Introduction to Graph Neural Networks

Machine Learning with Graphs

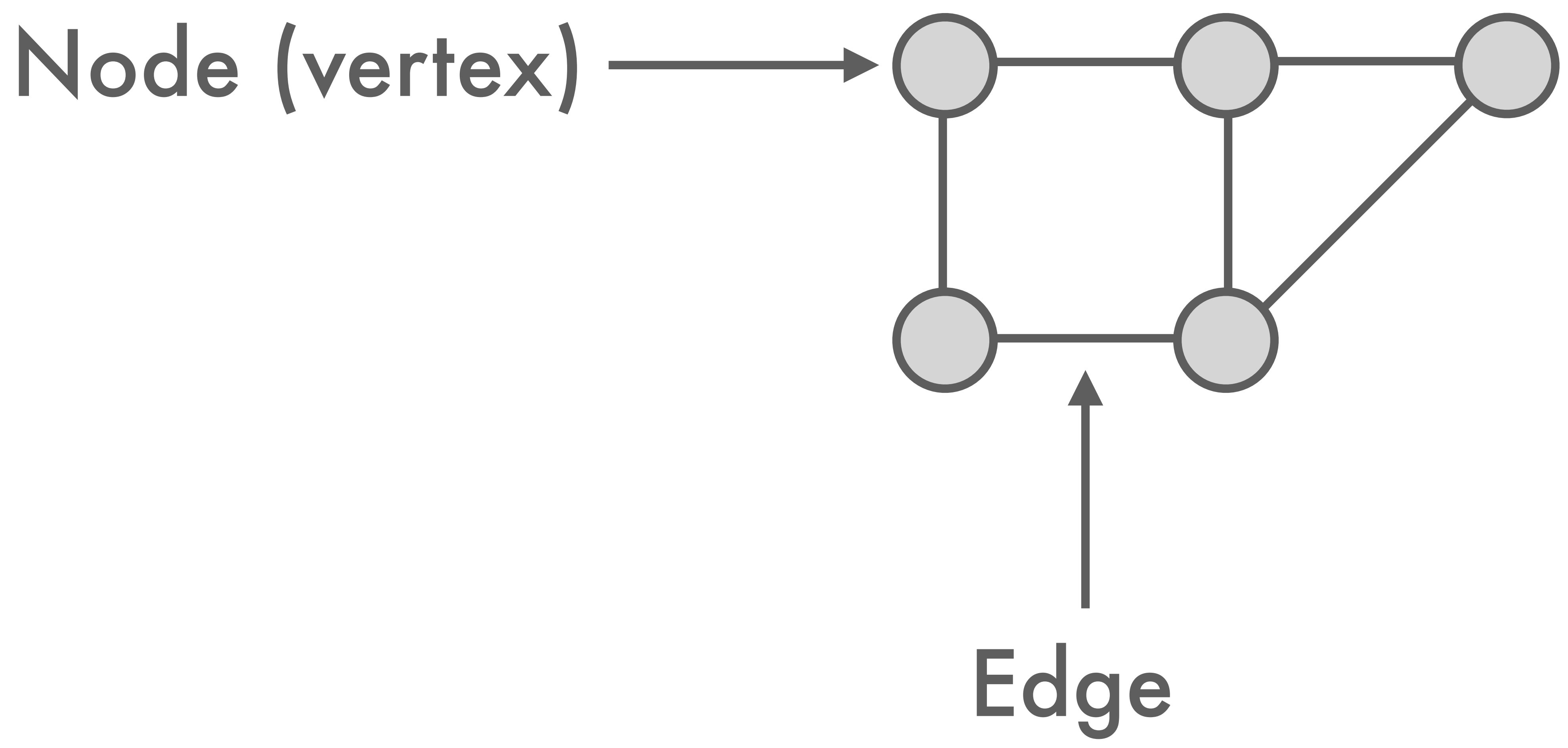
Christopher Morris, McGill University and Mila - Quebec AI Institute
www.christophermorris.info @chrsmrrs

Learning objectives

Understand learning with graphs and Graph Neural Networks:

- Understand specific challenges of graph-structured data
- Understand basic algorithms for learning with graphs
- Learn about common Graph Neural Network layers
- Understand limitations of Graph Neural Networks
- Learn how to overcome limitations of Graph Neural Networks
- Understand how to implement Graph Neural Networks using PyTorch Geometric

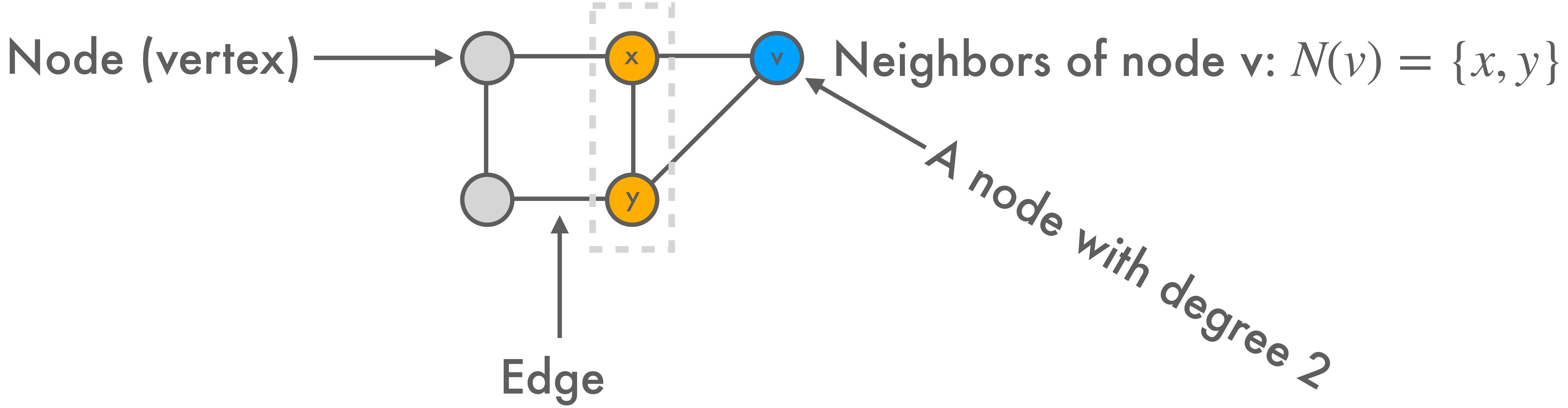
Basic definitions from graph theory



Defintion: Graph

A G is a pair $(V(G), E(G))$ with a set of nodes $V(G)$ and a set of edges $E(G) = \{(v, w) \mid v \neq w\}$.

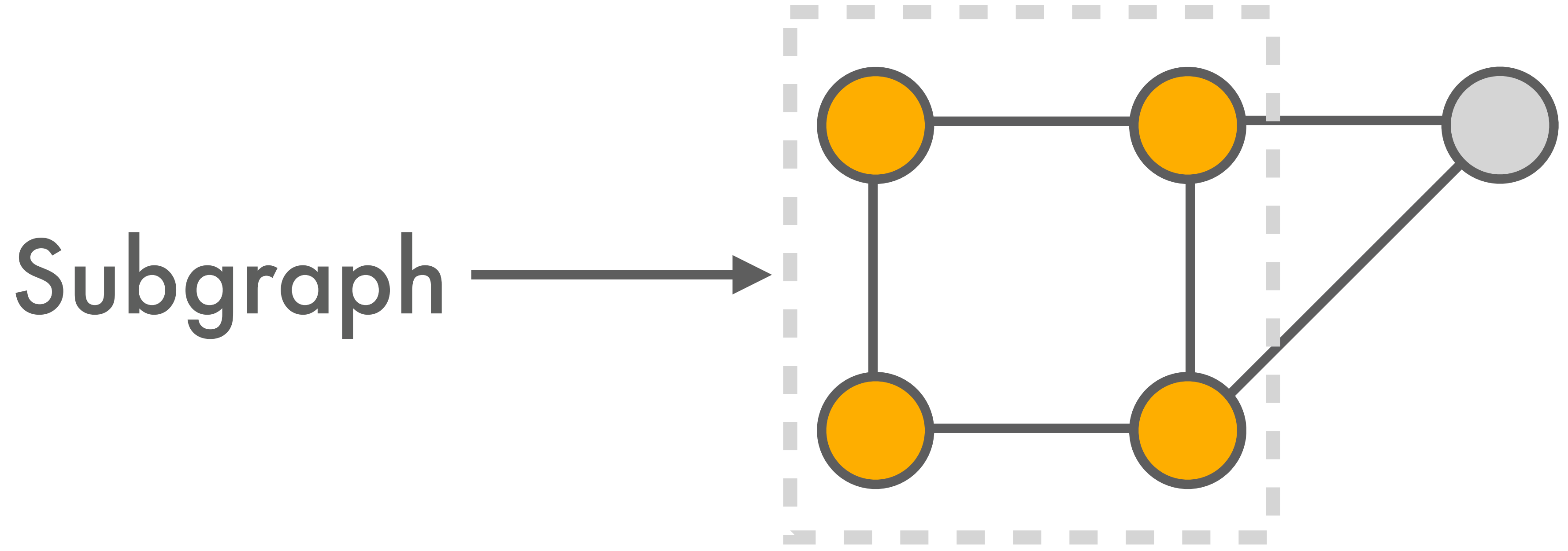
Basic definitions from graph theory



Defintion: Graph

A G is a pair $(V(G), E(G))$ with a set of nodes $V(G)$ and a set of edges $E(G) = \{(v, w) \mid v \neq w\}$.

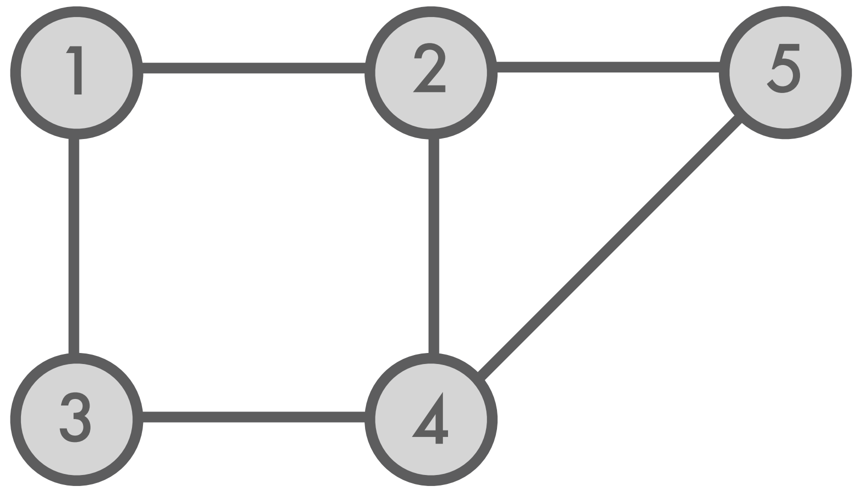
Basic definitions from graph theory



Defintion: Subgraph

Let G be a graph and subset $S \subseteq V(G)$, then (S, E_S) is a *subgraph* of G with $E_S = \{(u, v) \mid u, v \in S\} \subseteq E(G)$.

Basic definitions from graph theory



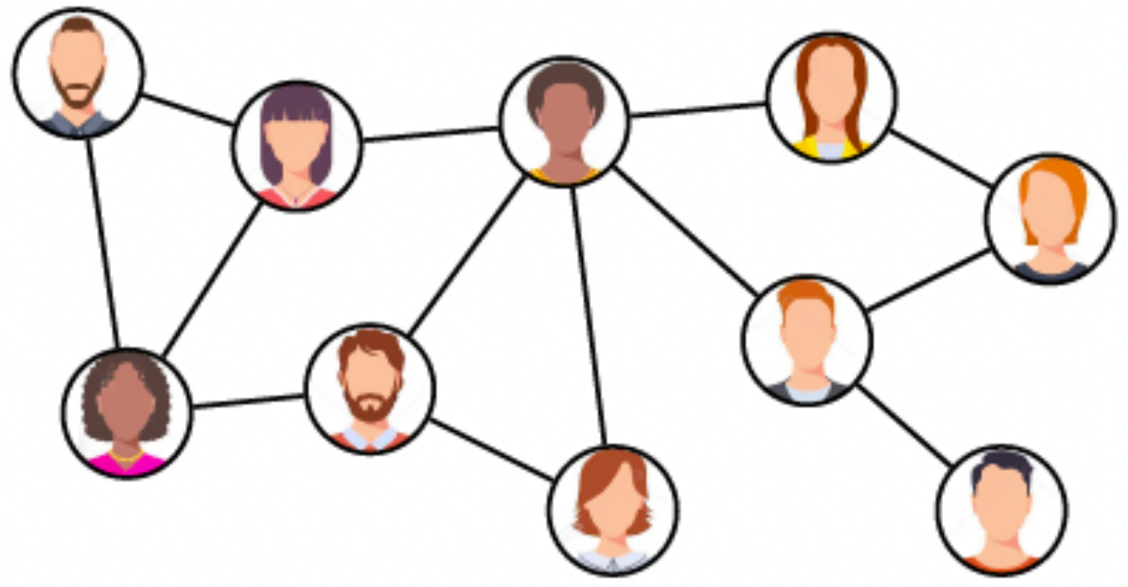
Graph

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

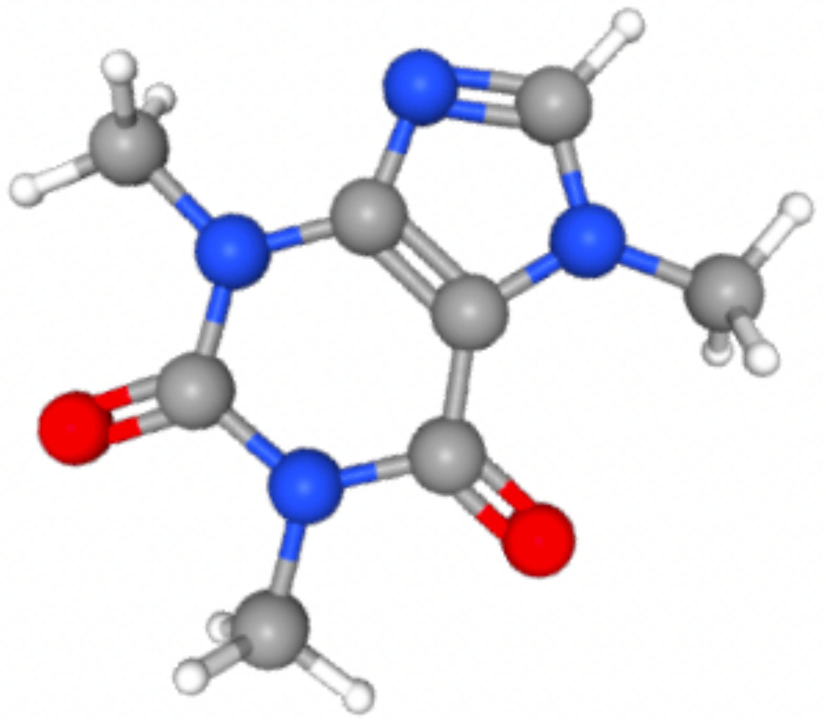
Adjacency matrix

Motivation: Graph data

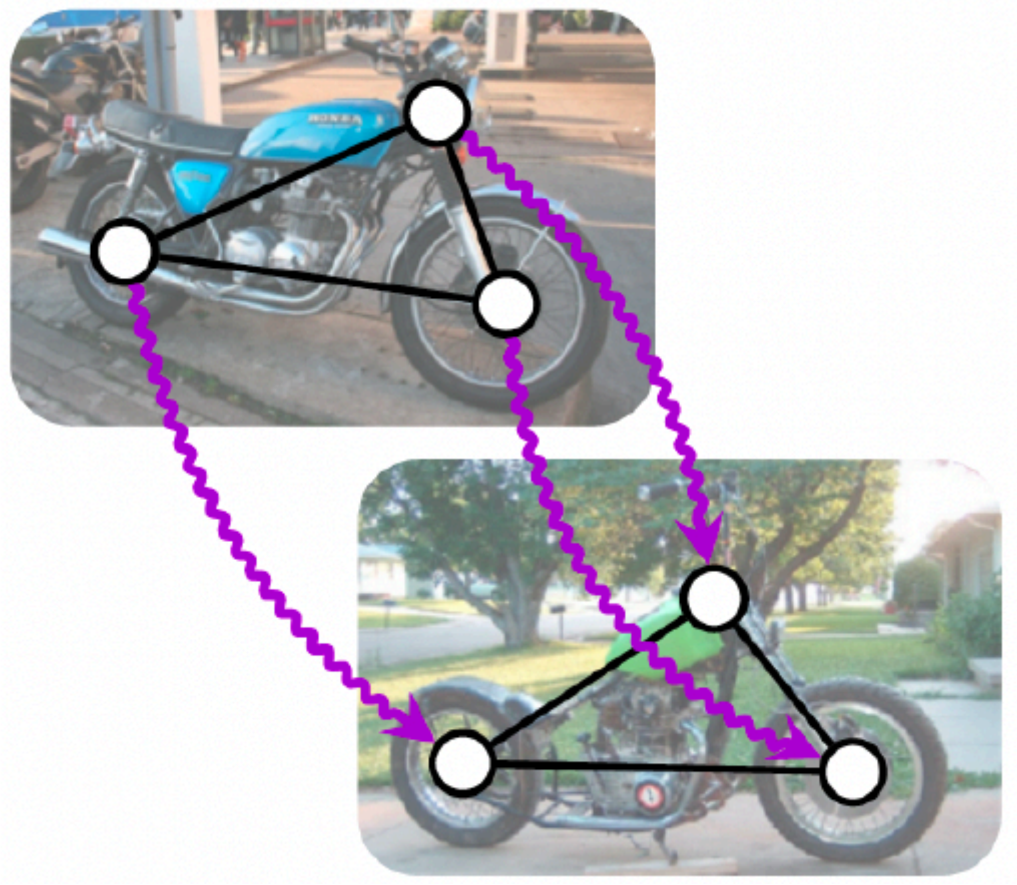
Graphs are everywhere...



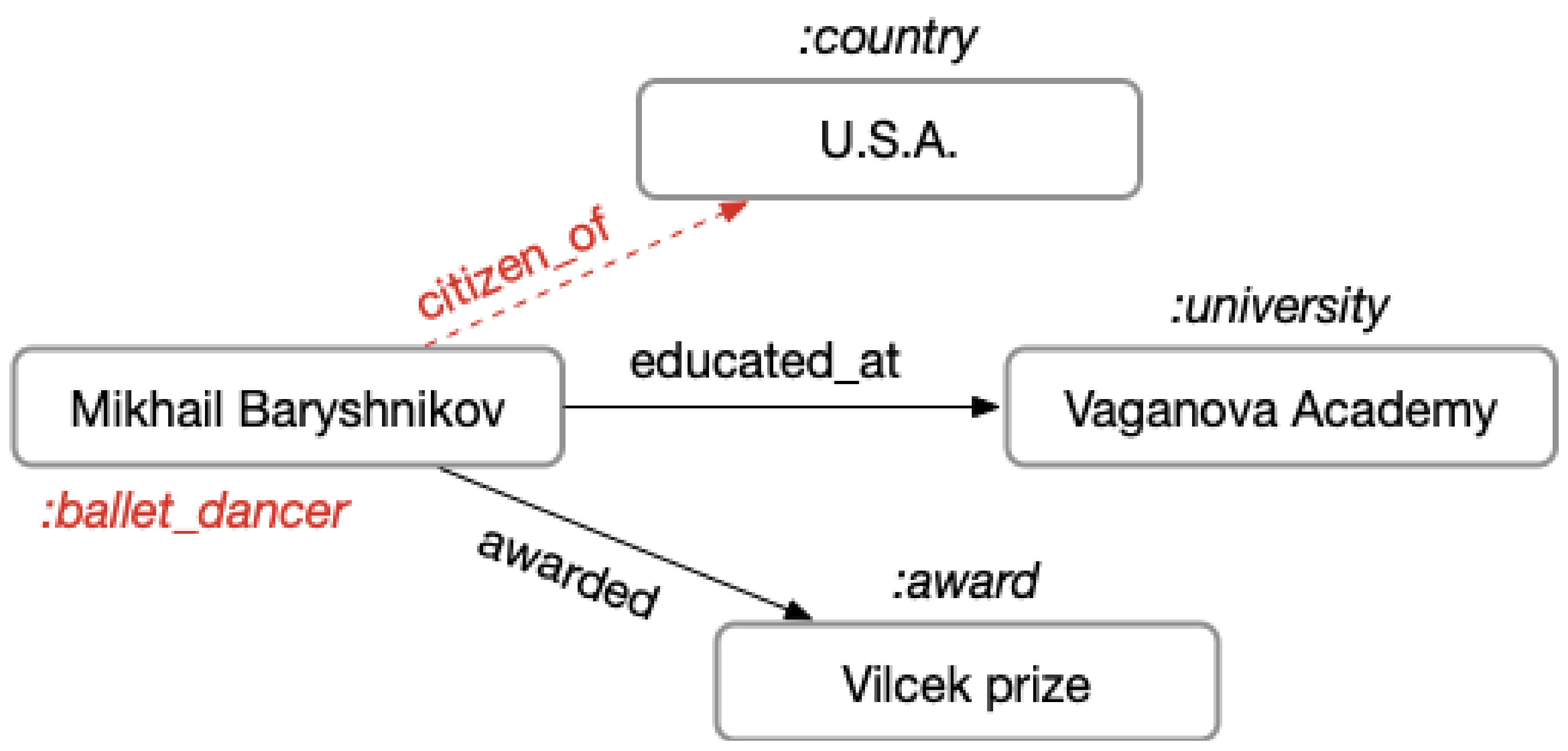
Social Networks



Chemical Molecules



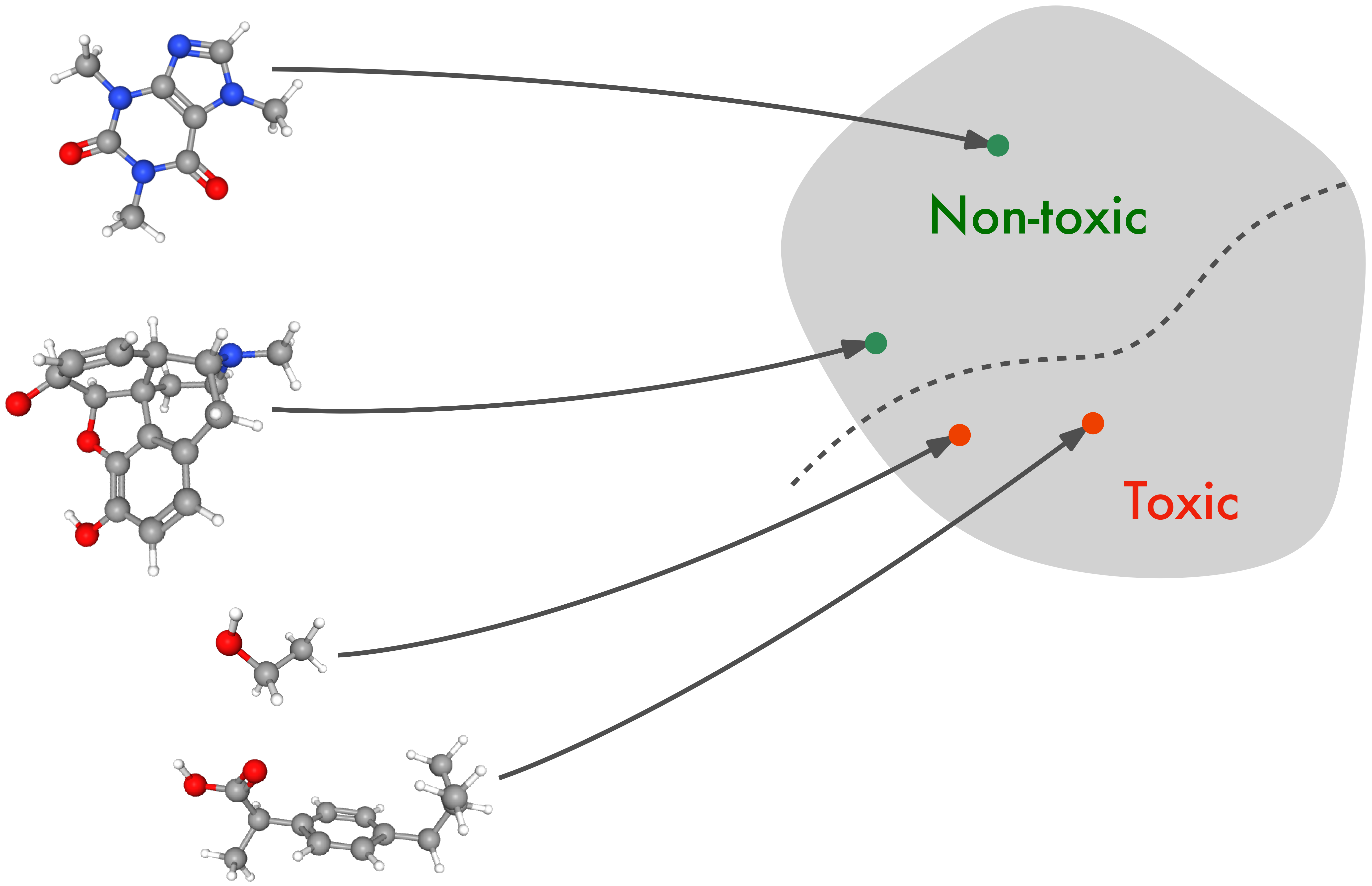
Computer Vision



Knowledge Bases

Motivation: A first example

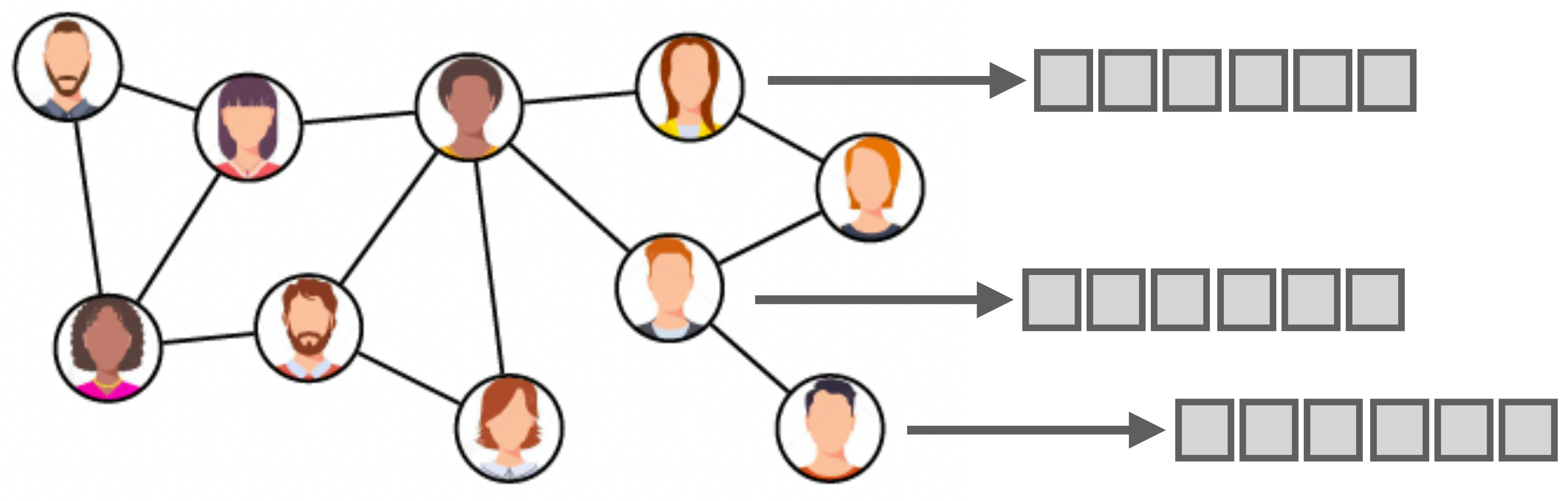
Learning of molecular properties



Learning with graphs: Two regimes

Node-level versus graph-level learning tasks

Node-level prediction

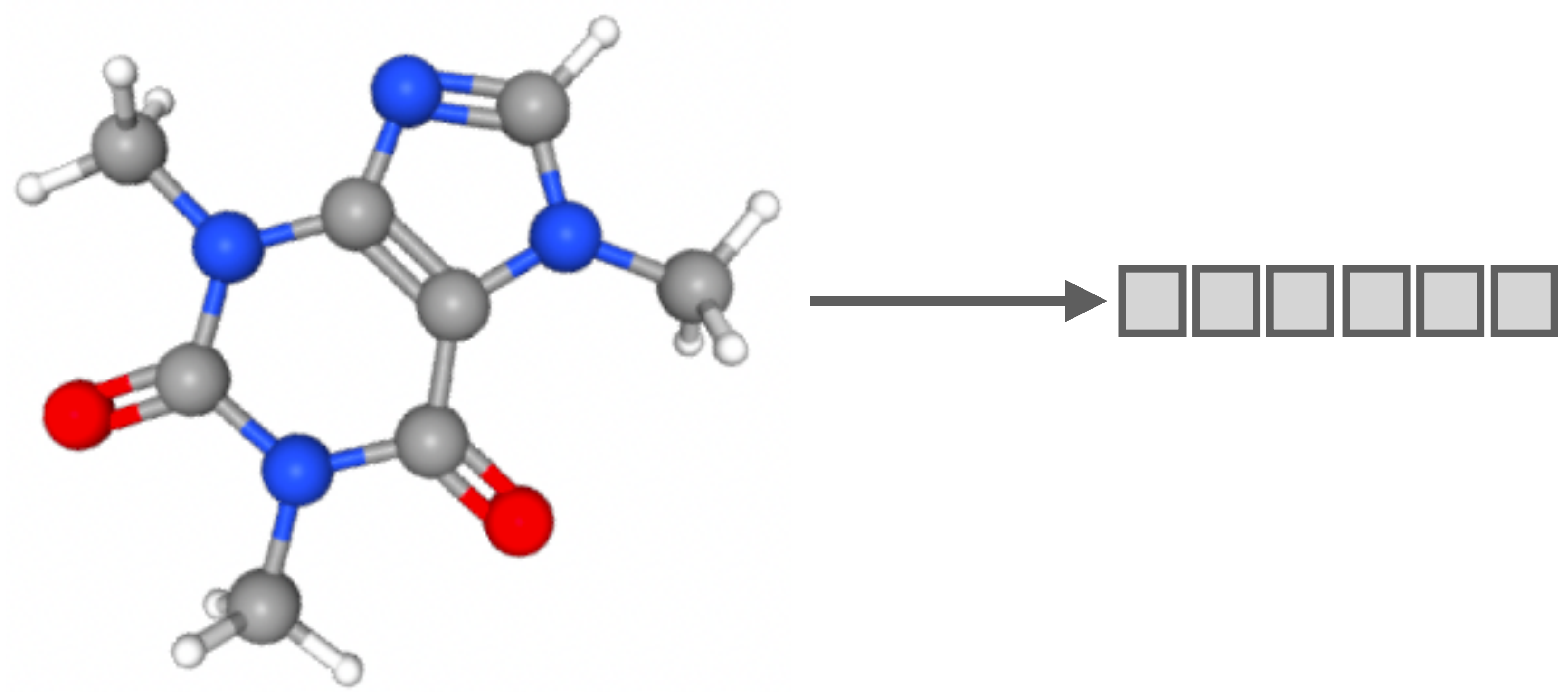


Social Networks

Make prediction for every node in the graph

versus

Graph-level prediction

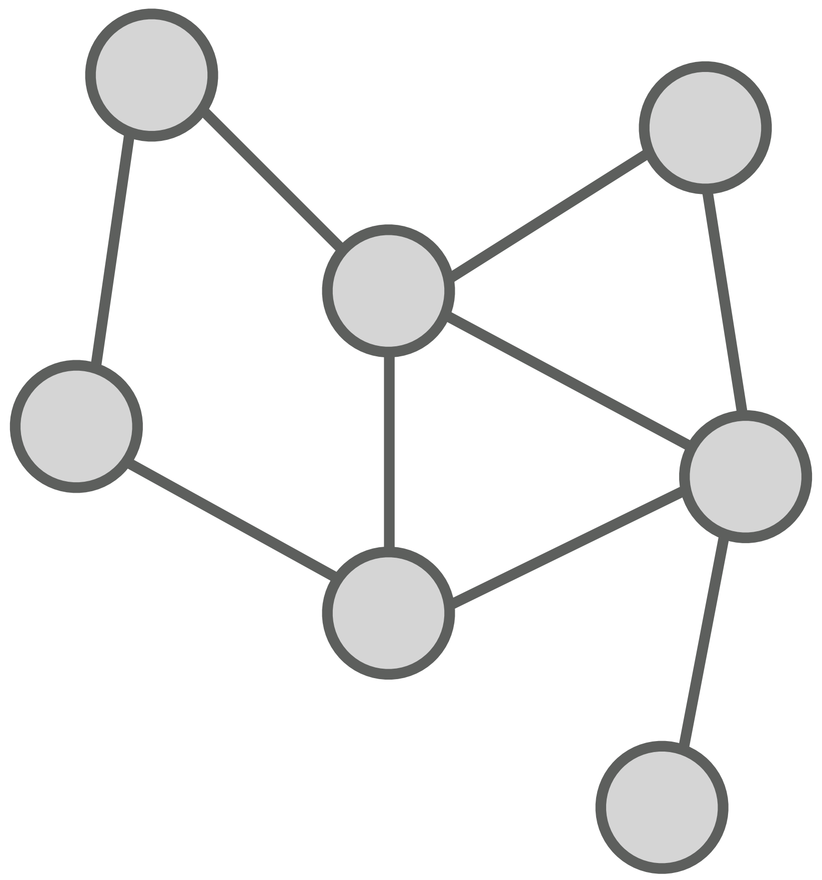


Chemical Molecules

Make prediction for whole graphs

Challenges of graph-structured data

Graphs versus images



Graph:
Non-regular structure

versus

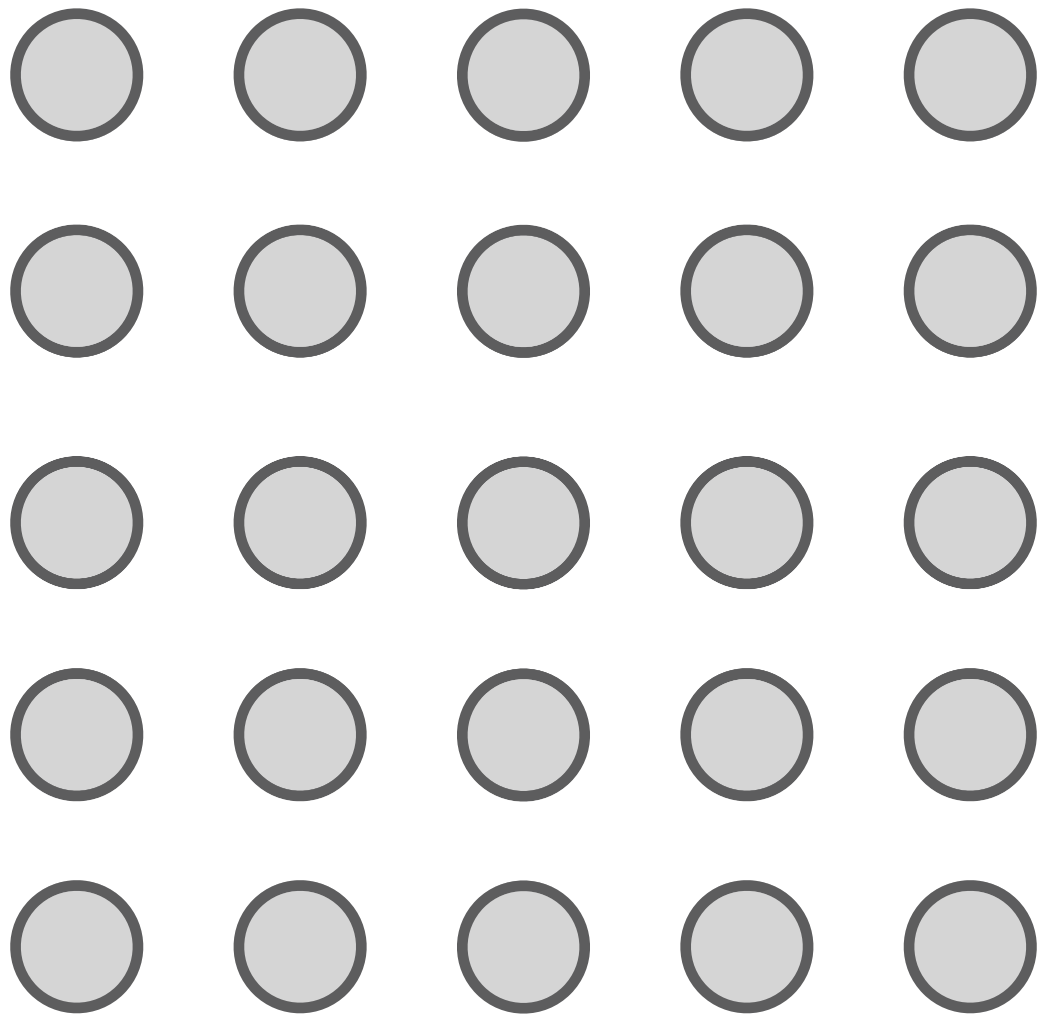
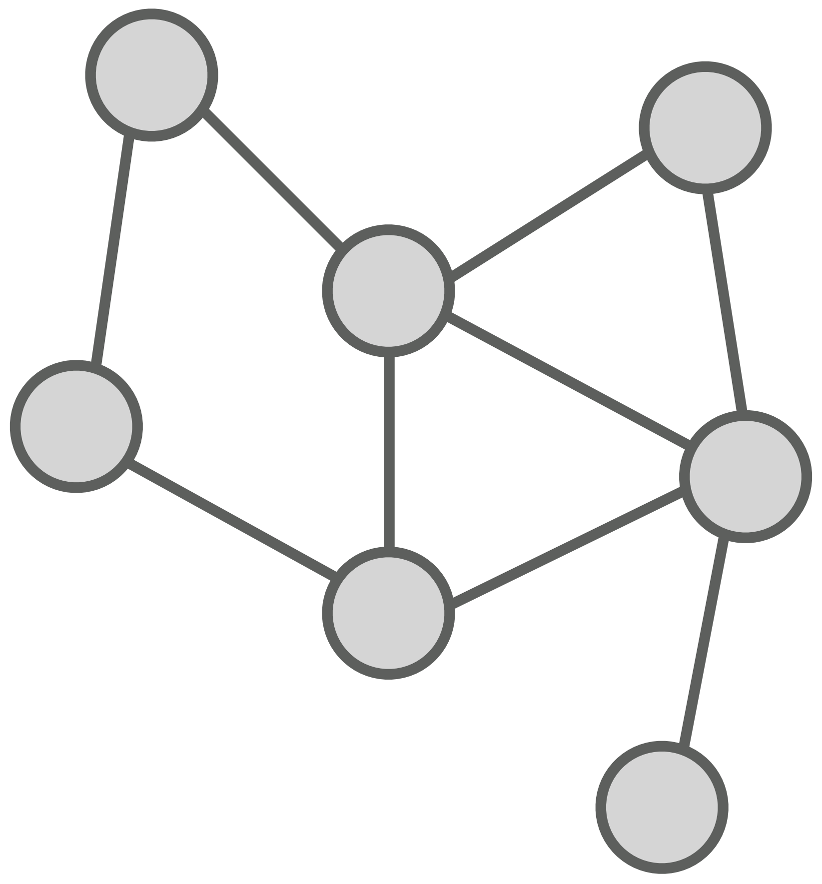


Image:
Regular structure

Insight

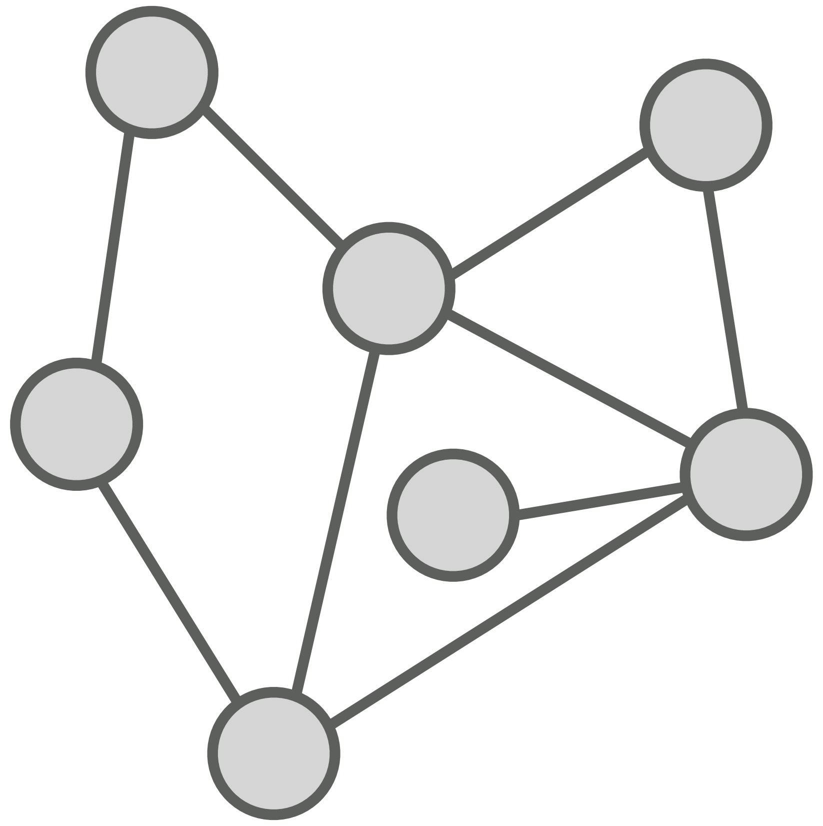
Graphs *do not* have a *regular structure*.

Challenges of graph-structured data



Graph G

versus

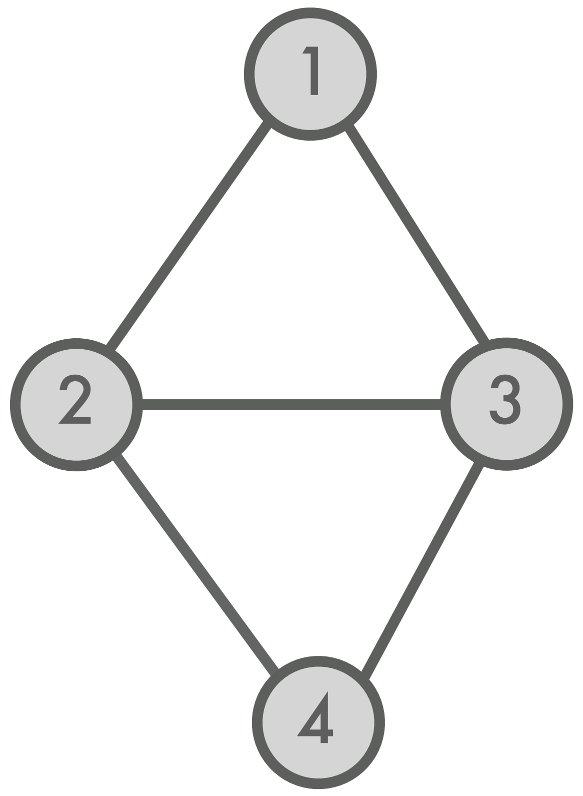


Graph H

Insight

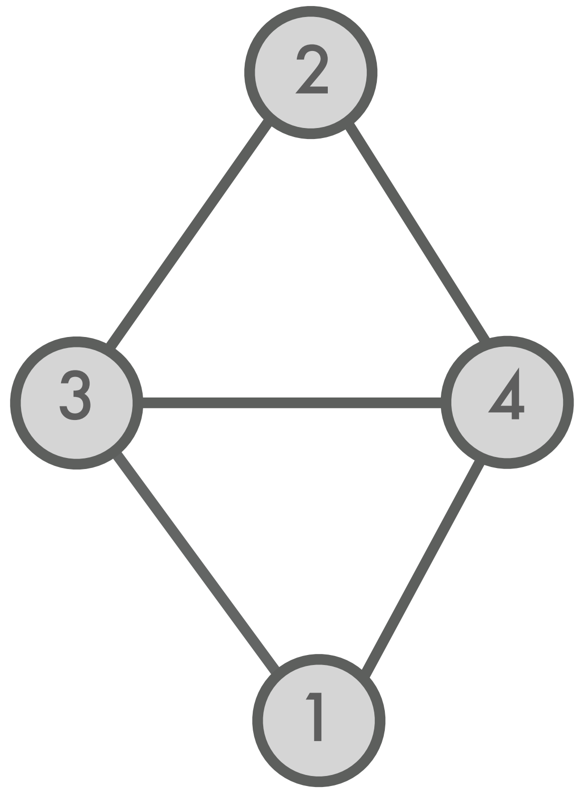
Graphs *do not* have a *unique representation*.

Challenges of graph-structured data



Graph G

versus



Graph H

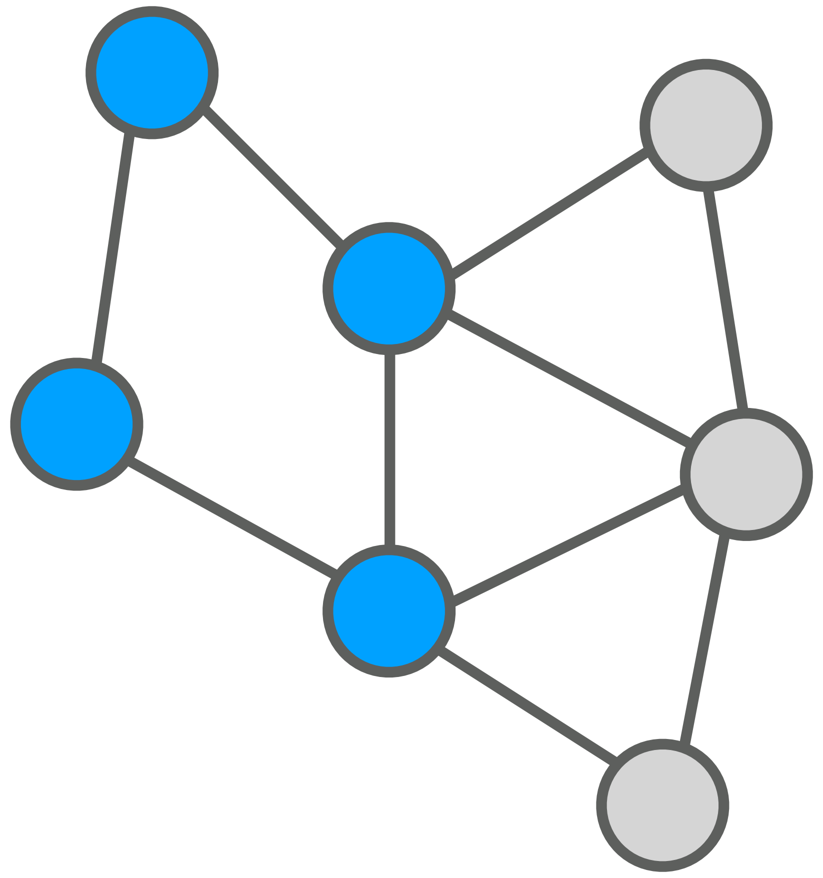
Insight

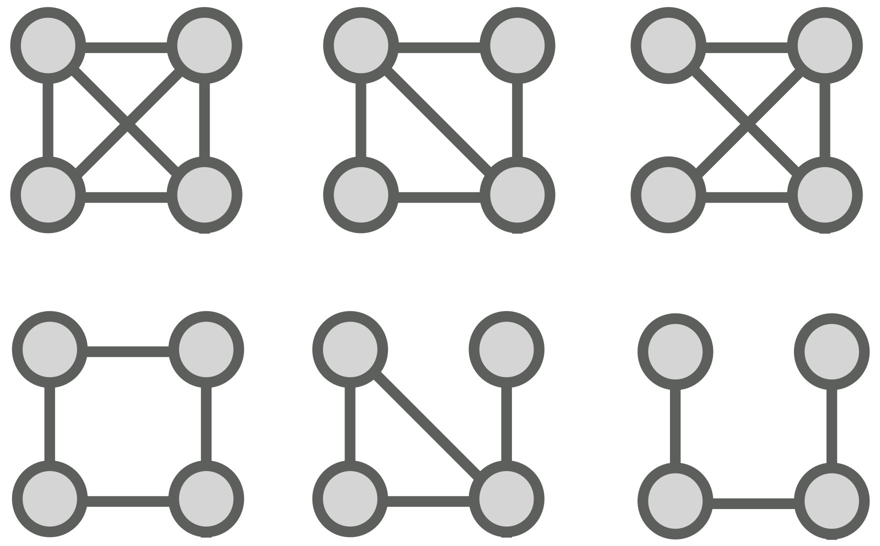
Graphs *do not* have a *unique representation*.

Pre-neural approaches to learning with graphs

Pre-neural approaches to learning with graphs

Subgraph-based approaches





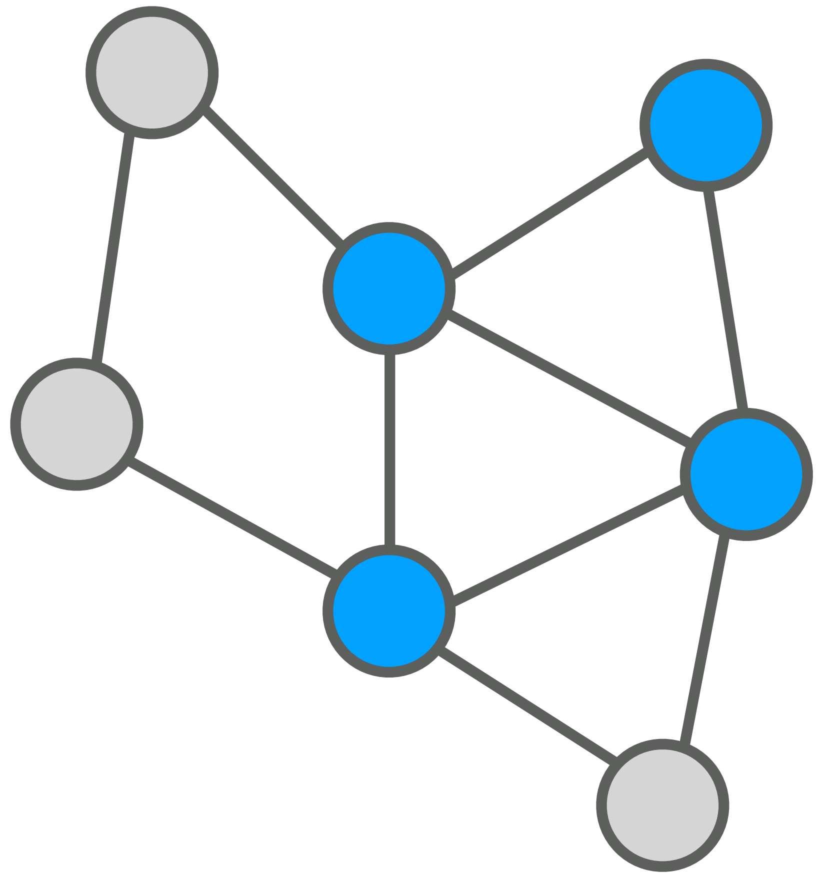
Connected graphs on 4 nodes

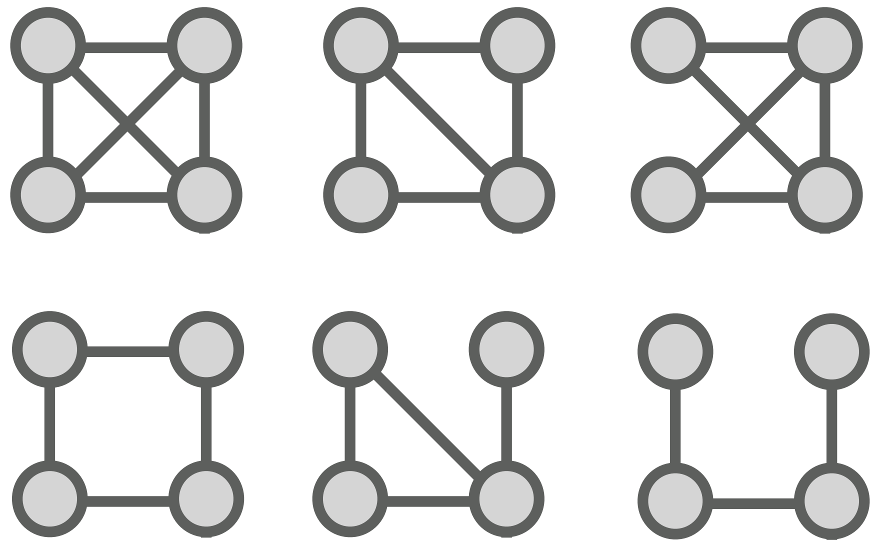
Idea

Count different connected subgraphs, e.g., on 4 nodes.

Pre-neural approaches to learning with graphs

Subgraph-based approaches





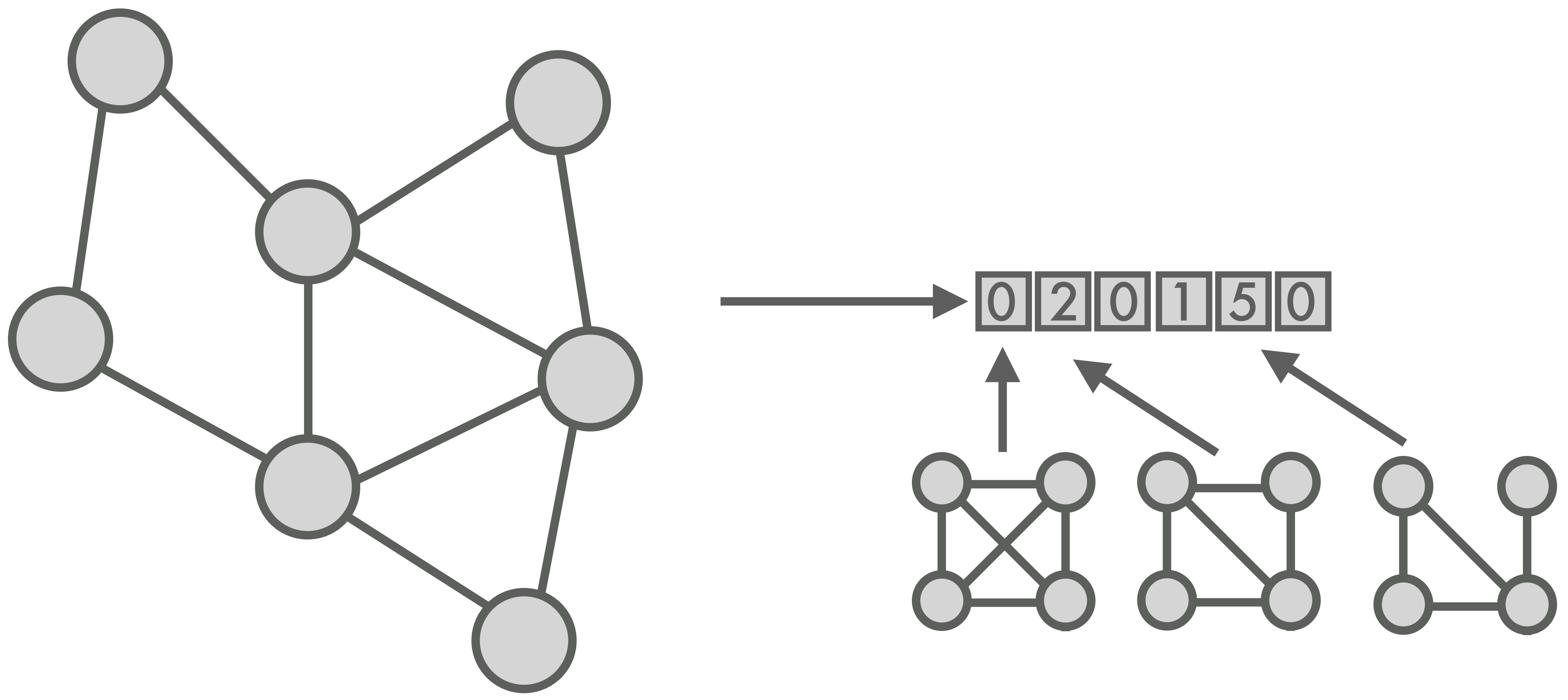
Connected graphs on 4 nodes

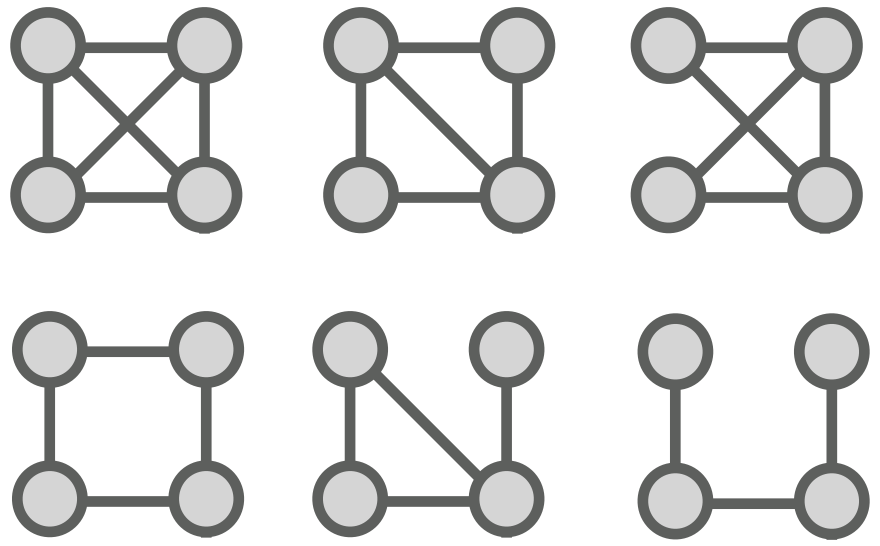
Idea

Count different connected subgraphs, e.g., on 4 nodes.

Pre-neural approaches to learning with graphs

Subgraph-based approaches





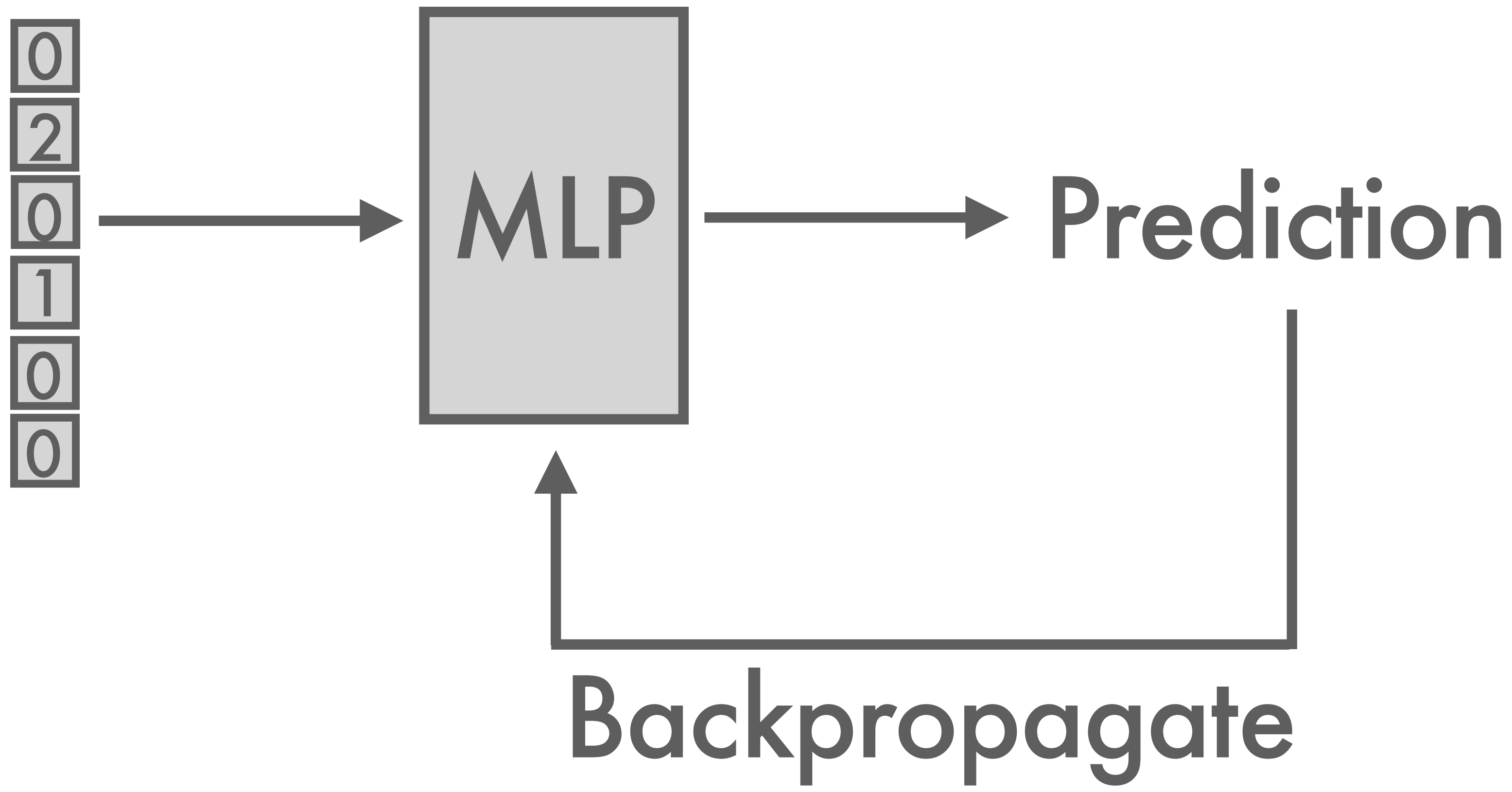
Connected graphs on 4 nodes

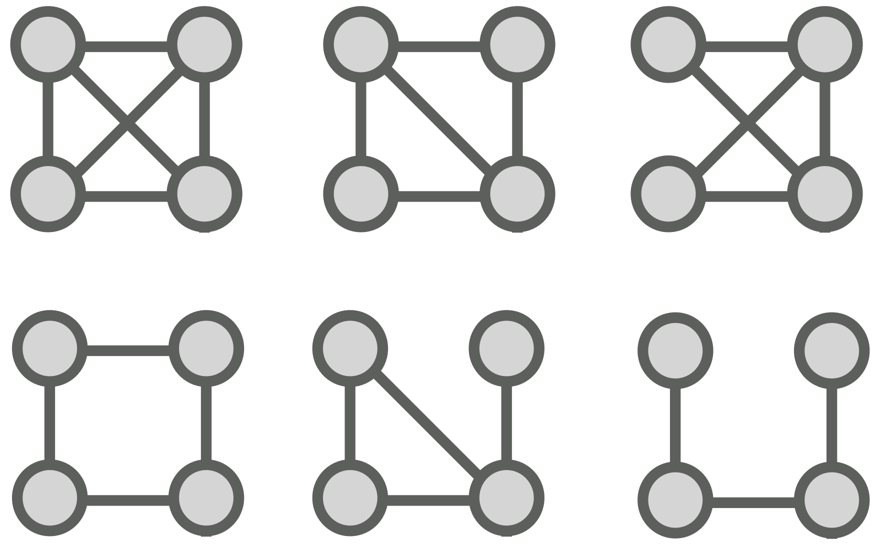
Idea

Count different connected subgraphs, e.g., on 4 nodes.

Pre-neural approaches to learning with graphs

Subgraph-based approaches





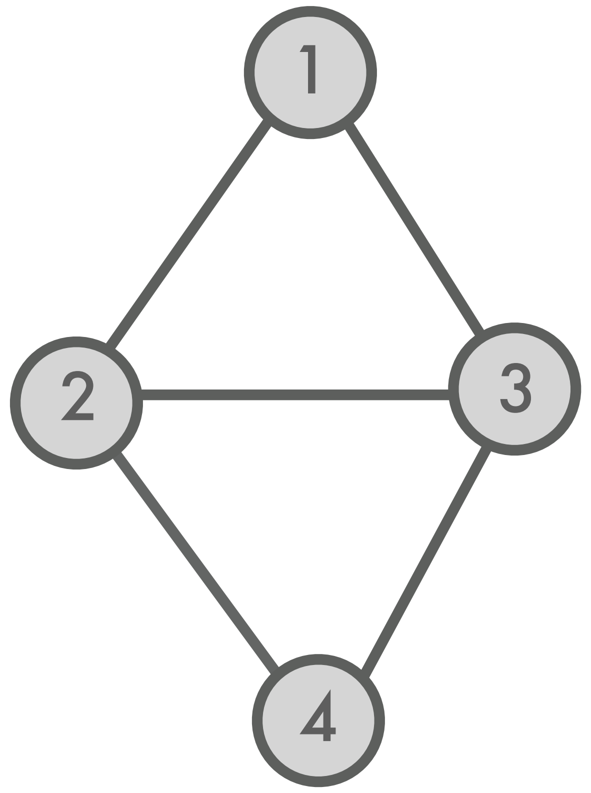
Connected graphs on 4 nodes

Idea

Count different connected subgraphs, e.g., on 4 nodes.

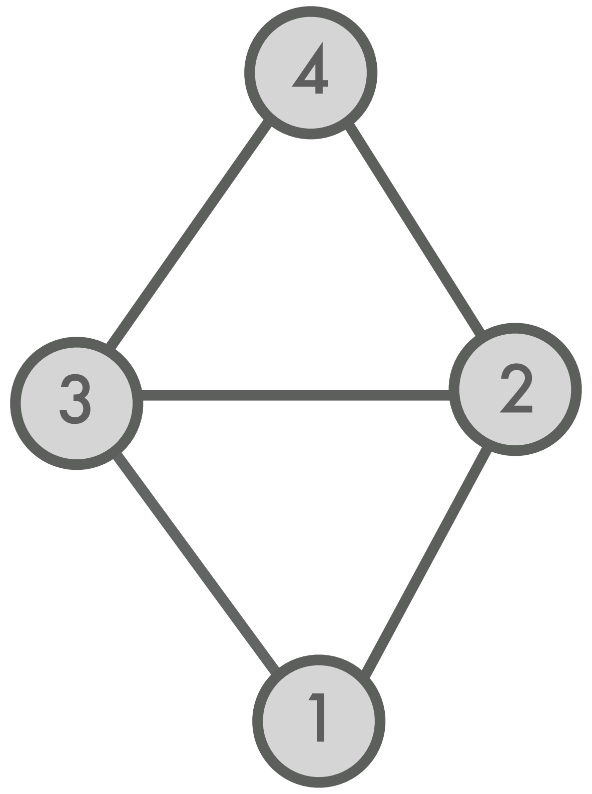
Weisfeiler-Leman Algorithm

A simple algorithm for the *graph isomorphism problem*



Graph G

versus



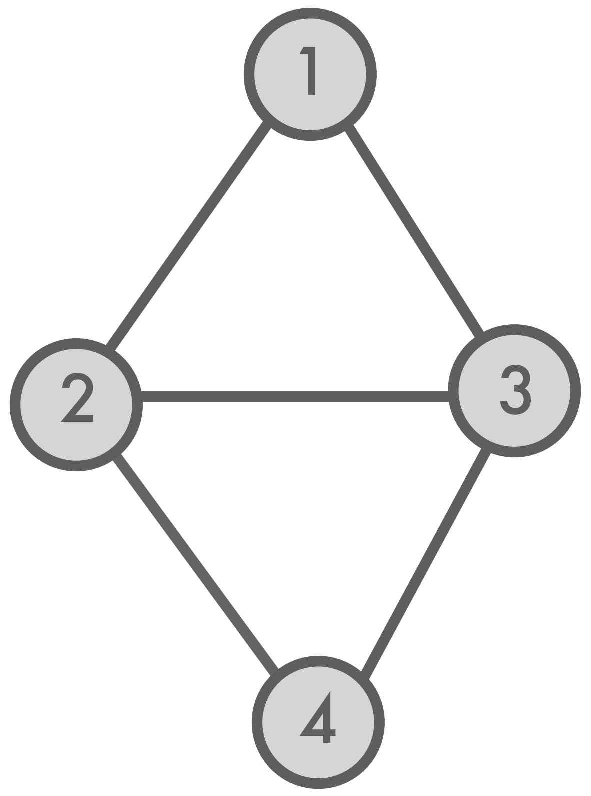
Graph H

Defintion: Graph isomorphism

Two graphs G, H are *isomorphic* if there exists a bijection $\phi\colon V(G) \to V(H)$ such that $(u, v) \in E(G)$ if and only if $(\phi(u), \phi(v)) \in E(H)$.

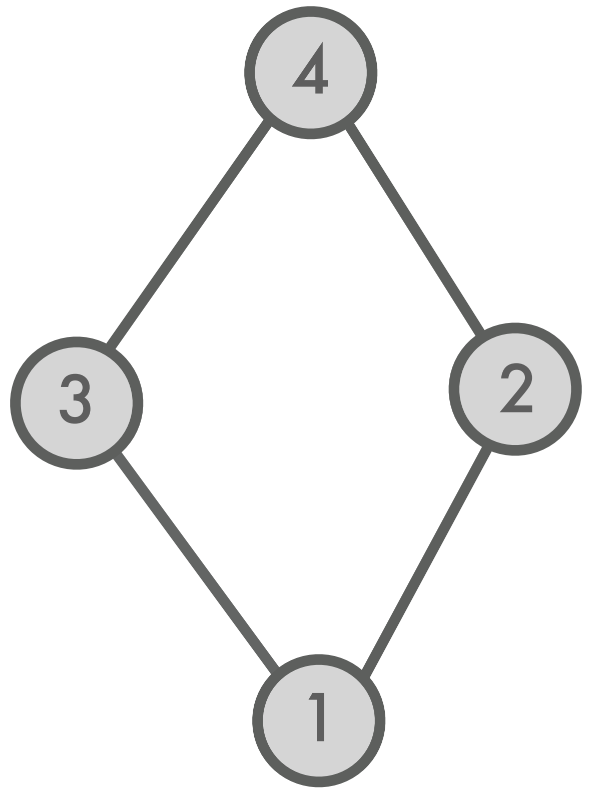
Weisfeiler-Leman Algorithm

A simple algorithm for the *graph isomorphism problem*



Graph G

versus



Graph H

Defintion: Graph isomorphism

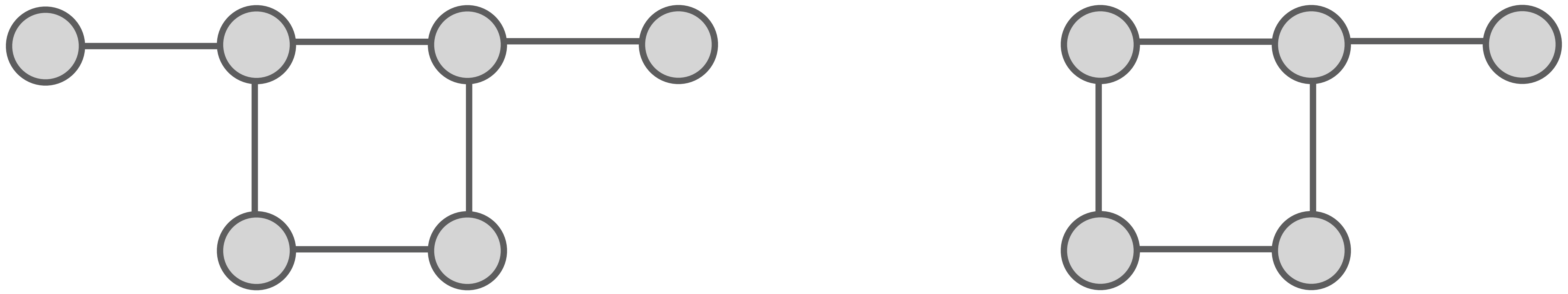
Two graphs G, H are *isomorphic* if there exists a bijection $\phi\colon V(G) \to V(H)$ such that $(u, v) \in E(G)$ if and only if $(\phi(u), \phi(v)) \in E(H)$.

Weisfeiler-Leman Algorithm

A simple algorithm for the *graph isomorphism problem*

Idea of the algorithm

Iteratively colors nodes based on colors of neighbors.



Weisfeiler-Leman Algorithm

A simple algorithm for the *graph isomorphism problem*

Idea of the algorithm

Iteratively colors nodes based on colors of neighbors.

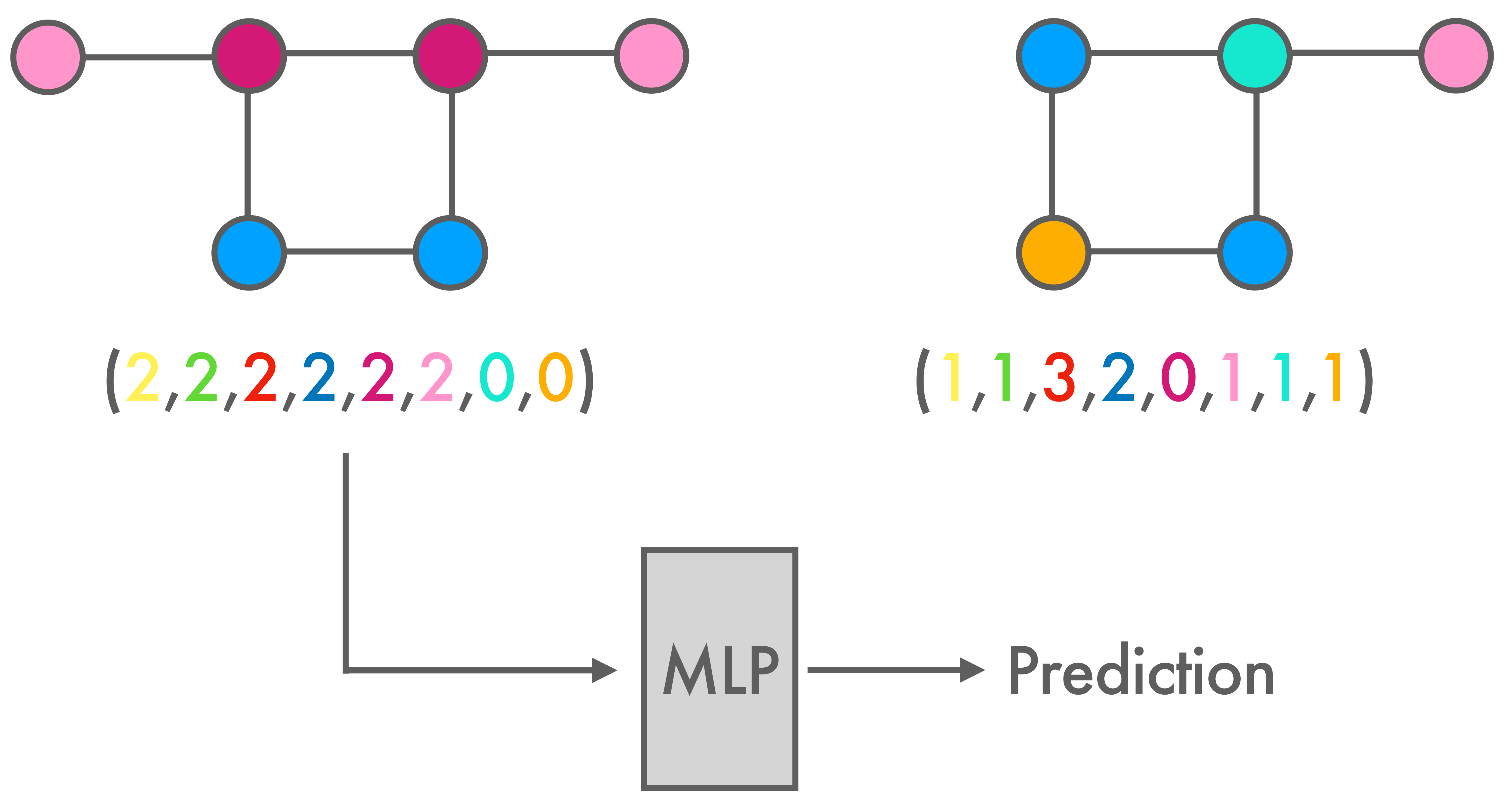
(2,2,2,0,0,0,0,0) (1,1,3,0,0,0,0,0)

Weisfeiler-Leman Algorithm

A simple algorithm for the *graph isomorphism problem*

Idea of the algorithm

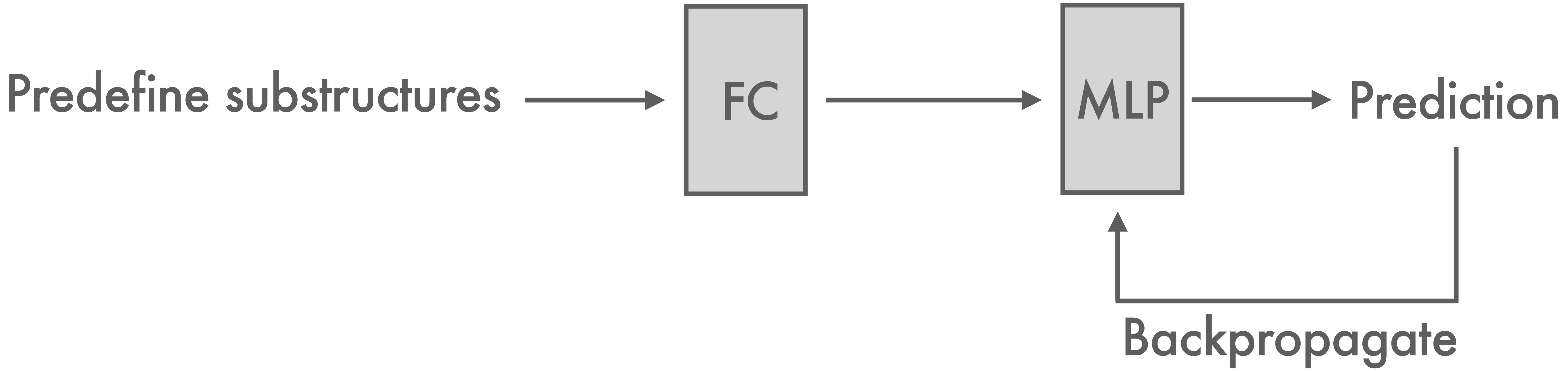
Iteratively colors nodes based on colors of neighbors.



Pre-neural approaches to learning with graphs

Idea of the algorithms

1. Extract substructures out of graph
2. Construct feature vector
3. Feed feature vector into MLP and train



Insight

Feature extraction is fixed and not part of the learning tasks.

Pre-neural approaches to learning with graphs

Idea of the algorithms

1. Extract substructures out of graph
2. Construct feature vector
3. Feed feature vector into MLP and train

- A Survey on Graph Kernels. Nils M. Kriege, Fredrik D. Johansson, Christopher Morris. Applied Network Science, Machine learning with graphs, 2020.
- K. M. Borgwardt, E. Ghisu, F. Llinares-López, L. O'Bray, and B. Rieck. Graph Kernels: State-of-the-Art and Future Challenges. Foundations and Trends in Machine Learning, 2020.

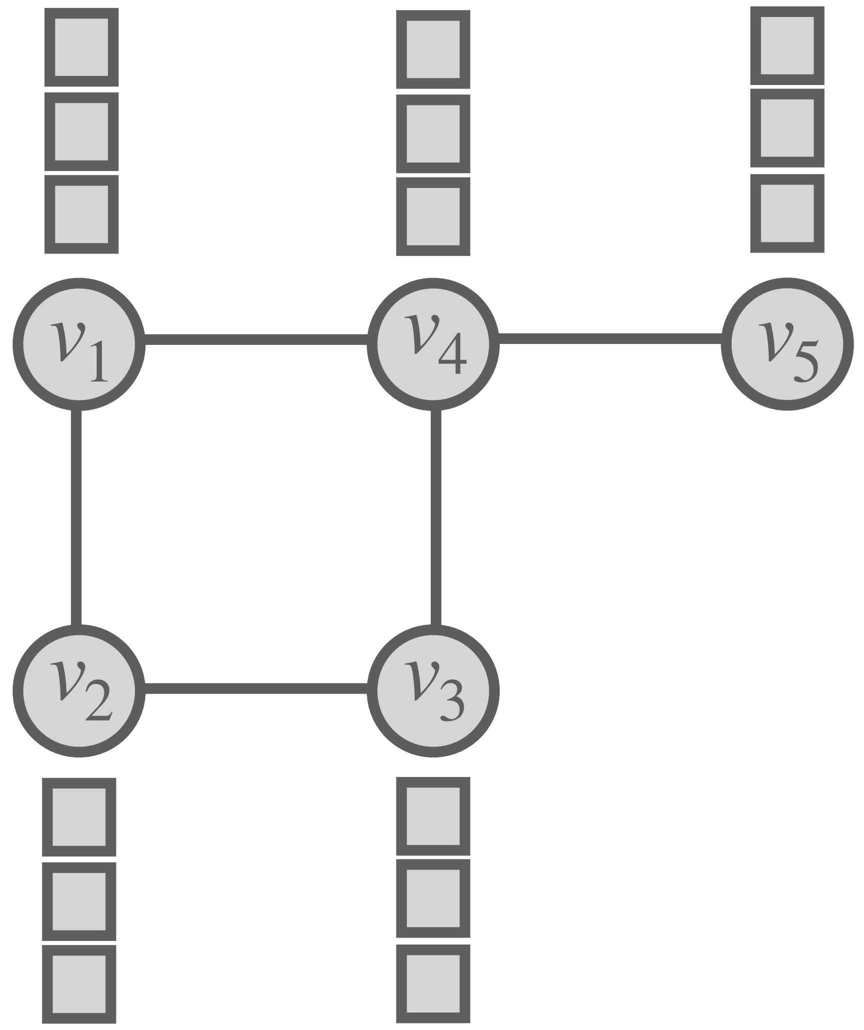
Introduction to Graph Neural Networks

Graph neural networks (GNNs)

Idea of graph neural networks

Aim

Learn d-dimensional vectorial representation of each node.



Idea of GNNs

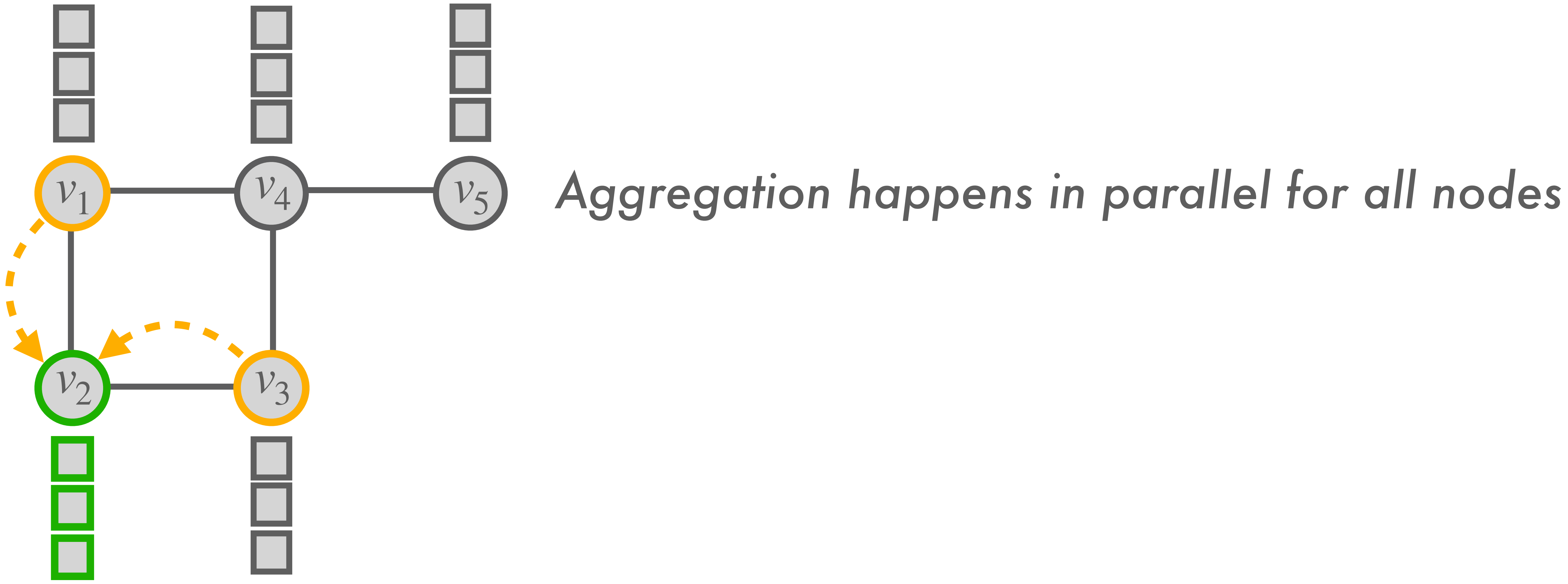
In each layer, aggregate features of neighbors to update feature of a node.

Graph neural networks (GNNs)

Idea of graph neural networks

Aim

Learn d-dimensional vectorial representation of each node.



Idea of GNNs

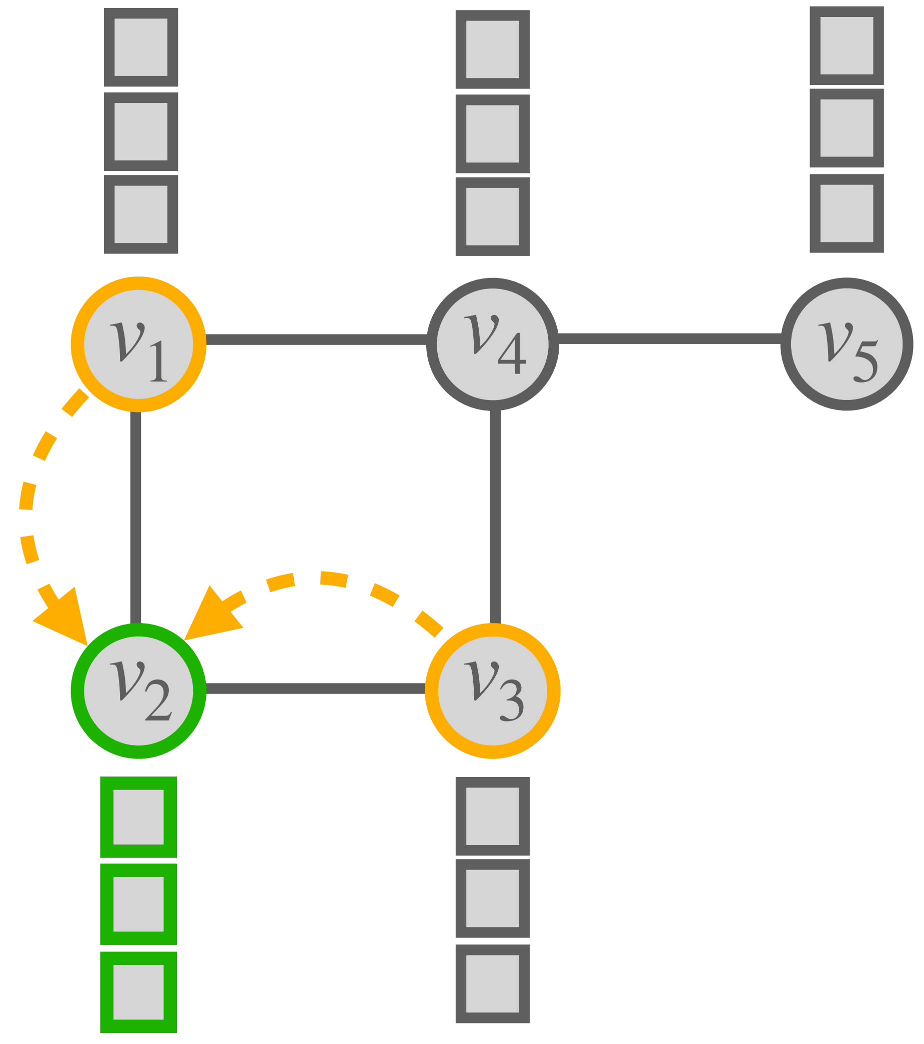
In each layer, aggregate features of neighbors to update feature of a node.

Graph neural networks (GNNs)

Idea of graph neural networks

Idea of GNNs

In each layer, aggregate features of neighbors to update feature of a node.



Aggregation happens in parallel for all nodes

Parameter matrices $\in \mathbb{R}^{d \times d}$

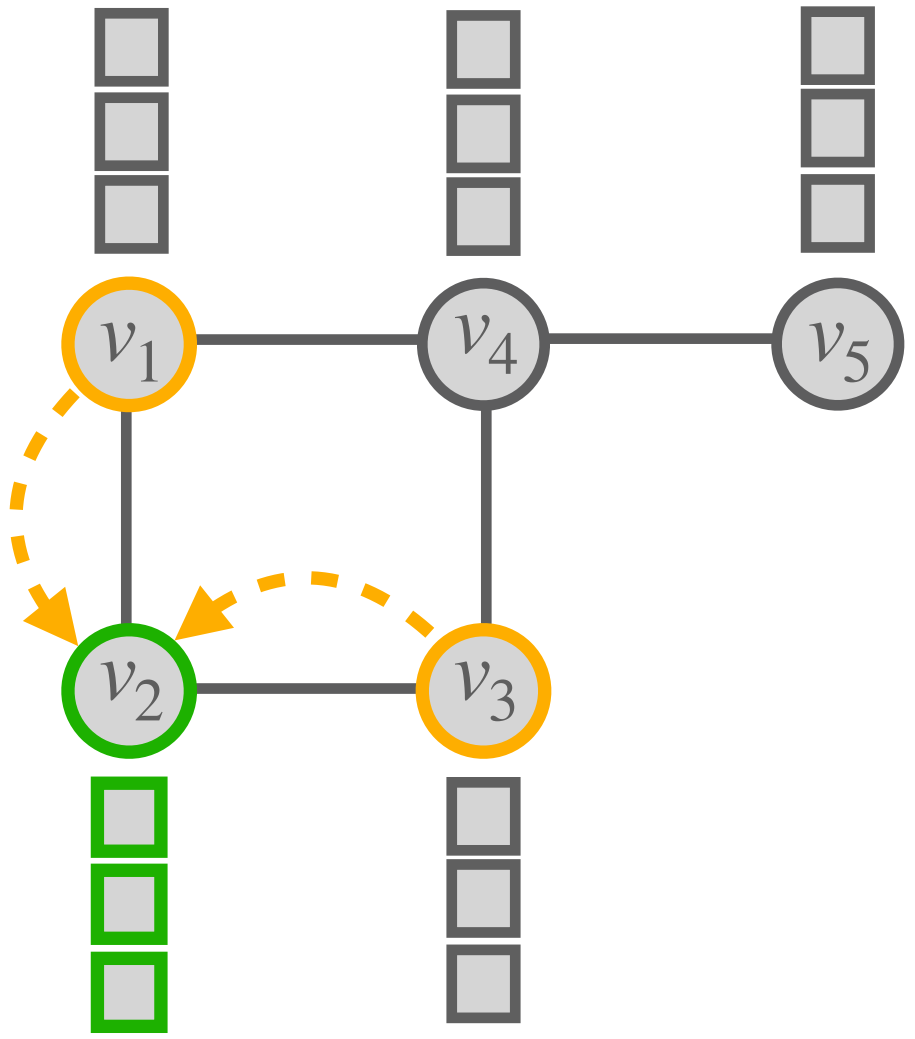
$$f^{(l)}(v) = \sigma\left(W_1 \cdot f^{(l-1)}(v) + W_2 \cdot \sum_{w \in N(v)} f^{(l-1)}(w)\right)$$

Graph neural networks (GNNs)

Idea of graph neural networks

Idea of GNNs

In each layer, aggregate features of neighbors to update feature of a node.

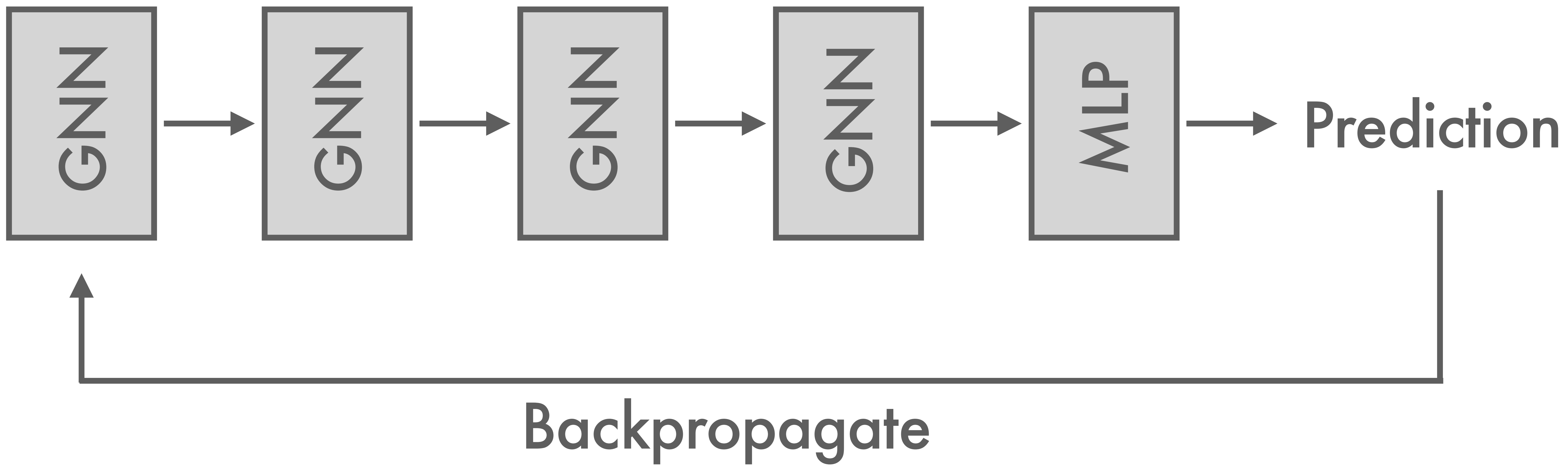


Aggregation happens in parallel for all nodes

$$f^{(l)}(v) = f_{\text{merge}}^{W_1}\Big(f^{(l-1)}(v), f_{\text{aggr}}^{W_2}\big(\{\!\{ f^{(l-1)}(w) \mid w \in N(v) \}\!\}\big)\Big)$$

Graph neural networks (GNNs)

Big picture



Training of GNNs

Train parameters of GNNs layers and MLP using gradient descent.

Graph neural networks (GNNs)

Flavors of Graph Neural Networks

- Simple GNN layer: $f^{(l)}(v) = \sigma\left(W_1 \cdot f^{(l-1)}(v) + W_2 \cdot \sum_{w \in N(v)} f^{(l-1)}(w) \right)$
- Message-passing NNs: $f^{(l)}(v) = f^{W_1}_{\textsf{merge}}\left(f^{(l-1)}(v), f^{W_2}_{\textsf{aggr}}\left(\{\!\{ f^{(l-1)}(w) \mid w \in N(v) \}\!\} \right) \right)$
- Graph Convolutional NNs: $f^{(l)}(v) = \sigma\left(W_1 \cdot \frac{1}{|N(v)| + 1} \sum_{w \in N(v) \cup \{v\}} \frac{1}{\sqrt{d_v}\sqrt{d_w}} f^{(l-1)}(w) \right)$
- Another 1000 more...

Graph neural networks (GNNs)

GraphSage

$$o^{(l)}(v) = W_1 \cdot f^{(l-1)}(v) + W_2 \cdot \frac{1}{|N(v)|} \sum_{w \in N(v)} f^{(l-1)}(w)$$

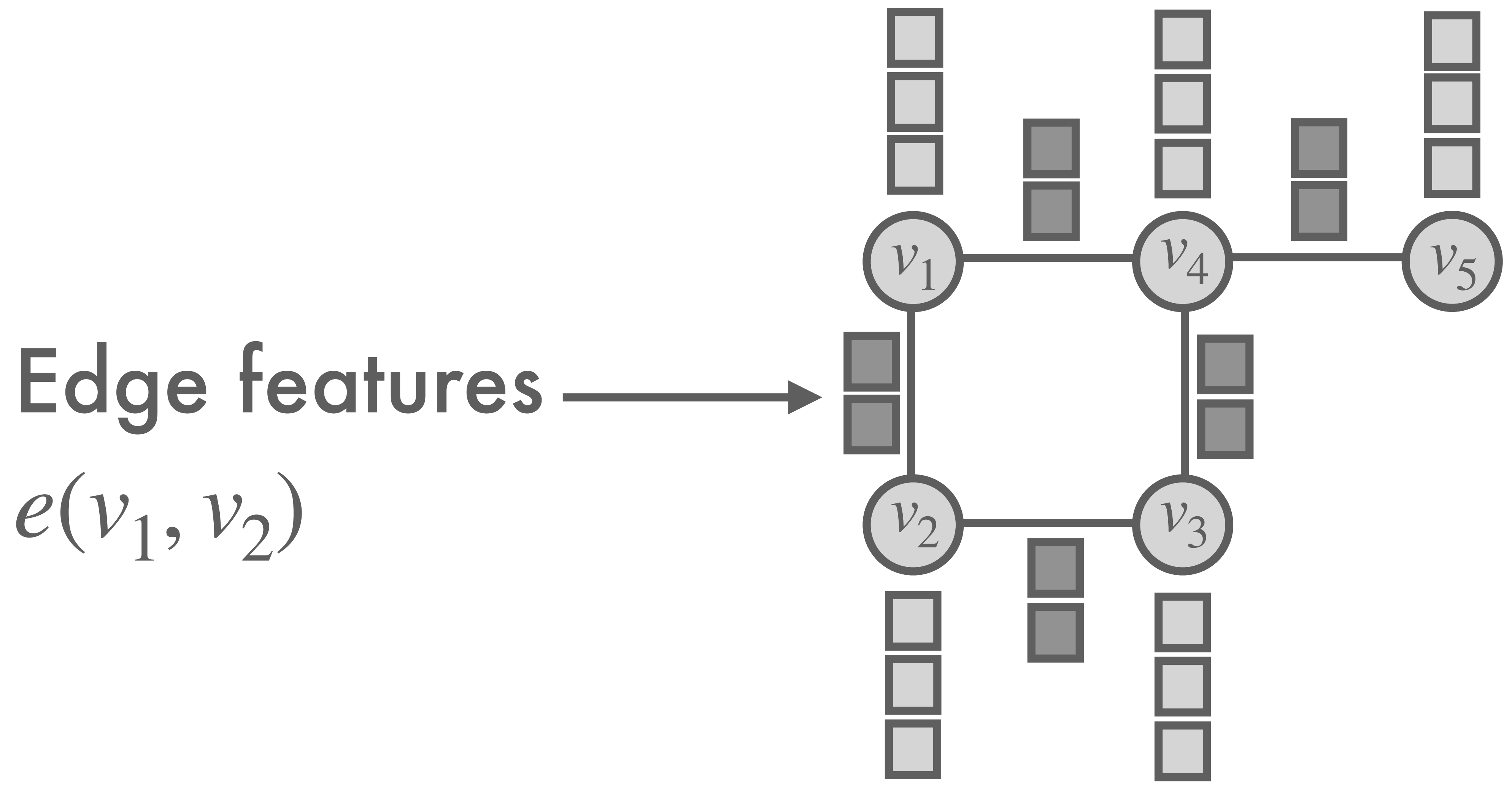
$$f^{(l)}(v) = \sigma\left(\frac{o^{(l)}}{\|o^{(l)}\|_2}\right)$$

Normalize features by ℓ_2 norm

Inductive Representation Learning on Large Graphs. W.L. Hamilton, R. Ying, and J. Leskovec. NeurIPS 2017

Graph neural networks (GNNs)

GNNs for graphs with edge features I

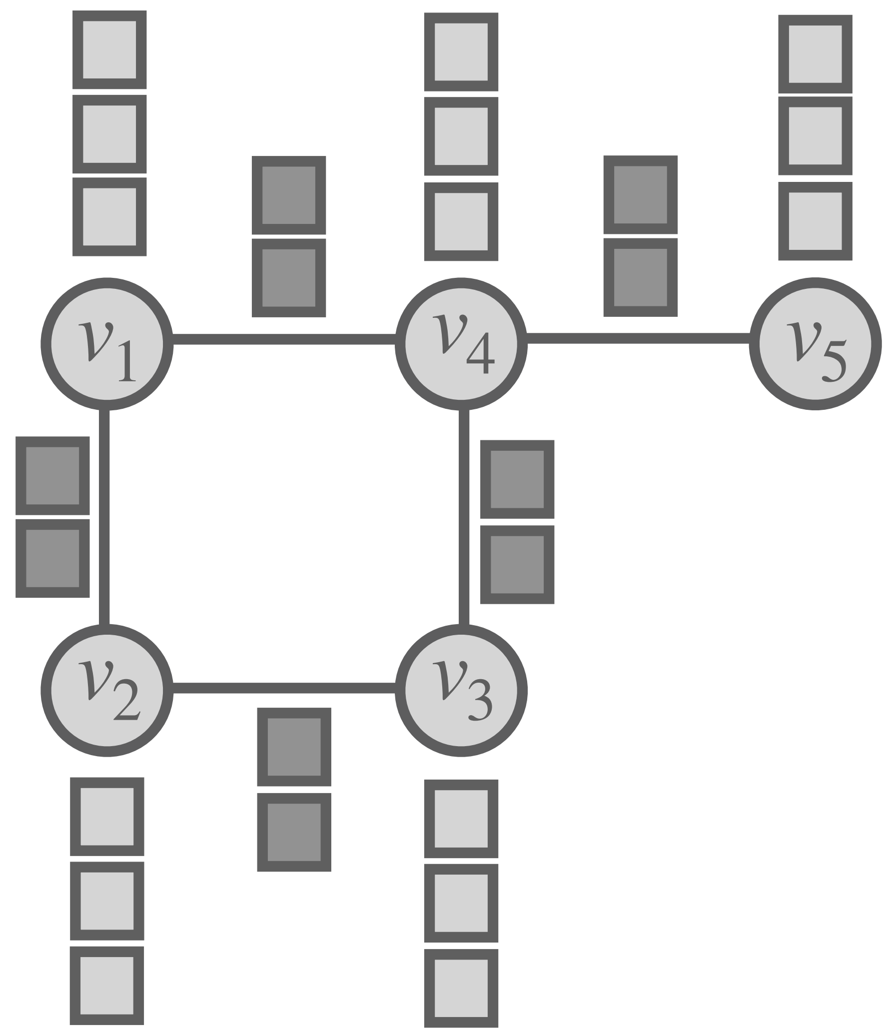


Concatenate edge feature with neighboring node feature:

$$f^{(l)}(v) = \sigma\left(W_1 \cdot f^{(l-1)}(v) + W_2 \cdot \sum_{w \in N(v)} \left[e(v,w), f^{(l-1)}(w) \right] \right)$$

Graph neural networks (GNNs)

GNNs for graphs with edge features II



Edge features $e(v_1, v_2)$

Use MLP to map edge feature to matrix and multiple with node feature

$$f^{(l)}(v) = \sigma\left(W_1 \cdot f^{(l-1)}(v) + W_2 \cdot \sum_{w \in N(v)} \text{MLP}(e(v,w)) \cdot f^{(l-1)}(w)\right)$$

Matrix $\in \mathbb{R}^{d \times d}$ Matrix $\in \mathbb{R}^{d \times 1}$

Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs. Martin Simonovsky, Nikos Komodakis. CVPR 2017

Graph neural networks (GNNs)

Graph Attention Networks (GAT)

Intuition behind GAT

Weight neighboring features differently during aggregation

$$f^{(l)}(v) = \sigma\left(\alpha_{v,v} W_1 \cdot f^{(l-1)}(v) + \sum_{w \in N(v)} \alpha_{v,w} W_2 \cdot f^{(l-1)}(w)\right)$$

Attention weight $\in \mathbb{R}$

Reminder softmax: $\frac{\exp(x_i)}{\sum_{j=1}^{K} \exp(x_j)}$

$$\alpha_{v,w} = \frac{\exp\left(\sigma\left(a^\top [W_3 \cdot f^{(l)}(v), W_3 \cdot f^{(l)}(w)]\right)\right)}{\sum_{w \in N(v) \cup \{v\}} \exp\left(\sigma\left(a^\top [W_3 \cdot f^{(l)}(v), W_3 \cdot f^{(l)}(w)]\right)\right)}$$

Graph Attention Networks. Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, Yoshua Bengio. ICLR 2018

Graph neural networks (GNNs)

Graph Isomorphism Networks (GIN)

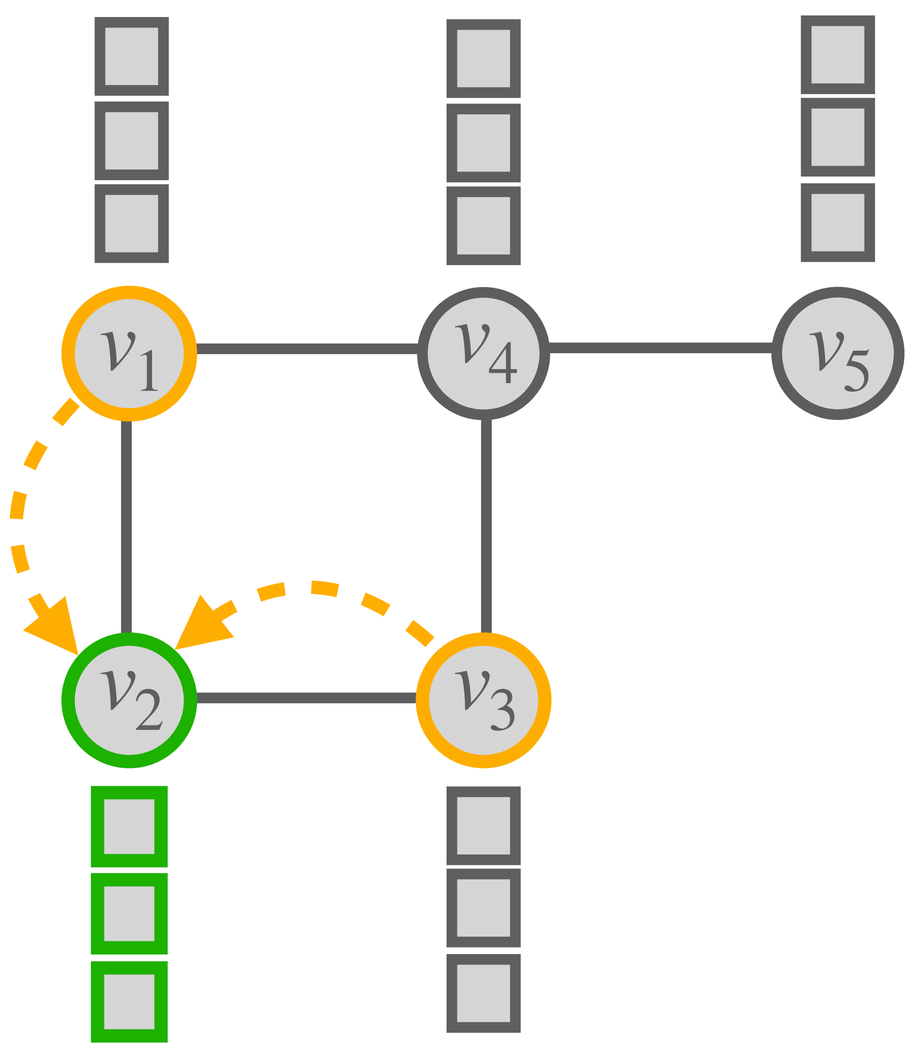
$$f^{(l)}(v) = \mathsf{MLP}\left((1+\epsilon)\cdot f^{(l-1)}(v) + \sum_{w\in N(v)} f^{(l-1)}(w)\right)$$

Learnable scalar $\in \mathbb{R}$

How Powerful are Graph Neural Networks?. Keyulu Xu, Weihua Hu, Jure Leskovec, Stefanie Jegelka. ICLR 2019

Graph neural networks (GNNs)

Pooling layers I



Question

How do we go from node features to a single graph feature?

Graph neural networks (GNNs)

Pooling layers II

- Sum pooling: $f(G) = \sum_{v \in V(G)} f^{(L)}(v)$

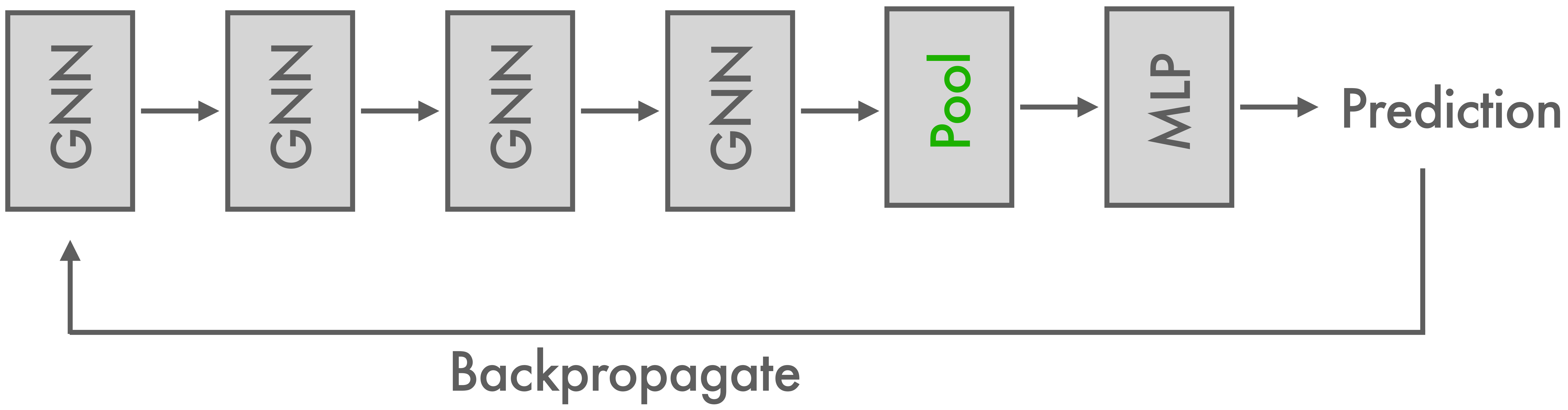
- Mean pooling: $f(G) = \frac{1}{|V(G)|} \sum_{v \in V(G)} f^{(L)}(v)$

- Max pooling: $f(G) = \max \left(\sum_{v \in V(G)} f^{(L)}(v) \right)$

- Many more sophisticated ones, e.g., based on differentiable clustering

Graph neural networks (GNNs)

GNNs with pooling



Training of GNNs

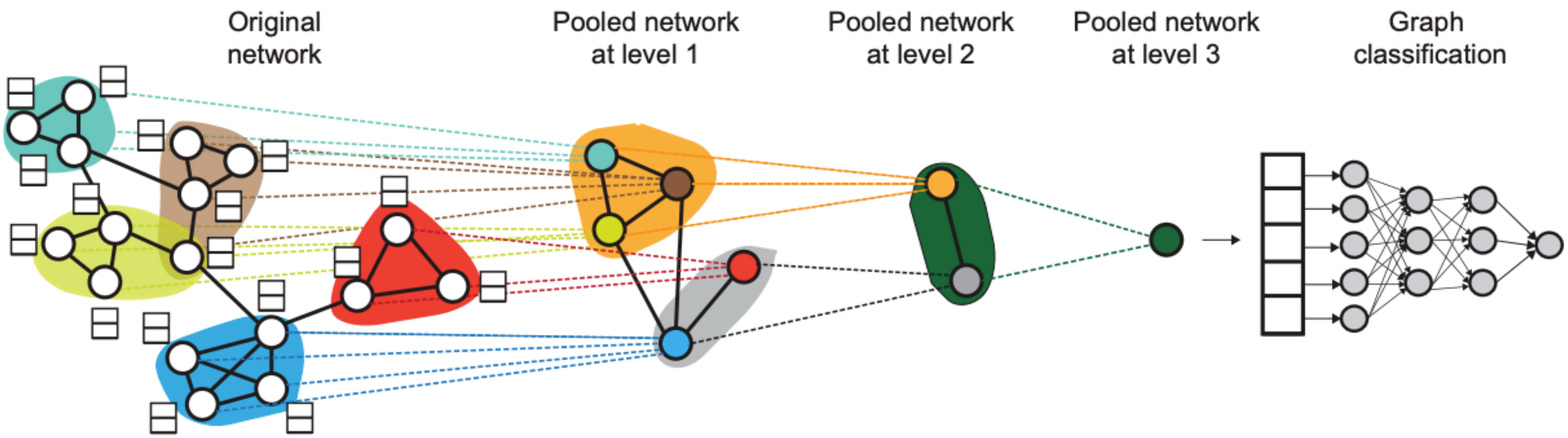
Train parameters of GNNs layers and MLP using gradient descent.

Graph neural networks (GNNs)

Pooling layers III, DiffPool

Intuition behind DiffPool

Coarsen graphs by clustering similar nodes together.



Hierarchical Graph Representation Learning with Differentiable Pooling. Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, Jure Leskovec. NeurIPS 2018.

Graph neural networks (GNNs)

Pooling layers III, DiffPool

Intuition behind DiffPool

Coarsen graphs by clustering similar nodes together.

new clusters

old clusters

$$\begin{bmatrix} 0.4 & 0.5 & 0.1 \\ 0.2 & 0.2 & 0.6 \\ 0.8 & 0.1 & 0.1 \\ 0.3 & 0.6 & 0.1 \end{bmatrix}$$

Reminder softmax: $\frac{\exp(x_i)}{\sum_{j=1}^{K} \exp(x_j)}$

$$S^{(l)} = \text{softmax}\left(\text{GNN}_{\text{Pool}}(A^{(l)}, F^{(l)})\right)$$

Graph representation at iteration l

Features at iteration l

Hierarchical Graph Representation Learning with Differentiable Pooling. Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, Jure Leskovec. NeurIPS 2018.

Graph neural networks (GNNs)

Pooling layers III, DiffPool

Intuition behind DiffPool

Coarsen graphs by clustering similar nodes together.

$$F^{(l+1)} = \underbrace{\begin{bmatrix} 0.4 & 0.2 & 0.8 & 0.3 \\ 0.5 & 0.2 & 0.1 & 0.6 \\ 0.1 & 0.6 & 0.1 & 0.1 \end{bmatrix}}_{\#\text{ new clusters} \times \#\text{ old clusters}} \bullet \underbrace{\begin{bmatrix} 4.2 & 2.5 & 0.1 \\ 1.2 & 4.2 & 0.6 \\ 2.8 & 6.1 & 4.1 \\ 3.3 & 4.6 & 4.1 \end{bmatrix}}_{F^{(l)}:\ \#\text{ old clusters} \times \#\text{ features}}$$

$$A^{(l+1)} = {S^{(l)}}^T A^{(l)} S^{(l)}$$

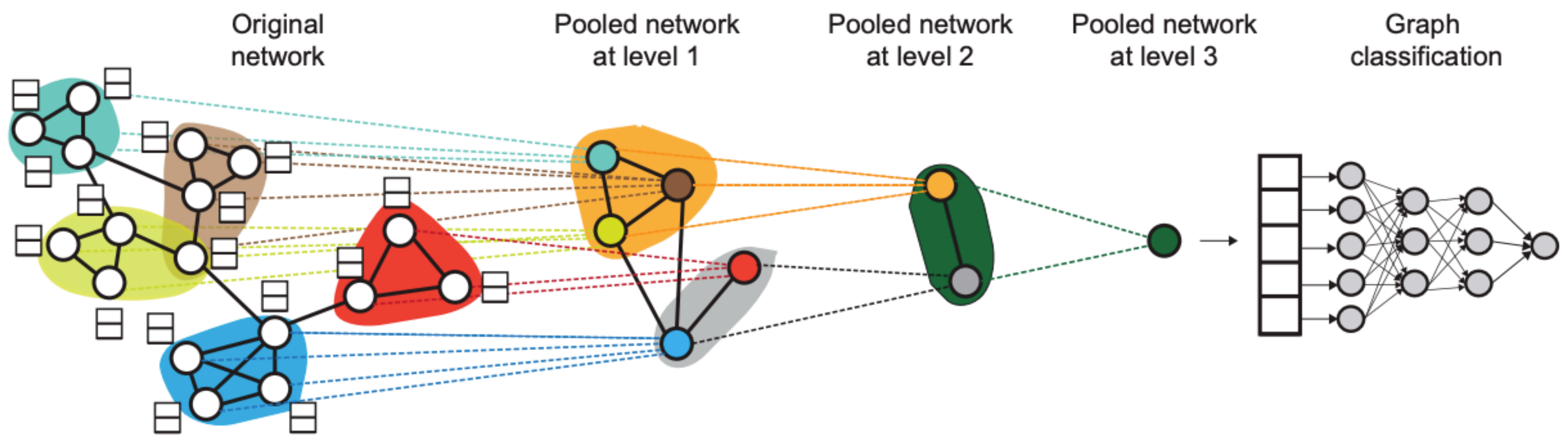
Hierarchical Graph Representation Learning with Differentiable Pooling. Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, Jure Leskovec. NeurIPS 2018.

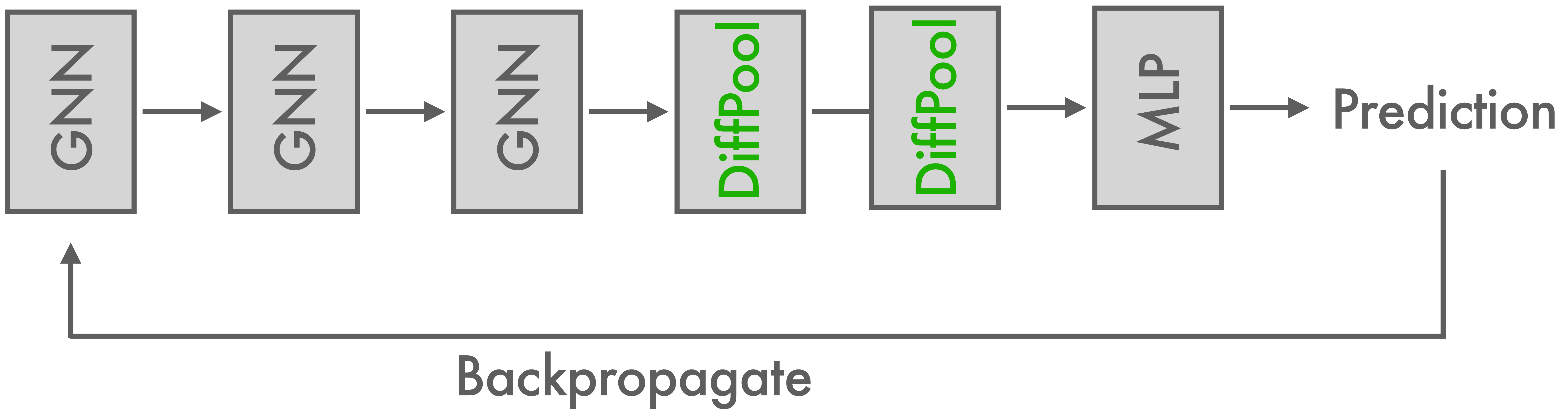
Graph neural networks (GNNs)

Pooling layers III, DiffPool

Intuition behind DiffPool

Coarsen graphs by clustering similar nodes together.





Hierarchical Graph Representation Learning with Differentiable Pooling. Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, Jure Leskovec. NeurIPS 2018.

Limitations of GNNs

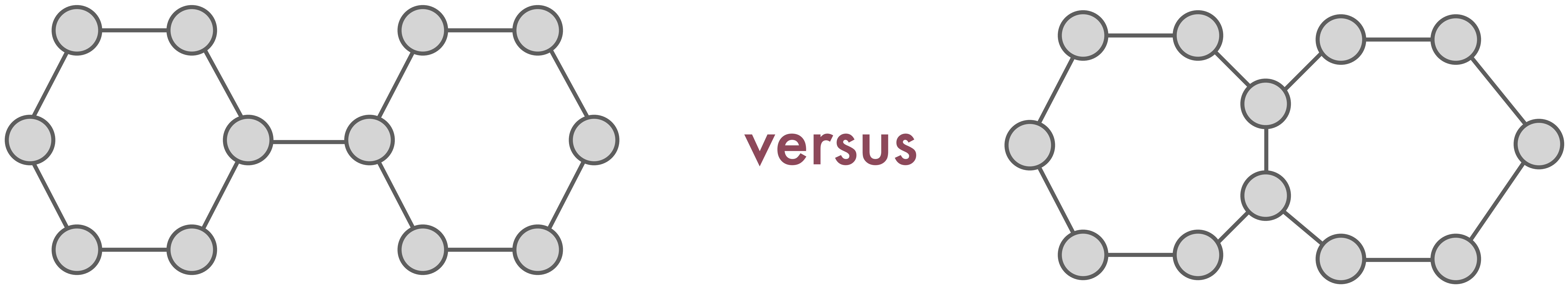
Graph neural networks (GNNs)

Limits of Graph Neural Networks

Questions

What are the limitations of graph neural networks?

- *Do there exist non-isomorphic graphs that cannot be distinguished by any possible GNN?*



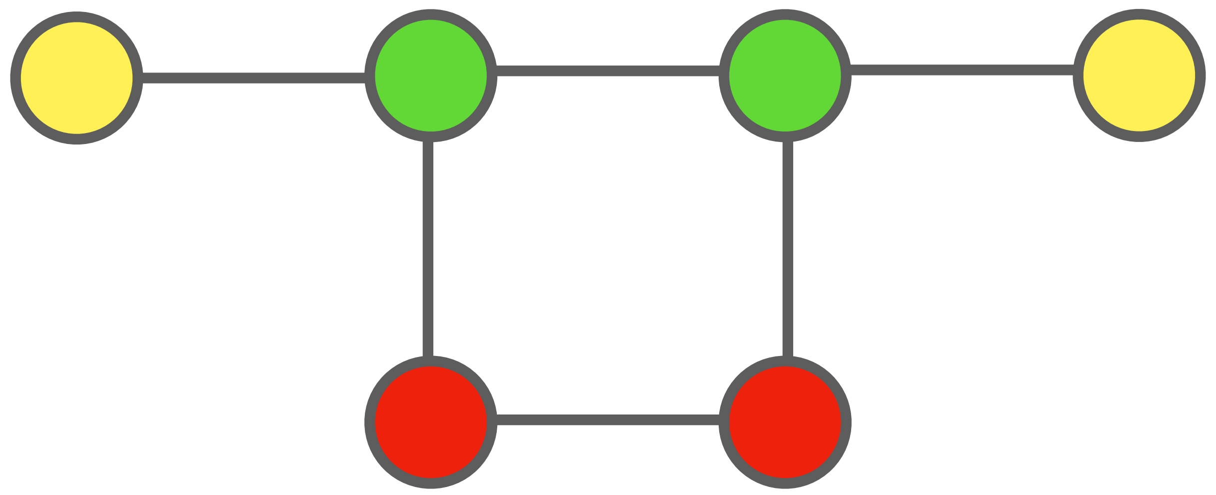
$$f^{(t)}(v) = f^{W_1}_{\text{merge}}\Big(f^{(t-1)}(v), f^{W_2}_{\text{aggr}}\big(\{\!\!\{ f^{(t-1)}(w) \mid w \in N(v) \}\!\!\}\big)\Big)$$

Weisfeiler-Leman Algorithm

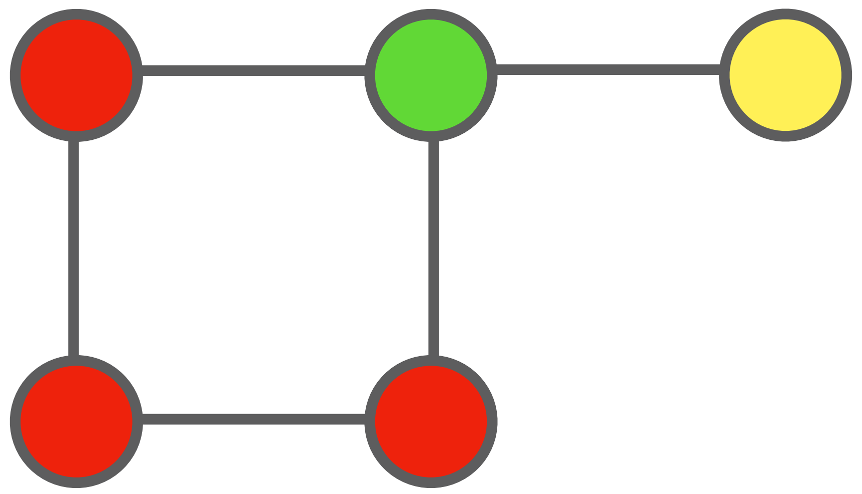
A simple algorithm for the *graph isomorphism problem*

Idea of the algorithm

Iteratively colors nodes based on colors of neighbors.



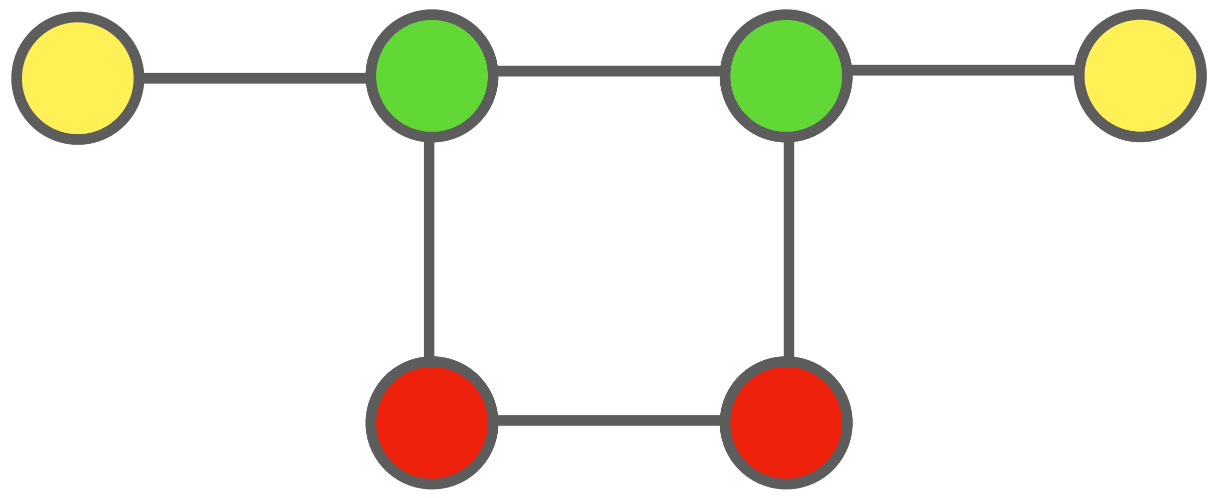
(2,2,2,0,0,0,0,0)



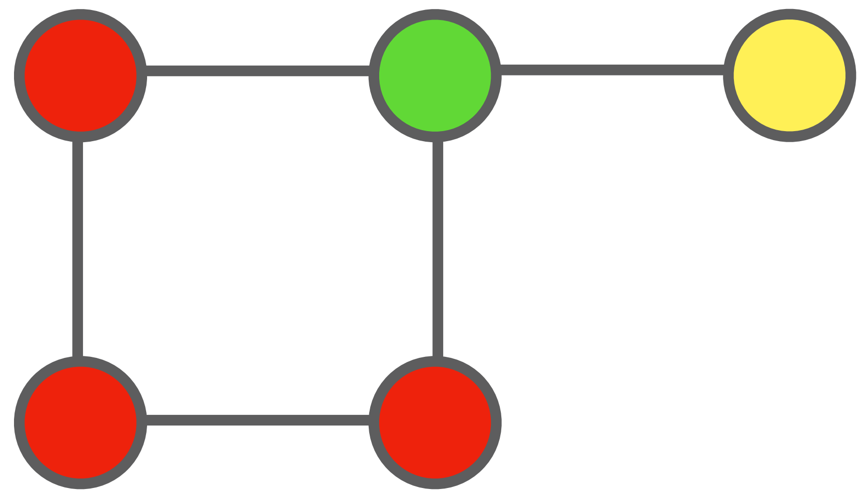
(1,1,3,0,0,0,0,0)

Graph neural networks (GNNs)

Limits of Graph Neural Networks



(2,2,2,0,0,0,0,0)



(1,1,3,0,0,0,0,0)

Coloring rule of the WL

$$c^{(t)}(v) = \mathsf{recolor}\Big(c^{(t-1)}(v), \{\!\{c^{(t-1)}(w) \mid w \in N(v)\}\!\}\Big)$$

versus

General form of GNNs

$$f^{(t)}(v) = f^{W_1}_{\mathsf{merge}}\Big(f^{(t-1)}(v), f^{W_2}_{\mathsf{aggr}}\big(\{\!\{f^{(t-1)}(w) \mid w \in N(v)\}\!\}\big)\Big)$$

Graph neural networks (GNNs)

Limits of Graph Neural Networks

Coloring rule of the WL

$$c^{(t)}(v) = \mathsf{hash}\Big(c^{(t-1)}(v), \{\!\{ c^{(t-1)}(w) \mid w \in N(v) \}\!\}\Big)$$

versus

General form of GNNs

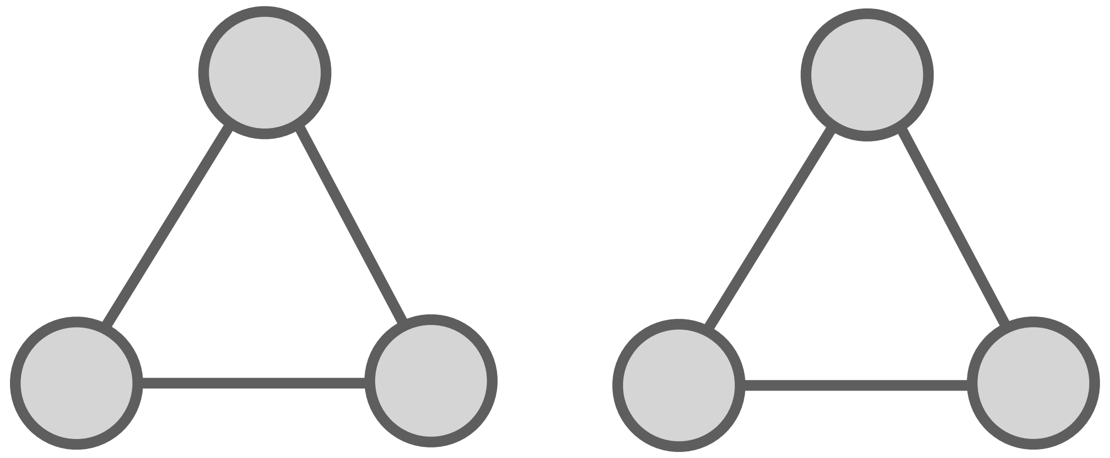
$$f^{(t)}(v) = f^{W_1}_{\mathsf{merge}}\Big(f^{(t-1)}(v), f^{W_2}_{\mathsf{aggr}}\big(\{\!\{ f^{(t-1)}(w) \mid w \in N(v) \}\!\}\big)\Big)$$

Theorem (Informal)

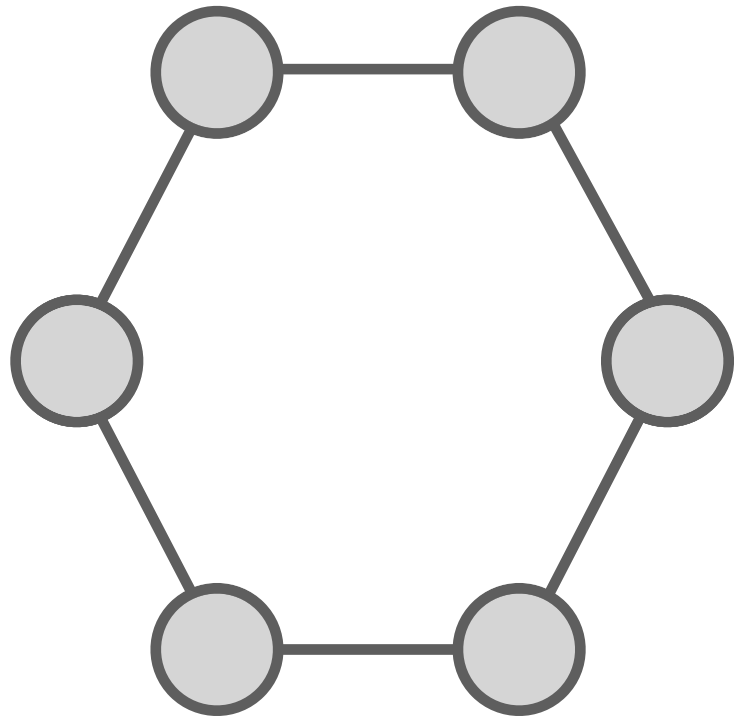
GNNs cannot be expressive than the WL algorithm in terms of distinguishing non-isomorphic graphs.

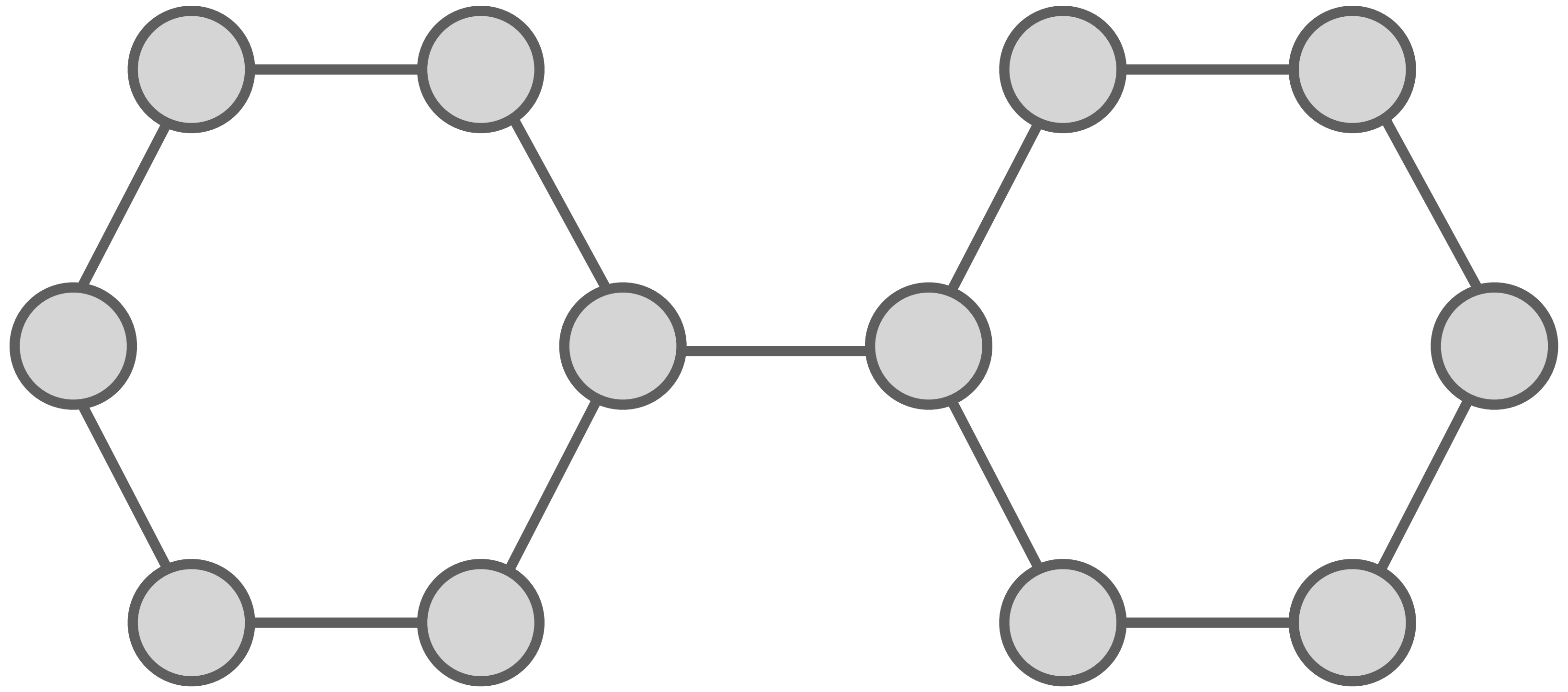
Graph neural networks (GNNs)

Limits of Graph Neural Networks

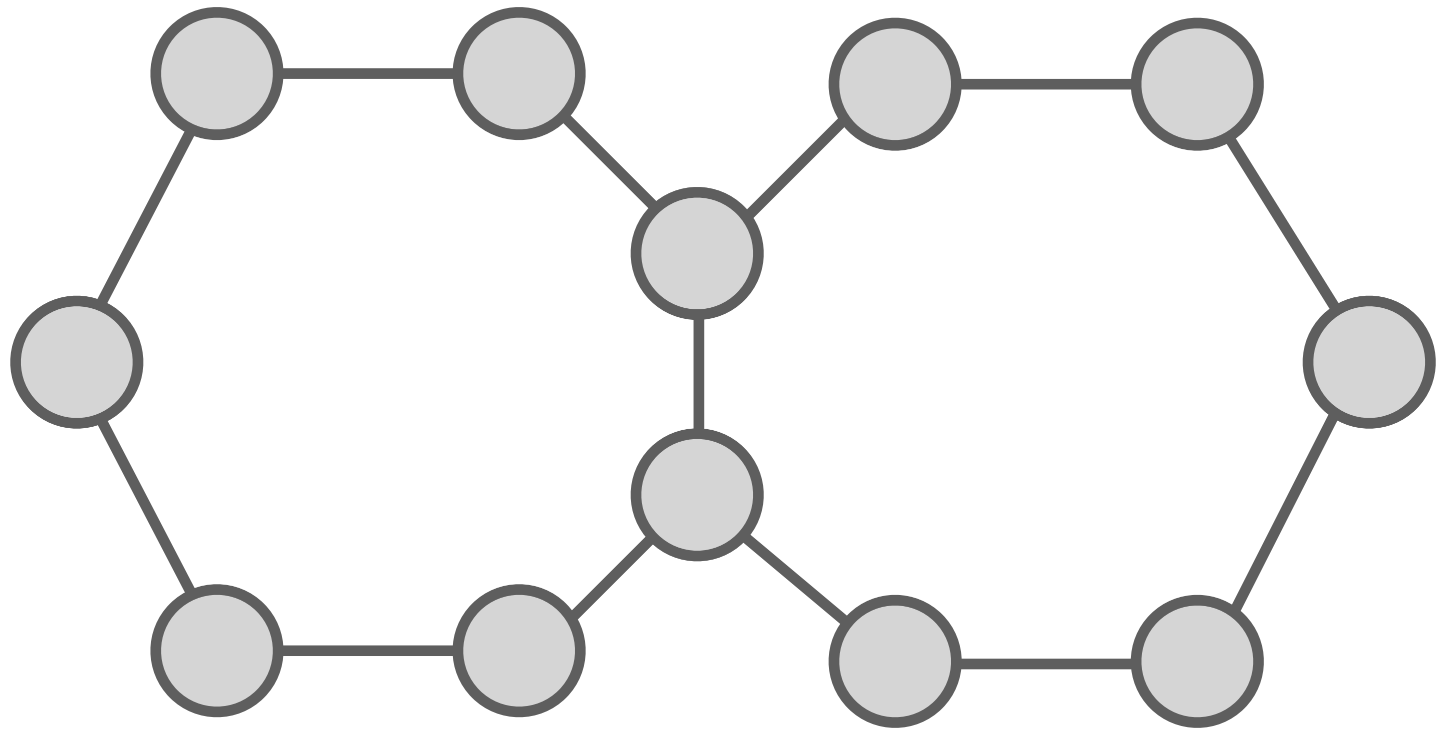


versus





versus

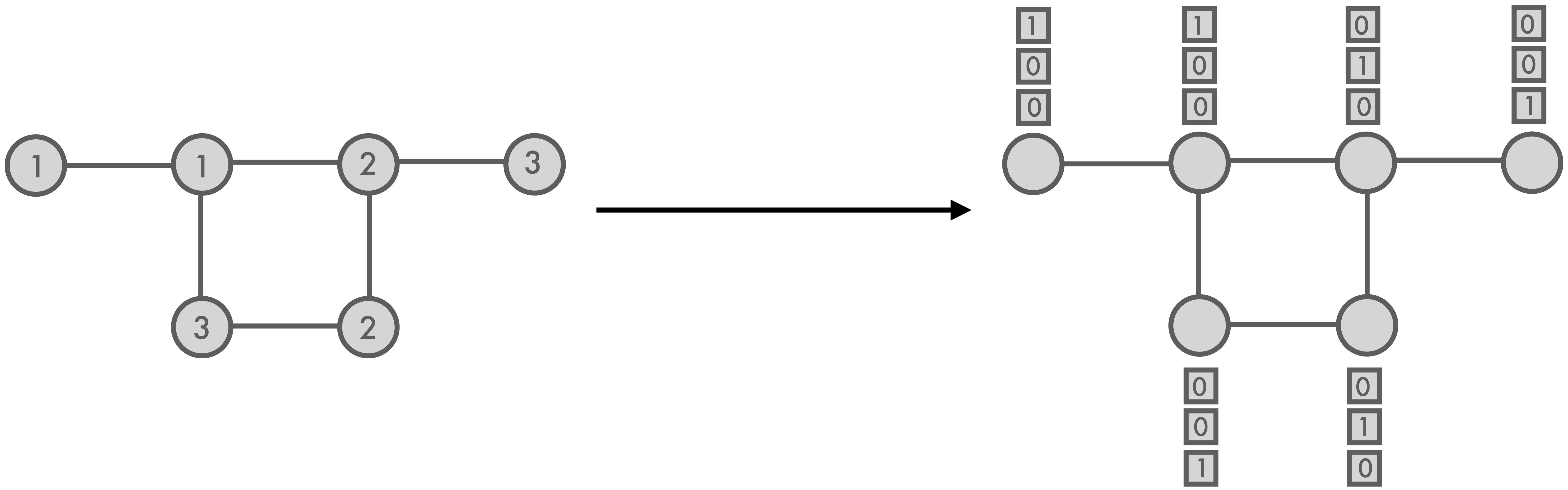


Graph neural networks (GNNs)

Limits of Graph Neural Networks

Theorem (Informal)

GNNs *cannot be expressive* than the WL algorithm in terms of *distinguishing non-isomorphic graphs.*



Features are *consistent* with labels of the graphs

Graph neural networks (GNNs)

Limits of Graph Neural Networks

Theorem (Informal)

GNNs *cannot be expressive* than the WL algorithm in terms of *distinguishing non-isomorphic graphs.*

$$c^{(t)}(v) = \mathsf{hash}\left(c^{(t-1)}(v), \{\!\!\{ c^{(t-1)}(w) \mid w \in N(v) \}\!\!\}\right)$$

$$f^{(t)}(v) = f^{W_1}_{\mathsf{merge}}\left(f^{(t-1)}(v), f^{W_2}_{\mathsf{aggr}}\left(\{\!\!\{ f^{(t-1)}(w) \mid w \in N(v) \}\!\!\}\right)\right)$$

Theorem (Formal)

Let G be labeled graph. Then for all $t \geq 0$ and all consistent features $f^{(0)}$ and choices of parameters $W^{(t)}$

$$c^{(t)}(v) = c^{(t)}(w) \text{ implies } f^{(t)}(v) = f^{(t)}(w)$$

for all nodes v and w.

Graph neural networks (GNNs)

Limits of Graph Neural Networks

Theorem (Formal)

Let G be labeled graph. Then for all $t \geq 0$ and all consistent features $f^{(0)}$ and choices of parameters $W^{(t)}$

$$c^{(t)}(v) = c^{(t)}(w) \text{ implies } f^{(t)}(v) = f^{(t)}(w)$$

for all nodes v and w.

Proof sketch.

Induction on the number of iterations or layers.

Case $t = 0$: Since we assumed consistent features by assumption it follows that

$$c^{(0)}(v) = c^{(0)}(w) \text{ implies } f^{(0)}(v) = f^{(0)}(w)$$

for all nodes v and w .

Proof sketch (cont.).

Case $t > 0$: Let v and w be two nodes and $t \geq 0$. Now assume $c^{(t+1)}(v) = c^{(t+1)}(w)$.
Assume for induction that

$$c^{(t)}(v) = c^{(t)}(w) \text{ implies } f^{(t)}(v) = f^{(t)}(w)$$

for all nodes v and w .

Proof sketch (cont.).
Case $t > 0$: Let v and w be two nodes and $t \geq 0$. Now assume $c^{(t+1)}(v) = c^{(t+1)}(w)$.
Assume for induction that

$$c^{(t)}(v) = c^{(t)}(w) \text{ implies } f^{(t)}(v) = f^{(t)}(w)$$

for all nodes v and w.

By assumption, we know that $c^{(t)}(v) = c^{(t)}(w)$ and

$$\{\!\{ c^{(t)}(e) \mid e \in N(v) \}\!\} = \{\!\{ c^{(t)}(e) \mid e \in N(w) \}\!\}$$

for all nodes v and w.

Proof sketch (cont.).
Let

$$M_v = \{\!\{ f^{(t)}(e) \mid e \in N(v) \}\!\} \quad \text{and} \quad M_w = \{\!\{ f^{(t)}(e) \mid e \in N(w) \}\!\}.$$

By induction hypothesis, we know that

$$M_v = M_w \quad \text{and} \quad f^{(t)}(v) = f^{(t)}(w).$$

Proof sketch (cont.).
Let

$$M_v = \{\!\{ f^{(t)}(e) \mid e \in N(v) \}\!\} \quad \text{and} \quad M_w = \{\!\{ f^{(t)}(e) \mid e \in N(w) \}\!\}.$$

By induction hypothesis, we know that

$$M_v = M_w \quad \text{and} \quad f^{(t)}(v) = f^{(t)}(w).$$

Hence, independent of choice for $f^{W_1}_{\text{merge}}$ and $f^{W_1}_{\text{aggr}}$ it follows that

$$f^{(t+1)}(v) = f^{(t+1)}(w).$$

Hence, it follows that

$$c^{(t+1)}(v) = c^{(t+1)}(w) \text{ implies } f^{(t+1)}(v) = f^{(t+1)}(w).$$

Graph neural networks (GNNs)

Limits of Graph Neural Networks

Coloring rule of the WL

$$c^{(t)}(v) = \text{hash}\left(c^{(t-1)}(v), \{\!\{c^{(t-1)}(w) \mid w \in N(v)\}\!\}\right)$$

versus

General form of GNNs

$$f^{(t)}(v) = f^{W_1}_{\text{merge}}\left(f^{(t-1)}(v), f^{W_2}_{\text{aggr}}\left(\{\!\{f^{(t-1)}(w) \mid w \in N(v)\}\!\}\right)\right)$$

Theorem (Informal)

There exists choices of $f^{W_1}_{\text{merge}}$ and $f^{W_2}_{\text{aggr}}$ such that

$$c^{(t)}(v) = c^{(t)}(w) \text{ if and only if } f^{(t)}(v) = f^{(t)}(w)$$

for all nodes v and w.

Graph neural networks (GNNs)

Limits of Graph Neural Networks

Theorem (Informal)

There exists choices of $f_{\text{merge}}^{W_1}$ *and* $f_{\text{aggr}}^{W_2}$ *such that*

$$c^{(t)}(v) = c^{(t)}(w) \text{ if and only if } f^{(t)}(v) = f^{(t)}(w)$$

for all nodes v *and* w.

Lemma (Informal)

Let $m > 0$, *let* $X \subseteq (0,1)$ *be a non-empty finite set. Then there exists a function* $d: X \to (0,1)$ *such that for all multisets* M, M' *with cardinality at most* m *and* $M \neq M'$ *it holds that*

$$\sum_{x \in M} d(x) \neq \sum_{x \in M'} d(x).$$

Graph neural networks (GNNs)

Limits of Graph Neural Networks

Lemma (Informal)

Let $m > 0$, *let* $X \subseteq (0,1)$ *be a non-empty finite set. Then there exists a function* $d: X \to (0,1)$ *such that for all multisets* M, M' *with cardinality at most* m *and* $M \neq M'$ *it holds that*

$$\sum_{x \in M} d(x) \neq \sum_{x \in M'} d(x).$$

Sketch of the proof sketch.

$$f^{(l)}(v) = g\left(W_1 \cdot f^{(l-1)}(v) + C \cdot \sum_{w \in N(v)} d(f^{(l-1)}(w))\right)$$

Multiset of neighbors gets unique representation

Graph neural networks (GNNs)

Limits of Graph Neural Networks

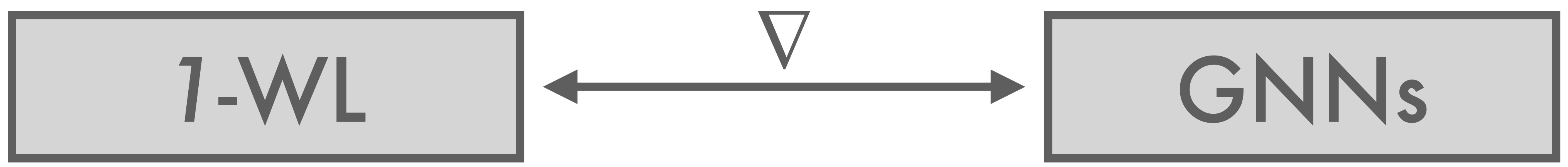
- How Powerful are Graph Neural Networks?. Keyulu Xu, Weihua Hu, Jure Leskovec, Stefanie Jegelka. ICLR 2019
- Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks.
  Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, Martin Grohe. AAAI 2019

Graph neural networks (GNNs)

Limits of Graph Neural Networks

Insight

1-WL and GNN have the same power in distinguishing non-isomorphic graphs.



Insight

Limits of the *1*-WL are well-understood.

V. Arvind, J. Köbler, G. Rattan, and O. Verbitsky. "On the Power of Color Refinement". International Symposium onFundamentals of Computation Theory 2015

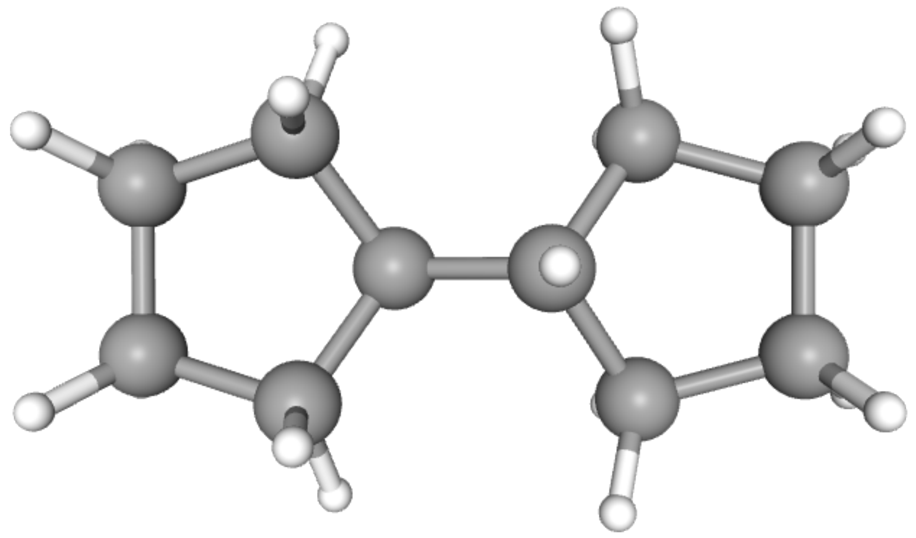
Graph neural networks (GNNs)

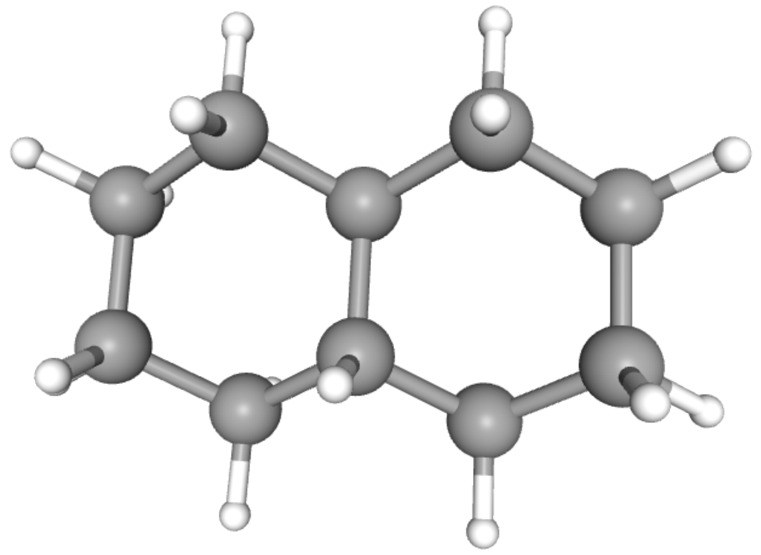
Limits of Graph Neural Networks

Insight

GNNs cannot distinguish very basic graph properties, e.g.,

- Cycles of different lengths
- Triangle counts
- Regular graphs
- ...





Questions

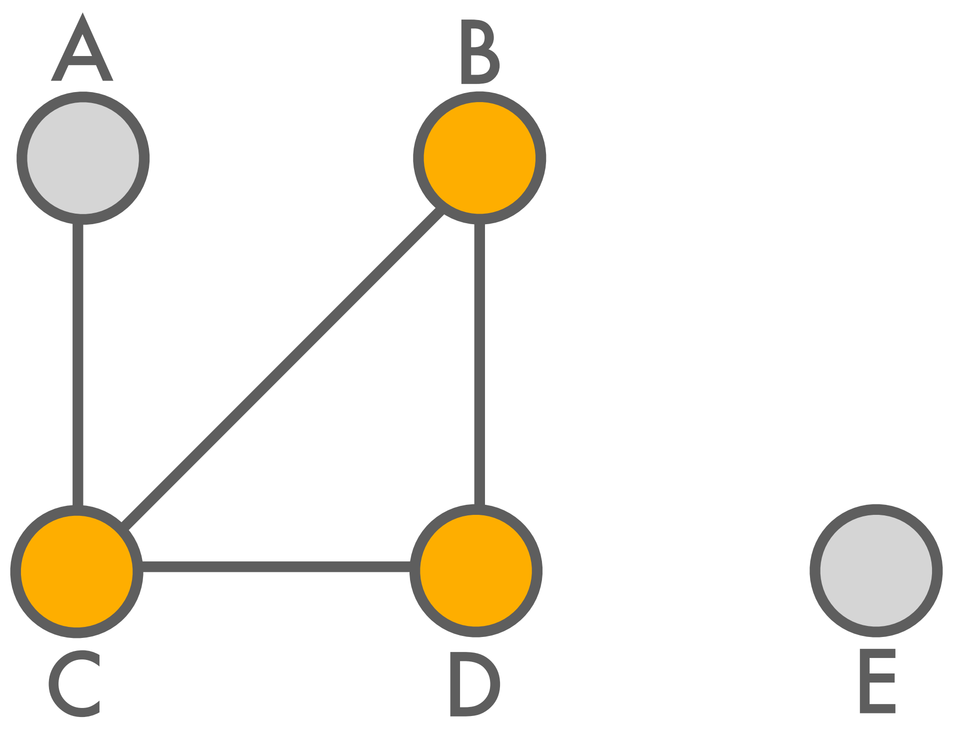
How can we overcome the limitations of GNNs?

Graph neural networks (GNNs)

k-dimensionaler Weisfeiler-Leman algorithm

k-dimensionaler Weisfeiler-Leman algorithm (Babai et al.)

- Colors *k*-tuples defined over the set of vertices
- Strictly more power as *k* increases



Idea of the algorithm

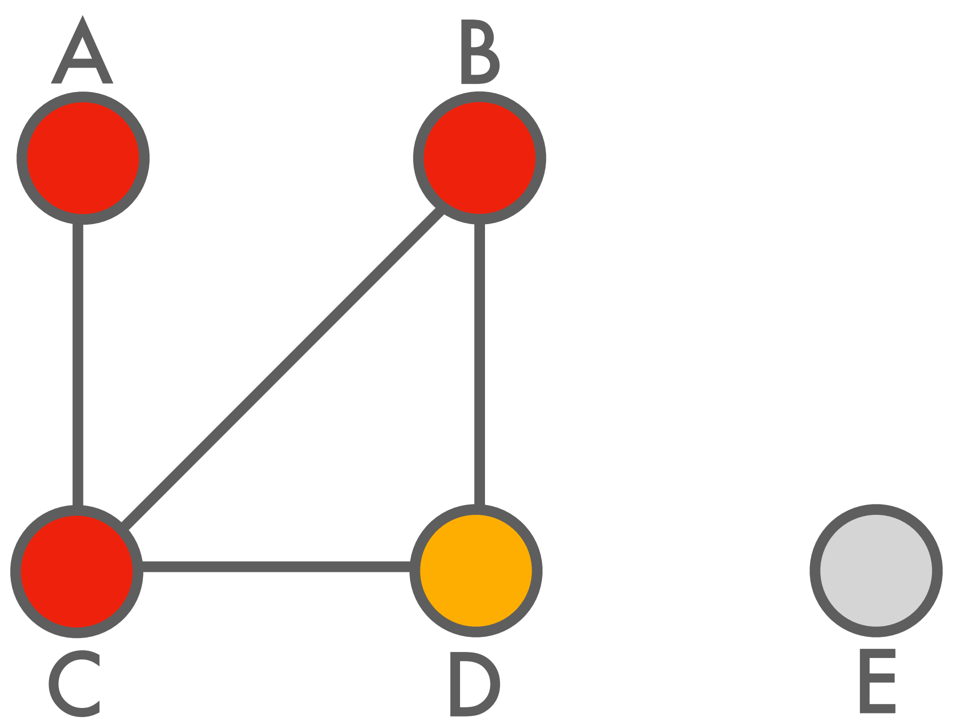
1. *Initially:* Two tuples get the same color if the induced subgraphs are isomorphic
2. *Iteration:* Two tuples get the same color if they have an equally colored neighbourhood

Graph neural networks (GNNs)

k-dimensionaler Weisfeiler-Leman algorithm

k-dimensionaler Weisfeiler-Leman algorithm (Babai et al.)

- Colors *k*-tuples defined over the set of vertices
- Strictly more power as *k* increases



Idea of the algorithm

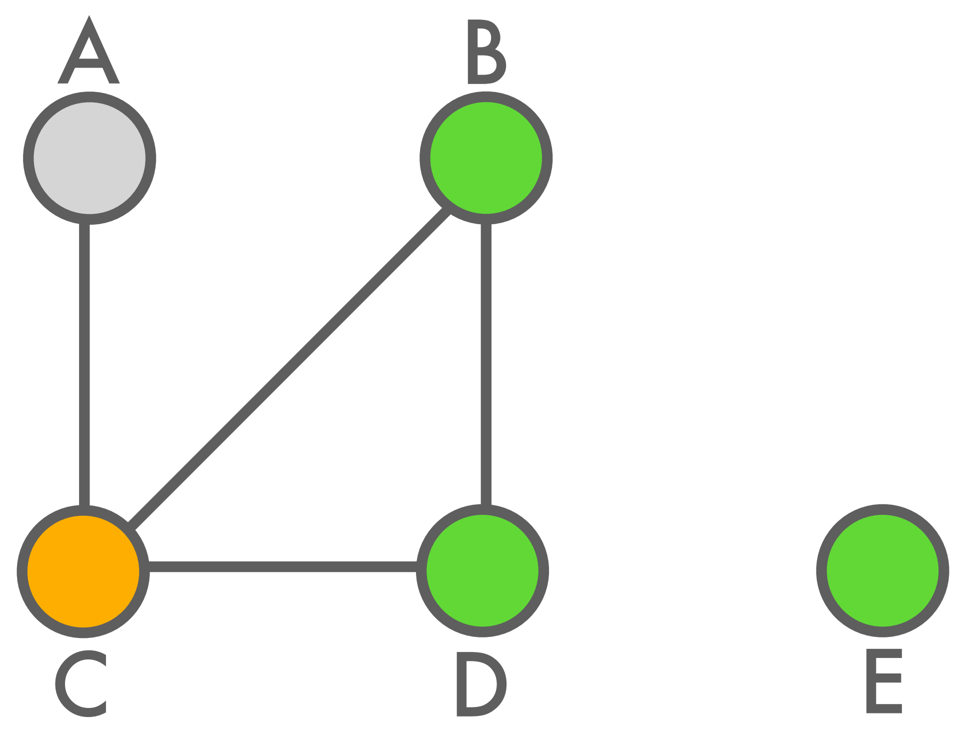
1. *Initially:* Two tuples get the same color if the induced subgraphs are isomorphic
2. *Iteration:* Two tuples get the same color if they have an equally colored neighbourhood

Graph neural networks (GNNs)

k-dimensionaler Weisfeiler-Leman algorithm

k-dimensionaler Weisfeiler-Leman algorithm (Babai et al.)

- Colors *k*-tuples defined over the set of vertices
- Strictly more power as *k* increases



Idea of the algorithm

1. *Initially:* Two tuples get the same color if the induced subgraphs are isomorphic
2. *Iteration:* Two tuples get the same color if they have an equally colored neighbourhood

Graph neural networks (GNNs)

Higher-order GNNs

k-dimensionaler Weisfeiler-Leman algorithm (Babai et al.)

- Colors k-tuples defined over the set of vertices
- Strictly more power as k increases

Idea

Derive k-dimensional Graph Neural Networks

$$f^{(l)}(t) = \mathsf{MLP}\Big([W_1 \cdot f^{(l-1)}(t) + W_2 \cdot \sum_{s \in N_i(t)} f^{(l-1)}(t)]_{i \in [k]}\Big),$$

where t is a k-tuple.

Graph neural networks (GNNs)

Higher-order GNNs

Idea

Derive k-dimensional Graph Neural Networks

$$f^{(l)}(t) = \sigma\Big(\mathsf{MLP}\big([W_1 \cdot f^{(l-1)}(t) + W_2 \cdot \sum_{s \in N_i(v)} f^{(l-1)}(t)]_{i \in [k]}\big)\Big),$$

where t is a k-tuple.

Theorem (Informal)

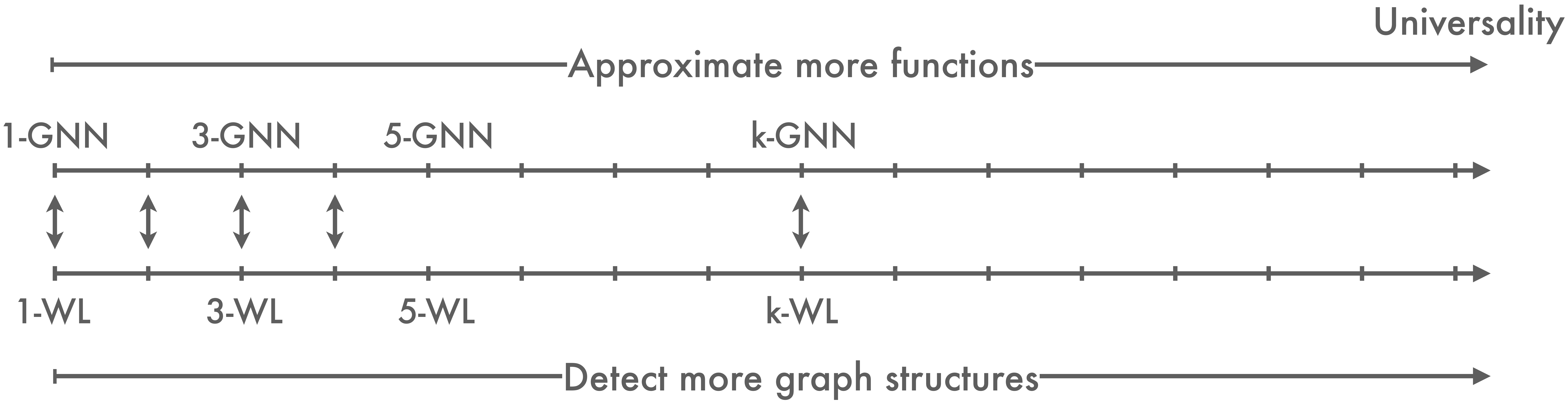
The k-order GNN architecture has the same expressivity as the k-WL in terms of distinguishing non-isomorphic graphs.

Graph neural networks (GNNs)

Higher-order GNNs

Theorem (Informal)

The *k*-order GNN architecture has the same expressivity as the *k*-WL in terms of distinguishing non-isomorphic graphs.



Waiss Azizian, Marc Lelarge. Expressive Power of Invariant and Equivariant Graph Neural Networks. ICLR 2021

Graph neural networks (GNNs)

Limits of Graph Neural Networks

Problem

The k-WL's and k-order GNN's memory complexity is in $\Omega(n^k)$.

Challenge

Design enhanced GNNs that overcome *1*-WL limitation but are still scalable.

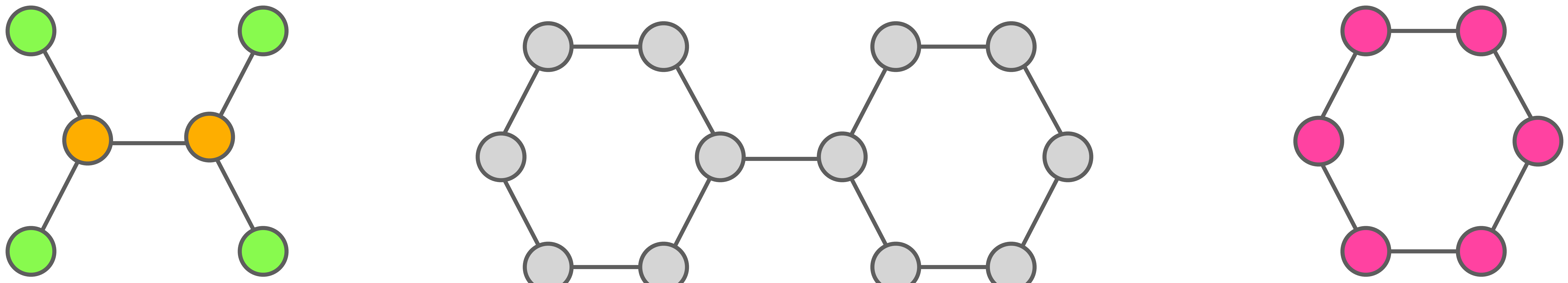
Graph neural networks (GNNs)

Limits of Graph Neural Networks

Subgraph GNNs

Enhance node features with subgraph information

- Fix a number of subgraphs in advance
- Compute “role” (formally, *automorphism type*) of each node with regards to these subgraphs



Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, Michael M. Bronstein.
Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting. CoRR abs/2006.09252 (2020)

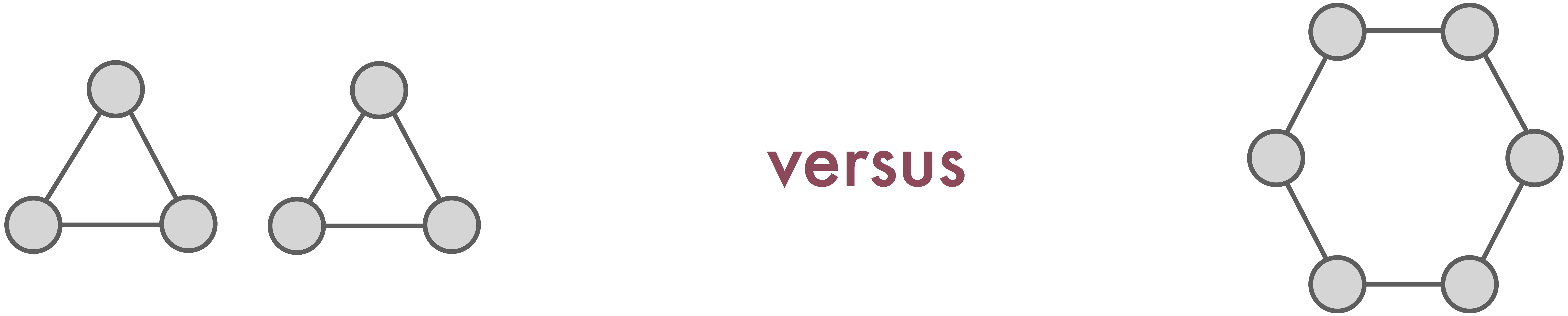
Graph neural networks (GNNs)

Limits of Graph Neural Networks

k-reconstruction GNNs

Break up symmetries of *1*-WL by removing nodes

- Remove every *k*-node subgraph from a given graph
- Use GNN to compute representation for resulting graph
- Pool together resulting representation



Leonardo Cotta, Christopher Morris, Bruno Ribeiro. Reconstruction for Powerful Graph Representations. NeurIPS 2021

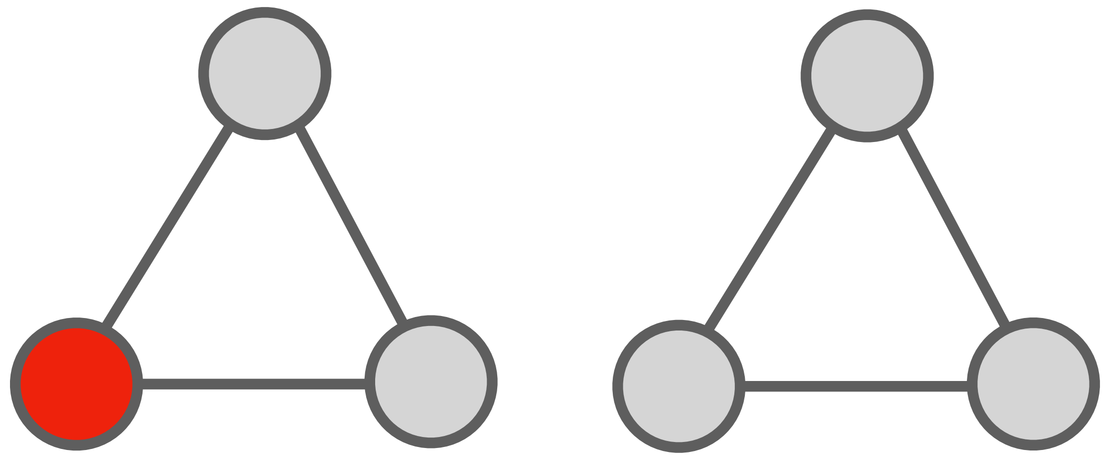
Graph neural networks (GNNs)

Limits of Graph Neural Networks

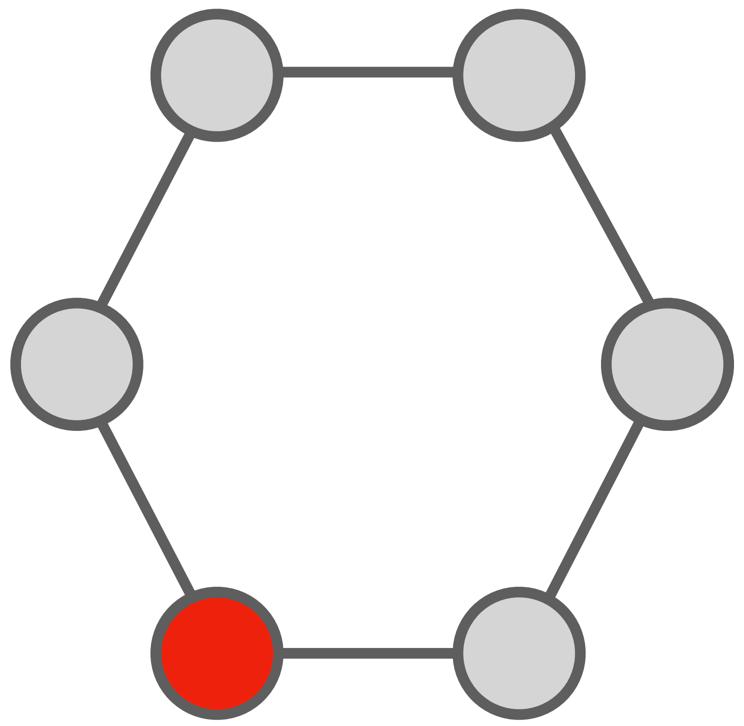
k-reconstruction GNNs

Break up symmetries of 1-WL by removing node

- Remove every *k*-node subgraph from a given graph
- Use GNN to compute representation for resulting graph
- Pool together resulting representation



versus



Leonardo Cotta, Christopher Morris, Bruno Ribeiro. Reconstruction for Powerful Graph Representations. NeurIPS 2021

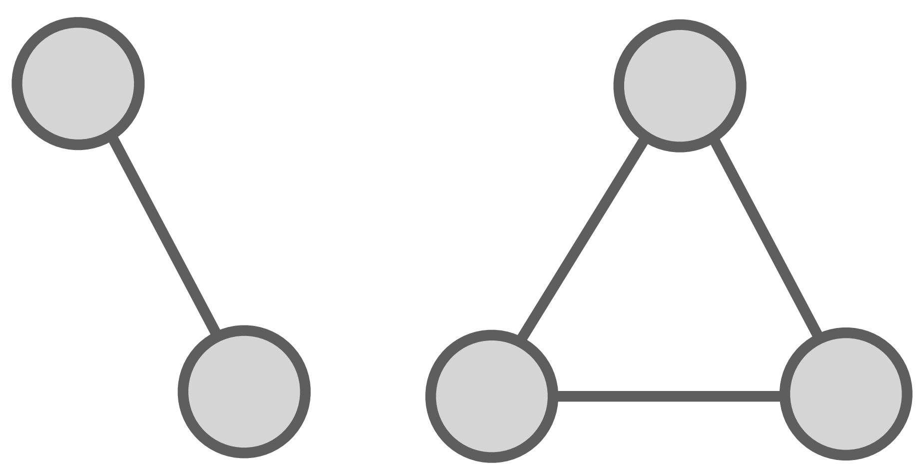
Graph neural networks (GNNs)

Limits of Graph Neural Networks

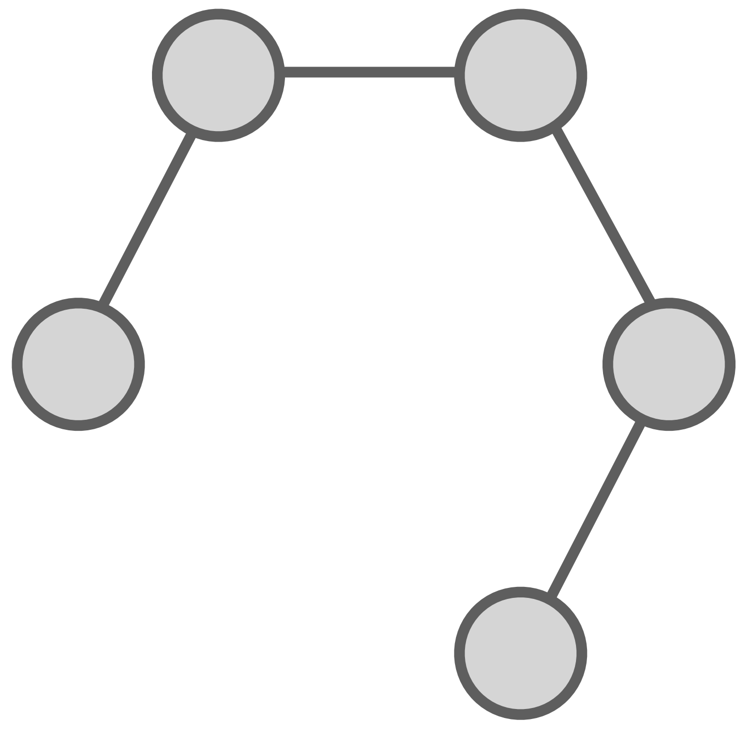
k-reconstruction GNNs

Break up symmetries of 1-WL by removing node

- Remove every k-node subgraph from a given graph
- Use GNN to compute representation for resulting graph
- Pool together resulting representation



versus



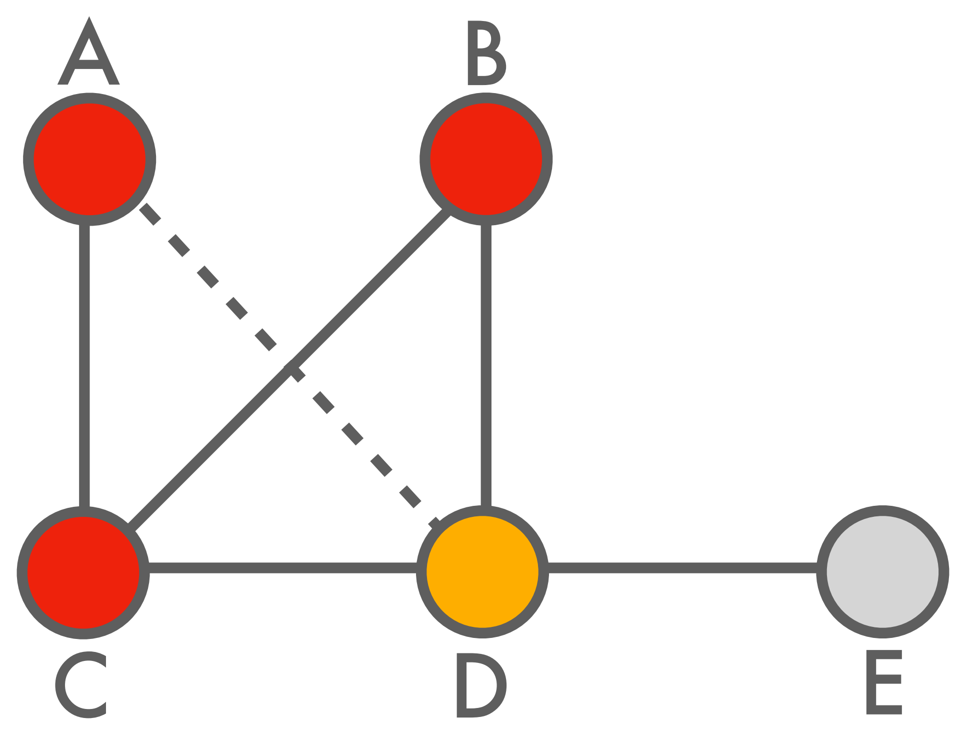
Leonardo Cotta, Christopher Morris, Bruno Ribeiro. Reconstruction for Powerful Graph Representations. NeurIPS 2021

Graph neural networks (GNNs)

Limits of Graph Neural Networks

Local k-WL

Consider only certain neighbors of a k-tuple.



Idea of the local algorithm

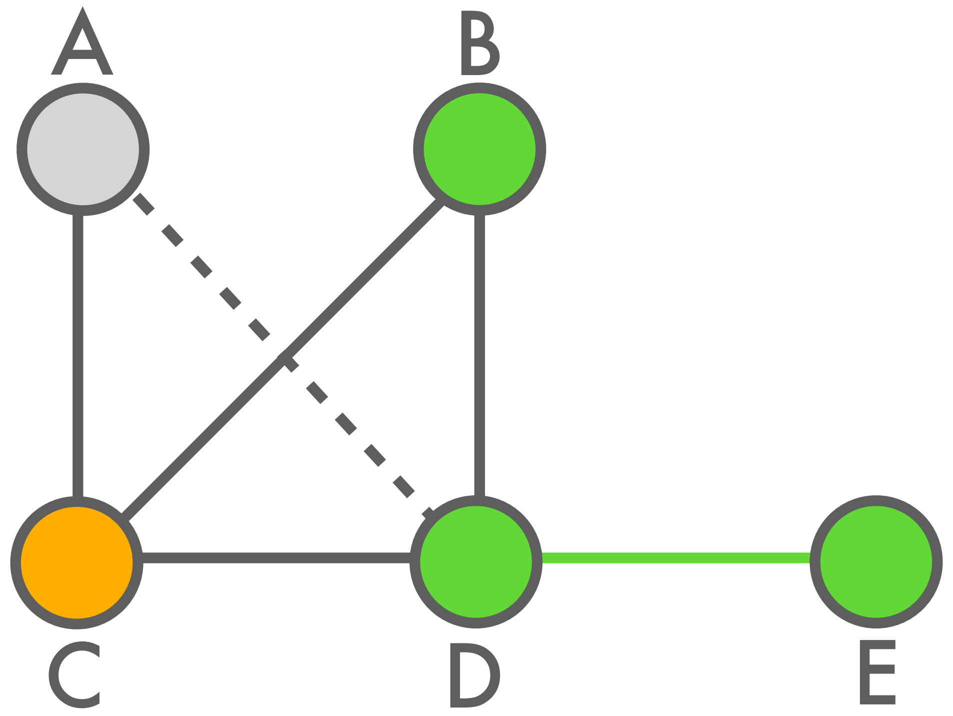
1. *Initial:* Two tuples get the same color if the induced subgraphs are isomorphic
2. *Iteration:* Two tuples get the same color if they have an equally colored *local neighbourhood*

Graph neural networks (GNNs)

Limits of Graph Neural Networks

Local k-WL

Consider only certain neighbors of a k-tuple.



Idea of the local algorithm

1. *Initial:* Two tuples get the same color if the induced subgraphs are isomorphic
2. *Iteration:* Two tuples get the same color if they have an equally colored *local neighbourhood*

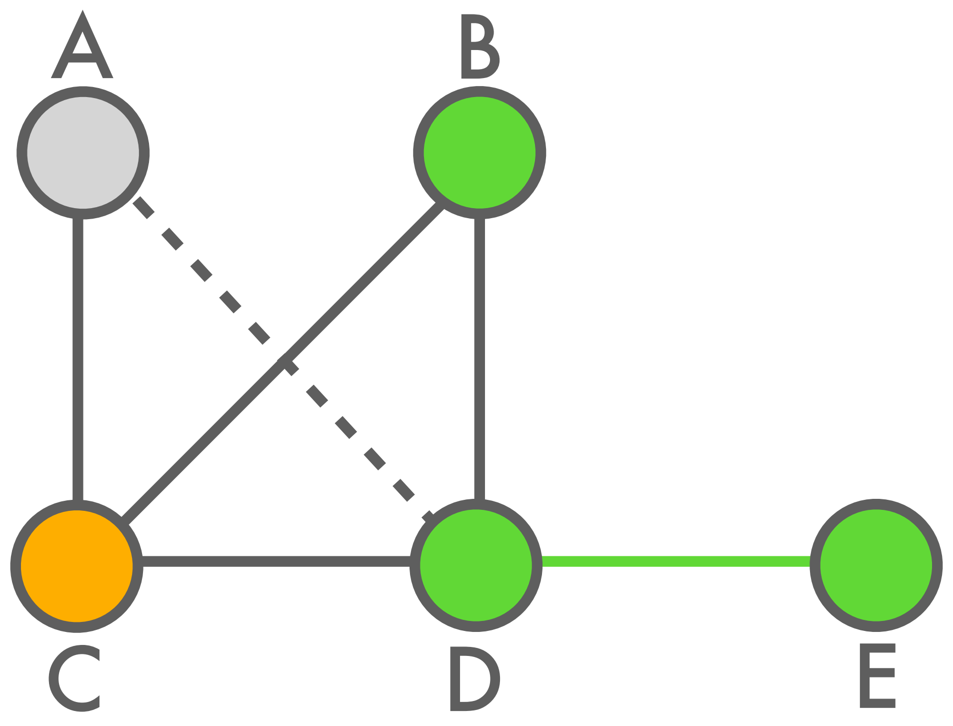
Graph neural networks (GNNs)

Limits of Graph Neural Networks

Local *k*-WL

Consider only local neighbors of a *k*-tuple.

- Takes sparsity of underlying graph into account
- Has the same power as ordinary *k*-WL, but more iterations are needed



Graph neural networks (GNNs)

Limits of Graph Neural Networks

Idea

Derive local *k*-dimensional Graph Neural Networks

$$f^{(l)}(t) = \mathsf{MLP}\Big([W_1 \cdot f^{(l-1)}(t) + W_2 \cdot \sum_{s \in N_i^L(t)} f^{(l-1)}(t)]_{i \in [k]}\Big),$$

Where *t* is *k*-tuple.

Christopher Morris, Gaurav Rattan, Petra Mutzel.
Weisfeiler and Leman go sparse: Towards scalable higher-order graph embeddings. NeurIPS 2020

Graph neural networks (GNNs)

Limits of Graph Neural Networks

- Christopher Morris, Yaron Lipman, Haggai Maron, Bastian Rieck, Nils M. Kriege, Martin Grohe, Matthias Fey, Karsten M. Borgwardt. Weisfeiler and Leman go Machine Learning: The Story so far. CoRR abs/2112.09992 (2021)

Implementing GNNs

Graph neural networks (GNNs)

Implementation of GNNs

Implementation Frameworks

Nowadays there exist quite a few good frameworks

- PyTorch Geometric (PyG, based on PyTorch, `www.pyg.org`)
- Deep Graph Library (DGL, based on PyTorch and TensorFlow, `www.dgl.ai`)
- Spektral (based on Keras, `www.graphneural.network`)

Challenge

Implement simple GNN layer in PyG:

$$f^{(l)}(v) = \sigma\left(W_1 \cdot f^{(l-1)}(v) + W_2 \cdot \sum_{w \in N(v)} f^{(l-1)}(w)\right)$$

Graph neural networks (GNNs)

Implementation of GNNs

Challenge

Implement simple GNN layer in PyG:

$$f^{(l)}(v) = \sigma\left(W_1 \cdot f^{(l-1)}(v) + W_2 \cdot \sum_{w \in N(v)} f^{(l-1)}(w)\right)$$

```
class SimpleLayer(MessagePassing):
    def __init__(self, in_channels, out_channels):
        super().__init__(aggr='add')

        self.w_1 = torch.nn.Linear(in_channels, out_channels)
        self.w_2 = torch.nn.Linear(in_channels, out_channels)

    def forward(self, features, edge_index):

        features_new =  self.w_2(features)
        feature_self = self.w_1(features)

        out = feature_self + self.propagate(edge_index, x=features_new)

        return out
```

Graph neural networks (GNNs)

Implementation of GNNs

Challenge

Implement simple GNN layer in PyG:

$$f^{(l)}(v) = \sigma\left(W_1 \cdot f^{(l-1)}(v) + W_2 \cdot \sum_{w \in N(v)} f^{(l-1)}(w)\right)$$

```
class SimpleArchitecture(torch.nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = SimpleLayer(dataset.num_node_features, 16)
        self.conv2 = SimpleLayer(16, dataset.num_classes)

    def forward(self, data):
        features, edge_index = data.x, data.edge_index

        features = self.conv1(features, edge_index)
        features = F.relu(features)
        features = self.conv2(features, edge_index)

        return F.log_softmax(features, dim=1)
```

Graph neural networks (GNNs)

Implementation of GNNs

Challenge

Implement simple GNN layer in PyG:

$$f^{(l)}(v) = \sigma\left(W_1 \cdot f^{(l-1)}(v) + W_2 \cdot \sum_{w \in N(v)} f^{(l-1)}(w)\right)$$

```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = SimpleArchitecture().to(device)
data = dataset[0].to(device)

optimizer = torch.optim.Adam(model.parameters(), lr=0.01, weight_decay=5e-4)

model.train()
for epoch in range(200):
    optimizer.zero_grad()
    out = model(data)
    loss = F.nll_loss(out[data.train_mask], data.y[data.train_mask])
    loss.backward()
    optimizer.step()
```

Conclusion

Key take aways

1. Challenges of learning with graphs: *Graphs due not have a unique representation*
2. Learned about basic algorithms for extracting features out of graphs
    1. Substructure counting
    2. Weisfeiler-Leman algorithm
3. Learned about common GNN layers
4. Learned about the limitations of GNNs, i.e., they are limited by the Weisfeiler-Leman algorithm
5. Learned how to overcome the limitations of GNNs
6. Learned how to implement a GNN layer in PyTorch Geometric