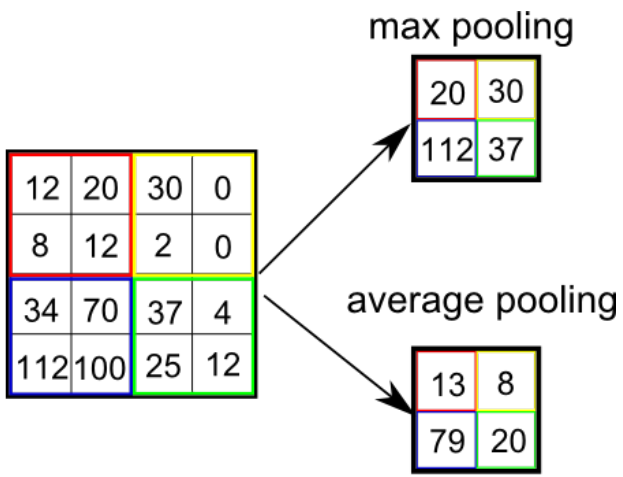
GCN will fail to distinguish proportionally equivalent multi-sets.

Not injective!

Case Study 2: GraphSAGE-maxpool

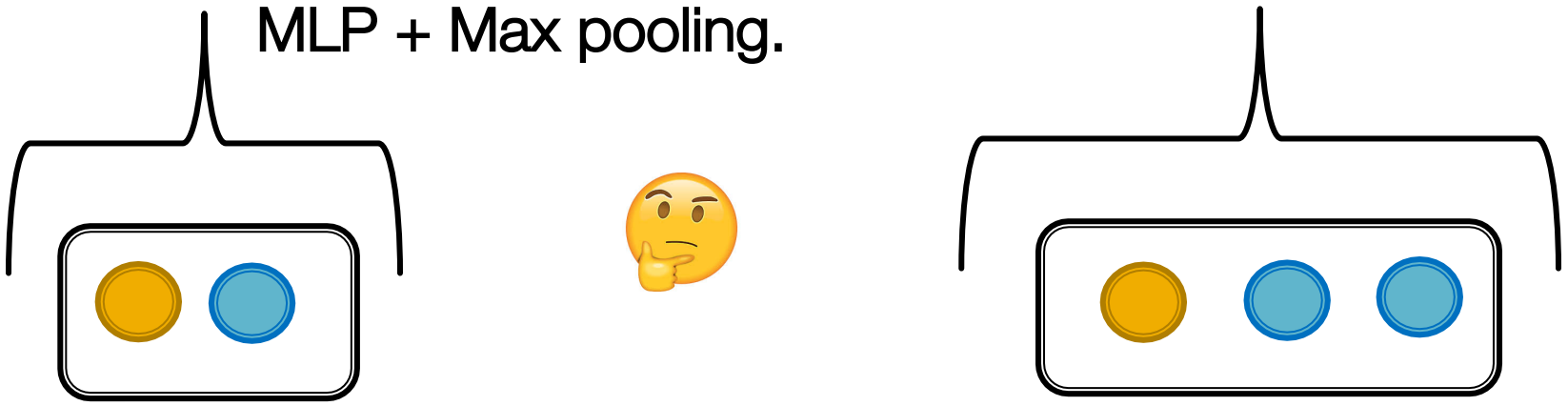
Recall: GraphSAGE uses max pooling.





Case Study2: GraphSAGE-maxpool

Recall: GraphSAGE uses max pooling.



GraphSAGE will even fail to distinguish multi-set with the same distinct elements.

Not injective!

Case Study2: GraphSAGE-maxpool

Recall: GraphSAGE uses max pooling.

How can we design injective multi-set function using neural networks?

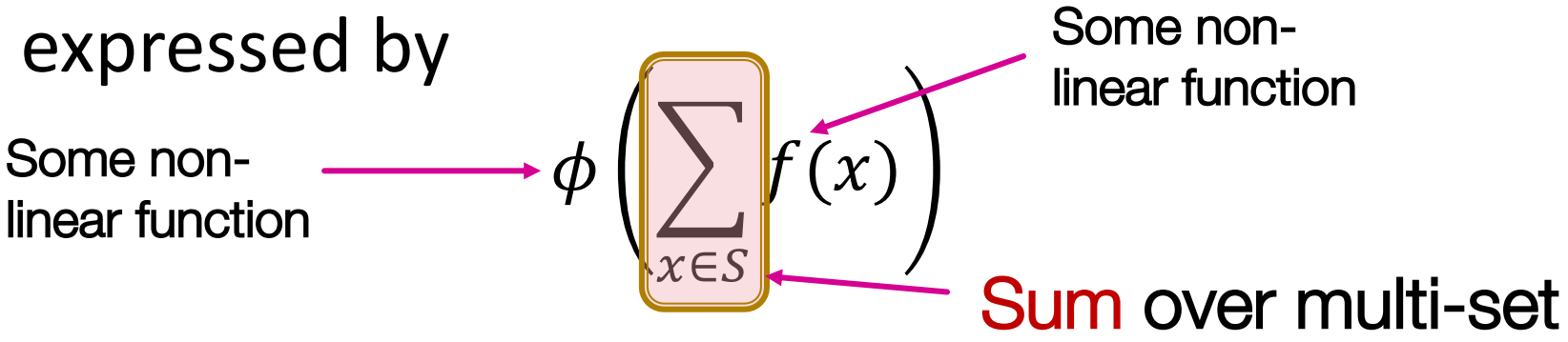
GCN will even fail to distinguish multi-set with the same distinct elements.

Not injective!

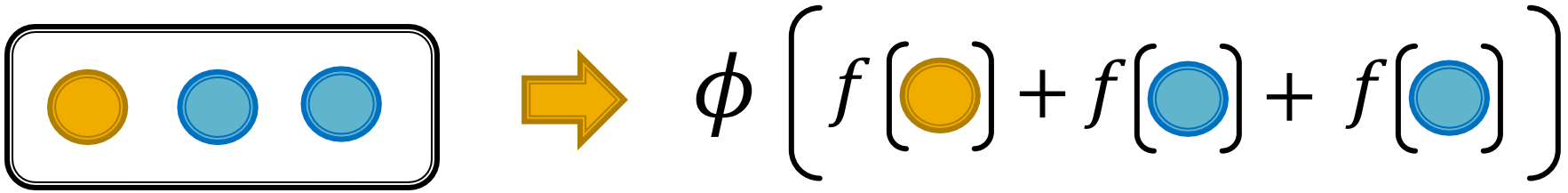
Injective multi-set function

Theorem

Any injective multi-set function can be expressed by



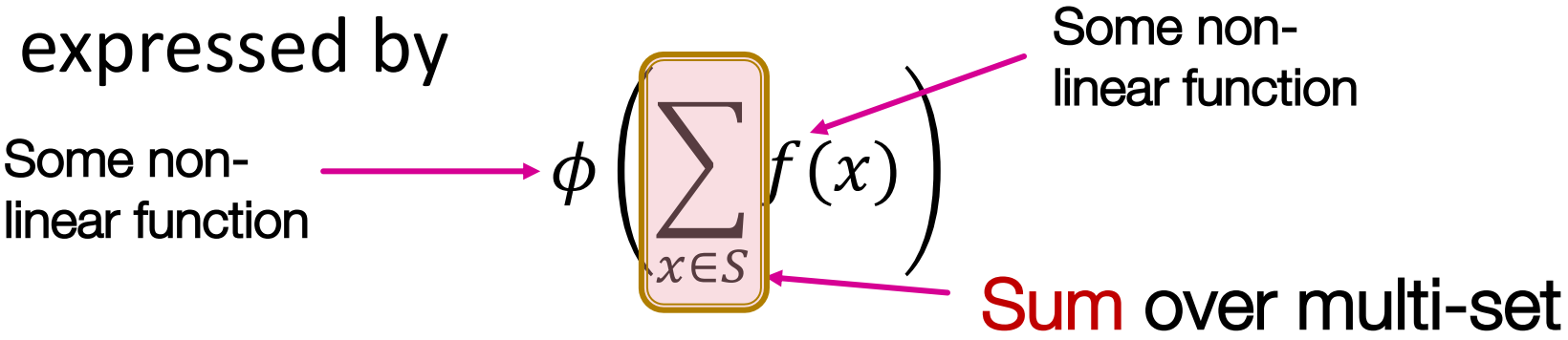
S : multi-set



Injective multi-set function

Theorem

Any injective multi-set function can be expressed by



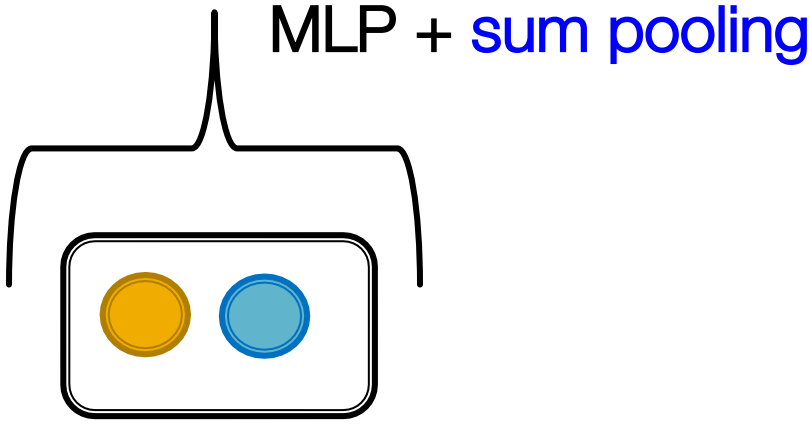
We can model ϕ and f using Multi-Layer-Perceptron (MLP). Note: MLP is a universal approximator.

$$MLP_\phi\left(MLP_f(\bigcirc) + MLP_f(\bigcirc) + MLP_f(\bigcirc)\right)$$

Most discriminative GNN

Graph Isomorphism Network (GIN)

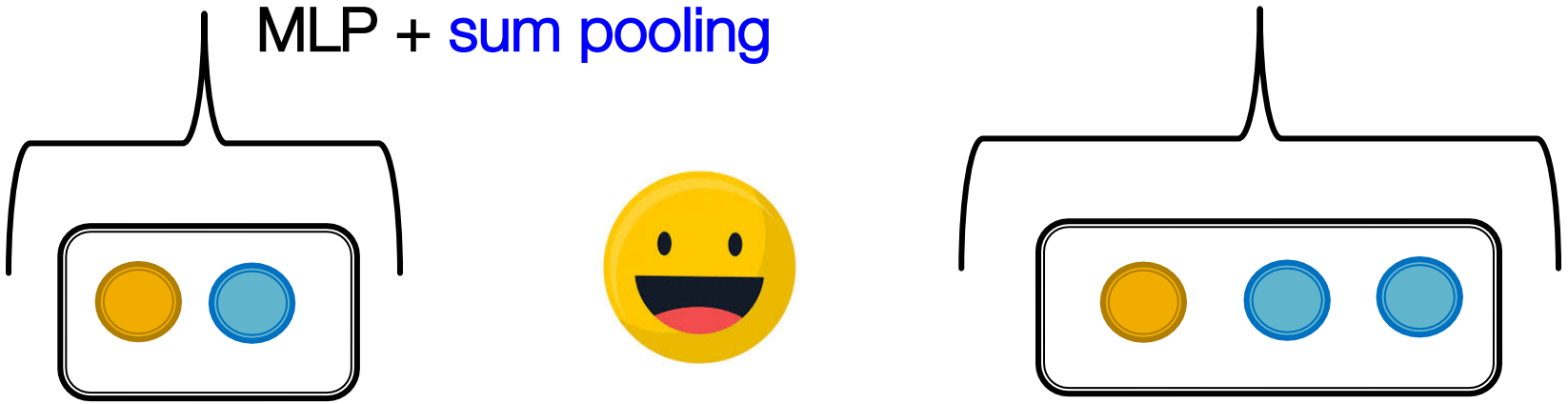
[Xu+ ICLR'2019]



Most discriminative GNN

Graph Isomorphism Network (GIN)

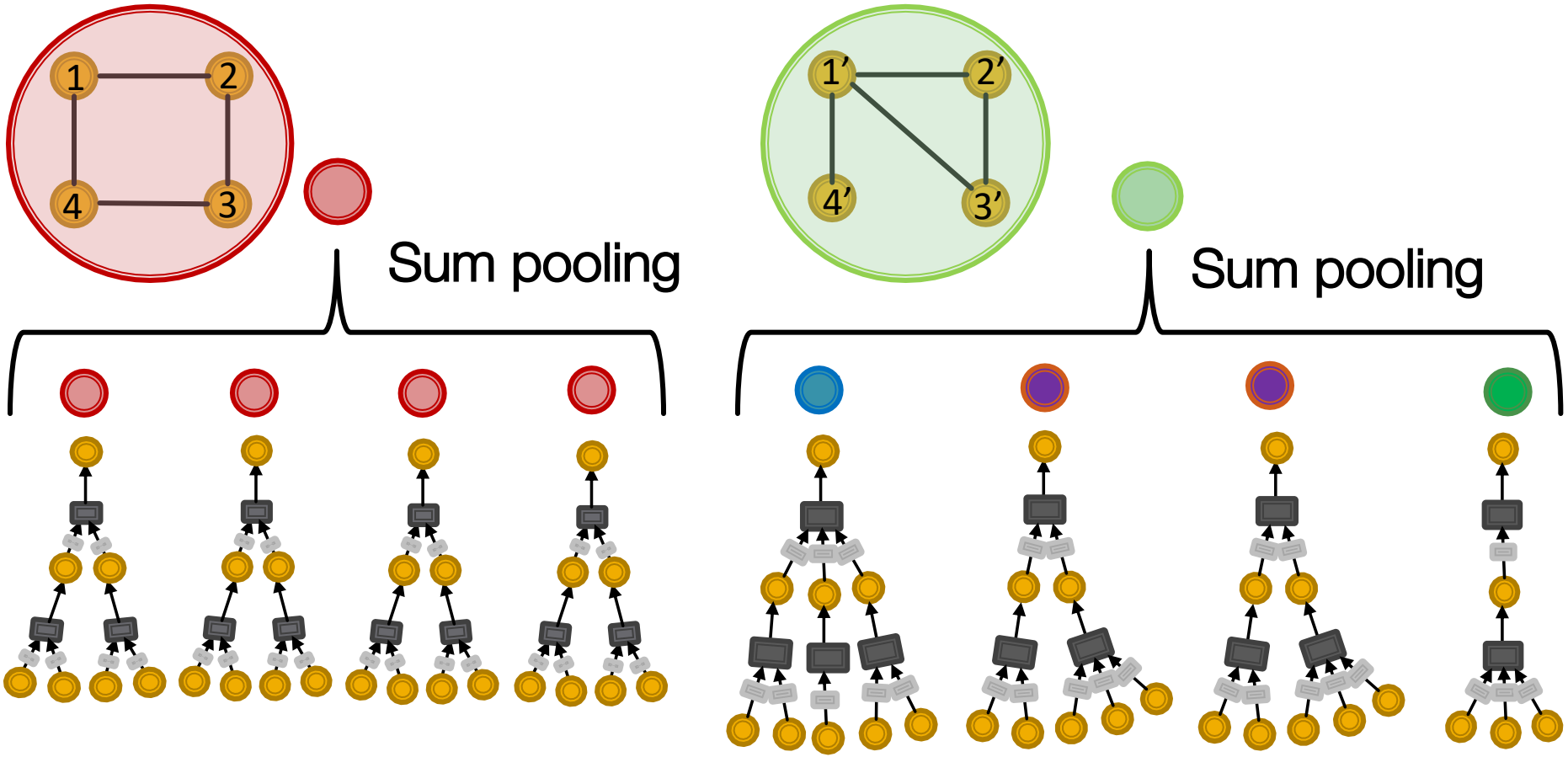
[Xu+ ICLR'2019]



The GIN's neighbor aggregation is injective!

Graph pooling in GIN

- Graph pooling is also a function over multiset. Sum pooling can give **injective graph pooling**!



Most discriminative GNN

So far: GIN achieves maximal discriminative power by using injective neighbor aggregation.

How powerful is this?

WL Graph Isomorphism Test

- GIN is closely related to **Weisfeiler-Lehman (WL) Graph Isomorphism Test** (1968).

- WL test is known to be capable of distinguishing **most** of real-world graphs.

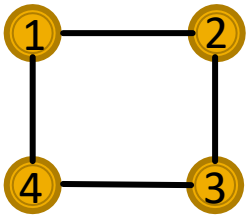
WL Graph Isomorphism Test

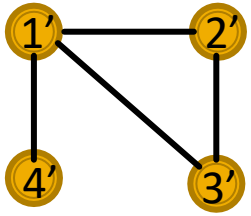
- GIN is closely related to **Weisfeiler-Lehman (WL) Graph Isomorphism Test** (1968).
- WL test is known to be capable of distinguishing most of real-world graphs.
- Next: **We will show GIN is as discriminative as the WL test**.

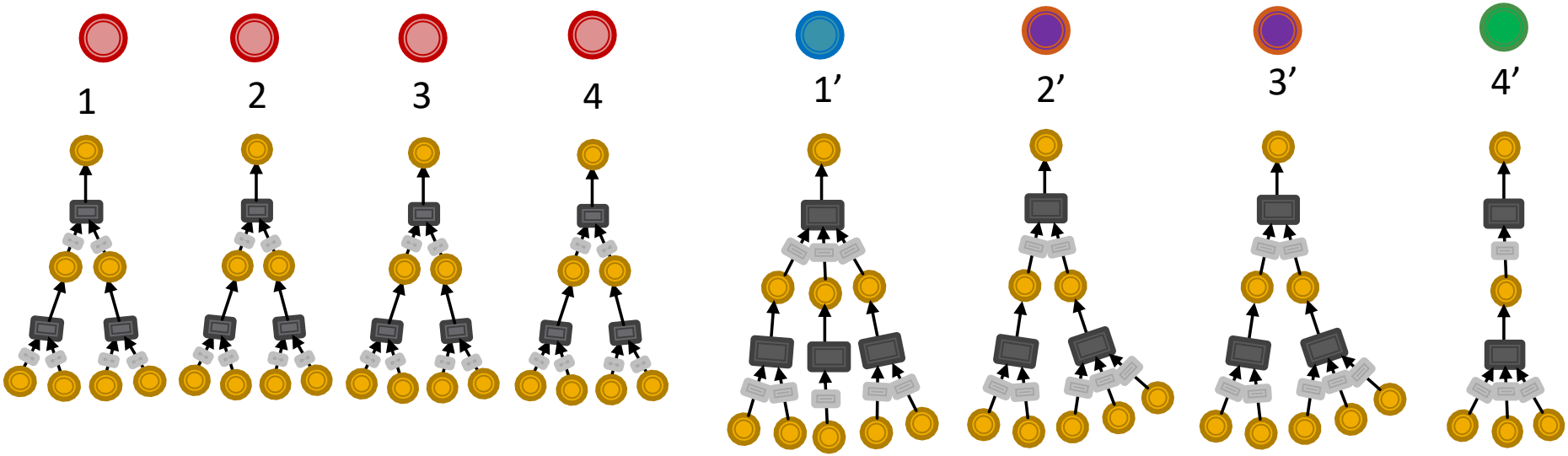
WL Graph Isomorphism Test

- WL first maps different rooted subtrees to different colors

Similar to Injective neighbor aggregation in GIN



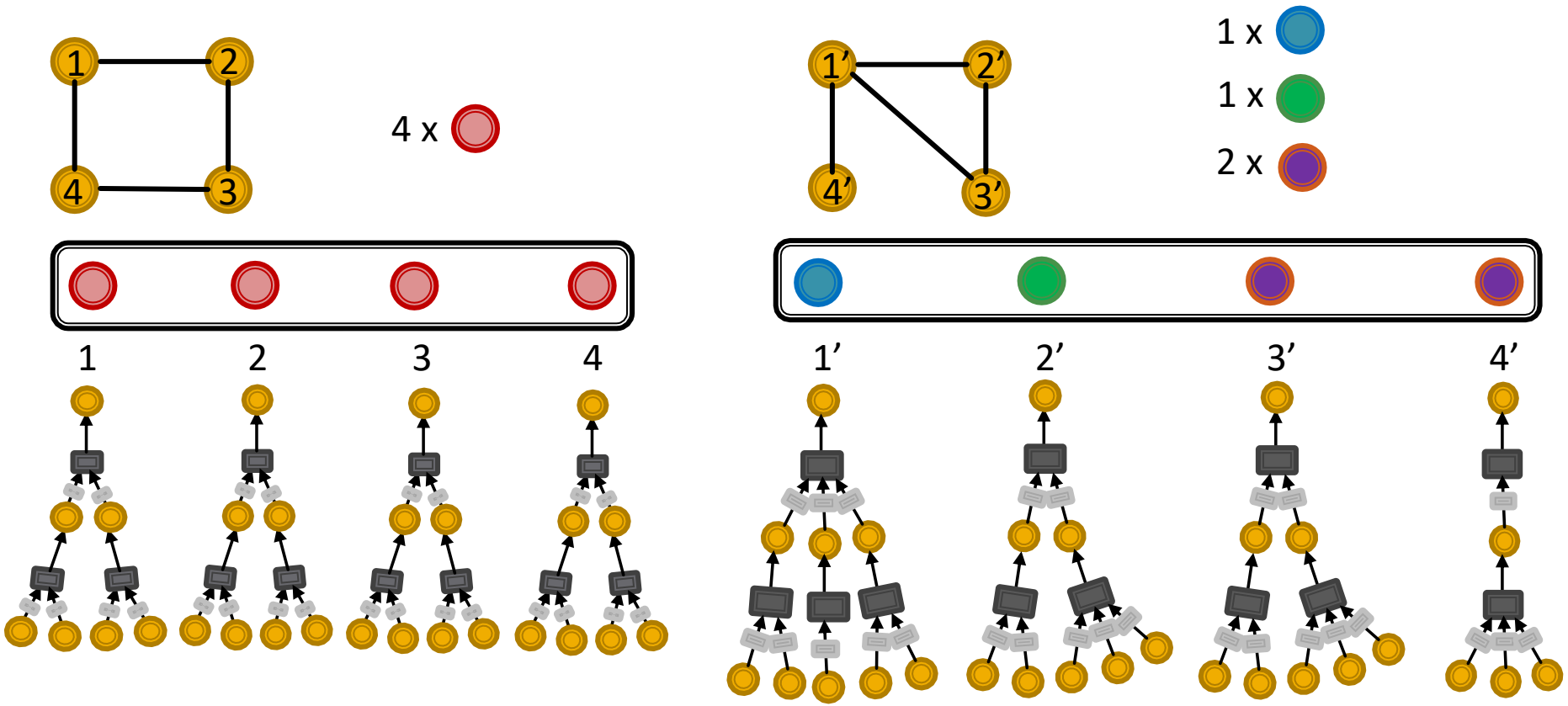




WL Graph Isomorphism Test

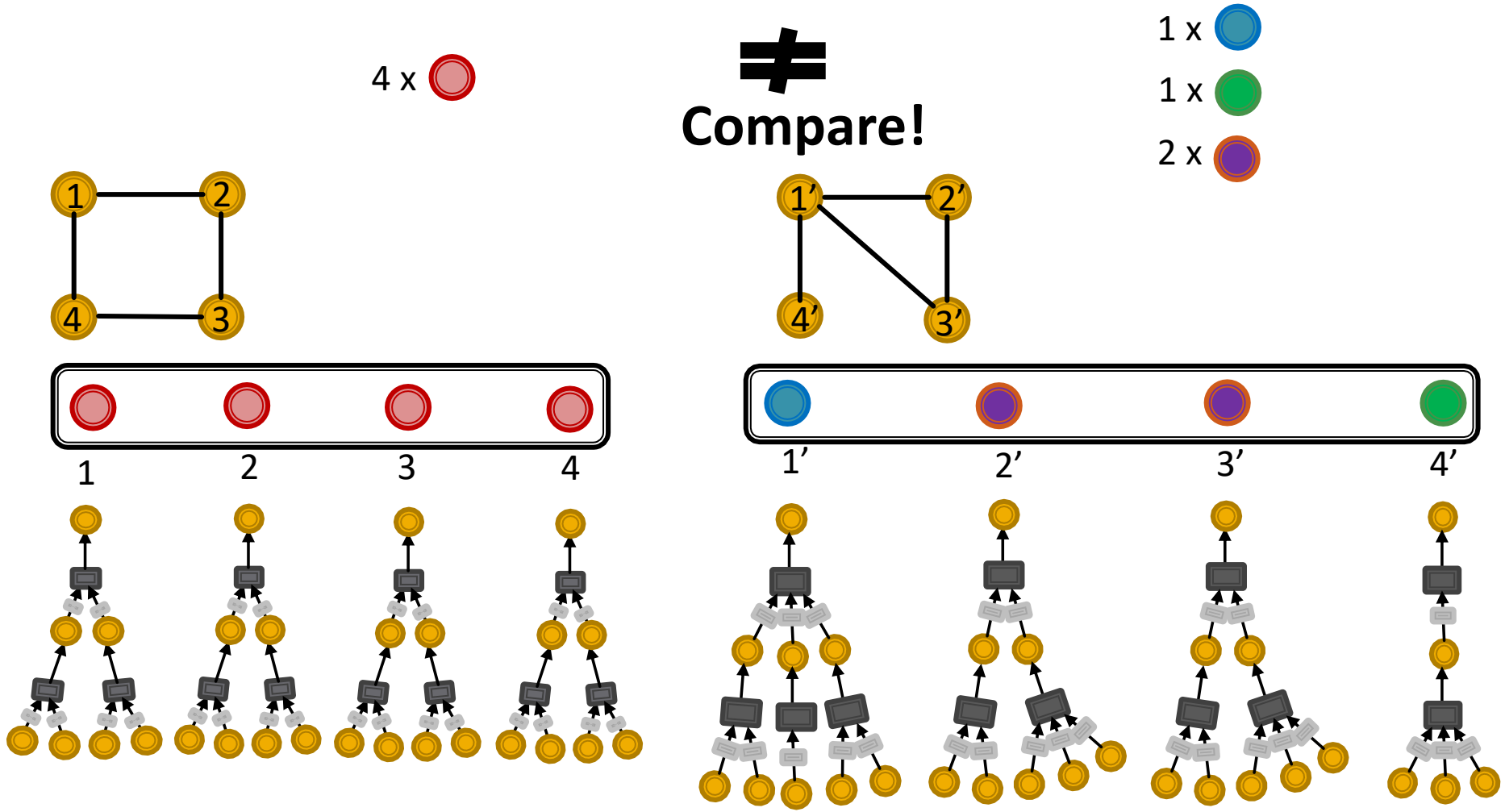
- WL then counts different colors.

Similar to graph pooling in GNN



WL Graph Isomorphism Test

- Finally, WL compares the count



WL Graph Isomorphism Test

WL test and GIN are operationally equivalent.

Graphs that WL test can distinguish

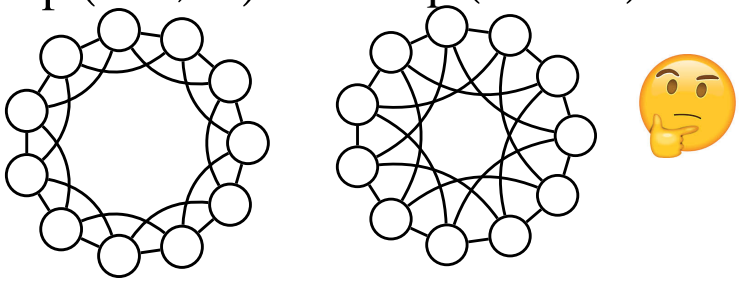
⇔ **Graphs that GIN can distinguish**

Relation to Graph Isomorphism test

Observation

- GINs have the same discriminative power as the WL graph isomorphism test.
- WL test has been known to distinguish most of the graphs, except for some corner cases.

$\mathcal{G}_{\text{skip}}(11, 2)$ $\mathcal{G}_{\text{skip}}(11, 3)$



The two graphs look the same for WL test because all the nodes have the same local subtree structure!

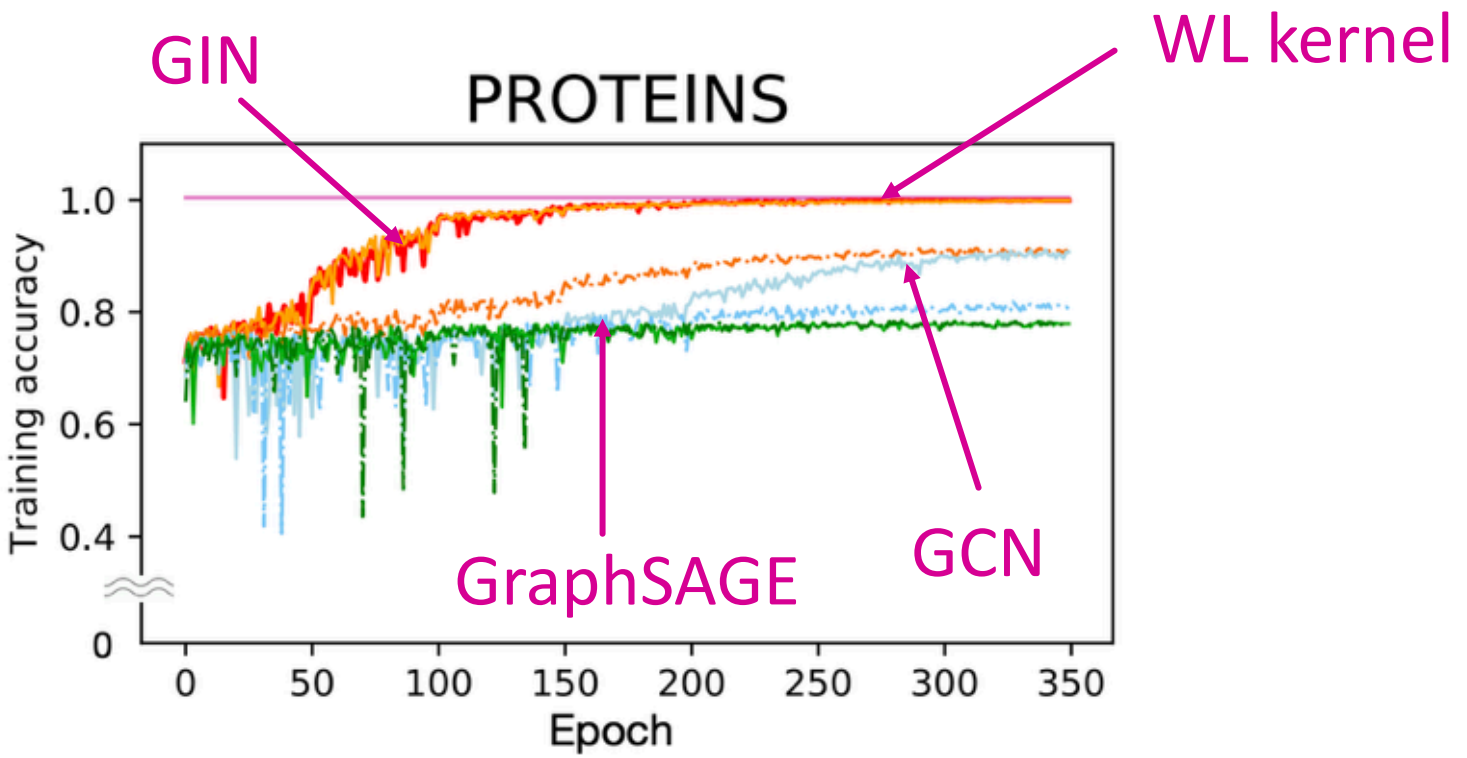
Follow-up work to resolve corner cases but with **exponential time complexity:** [Murphy+ ICML 2019]

Experiments: Training accuracy

- Graph classification: social and bio/chem graphs

Training accuracy of different GNN architectures.

GIN fits training data much better than GCN, GraphSAGE.

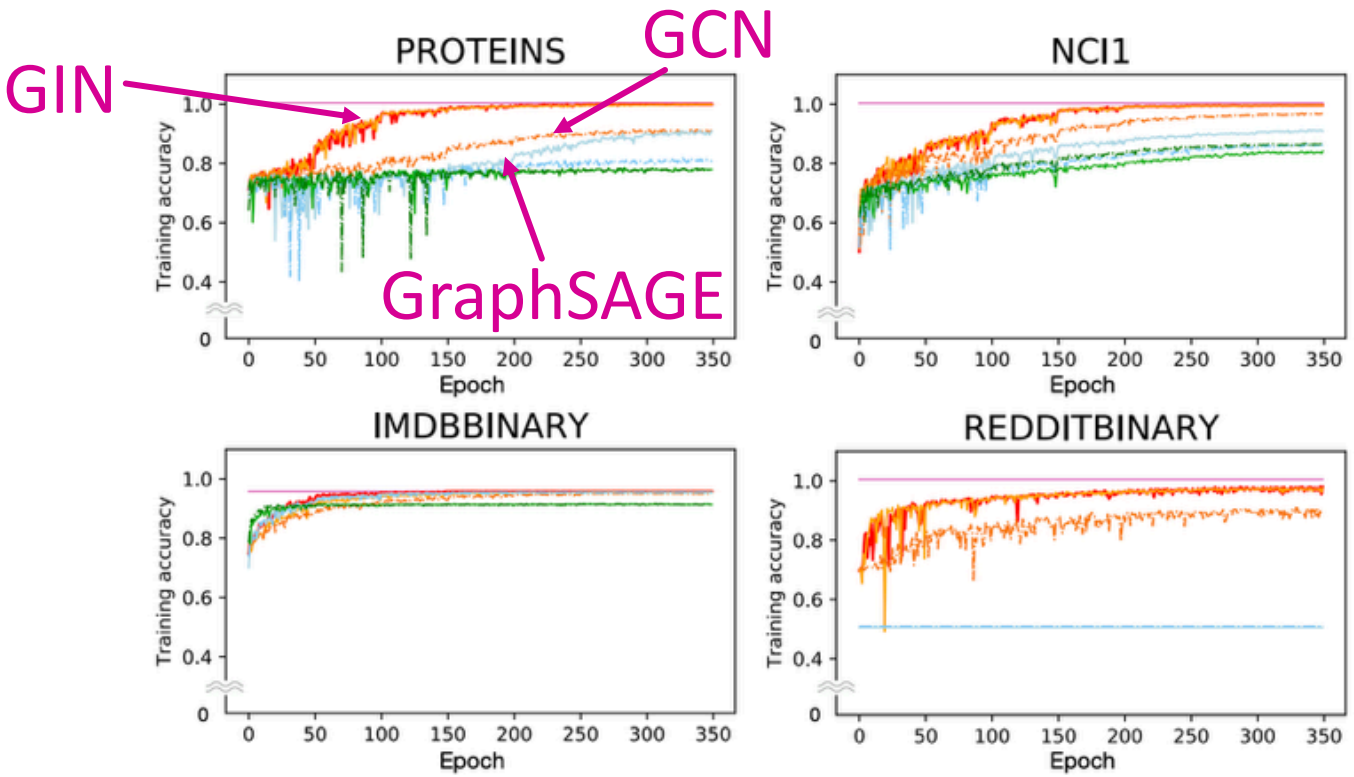


Experiments: Training accuracy

- Graph classification: social and bio/chem graphs

Training accuracy of different GNN architectures.

Same trend across datasets!



Experiments: Test accuracy

- Graph classification: social and bio/chem graphs

GIN outperforms existing GNNs also in terms of **test accuracy** because it can better capture graph structure.

Data / Model	**GIN (Powerful)**	**GCN (Mean)**	**GraphSAGE (Max)**
RDT-B	**92.4**	**50.0**	**50.0**
IMDB-B	**75.1**	74.0	72.3
NCI1	**82.7**	80.2	77.7
MUTAG	**89.4**	85.6	85.1

Reddit dataset **does not** have node features!

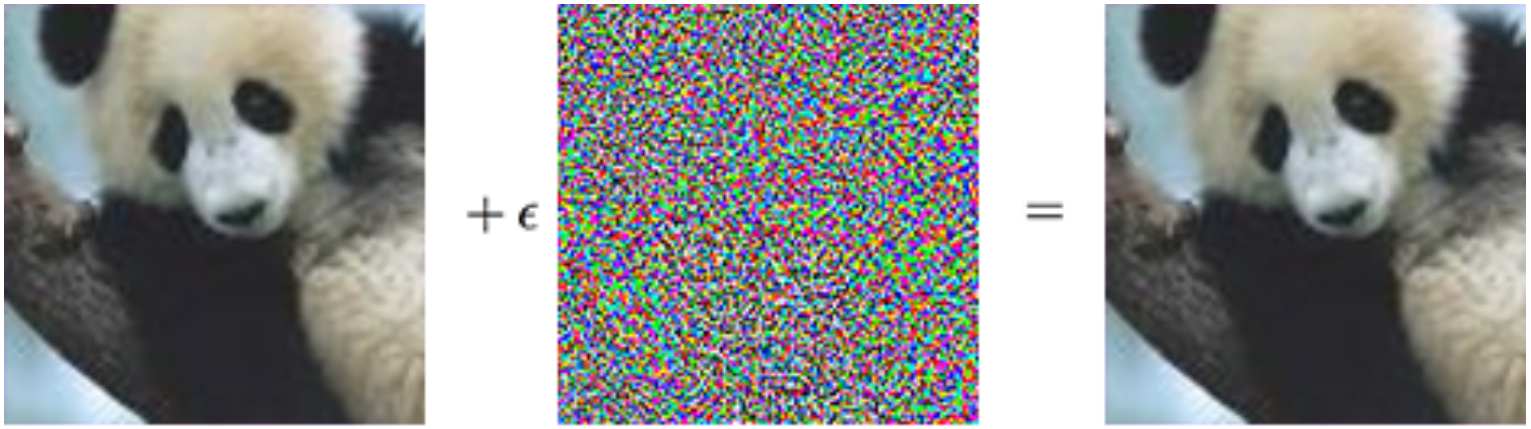
Summary of the first part

- Existing GNNs use **non-injective neighbor aggregation**, thus have **low discriminative power**
- GIN uses **injective neighbor aggregation**, and is as discriminative as the WL graph isomorphism test
- GIN achieves state-of-the-art test performance in graph classification

2. Vulnerability of GNNs to noise in graph data

Adversarial Attacks in DNNs

- Deep Neural Networks are **vulnerable to adversarial attacks**!
- Attacks are often implemented as **imperceptible noise** that changes the prediction



Attacks on Graph Domains

- **Adversaries** are **very common** in applications of graph neural networks, e.g., search engines, recommender systems, social networks, etc.

- These adversaries **will exploit** any exposed vulnerabilities!

Attacks on Graph Domains

- **Adversaries** are **very common** in applications of graph neural networks. e.g. search engines, recommender systems, social networks.
- These adversaries will exploit any vulnerabilities exposed.

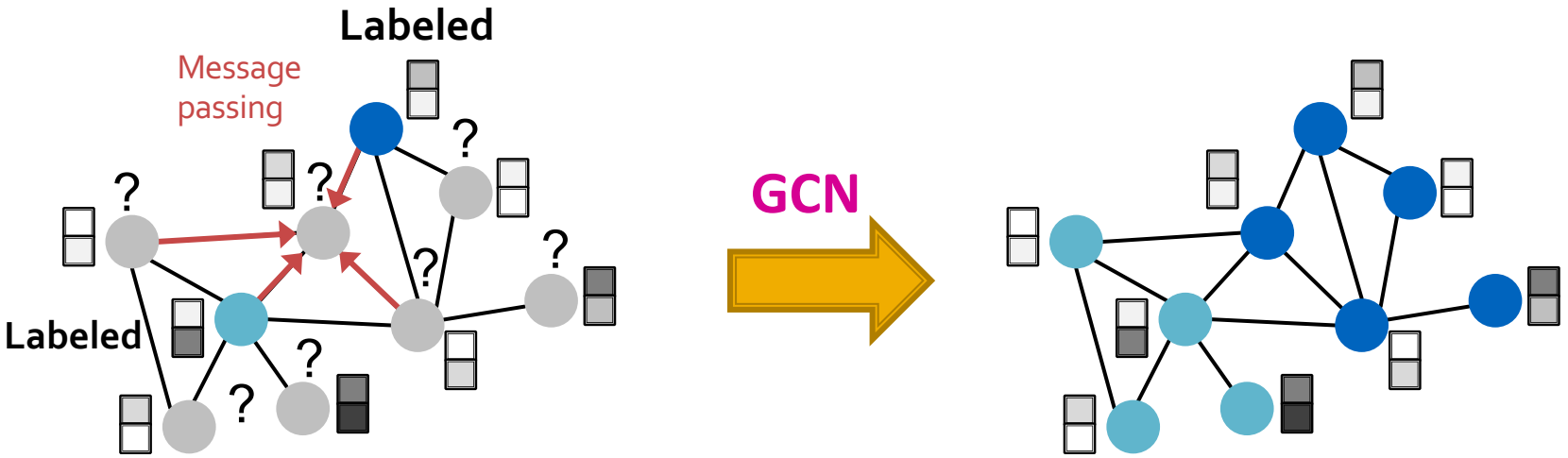
Are GNNs robust to adversarial attacks?

Semi-Supervised Node Classification

- Here we focus on **semi-supervised node classification** using **Graph Convolutional Neural Networks (GCN)** [Kipf+ ICLR'2017].

Input: Partially labeled attributed graph

Goal: Predict labels of unlabeled nodes



?: Unlabeled

GCN for Semi-Supervised Node Classification

$A \in \{0,1\}^{N \times N}$: Adjacency matrix

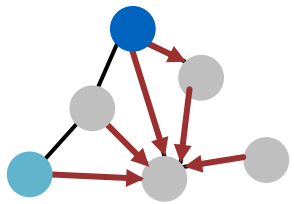
$X \in \{0,1\}^{N \times D}$: (binary) node attributes

$\hat{A} \equiv D^{-\frac{1}{2}}(A + I)\, D^{-\frac{1}{2}}$: Renormalized adjacency matrix

Classification Model:

Two-step GCN message passing

$$softmax\left(\hat{A}\, ReLU\left(\hat{A}XW^{(1)}\right)W^{(2)}\right)$$

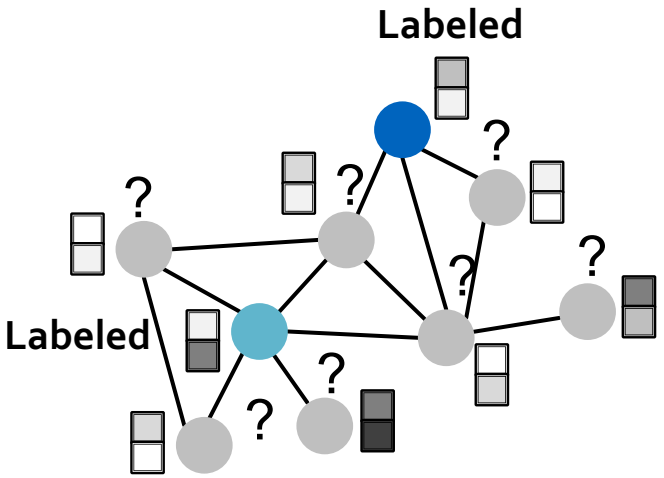


Training:

Minimize cross entropy loss on labeled data

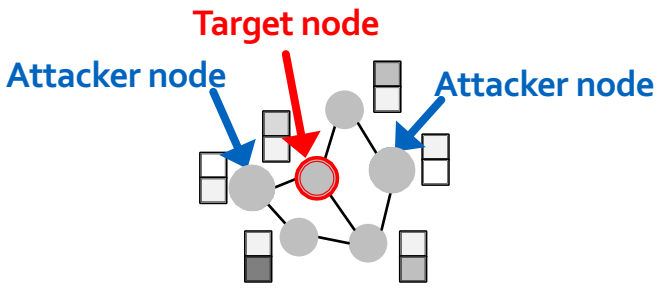
Testing:

Apply the model to predict unlabeled data



Attack possibilities

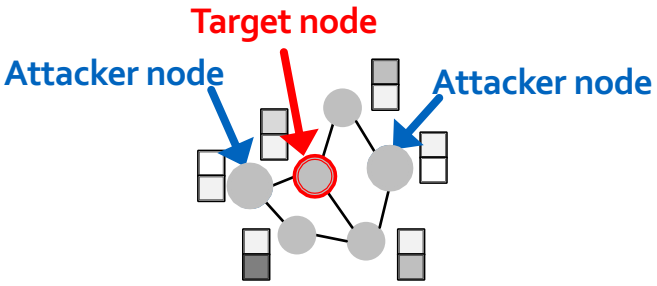
- What are attack possibilities in real world?



Target node $t \in V$: node whose classification label we want to change
Attacker nodes $S \subset V$: nodes the attacker can modify

Attack possibilities

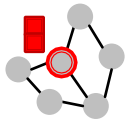
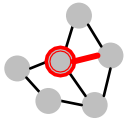
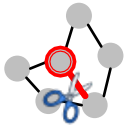
- What are attack possibilities in real world?



Target node $t \in V$: node whose classification label we want to change
Attacker nodes $S \subset V$: nodes the attacker can modify

Direct attack ($S = \{t\}$)

- Modify the **target**'s features
- Add connections to the **target**
- Remove connections from the **target**

Example

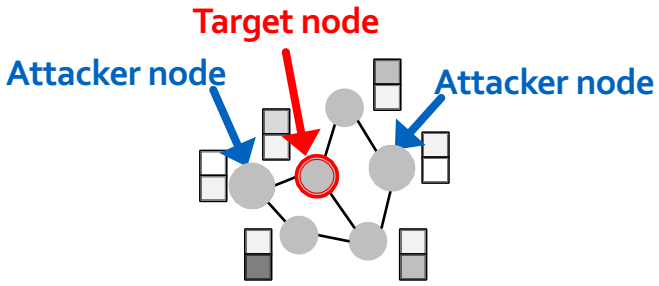
Change website content

Buy likes/ followers

Unfollow untrusted users

Attack possibilities

- What are attack possibilities in real world?

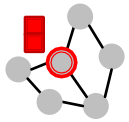


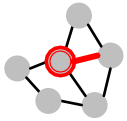
Target node $t \in V$: node whose classification label we want to change
Attacker nodes $S \subset V$: nodes the attacker can modify

Direct attack ($S = \{t\}$)

- Modify the **target**'s features
- Add connections to the **target**
- Remove connections from the **target**

Example

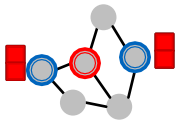

Change website content


Buy likes/ followers


Unfollow untrusted users

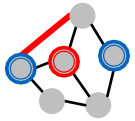
Indirect attack ($t \notin S$)

- Modify the **attackers**' features

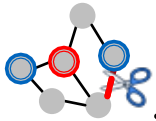


Example
Hijack friends of target

- Add connections to the **attackers**



- Remove connections from the **attackers**



Create a link/ spam farm

Attack possibilities

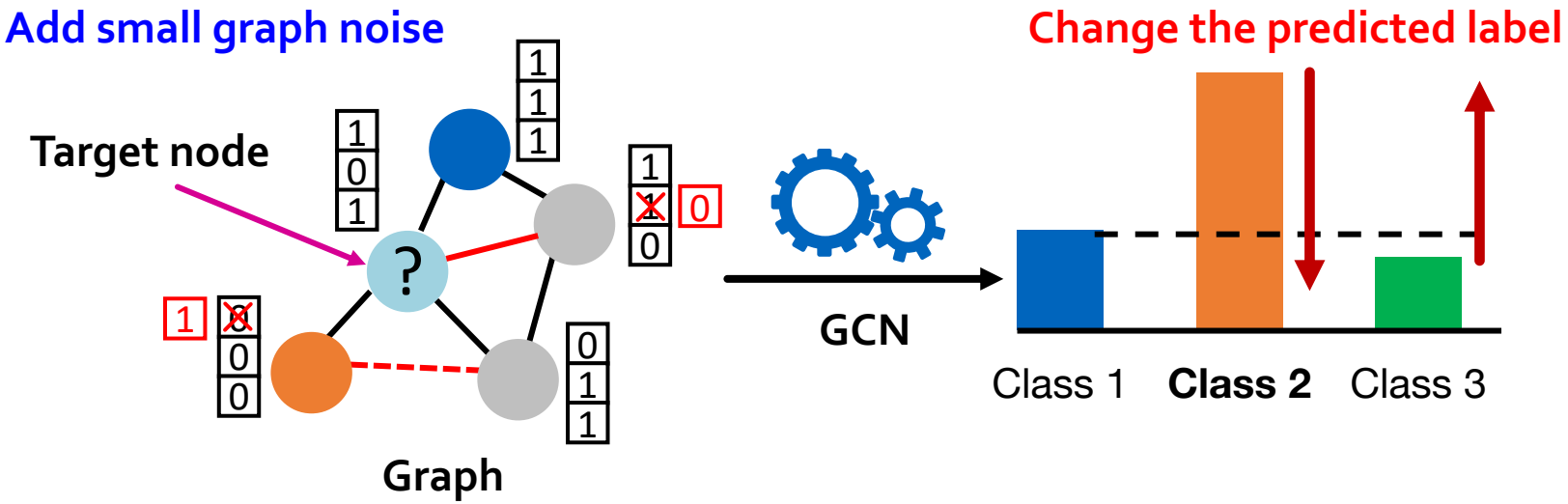
- What are attack possibilities in real world?

How to mathematically formalize these attack possibilities?

Nettack: High Level Idea

- Zügner+, *Adversarial Attacks on Neural Networks for Graph Data*, KDD'18

Maximize (Change of predicted labels of target node)
Subject to (Limited noise in the graph)



Mathematical Formulation

- Find a **modified graph** that maximizes the change of predicted labels of target node

Let's parse the objective function!

$$\arg\max_{A',X'} \max_{c \neq c_{old}} \log Z^*_{v,c} - \log Z^*_{v,c_{old}}$$

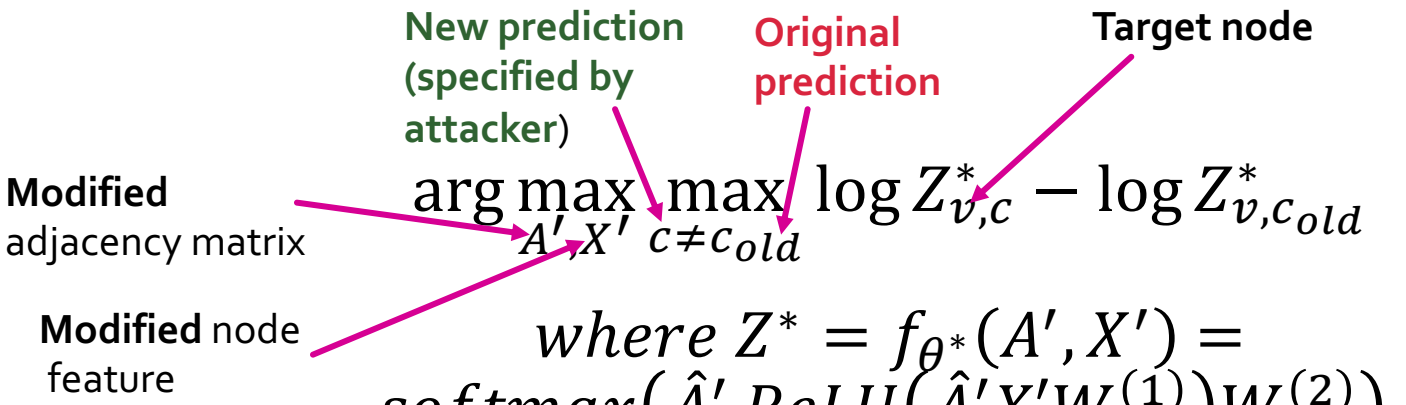
$$where\ Z^* = f_{\theta^*}(A', X') = softmax\left(\hat{A}'\ ReLU\left(\hat{A}'X'W^{(1)}\right)W^{(2)}\right),$$

$$with\ \ \theta^* = \arg\min_{\theta} \mathcal{L}\,(\theta; A', X')$$

$$s.t.\,(A', X') \approx (A, X)$$

Mathematical Formulation

- Find a **modified graph** that maximizes the change of predicted labels of target node



$$\arg\max_{A',X'} \max_{c \neq c_{old}} \log Z^*_{v,c} - \log Z^*_{v,c_{old}}$$

$$where\ Z^* = f_{\theta^*}(A', X') = softmax\left(\hat{A}'\ ReLU\left(\hat{A}'X'W^{(1)}\right)W^{(2)}\right),$$

$$with\ \ \theta^* = \arg\min_{\theta} \mathcal{L}\,(\theta; A', X')$$

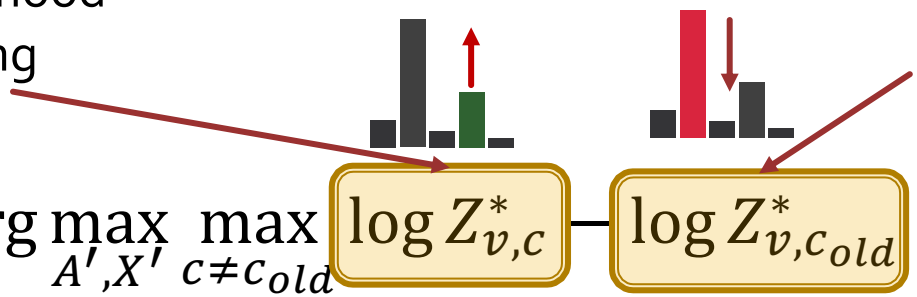
$$s.t.\,(A', X') \approx (A, X)$$

Mathematical Formulation

- Find a **modified graph** that maximizes the change of predicted labels of target node

Increase the loglikelihood of target node v being predicted as c

Decrease the loglikelihood of target node v being predicted as c_{old}



$$\arg\max_{A',X'} \max_{c \neq c_{old}} \log Z^*_{v,c} - \log Z^*_{v,c_{old}}$$

$$where\ Z^* = f_{\theta^*}(A', X') = softmax\left(\hat{A}'\ ReLU\left(\hat{A}'X'W^{(1)}\right)W^{(2)}\right),$$

$$with\ \ \theta^* = \arg\min_{\theta} \mathcal{L}\,(\theta; A', X')$$

$$s.t.\,(A', X') \approx (A, X)$$

Mathematical Formulation

- Find a **modified graph** that maximizes the change of predicted labels of target node

GCN is **trained on modified graph**, which is then used to predict labels of the target node.

$$\arg\max_{A',X'} \max_{c \neq c_{old}} \log Z^*_{v,c} - \log Z^*_{v,c_{old}}$$

$$where\ Z^* = f_{\theta^*}(A', X') = softmax\left(\hat{A}'\ ReLU\left(\hat{A}'X'W^{(1)}\right)W^{(2)}\right),$$

$$with\ \ \theta^* = \arg\min_{\theta} \mathcal{L}\,(\theta; A', X')$$

$$s.t.\,(A', X') \approx (A, X)$$

Mathematical Formulation

- Find a **modified graph** that maximizes the change of predicted labels of target node

The **modified graph** should be close to the original graph.
E.g., Fixed budget of edge deletion/addition, node attribute perturbation

$$\arg\max_{A',X'} \max_{c \neq c_{old}} \log Z^*_{v,c} - \log Z^*_{v,c_{old}}$$

$$where\ Z^* = f_{\theta^*}(A', X') = softmax\left(\hat{A}'\ ReLU\left(\hat{A}'X'W^{(1)}\right)W^{(2)}\right),$$

$$with\ \ \theta^* = \arg\min_{\theta} \mathcal{L}\,(\theta; A', X')$$

$$s.t.\,(A', X') \approx (A, X)$$

Tractable Optimization

- **In practice**, we cannot exactly solve the optimization problem because…
  - Graph modification is **discrete** (cannot use simple gradient descent to optimize)
  - Inner loop involves expensive re-training of GCN

$$\arg\max_{A',X'} \max_{c \neq c_{old}} \log Z^*_{v,c} - \log Z^*_{v,c_{old}}$$

$$where\ Z^* = f_{\theta^*}(A', X') = softmax\left(\hat{A}'\ ReLU\left(\hat{A}'X'W^{(1)}\right)W^{(2)}\right),$$

$$with\ \ \theta^* = \arg\min_{\theta} \mathcal{L}\left(\theta; A', X'\right)$$

$$s.t.\ (A', X') \approx (A, X)$$

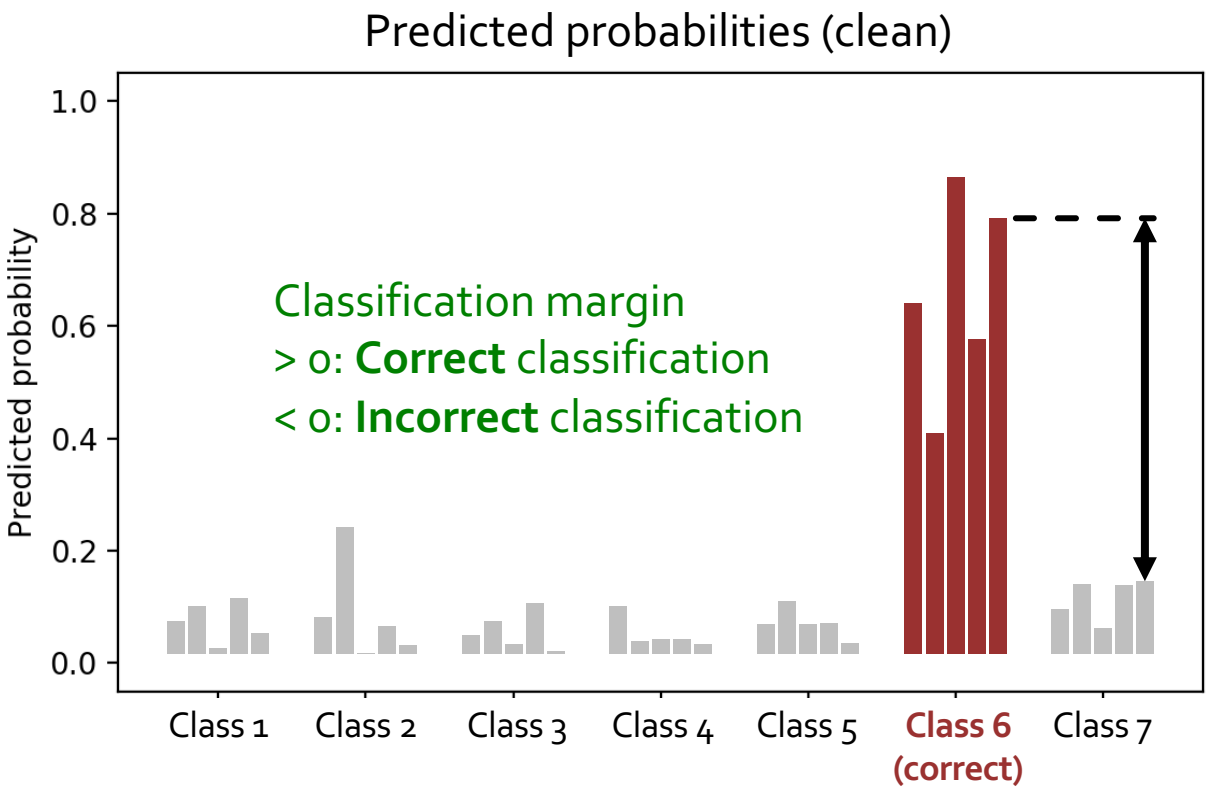
Tractable optimization

- Some heuristics have been proposed to efficiently obtain an **approximate solution**.
- For example:
  - **Greedily** choosing the step-by-step graph modification
  - Simplifying GCN by removing ReLU activation (to work in closed form)
  - Etc.

More details in Zügner+ KDD'2018!

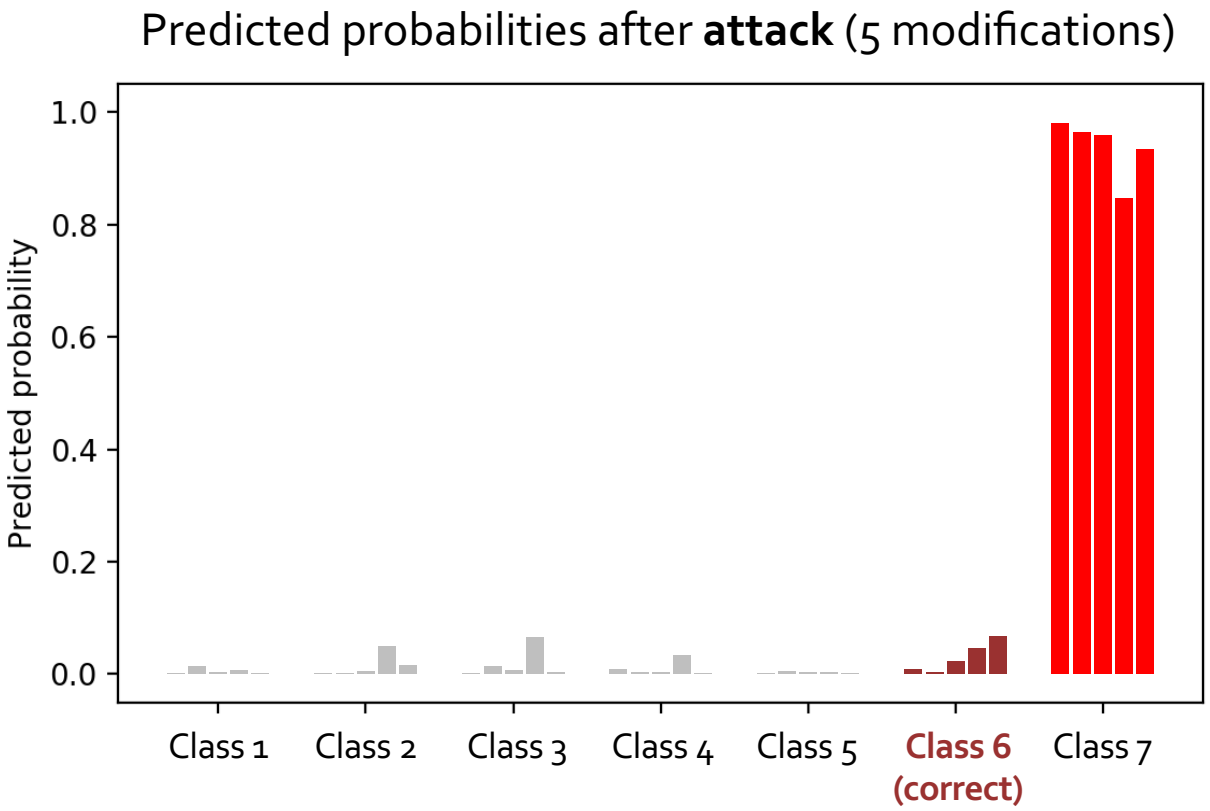
Nettack Experiments

- Semi-supervised node classification with GCN
  - Class predictions for a **single node**, produced by 5 GCNs with different random initializations



Experiments

- The GCN prediction is **easily manipulated** by only 5 modifications of graph structure (|V|=~2k, |E|=~5k)



Summary of the second part

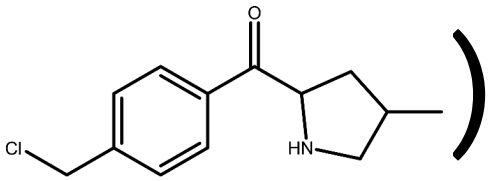
- The prediction of GCN is easily manipulated by only 5 modifications of graph structure.

GNNs are not robust to adversarial attacks!

3. Open questions & Future directions

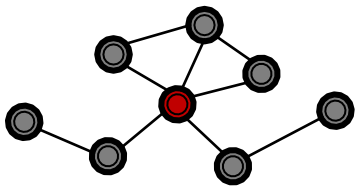
GNNs for Science Domains

- **Chemistry:** Molecular graphs
  - Molecular property prediction

GNN(

) = toxic?

- **Biology:** Protein-Protein Interaction Networks
  - Protein function prediction

GNN () = adenylate cyclase activity?

Challenges of Applying GNNs

- Scarcity of labeled data
  - Labels require **expensive experiments**

    → Models overfit to small training datasets

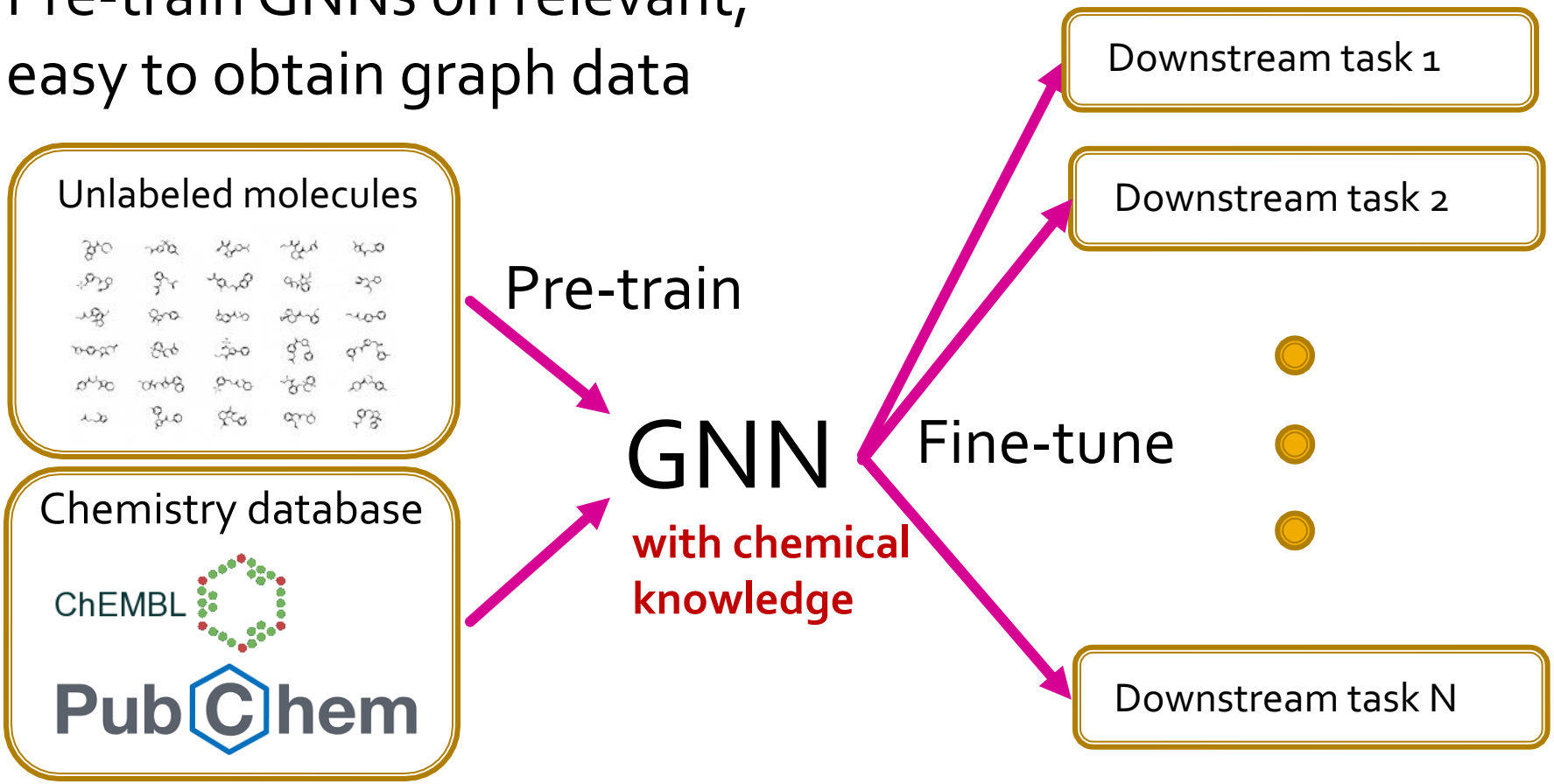
- Out-of-distribution prediction
  - Test examples are **very different from training** in scientific discovery

    → Models typically perform poorly

Pre-training for GNNs

- **Pre-training GNNs** [Hu+ 2019]

Pre-train GNNs on relevant, easy to obtain graph data



Making GNNs Robust

- We have seen how to attack GNNs
- Open question:

  How to defend against the attacks?

Challenges:

- Tractable optimization on discrete graph data
- Achieving good trade-off between Accuracy and Robustness