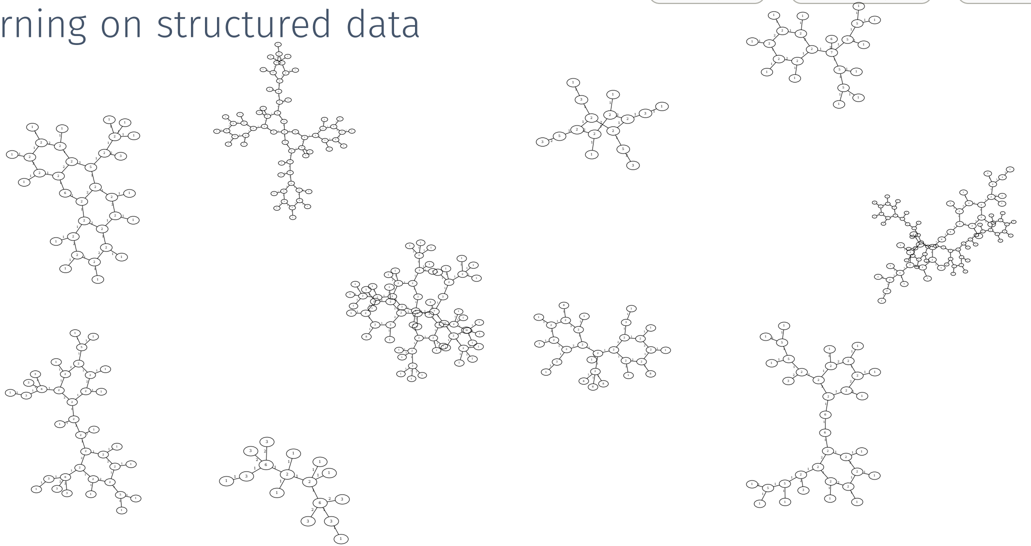
Expressive Graph Embeddings via Homomorphism Counts

Pascal Welke

CAIML Seminar on 25. November 2024

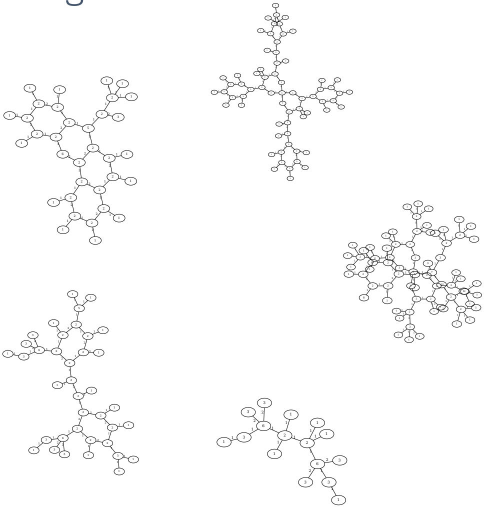
Hu et al (2020) Morris et al (2020) Dwivedi et al (2022)

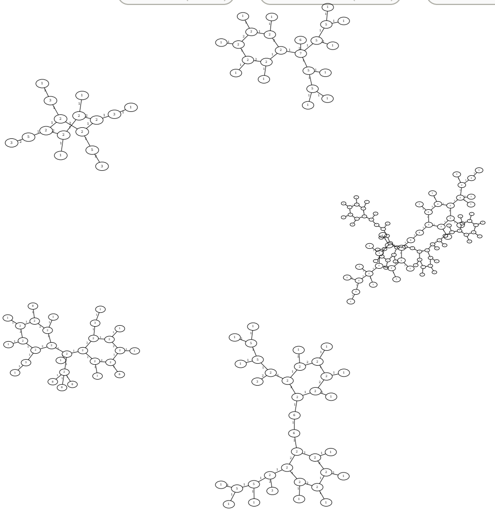
Learning on structured data



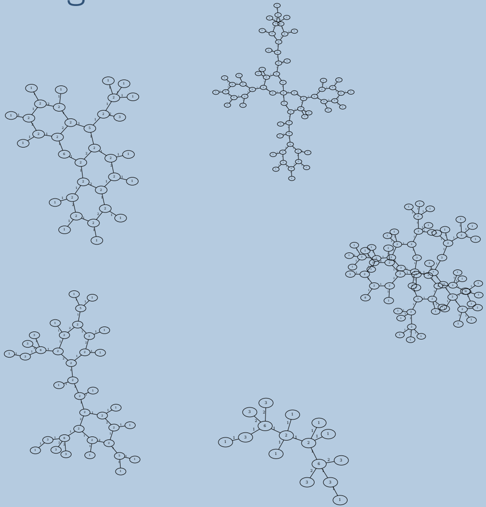
Hu et al (2020) Morris et al (2020) Dwivedi et al (2022)

Learning on structured data

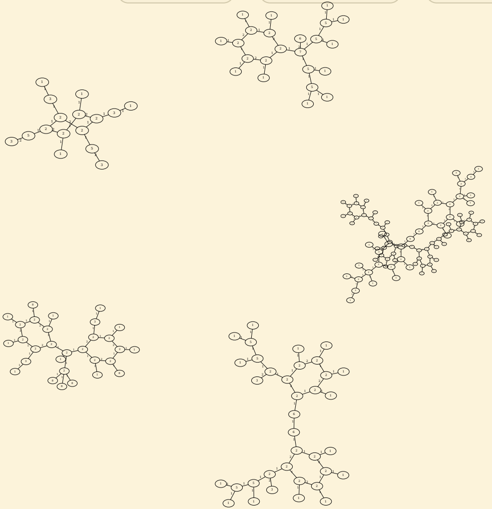




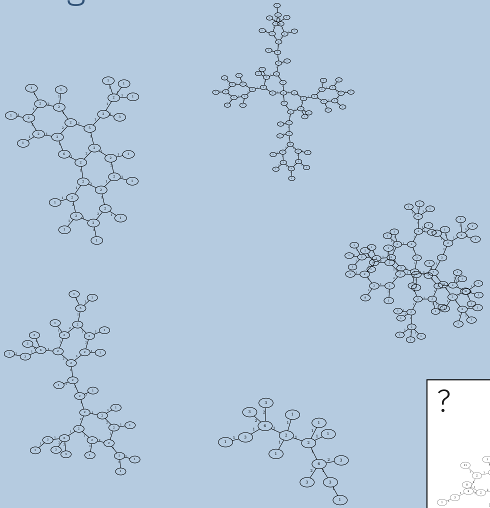
Learning on structured data

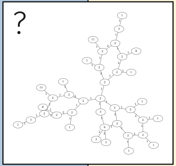


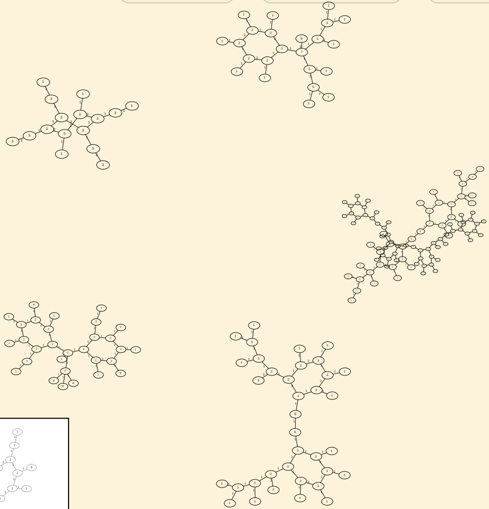
Hu et al (2020) Morris et al (2020) Dwivedi et al (2022)



Learning on structured data

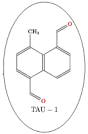




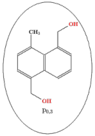


Learning on structured data

chemistry prof.

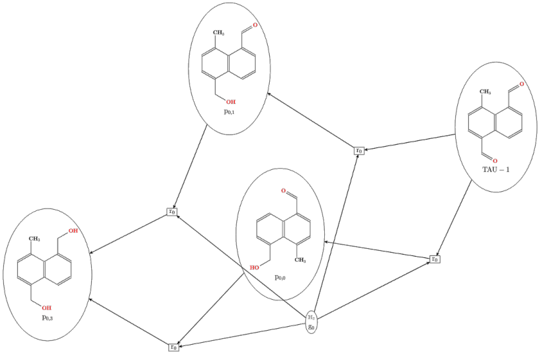


?



Learning on structured data

me



Problems in graph learning

Neural methods achieve remarkable results in graph learning

Problems in graph learning

Neural methods achieve remarkable results in graph learning

- molecule synthesis and prediction
- modeling of human social behavior
- …

Problems in graph learning

Neural methods achieve remarkable results in graph learning

- molecule synthesis and prediction
- modeling of human social behavior
- ...

but come with

Problems in graph learning

Neural methods achieve remarkable results in graph learning

- molecule synthesis and prediction
- modeling of human social behavior
- ...

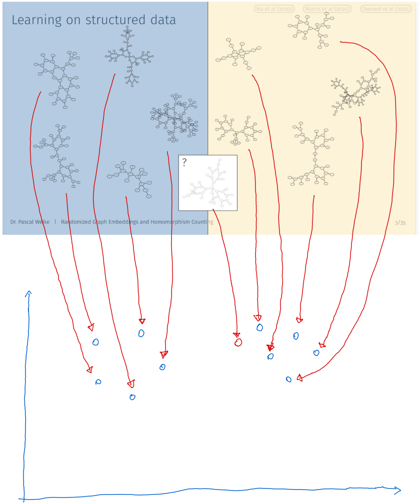
but come with

- significant resource demands
- too much complexity to be interpretable
- which hinders application in many scenarios

Graph Representation Learning

The goal

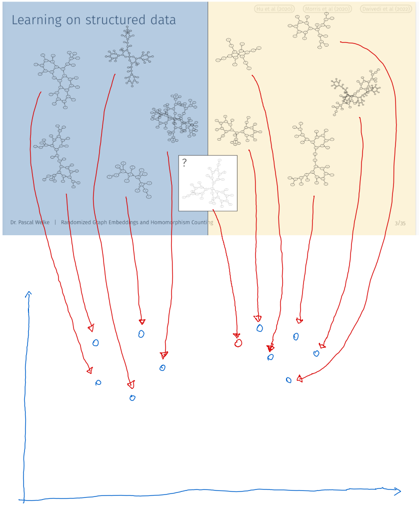
Vectorial graph representations
that



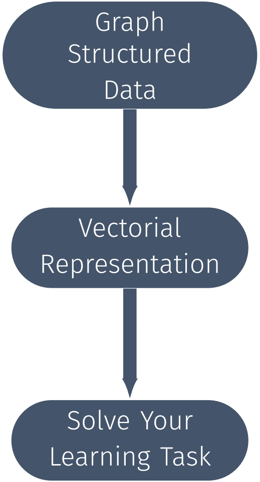
The goal

Vectorial graph representations that

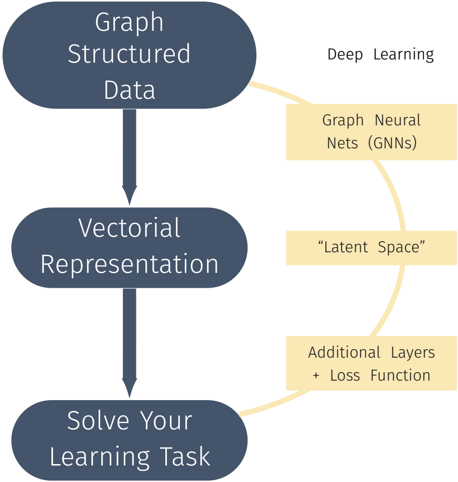
- yield semantically and structurally meaningful distances
- are interpretable
- are adaptable to given data



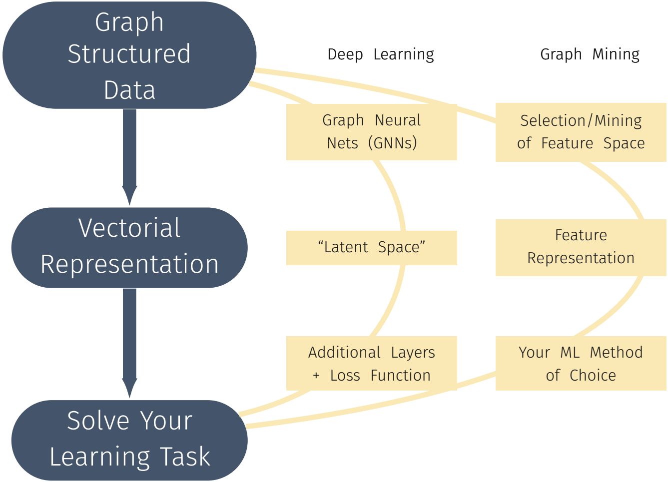
Graph representation learning



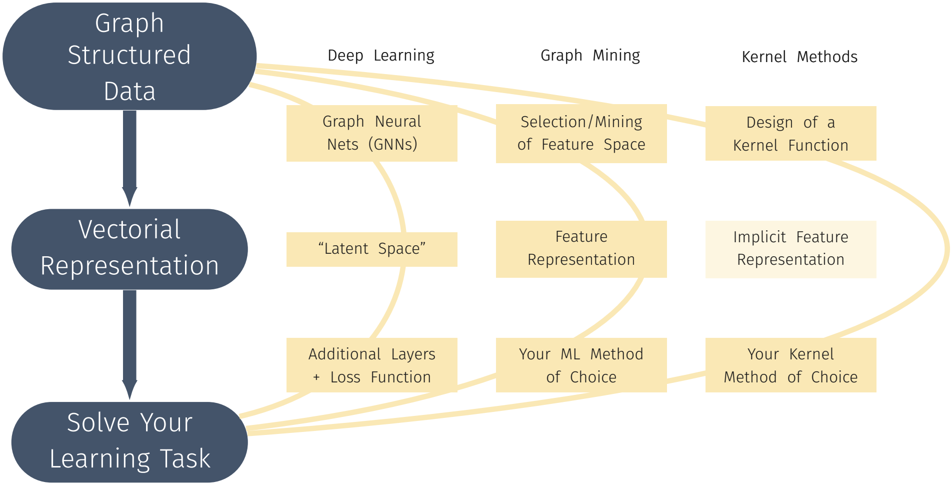
Graph representation learning



Graph representation learning

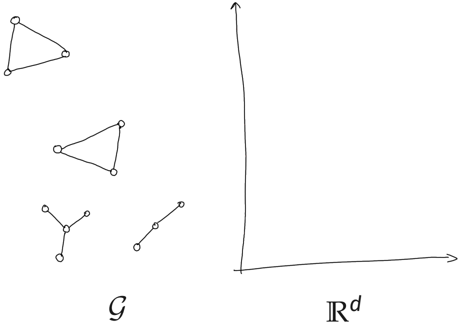


Graph representation learning



The problem with vectorial graph representations

We want our graph representation function ϕ to be



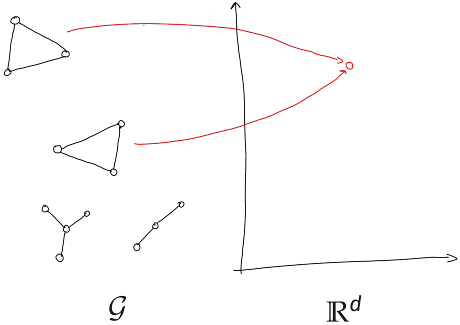
The problem with vectorial graph representations

We want our graph representation function ϕ to be

- permutation-invariant

  for all isomorphic graphs

$$G \simeq H : \; \phi(G) = \phi(H)$$



The problem with vectorial graph representations

We want our graph representation function ϕ to be

- permutation-invariant

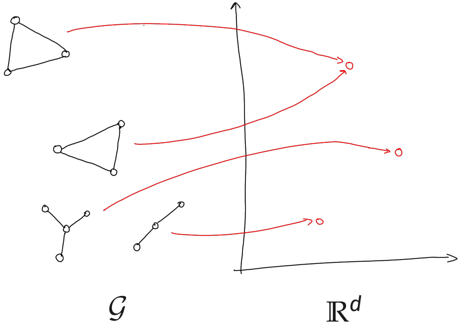
  for all isomorphic graphs

$$G \simeq H: \ \phi(G) = \phi(H)$$

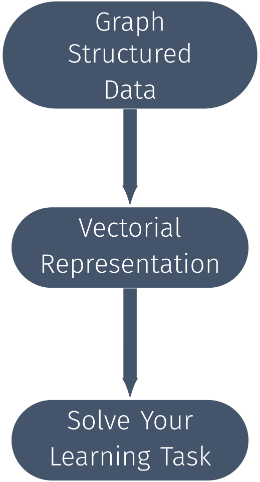
- complete

  for all non-isomorphic graphs

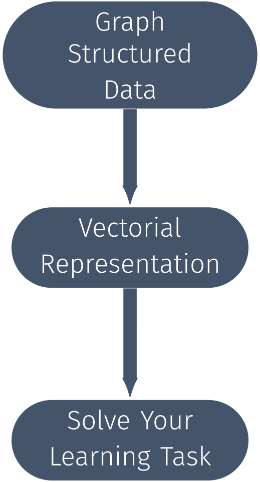
$$G \not\simeq H: \ \phi(G) \neq \phi(H)$$

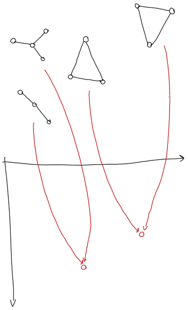


Why do we care?



Why do we care?





Why do we care?

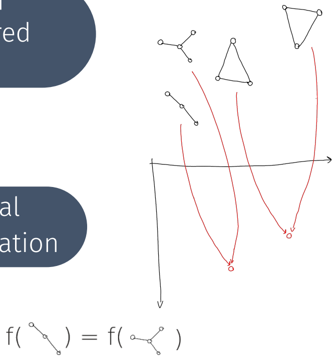
Graph Structured Data

↓

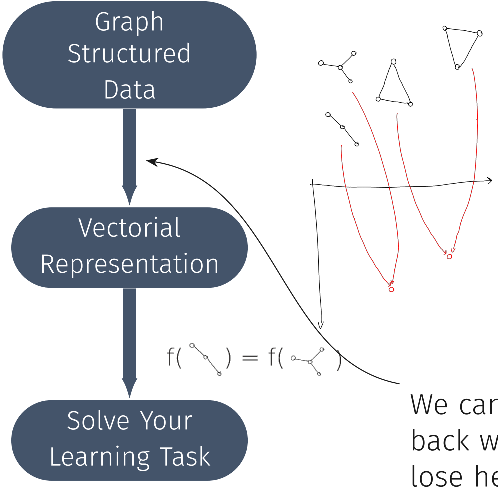
Vectorial Representation

↓

Solve Your Learning Task



Why do we care?



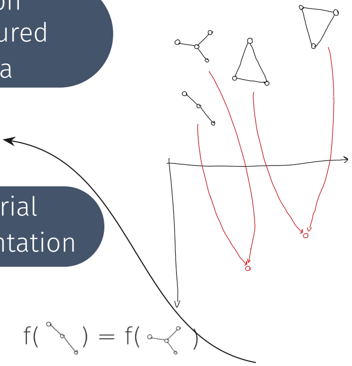
We cannot get back what we lose here

Why do we care?

Graph Structured Data

Vectorial Representation

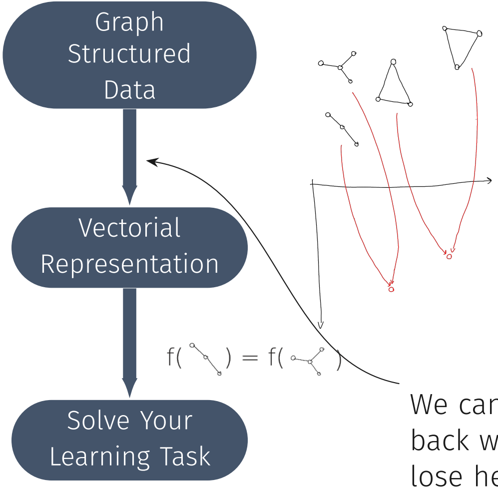
Solve Your Learning Task



We cannot get back what we lose here

- Unfortunately computing any permutation invariant and complete embedding (or kernel) is as hard as deciding graph isomorphism

Why do we care?



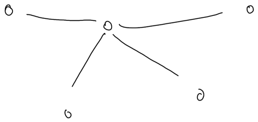
We cannot get back what we lose here

- Unfortunately computing any permutation invariant and complete embedding (or kernel) is as hard as deciding graph isomorphism
- **Typical solution**: drop completeness for efficiency
  - most practical graph kernels, GNNs, Weisfeiler Leman test, …

Message Passing and the Weisfeiler Leman Algorithm

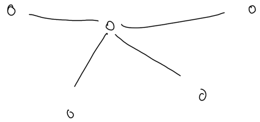
Message Passing

- Let's assume that we have some feature representation $r_0 : V(G) \to \mathbb{R}^d$ for the vertices in our graph



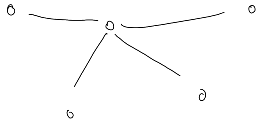
Message Passing

- Let's assume that we have some feature representation $r_0 : V(G) \to \mathbb{R}^d$ for the vertices in our graph
- It is reasonable that in many situations neighboring vertices influence each other



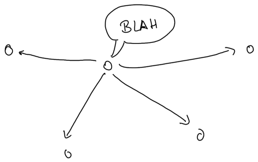
Message Passing

- Let's assume that we have some feature representation $r_0 : V(G) \to \mathbb{R}^d$ for the vertices in our graph
- It is reasonable that in many situations neighboring vertices influence each other
- Consider a social network where users spread their content along connections to their affiliates



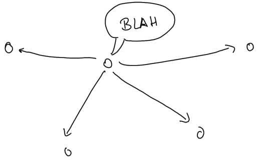
Message Passing

- Let's assume that we have some feature representation $r_0 : V(G) \to \mathbb{R}^d$ for the vertices in our graph
- It is reasonable that in many situations neighboring vertices influence each other
- Consider a social network where users spread their content along connections to their affiliates
- In turn, neighbors might be influenced by that and hence spread (a variant of) that information (aka. "retweet")



Message Passing

- Let's assume that we have some feature representation $r_0 : V(G) \to \mathbb{R}^d$ for the vertices in our graph
- It is reasonable that in many situations neighboring vertices influence each other
- Consider a social network where users spread their content along connections to their affiliates
- In turn, neighbors might be influenced by that and hence spread (a variant of) that information (aka. "retweet")
- Message passing models this kind of behavior as a simultaneous round based process



The message passing framework

$$r_{k+1}(v) = \mathrm{upd}_k\left(r_k(v),\ \mathrm{agg}_k\left(\{\{r_k(w) \mid w \in N(v)\}\}\right)\right)$$

where

The message passing framework

$$r_{k+1}(v) = \mathrm{upd}_k\left(r_k(v),\ \mathrm{agg}_k\left(\{\{r_k(w) \mid w \in N(v)\}\}\right)\right)$$

where

- $v \in V(G)$ is a vertex

The message passing framework

$$r_{k+1}(v) = \mathrm{upd}_k\left(r_k(v),\ \mathrm{agg}_k\left(\{\{r_k(w) \mid w \in N(v)\}\}\right)\right)$$

where

- $v \in V(G)$ is a vertex
- $k \geq 0$

The message passing framework

$$r_{k+1}(v) = \text{upd}_k\left(r_k(v),\ \text{agg}_k\left(\{\{r_k(w) \mid w \in N(v)\}\}\right)\right)$$

where

- $v \in V(G)$ is a vertex
- $k \geq 0$
- $r_{k+1} : V(G) \to \mathcal{X}_{k+1}$ for all $k \geq 0$

The message passing framework

$$r_{k+1}(v) = \mathrm{upd}_k\left(r_k(v),\ \mathrm{agg}_k\left(\{\{r_k(w) \mid w \in N(v)\}\}\right)\right)$$

where

- $v \in V(G)$ is a vertex
- $k \geq 0$
- $r_{k+1} : V(G) \to \mathcal{X}_{k+1}$ for all $k \geq 0$
- $r_0 : V(G) \to \mathcal{X}_0$ is assumed to be given

The message passing framework

$$r_{k+1}(v) = \mathrm{upd}_k\left(r_k(v),\ \mathrm{agg}_k\left(\{\{r_k(w) \mid w \in N(v)\}\}\right)\right)$$

where

- $v \in V(G)$ is a vertex
- $k \geq 0$
- $r_{k+1} : V(G) \to \mathcal{X}_{k+1}$ for all $k \geq 0$
- $r_0 : V(G) \to \mathcal{X}_0$ is assumed to be given
- $\{\{r_k(w) \mid w \in N(v)\}\}$ is the multiset of (old) representations of the neighbors of $v \in V(G)$

The message passing framework

$$r_{k+1}(v) = \mathrm{upd}_k\left(r_k(v),\ \mathrm{agg}_k\left(\{\{r_k(w) \mid w \in N(v)\}\}\right)\right)$$

where

- $v \in V(G)$ is a vertex
- $k \geq 0$
- $r_{k+1} : V(G) \to \mathcal{X}_{k+1}$ for all $k \geq 0$
- $r_0 : V(G) \to \mathcal{X}_0$ is assumed to be given
- $\{\{r_k(w) \mid w \in N(v)\}\}$ is the multiset of (old) representations of the neighbors of $v \in V(G)$
- $\mathrm{agg}_k : \mathbb{N}^{\mathcal{X}_k} \to \mathcal{X}'_k$ *aggregates* a set of (old) representations to some value

The message passing framework

$$r_{k+1}(v) = \mathrm{upd}_k\left(r_k(v),\ \mathrm{agg}_k\left(\{\{r_k(w) \mid w \in N(v)\}\}\right)\right)$$

where

- $v \in V(G)$ is a vertex
- $k \geq 0$
- $r_{k+1} : V(G) \to \mathcal{X}_{k+1}$ for all $k \geq 0$
- $r_0 : V(G) \to \mathcal{X}_0$ is assumed to be given
- $\{\{r_k(w) \mid w \in N(v)\}\}$ is the multiset of (old) representations of the neighbors of $v \in V(G)$

- $\mathrm{agg}_k : \mathbb{N}^{\mathcal{X}_k} \to \mathcal{X}'_k$ *aggregates* a set of (old) representations to some value
- $\mathrm{upd}_k : \mathcal{X}_k \times \mathcal{X}'_k \to \mathcal{X}_{k+1}$ *updates* the representation of v given its old representation and the aggregate of the neighbors

The message passing framework

$$r_{k+1}(v) = \mathsf{upd}_k\left(r_k(v),\ \mathsf{agg}_k\left(\{\{r_k(w) \mid w \in N(v)\}\}\right)\right)$$

where

- $v \in V(G)$ is a vertex
- $k \geq 0$
- $r_{k+1} : V(G) \to \mathcal{X}_{k+1}$ for all $k \geq 0$
- $r_0 : V(G) \to \mathcal{X}_0$ is assumed to be given
- $\{\{r_k(w) \mid w \in N(v)\}\}$ is the multiset of (old) representations of the neighbors of $v \in V(G)$
- $\mathsf{agg}_k : \mathbb{N}^{\mathcal{X}_k} \to \mathcal{X}'_k$ *aggregates* a set of (old) representations to some value
- $\mathsf{upd}_k : \mathcal{X}_k \times \mathcal{X}'_k \to \mathcal{X}_{k+1}$ *updates* the representation of v given its old representation and the aggregate of the neighbors

We will omit k in the notation of upd_k and agg_k when $\mathsf{upd}_0 = \mathsf{upd}_1 = \ldots$

Weisfeiler Leman (1)

$$r_{k+1}^{WL}(v) = \#_k \left(r_k^{WL}(v),\ \left\{ \left\{ r_k^{WL}(w) \mid w \in N(v) \right\} \right\} \right)$$

Weisfeiler Leman (1)

$$r_{k+1}^{WL}(v) = \#_k\left(r_k^{WL}(v),\ \left\{\left\{r_k^{WL}(w) \mid w \in N\left(v\right)\right\}\right\}\right)$$

Where

Weisfeiler Leman (1)

$$r_{k+1}^{WL}(v) = \#_k\left(r_k^{WL}(v),\ \left\{\left\{r_k^{WL}(w) \mid w \in N(v)\right\}\right\}\right)$$

Where

- $r_0^{WL} : V(G) \to \mathcal{X}_0$ maps to a discrete space

Weisfeiler Leman (1)

$$r_{k+1}^{WL}(v) = \#_k\left(r_k^{WL}(v),\ \left\{\left\{r_k^{WL}(w) \mid w \in N(v)\right\}\right\}\right)$$

Where

- $r_0^{WL} : V(G) \to \mathcal{X}_0$ maps to a discrete space
- $\#_k : \mathcal{X}_k \times \mathbb{N}^{\mathcal{X}_k} \to \mathcal{X}_{k+1}$ is a perfect hash function

Weisfeiler Leman (2)

$$r_{k+1}^{WL}(v) = \#_k \left(r_k^{WL}(v),\ \left\{ \left\{ r_k^{WL}(w) \mid w \in N\left(v\right) \right\} \right\} \right)$$

Weisfeiler Leman (2)

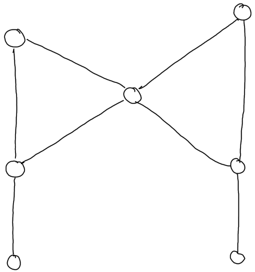
$$r_{k+1}^{WL}(v) = \#_k \left(r_k^{WL}(v),\ \left\{ \left\{ r_k^{WL}(w) \mid w \in N(v) \right\} \right\} \right)$$

Let's think of the hash values as colors of vertices

Weisfeiler Leman (2)

$$r_{k+1}^{WL}(v) = \#_k\left(r_k^{WL}(v),\ \left\{\left\{r_k^{WL}(w) \mid w \in N(v)\right\}\right\}\right)$$

Let's think of the hash values as colors of vertices

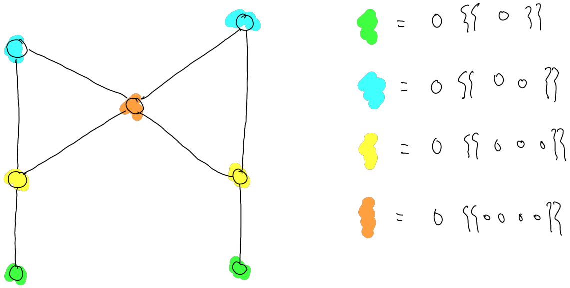


$k = 0$

Weisfeiler Leman (2)

$$r_{k+1}^{WL}(v) = \#_k \left(r_k^{WL}(v),\ \left\{\left\{ r_k^{WL}(w) \mid w \in N(v) \right\}\right\} \right)$$

Let's think of the hash values as colors of vertices

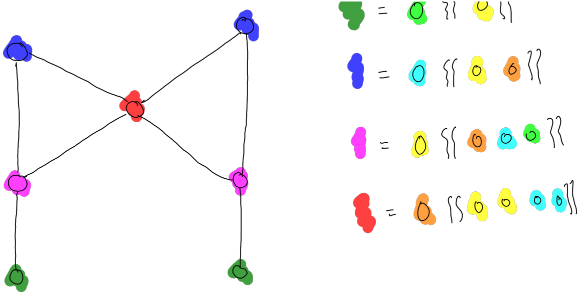


$k = 1$

Weisfeiler Leman (2)

$$r_{k+1}^{WL}(v) = \#_k \left(r_k^{WL}(v),\ \left\{ \left\{ r_k^{WL}(w) \mid w \in N(v) \right\} \right\} \right)$$

Let's think of the hash values as colors of vertices



$k = 2$

Message passing graph neural networks (1)

$$r_{k+1}^{MPNN}(v) = \mathrm{MLP}_k^{\mathrm{UPD}}\left(r_k^{MPNN}(v),\ \mathrm{MLP}_k\left(\sum_{w \in N(v)} r_k^{MPNN}(w)\right)\right)$$

Message passing graph neural networks (1)

$$r_{k+1}^{MPNN}(v) = \mathrm{MLP}_k^{\mathrm{UPD}}\left(r_k^{MPNN}(v),\ \mathrm{MLP}_k\left(\sum_{w \in N(v)} r_k^{MPNN}(w)\right)\right)$$

Where

Message passing graph neural networks (1)

$$r_{k+1}^{MPNN}(v) = \mathrm{MLP}_k^{\mathrm{UPD}}\left(r_k^{MPNN}(v),\ \mathrm{MLP}_k\left(\sum_{w \in N(v)} r_k^{MPNN}(w)\right)\right)$$

Where

- $r_0^{MPNN} : V(G) \to \mathbb{R}^d$

Message passing graph neural networks (1)

$$r_{k+1}^{MPNN}(v) = \mathrm{MLP}_k^{\mathrm{UPD}}\left(r_k^{MPNN}(v),\ \mathrm{MLP}_k\left(\sum_{w \in N(v)} r_k^{MPNN}(w)\right)\right)$$

Where

- $r_0^{MPNN} : V(G) \to \mathbb{R}^d$
- $\mathrm{MLP}_k^{\mathrm{AGG}} : \mathbb{R}^d \to \mathbb{R}^d$ is a multilayer perceptron

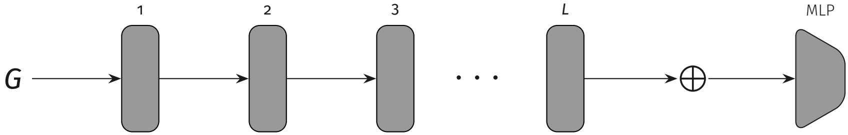
Message passing graph neural networks (1)

$$r_{k+1}^{MPNN}(v) = \mathrm{MLP}_k^{\mathrm{UPD}}\left(r_k^{MPNN}(v),\ \mathrm{MLP}_k\left(\sum_{w \in N(v)} r_k^{MPNN}(w)\right)\right)$$

Where

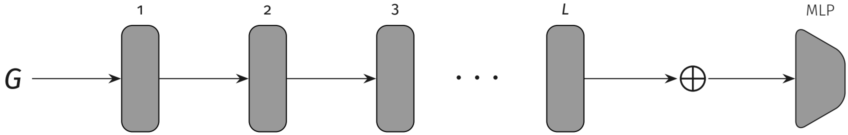
- $r_0^{MPNN} : V(G) \to \mathbb{R}^d$
- $\mathrm{MLP}_k^{\mathrm{AGG}} : \mathbb{R}^d \to \mathbb{R}^d$ is a multilayer perceptron
- $\mathrm{MLP}_k^{\mathrm{UPD}} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_d$ is a multilayer perceptron

Message passing graph neural networks (2)



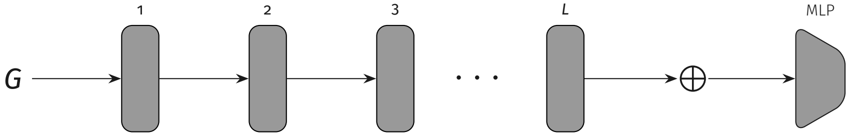
- MPNN layers are stacked on top of each other

Message passing graph neural networks (2)



- MPNN layers are stacked on top of each other
- Graph level tasks are solved by summing together all node representations, then a final MLP

Message passing graph neural networks (2)

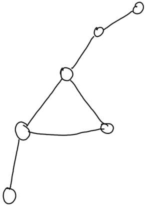


- MPNN layers are stacked on top of each other
- Graph level tasks are solved by summing together all node representations, then a final MLP
- Training can be done with gradient descent

Message Passing and the Weisfeiler Leman Algorithm | Issues

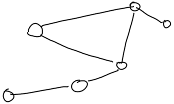
Isomorphic Graphs have Identical WL Label Histograms

G



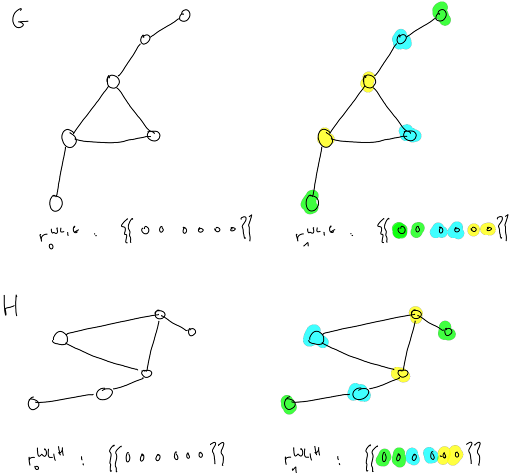
$r_0^{WL,G}: \{\{\circ\ \circ\ \circ\ \circ\ \circ\ \circ\}\}$

H

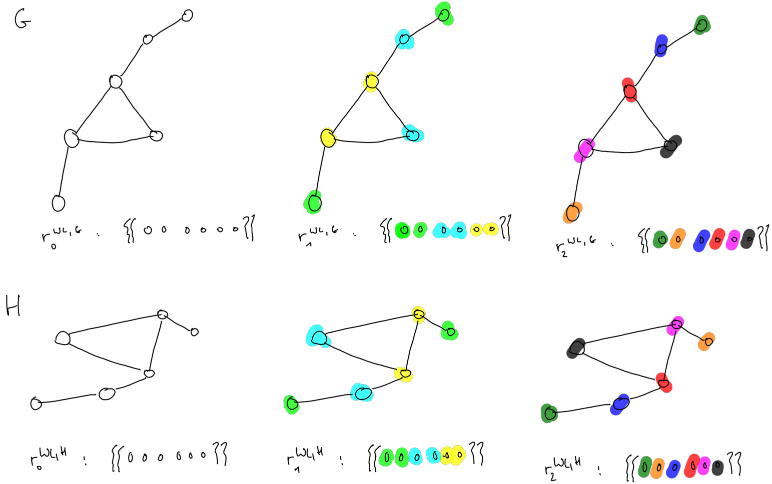


$r_0^{WL,H}: \{\{\circ\ \circ\ \circ\ \circ\ \circ\ \circ\}\}$

Isomorphic Graphs have Identical WL Label Histograms

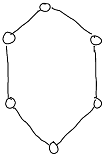


Isomorphic Graphs have Identical WL Label Histograms



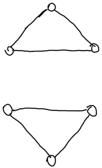
Nonisomorphic Graphs Can Have Identical Label Histograms

G



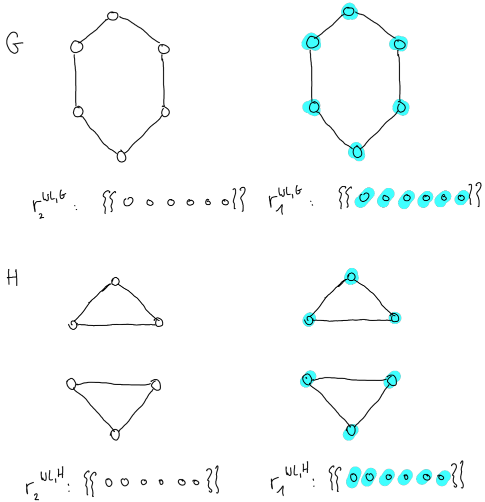
$r_2^{WL,G}$: $\{\{\circ\ \circ\ \circ\ \circ\ \circ\ \circ\}\}$

H

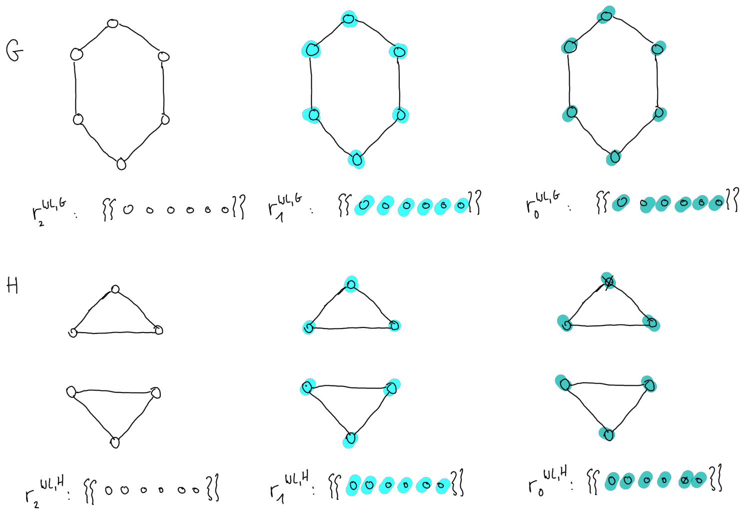


$r_2^{WL,H}$: $\{\{\circ\ \circ\ \circ\ \circ\ \circ\ \circ\}\}$

Nonisomorphic Graphs Can Have Identical Label Histograms



Nonisomorphic Graphs Can Have Identical Label Histograms



The connection between WL and MPNNs

$$r_k^{\mathrm{WL}}(G) = r_k^{\mathrm{WL}}(H) \Longrightarrow r_k^{\mathrm{MPNN}}(G) = r_k^{\mathrm{MPNN}}(H)$$

The connection between WL and MPNNs

$$r_k^{\mathrm{WL}}(G) = r_k^{\mathrm{WL}}(H) \Longrightarrow r_k^{\mathrm{MPNN}}(G) = r_k^{\mathrm{MPNN}}(H)$$

- Whenever WL cannot distinguish two graphs, *any* MPNN cannot compute different representations

The connection between WL and MPNNs

$$r_k^{\mathrm{WL}}(G) = r_k^{\mathrm{WL}}(H) \Longrightarrow r_k^{\mathrm{MPNN}}(G) = r_k^{\mathrm{MPNN}}(H)$$

- Whenever WL cannot distinguish two graphs, *any* MPNN cannot compute different representations
- MPNNs are incomplete

The connection between WL and MPNNs

$$r_k^{\mathrm{WL}}(G) = r_k^{\mathrm{WL}}(H) \Longrightarrow r_k^{\mathrm{MPNN}}(G) = r_k^{\mathrm{MPNN}}(H)$$

- Whenever WL cannot distinguish two graphs, *any* MPNN cannot compute different representations
- MPNNs are incomplete
- Their incompleteness can be bounded by the incompleteness of the WL algorithm

Homomorphism Counts as Graph Representations

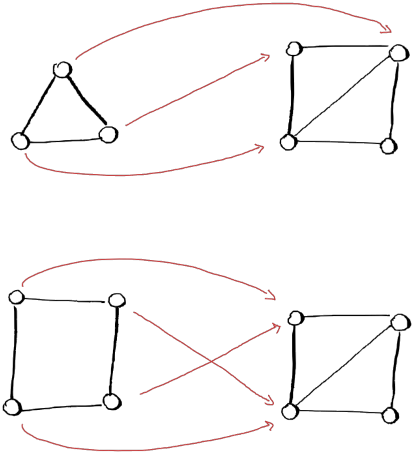
Homomorphism

A *homomorphism* from H to G is a function

$$h : V(H) \to V(G)$$

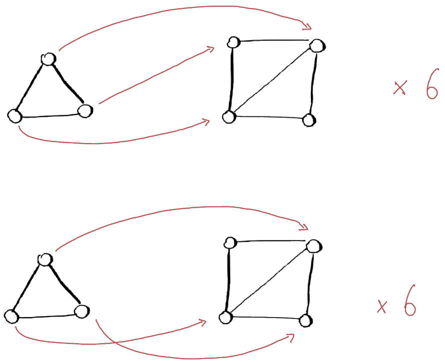
such that

$$(v, w) \in E(H) \implies (h(v), h(w)) \in V(G)$$



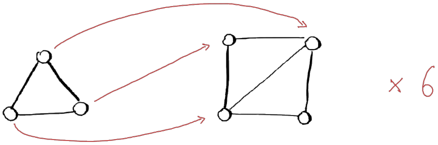
Counting Homomorphisms

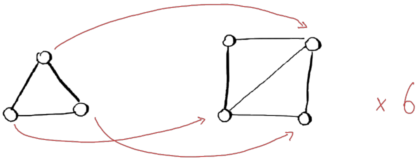
Given ***H*** and ***G***, we can ask *how many* homomorphisms exist from ***H*** to ***G***?



Counting Homomorphisms

Given $\boldsymbol{H}$ and $\boldsymbol{G}$, we can ask *how many* homomorphisms exist from $\boldsymbol{H}$ to $\boldsymbol{G}$?

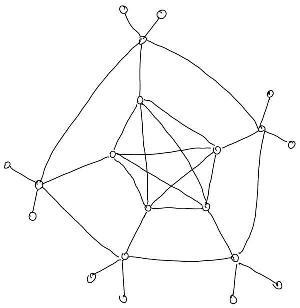




There are twelve homomorphisms from $\boldsymbol{H}$ to $\boldsymbol{G}$!

An intractable complete graph embedding

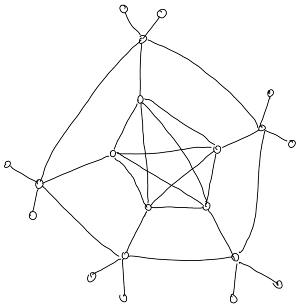
G



An intractable complete graph embedding

$\varphi_n(G)$

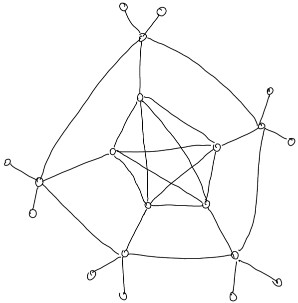
G

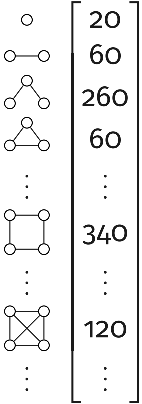


An intractable complete graph embedding

$\varphi_n(G)$

G





Theorem [Lovász 1967].
Two graphs G and H are isomorphic iff $\varphi_n(G) = \varphi_n(H)$

We can count homomorphisms (for some graphs) in practice!

- Homomorphism counting is fixed parameter tractable

We can count homomorphisms (for some graphs) in practice!

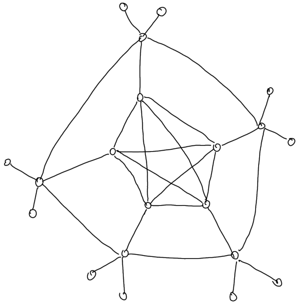
- Homomorphism counting is fixed parameter tractable
- The parameter is called tree-width

We can count homomorphisms (for some graphs) in practice!

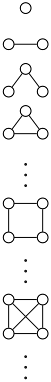
- Homomorphism counting is fixed parameter tractable
- The parameter is called tree-width
- If the pattern H has tree-width k, the homomorphisms from H to any G can be counted in $O(|V(G)|^k)$

An intractable complete graph embedding

G



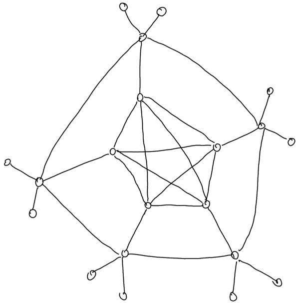
$\varphi_n(G)$



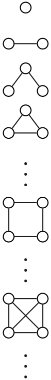
$$\begin{bmatrix} 20 \\ 60 \\ 260 \\ 60 \\ \vdots \\ 340 \\ \vdots \\ 120 \\ \vdots \end{bmatrix}$$

An intractable complete graph embedding

G



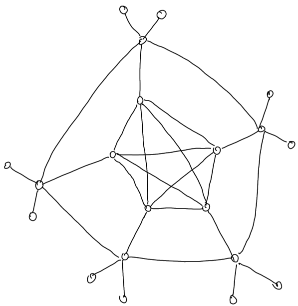
$\varphi_n(G)$

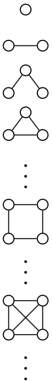


$$\begin{bmatrix} 20 \\ 60 \\ 260 \\ 60 \\ \vdots \\ 340 \\ \vdots \\ 120 \\ \vdots \end{bmatrix} \begin{array}{c} \\ \leftarrow \\ \\ \leftarrow \\ \\ \leftarrow \\ \\ \leftarrow \\ \\ \end{array}$$

~~An in~~tractable complete graph embedding

G





$$\varphi_n(G) = \begin{bmatrix} 20 \\ 60 \\ 260 \\ 60 \\ \vdots \\ 340 \\ \vdots \\ 120 \\ \vdots \end{bmatrix} \quad \begin{matrix} \\ \leftarrow \\ \\ \leftarrow \\ \\ \leftarrow \\ \\ \leftarrow \\ \\ \end{matrix} \quad \varphi_{\mathcal{F}}(G) = \begin{bmatrix} 0 \\ 60 \\ 0 \\ 60 \\ 0 \\ 340 \\ 0 \\ 120 \\ 0 \end{bmatrix}$$

How to select the patterns?

- Some patterns are more expensive than others

How to select the patterns?

- Some patterns are more expensive than others
- Some patterns might be more useful for the task at hand than others

How to select the patterns?

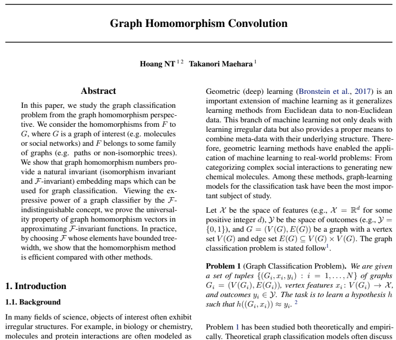
- Some patterns are more expensive than others
- Some patterns might be more useful for the task at hand than others

We will now see two variants how to select patterns

Graph Homomorphism Convolution (GHC)

NT and Maehara (2020)

- Introduce homomorphism counts as feature vectors of graphs



Graph Homomorphism Convolution

Hoang NT [1 2] Takanori Maehara [1]

Abstract

In this paper, we study the graph classification problem from the graph homomorphism perspective. We consider the homomorphisms from F to G, where G is a graph of interest (e.g. molecules or social networks) and F belongs to some family of graphs (e.g. paths or non-isomorphic trees). We show that graph homomorphism numbers provide a natural invariant (isomorphism invariant and $\mathcal{F}$-invariant) embedding maps which can be used for graph classification. Viewing the expressive power of a graph classifier by the $\mathcal{F}$-indistinguishable concept, we prove the universality property of graph homomorphism vectors in approximating $\mathcal{F}$-invariant functions. In practice, by choosing $\mathcal{F}$ whose elements have bounded tree-width, we show that the homomorphism method is efficient compared with other methods.

1. Introduction

1.1. Background

In many fields of science, objects of interest often exhibit irregular structures. For example, in biology or chemistry, molecules and protein interactions are often modeled as

Geometric (deep) learning (Bronstein et al., 2017) is an important extension of machine learning as it generalizes learning methods from Euclidean data to non-Euclidean data. This branch of machine learning not only deals with learning irregular data but also provides a proper means to combine meta-data with their underlying structure. Therefore, geometric learning methods have enabled the application of machine learning to real-world problems: From categorizing complex social interactions to generating new chemical molecules. Among these methods, graph-learning models for the classification task have been the most important subject of study.

Let $\mathcal{X}$ be the space of features (e.g., $\mathcal{X} = \mathbb{R}^d$ for some positive integer d), $\mathcal{Y}$ be the space of outcomes (e.g., $\mathcal{Y} = \{0, 1\}$), and $G = (V(G), E(G))$ be a graph with a vertex set $V(G)$ and edge set $E(G) \subseteq V(G) \times V(G)$. The graph classification problem is stated follow[1].

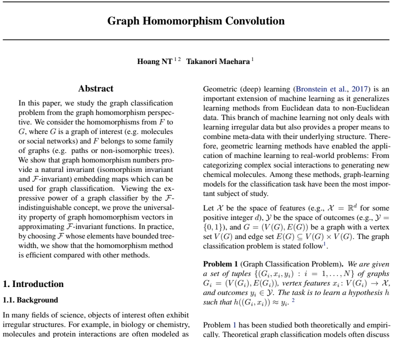
Problem 1 (Graph Classification Problem). *We are given a set of tuples* $\{(G_i, x_i, y_i) : i = 1, \dots, N\}$ *of graphs* $G_i = (V(G_i), E(G_i))$, *vertex features* $x_i : V(G_i) \to \mathcal{X}$, *and outcomes* $y_i \in \mathcal{Y}$. *The task is to learn a hypothesis* h *such that* $h((G_i, x_i)) \approx y_i$. [2]

Problem 1 has been studied both theoretically and empirically. Theoretical graph classification models often discuss

Graph Homomorphism Convolution (GHC)

NT and Maehara (2020)

- Introduce homomorphism counts as feature vectors of graphs
- Propose to select 'suitable, small' pattern set $\mathcal{F}$



Graph Homomorphism Convolution

Hoang NT [1 2] Takanori Maehara [1]

Abstract

In this paper, we study the graph classification problem from the graph homomorphism perspective. We consider the homomorphisms from F to G, where G is a graph of interest (e.g. molecules or social networks) and F belongs to some family of graphs (e.g. paths or non-isomorphic trees). We show that graph homomorphism numbers provide a natural invariant (isomorphism invariant and $\mathcal{F}$-invariant) embedding maps which can be used for graph classification. Viewing the expressive power of a graph classifier by the $\mathcal{F}$-indistinguishable concept, we prove the universality property of graph homomorphism vectors in approximating $\mathcal{F}$-invariant functions. In practice, by choosing $\mathcal{F}$ whose elements have bounded tree-width, we show that the homomorphism method is efficient compared with other methods.

1. Introduction

1.1. Background

In many fields of science, objects of interest often exhibit irregular structures. For example, in biology or chemistry, molecules and protein interactions are often modeled as

Geometric (deep) learning (Bronstein et al., 2017) is an important extension of machine learning as it generalizes learning methods from Euclidean data to non-Euclidean data. This branch of machine learning not only deals with learning irregular data but also provides a proper means to combine meta-data with their underlying structure. Therefore, geometric learning methods have enabled the application of machine learning to real-world problems: From categorizing complex social interactions to generating new chemical molecules. Among these methods, graph-learning models for the classification task have been the most important subject of study.

Let $\mathcal{X}$ be the space of features (e.g., $\mathcal{X} = \mathbb{R}^d$ for some positive integer d), $\mathcal{Y}$ be the space of outcomes (e.g., $\mathcal{Y} = \{0, 1\}$), and $G = (V(G), E(G))$ be a graph with a vertex set $V(G)$ and edge set $E(G) \subseteq V(G) \times V(G)$. The graph classification problem is stated follow[1].

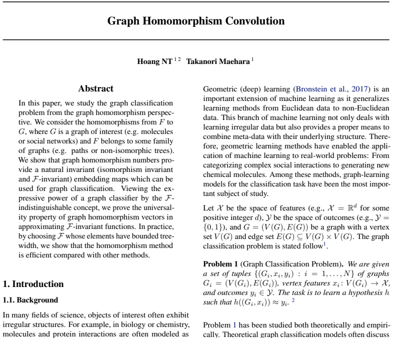
Problem 1 (Graph Classification Problem). *We are given a set of tuples* $\{(G_i, x_i, y_i) : i = 1, \ldots, N\}$ *of graphs* $G_i = (V(G_i), E(G_i))$, *vertex features* $x_i \colon V(G_i) \to \mathcal{X}$, *and outcomes* $y_i \in \mathcal{Y}$. *The task is to learn a hypothesis* h *such that* $h((G_i, x_i)) \approx y_i$. [2]

Problem 1 has been studied both theoretically and empirically. Theoretical graph classification models often discuss

Graph Homomorphism Convolution (GHC)

NT and Maehara (2020)

- Introduce homomorphism counts as feature vectors of graphs
- Propose to select 'suitable, small' pattern set $\mathcal{F}$
  - The first 13 trees



Graph Homomorphism Convolution

Hoang NT [1 2] Takanori Maehara [1]

Abstract

In this paper, we study the graph classification problem from the graph homomorphism perspective. We consider the homomorphisms from F to G, where G is a graph of interest (e.g. molecules or social networks) and F belongs to some family of graphs (e.g. paths or non-isomorphic trees). We show that graph homomorphism numbers provide a natural invariant (isomorphism invariant and $\mathcal{F}$-invariant) embedding maps which can be used for graph classification. Viewing the expressive power of a graph classifier by the $\mathcal{F}$-indistinguishable concept, we prove the universality property of graph homomorphism vectors in approximating $\mathcal{F}$-invariant functions. In practice, by choosing $\mathcal{F}$ whose elements have bounded tree-width, we show that the homomorphism method is efficient compared with other methods.

1. Introduction

1.1. Background

In many fields of science, objects of interest often exhibit irregular structures. For example, in biology or chemistry, molecules and protein interactions are often modeled as

Geometric (deep) learning (Bronstein et al., 2017) is an important extension of machine learning as it generalizes learning methods from Euclidean data to non-Euclidean data. This branch of machine learning not only deals with learning irregular data but also provides a proper means to combine meta-data with their underlying structure. Therefore, geometric learning methods have enabled the application of machine learning to real-world problems: From categorizing complex social interactions to generating new chemical molecules. Among these methods, graph-learning models for the classification task have been the most important subject of study.

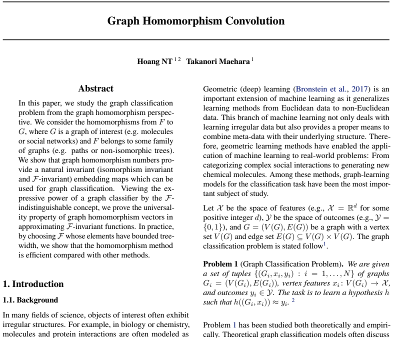
Let $\mathcal{X}$ be the space of features (e.g., $\mathcal{X} = \mathbb{R}^d$ for some positive integer d), $\mathcal{Y}$ be the space of outcomes (e.g., $\mathcal{Y} = \{0, 1\}$), and $G = (V(G), E(G))$ be a graph with a vertex set $V(G)$ and edge set $E(G) \subseteq V(G) \times V(G)$. The graph classification problem is stated follow[1].

Problem 1 (Graph Classification Problem). *We are given a set of tuples $\{(G_i, x_i, y_i) : i = 1, \dots, N\}$ of graphs $G_i = (V(G_i), E(G_i))$, vertex features $x_i \colon V(G_i) \to \mathcal{X}$, and outcomes $y_i \in \mathcal{Y}$. The task is to learn a hypothesis h such that $h((G_i, x_i)) \approx y_i$.* [2]

Problem 1 has been studied both theoretically and empirically. Theoretical graph classification models often discuss

Graph Homomorphism Convolution (GHC) NT and Maehara (2020)

- Introduce homomorphism counts as feature vectors of graphs
- Propose to select ’suitable, small’ pattern set $\mathcal{F}$
  - The first 13 trees
  - Cycles up to length 7



Graph Homomorphism Convolution

Hoang NT [1 2] Takanori Maehara [1]

Abstract

In this paper, we study the graph classification problem from the graph homomorphism perspective. We consider the homomorphisms from F to G, where G is a graph of interest (e.g. molecules or social networks) and F belongs to some family of graphs (e.g. paths or non-isomorphic trees). We show that graph homomorphism numbers provide a natural invariant (isomorphism invariant and $\mathcal{F}$-invariant) embedding maps which can be used for graph classification. Viewing the expressive power of a graph classifier by the $\mathcal{F}$-indistinguishable concept, we prove the universality property of graph homomorphism vectors in approximating $\mathcal{F}$-invariant functions. In practice, by choosing $\mathcal{F}$ whose elements have bounded tree-width, we show that the homomorphism method is efficient compared with other methods.

1. Introduction

1.1. Background

In many fields of science, objects of interest often exhibit irregular structures. For example, in biology or chemistry, molecules and protein interactions are often modeled as

Geometric (deep) learning (Bronstein et al., 2017) is an important extension of machine learning as it generalizes learning methods from Euclidean data to non-Euclidean data. This branch of machine learning not only deals with learning irregular data but also provides a proper means to combine meta-data with their underlying structure. Therefore, geometric learning methods have enabled the application of machine learning to real-world problems: From categorizing complex social interactions to generating new chemical molecules. Among these methods, graph-learning models for the classification task have been the most important subject of study.

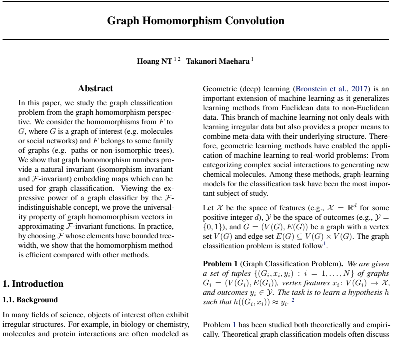
Let $\mathcal{X}$ be the space of features (e.g., $\mathcal{X} = \mathbb{R}^d$ for some positive integer d), $\mathcal{Y}$ be the space of outcomes (e.g., $\mathcal{Y} = \{0, 1\}$), and $G = (V(G), E(G))$ be a graph with a vertex set $V(G)$ and edge set $E(G) \subseteq V(G) \times V(G)$. The graph classification problem is stated follow[1].

Problem 1 (Graph Classification Problem). *We are given a set of tuples $\{(G_i, x_i, y_i) : i = 1, \dots, N\}$ of graphs $G_i = (V(G_i), E(G_i))$, vertex features $x_i \colon V(G_i) \to \mathcal{X}$, and outcomes $y_i \in \mathcal{Y}$. The task is to learn a hypothesis h such that $h((G_i, x_i)) \approx y_i$.* [2]

Problem 1 has been studied both theoretically and empirically. Theoretical graph classification models often discuss

Graph Homomorphism Convolution (GHC) (NT and Maehara (2020))

- Introduce homomorphism counts as feature vectors of graphs
- Propose to select 'suitable, small' pattern set $\mathcal{F}$
  - The first 13 trees
  - Cycles up to length 7
- Use an SVM with these features



Graph Homomorphism Convolution

Hoang NT [1 2] Takanori Maehara [1]

Abstract

In this paper, we study the graph classification problem from the graph homomorphism perspective. We consider the homomorphisms from F to G, where G is a graph of interest (e.g. molecules or social networks) and F belongs to some family of graphs (e.g. paths or non-isomorphic trees). We show that graph homomorphism numbers provide a natural invariant (isomorphism invariant and $\mathcal{F}$-invariant) embedding maps which can be used for graph classification. Viewing the expressive power of a graph classifier by the $\mathcal{F}$-indistinguishable concept, we prove the universality property of graph homomorphism vectors in approximating $\mathcal{F}$-invariant functions. In practice, by choosing $\mathcal{F}$ whose elements have bounded tree-width, we show that the homomorphism method is efficient compared with other methods.

1. Introduction

1.1. Background

In many fields of science, objects of interest often exhibit irregular structures. For example, in biology or chemistry, molecules and protein interactions are often modeled as

Geometric (deep) learning (Bronstein et al., 2017) is an important extension of machine learning as it generalizes learning methods from Euclidean data to non-Euclidean data. This branch of machine learning not only deals with learning irregular data but also provides a proper means to combine meta-data with their underlying structure. Therefore, geometric learning methods have enabled the application of machine learning to real-world problems: From categorizing complex social interactions to generating new chemical molecules. Among these methods, graph-learning models for the classification task have been the most important subject of study.

Let $\mathcal{X}$ be the space of features (e.g., $\mathcal{X} = \mathbb{R}^d$ for some positive integer d), $\mathcal{Y}$ be the space of outcomes (e.g., $\mathcal{Y} = \{0, 1\}$), and $G = (V(G), E(G))$ be a graph with a vertex set $V(G)$ and edge set $E(G) \subseteq V(G) \times V(G)$. The graph classification problem is stated follow[1].

Problem 1 (Graph Classification Problem). *We are given a set of tuples $\{(G_i, x_i, y_i) : i = 1, \dots, N\}$ of graphs $G_i = (V(G_i), E(G_i))$, vertex features $x_i : V(G_i) \to \mathcal{X}$, and outcomes $y_i \in \mathcal{Y}$. The task is to learn a hypothesis h such that $h((G_i, x_i)) \approx y_i$.* [2]

Problem 1 has been studied both theoretically and empirically. Theoretical graph classification models often discuss

GHC: Experimental results

Table 2. Classification accuracy over 10 experiments
(a) Synthetic datasets

METHODS	CSL	BIPARTITE	PAULUS25
Practical models			
GIN	10.00 ± 0.00	55.75 ± 7.91	7.14 ± 0.00
GNTK	10.00 ± 0.00	58.03 ± 6.84	7.14 ± 0.00
Theory models			
Ring-GNN	$10 \sim 80 \pm 15.7$	55.72 ± 6.95	7.15 ± 0.00
GHC-Tree	10.00 ± 0.00	52.68 ± 7.15	7.14 ± 0.00
GHC-Cycle	$\mathbf{100.0} \pm \mathbf{0.00}$	$\mathbf{100.0} \pm \mathbf{0.00}$	7.14 ± 0.00

(b) Benchmark datasets

METHODS	MUTAG	IMDB-BIN	IMDB-MUL
Practical models			
GNTK	89.46 ± 7.03	75.61 ± 3.98	51.91 ± 3.56
GIN	89.40 ± 5.60	70.70 ± 1.10	43.20 ± 2.00
PATCHY-SAN	89.92 ± 4.50	71.00 ± 2.20	45.20 ± 2.80
WL kernel	90.40 ± 5.70	73.80 ± 3.90	50.90 ± 3.80
Theory models			
Ring-GNN	78.07 ± 5.61	73.00 ± 5.40	48.20 ± 2.70
GHC-Tree	89.28 ± 8.26	72.10 ± 2.62	48.60 ± 4.40
GHC-Cycles	87.81 ± 7.46	70.93 ± 4.54	47.41 ± 3.67

GHC: Experimental results

Table 2. Classification accuracy over 10 experiments
(a) Synthetic datasets

METHODS	CSL	BIPARTITE	PAULUS25
Practical models			
GIN	10.00 ± 0.00	55.75 ± 7.91	7.14 ± 0.00
GNTK	10.00 ± 0.00	58.03 ± 6.84	7.14 ± 0.00
Theory models			
Ring-GNN	$10 \sim 80 \pm 15.7$	55.72 ± 6.95	7.15 ± 0.00
GHC-Tree	10.00 ± 0.00	52.68 ± 7.15	7.14 ± 0.00
GHC-Cycle	$\mathbf{100.0 \pm 0.00}$	$\mathbf{100.0 \pm 0.00}$	7.14 ± 0.00

(b) Benchmark datasets

METHODS	MUTAG	IMDB-BIN	IMDB-MUL
Practical models			
GNTK	89.46 ± 7.03	75.61 ± 3.98	51.91 ± 3.56
GIN	89.40 ± 5.60	70.70 ± 1.10	43.20 ± 2.00
PATCHY-SAN	89.92 ± 4.50	71.00 ± 2.20	45.20 ± 2.80
WL kernel	90.40 ± 5.70	73.80 ± 3.90	50.90 ± 3.80
Theory models			
Ring-GNN	78.07 ± 5.61	73.00 ± 5.40	48.20 ± 2.70
GHC-Tree	89.28 ± 8.26	72.10 ± 2.62	48.60 ± 4.40
GHC-Cycles	87.81 ± 7.46	70.93 ± 4.54	47.41 ± 3.67

- Good results on some synthetic datasets

GHC: Experimental results

Table 2. Classification accuracy over 10 experiments
(a) Synthetic datasets

METHODS	CSL	BIPARTITE	PAULUS25
Practical models			
GIN	10.00 ± 0.00	55.75 ± 7.91	7.14 ± 0.00
GNTK	10.00 ± 0.00	58.03 ± 6.84	7.14 ± 0.00
Theory models			
Ring-GNN	$10 \sim 80 \pm 15.7$	55.72 ± 6.95	7.15 ± 0.00
GHC-Tree	10.00 ± 0.00	52.68 ± 7.15	7.14 ± 0.00
GHC-Cycle	$\mathbf{100.0 \pm 0.00}$	$\mathbf{100.0 \pm 0.00}$	7.14 ± 0.00

(b) Benchmark datasets

METHODS	MUTAG	IMDB-BIN	IMDB-MUL
Practical models			
GNTK	89.46 ± 7.03	75.61 ± 3.98	51.91 ± 3.56
GIN	89.40 ± 5.60	70.70 ± 1.10	43.20 ± 2.00
PATCHY-SAN	89.92 ± 4.50	71.00 ± 2.20	45.20 ± 2.80
WL kernel	90.40 ± 5.70	73.80 ± 3.90	50.90 ± 3.80
Theory models			
Ring-GNN	78.07 ± 5.61	73.00 ± 5.40	48.20 ± 2.70
GHC-Tree	89.28 ± 8.26	72.10 ± 2.62	48.60 ± 4.40
GHC-Cycles	87.81 ± 7.46	70.93 ± 4.54	47.41 ± 3.67

- Good results on some synthetic datasets
- Competitive results on (smaller) benchmark datasets

GHC is incomplete

- GHC in practice requires a fixed, user defined choice of the pattern set $\mathcal{F}$

GHC is incomplete

- GHC in practice requires a fixed, user defined choice of the pattern set $\mathcal{F}$
- This allows to bound the expressivity of GHC by an extension of the WL algorithm: k-WL (Neuen (2024))

Expectation-Complete Graph Representations with Homomorphisms




ICML 2023

Pascal Welke*, Maximilian Thiessen*, Fabian Jogl, and Thomas Gärtner

TU Wien
Vienna | Austria
Research Unit Machine Learning

At a glance

- Expressiveness bounded by k-WL

At a glance

- Expressiveness bounded by k-WL

$\Rightarrow$ choice of architecture implies a fixed limit on what graphs can be distinguished

At a glance

- Expressiveness bounded by k-WL

$\Rightarrow$ choice of architecture implies a fixed limit on what graphs can be distinguished

- What can we do if we don't know anything about our datset?

At a glance

- Expressiveness bounded by k-WL

$\Rightarrow$ choice of architecture implies a fixed limit on what graphs can be distinguished

- What can we do if we don't know anything about our datset?

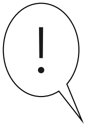
- We present an architecture which has no upper expressivity bound

At a glance

- Expressiveness bounded by k-WL

$\Rightarrow$ choice of architecture implies a fixed limit on what graphs can be distinguished

- What can we do if we don't know anything about our datset?



- We present an architecture which has no upper expressivity bound
- Asymptotically, our graph representation is complete.

At a glance

- Expressiveness bounded by k-WL

$\Rightarrow$ choice of architecture implies a fixed limit on what graphs can be distinguished

- What can we do if we don't know anything about our datset?

- We present an architecture which has no upper expressivity bound
- Asymptotically, our graph representation is complete.

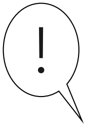
$\Rightarrow$ allows to adapt to challenging learning tasks without domain knowledge

At a glance

- Expressiveness bounded by $\boldsymbol{k}$-WL

$\Rightarrow$ choice of architecture implies a fixed limit on what graphs can be distinguished

- What can we do if we don't know anything about our datset?



- We present an architecture which has no upper expressivity bound
- Asymptotically, our graph representation is complete.

$\Rightarrow$ allows to adapt to challenging learning tasks without domain knowledge

$\Rightarrow$ works well in practice

What if we keep completeness . . .

. . . in expectation?

Expectation complete graph embeddings

Let $\phi_X : \mathcal{G} \to V$ depend on a random variable X drawn from a distr. $\mathcal{D}$ over a set $\mathcal{X}$

Expectation complete graph embeddings

Let $\phi_X : \mathcal{G} \to V$ depend on a random variable X drawn from a distr. $\mathcal{D}$ over a set $\mathcal{X}$

We call ϕ_X complete in expectation if the expectation

$$\mathop{\mathbb{E}}_{X \sim \mathcal{D}}[\phi_X(\cdot)] = \sum_{t \in \mathcal{X}} \Pr(X = t)\phi_t(\cdot)$$

is a complete graph embedding

What is the benefit?

Sampling $X_1, X_2, X_3, \ldots$ will eventually make the joint embedding

$$(\phi_{X_1}(G), \phi_{X_2}(G), \phi_{X_3}(G), \ldots)$$

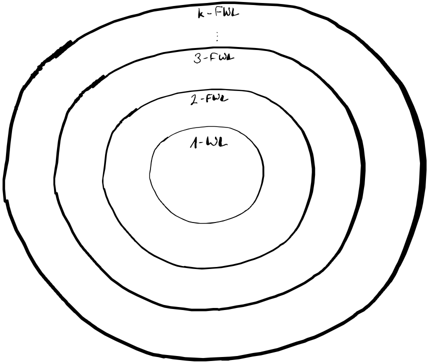
arbitrarily expressive

What is the benefit?

Sampling $X_1, X_2, X_3, \ldots$ will eventually make the joint embedding

$$(\phi_{X_1}(G), \phi_{X_2}(G), \phi_{X_3}(G), \ldots)$$

arbitrarily expressive

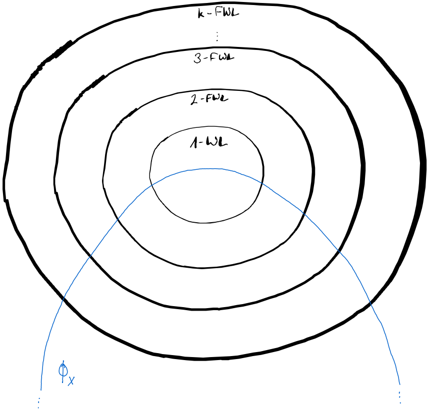


What is the benefit?

Sampling $X_1, X_2, X_3, \ldots$ will eventually make the joint embedding

$$(\phi_{X_1}(G), \phi_{X_2}(G), \phi_{X_3}(G), \ldots)$$

arbitrarily expressive

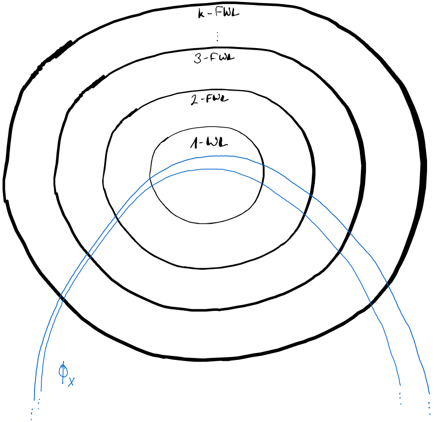


What is the benefit?

Sampling $X_1, X_2, X_3, \ldots$ will eventually make the joint embedding

$$(\phi_{X_1}(G), \phi_{X_2}(G), \phi_{X_3}(G), \ldots)$$

arbitrarily expressive

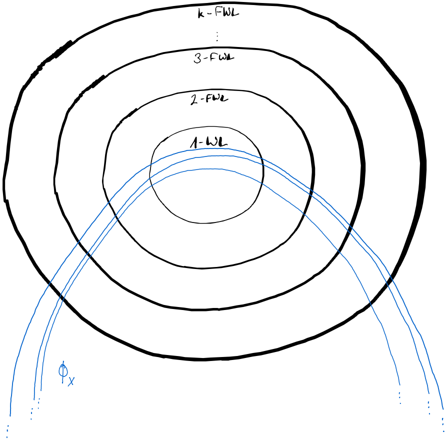


What is the benefit?

Sampling $X_1, X_2, X_3, \ldots$ will eventually make the joint embedding

$$(\phi_{X_1}(G), \phi_{X_2}(G), \phi_{X_3}(G), \ldots)$$

arbitrarily expressive



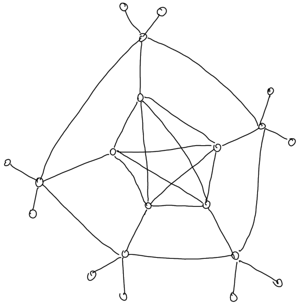
What if we keep completeness …

… in expectation

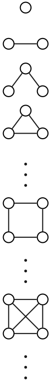
… efficiently

An intractable complete graph embedding

G



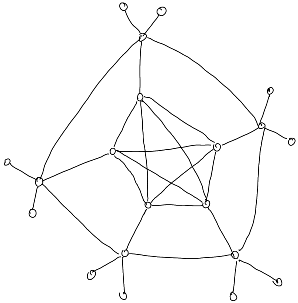
$\varphi_n(G)$

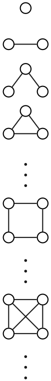


$$\begin{bmatrix}20\\60\\260\\60\\\vdots\\340\\\vdots\\120\\\vdots\end{bmatrix}$$

An intractable complete graph embedding

G



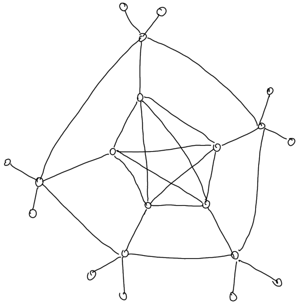


$$\varphi_n(G)$$

$$\begin{array}{cc} \begin{bmatrix} 20 \\ 60 \\ 260 \\ 60 \\ \vdots \\ 340 \\ \vdots \\ 120 \\ \vdots \end{bmatrix} & \begin{array}{c} \\ \leftarrow \\ \\ \leftarrow \\ \\ \leftarrow \\ \\ \leftarrow \\ \\ \end{array} \end{array}$$

~~An in~~tractable complete graph embedding

G



$$\varphi_n(G) = \begin{bmatrix} 20 \\ 60 \\ 260 \\ 60 \\ \vdots \\ 340 \\ \vdots \\ 120 \\ \vdots \end{bmatrix} \qquad \begin{matrix} \\ \leftarrow \\ \\ \leftarrow \\ \\ \leftarrow \\ \\ \leftarrow \\ \\ \end{matrix} \qquad \varphi_{\mathcal{F}}(G) = \begin{bmatrix} 0 \\ 60 \\ 0 \\ 60 \\ 0 \\ 340 \\ 0 \\ 120 \\ 0 \end{bmatrix}$$

Efficient and expectation-complete graph embeddings

- Homomorphism counting is fixed parameter tractable

Efficient and expectation-complete graph embeddings

- Homomorphism counting is fixed parameter tractable
- We design a distribution $\mathcal{D}$ that weights down expensive patterns

Efficient and expectation-complete graph embeddings

- Homomorphism counting is fixed parameter tractable
- We design a distribution $\mathcal{D}$ that weights down expensive patterns

Theorem (ICML 2023)

Computing the expectation-complete graph embedding $\phi_X(G)$ with $X \sim \mathcal{D}$ takes polynomial time in $V(G)$ in expectation for all $G \in \mathcal{G}_n$.

Efficient and expectation-complete graph embeddings

- Homomorphism counting is fixed parameter tractable
- We design a distribution $\mathcal{D}$ that weights down expensive patterns

Theorem (ICML 2023)

Computing the expectation-complete graph embedding $\phi_X(G)$ with $X \sim \mathcal{D}$ takes polynomial time in $V(G)$ in expectation for all $G \in \mathcal{G}_n$.

- We also showed

Efficient and expectation-complete graph embeddings

- Homomorphism counting is fixed parameter tractable
- We design a distribution $\mathcal{D}$ that weights down expensive patterns

Theorem (ICML 2023)

Computing the expectation-complete graph embedding $\phi_X(G)$ with $X \sim \mathcal{D}$ takes polynomial time in $V(G)$ in expectation for all $G \in \mathcal{G}_n$.

- We also showed
  - convergence results

Efficient and expectation-complete graph embeddings

- Homomorphism counting is fixed parameter tractable
- We design a distribution $\mathcal{D}$ that weights down expensive patterns

Theorem (ICML 2023)

Computing the expectation-complete graph embedding $\phi_X(G)$ with $X \sim \mathcal{D}$ takes polynomial time in $V(G)$ in expectation for all $G \in \mathcal{G}_n$.

- We also showed
  - convergence results
  - universal approximation results

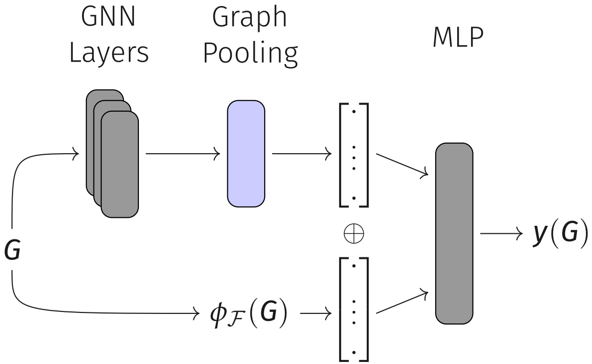
Efficient and expectation-complete GNNs

We can make any (message passing) GNN expectation-complete

Efficient and expectation-complete GNNs

We can make any (message passing) GNN expectation-complete

ICML 2023
LoG 2023



Empirical results

Table 1. Performance of different GNNs on 9 OGB benchmarks and `ZINC`. Baseline of a GNN with homorphism counts is the same GNN without homomorphism counts. Results for GNNs with homorphism counts are averaged over 9 different random samples of pattern graphs.

	Top 1 / 2 / 3	Beats baseline
GIN	0% / 0% / 0%	-
GIN+hom	0% / 10% / 10%	100%
GCN	0% / 0% / 0%	-
GCN+hom	10% / 10% / 20%	90%
GIN+F	0% / 10% / 50%	-
GIN+hom +F	20% / 40% / 70%	90%
GCN+F	0% / 50% / 60%	-
GCN+hom+F	70% / 80% / 90%	90%

Empirical results

Table 1. Performance of different GNNs on 9 OGB benchmarks and ZINC. Baseline of a GNN with homorphism counts is the same GNN without homomorphism counts. Results for GNNs with homorphism counts are averaged over 9 different random samples of pattern graphs.

	Top 1 / 2 / 3	Beats baseline
GIN	0% / 0% / 0%	-
GIN+hom	0% / 10% / 10%	100%
GCN	0% / 0% / 0%	-
GCN+hom	10% / 10% / 20%	90%
GIN+F	0% / 10% / 50%	-
GIN+hom +F	20% / 40% / 70%	90%
GCN+F	0% / 50% / 60%	-
GCN+hom+F	70% / 80% / 90%	90%

Table 2. Accuracy on synthetic data

Method	CSL	PAULUS25
GIN	10.00 ± 0.00	7.14 ± 0.00
GNTK	10.00 ± 0.00	7.14 ± 0.00
GHC-Tree	10.00 ± 0.00	7.14 ± 0.00
GHC-Cycle	100.0 ± 0.00	7.14 ± 0.00
WL	10.00 ± 0.00	7.14 ± 0.00
Ours	37.67 ± 9.11	100.0 ± 0.00

An open question and a recent answer

- Our runtime is polynomial in expectation, but

An open question and a recent answer

- Our runtime is polynomial in expectation, but
  - We can realistically sample 20-100 patterns
  - (that suffices in practice)

An open question and a recent answer

- Our runtime is polynomial in expectation, but
  - We can realistically sample 20-100 patterns
  - (that suffices in practice)
- How can we speedup the runtime while maintaining the theoretical properties?

An open question and a recent answer

- Our runtime is polynomial in expectation, but
  - We can realistically sample 20-100 patterns
  - (that suffices in practice)
- How can we speedup the runtime while maintaining the theoretical properties?

Estimating homomorphism counts instead of exact computation might work well

An open question and a recent answer

- Our runtime is polynomial in expectation, but
  - We can realistically sample 20-100 patterns
  - (that suffices in practice)
- How can we speedup the runtime while maintaining the theoretical properties?

Estimating homomorphism counts instead of exact computation might work well

- Beaujean et al (2021)

An open question and a recent answer

- Our runtime is polynomial in expectation, but
  - We can realistically sample 20-100 patterns
  - (that suffices in practice)
- How can we speedup the runtime while maintaining the theoretical properties?

Estimating homomorphism counts instead of exact computation might work well

- Beaujean et al (2021)
- BSc thesis 2023

An open question and a recent answer

- Our runtime is polynomial in expectation, but
  - We can realistically sample 20-100 patterns
  - (that suffices in practice)
- How can we speedup the runtime while maintaining the theoretical properties?

Estimating homomorphism counts instead of exact computation might work well

- Beaujean et al (2021)
- BSc thesis 2023
- KDD 2020

An open question and a recent answer

- Our runtime is polynomial in expectation, but
  - We can realistically sample 20-100 patterns
  - (that suffices in practice)
- How can we speedup the runtime while maintaining the theoretical properties?

Estimating homomorphism counts instead of exact computation might work well

- Beaujean et al (2021)
- BSc thesis 2023
- KDD 2020
- fast and precise in practice

Homomorphism Counts as Node Representations

Connecting homomorphism counting and message passing

- So far, message passing and homomorphism counting have touched, but not really interacted

Connecting homomorphism counting and message passing

- So far, message passing and homomorphism counting have touched, but not really interacted
- Homomorphism counts can also be included in the message passing

Rooted homomorphism counting

A *rooted graph* (G, v) is a graph G with a special root $v \in V(G)$

Rooted homomorphism counting

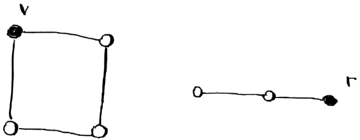
A *rooted graph* (G, v) is a graph G with a special root $v \in V(G)$

A rooted homomorphism h from (H, r) to (G, v) is a homomorphism h with $h(r) = v$

Rooted homomorphism counting

A *rooted graph* (G, v) is a graph G with a special root $v \in V(G)$

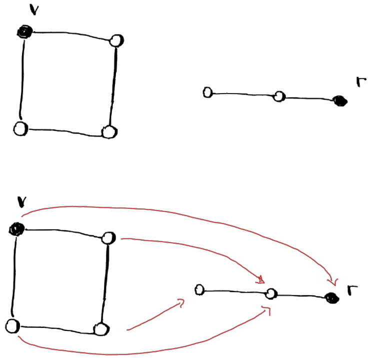
A rooted homomorphism h from (H, r) to (G, v) is a homomorphism h with $h(r) = v$



Rooted homomorphism counting

A *rooted graph* (G, v) is a graph G with a special root $v \in V(G)$

A rooted homomorphism h from (H, r) to (G, v) is a homomorphism h with $h(r) = v$

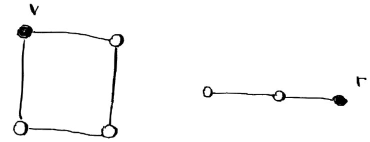


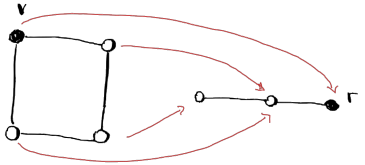
Rooted homomorphism counting

A *rooted graph* (G, v) is a graph G with a special root $v \in V(G)$

A rooted homomorphism h from (H, r) to (G, v) is a homomorphism h with $h(r) = v$

- We can now count rooted homomorphisms for any node v in G



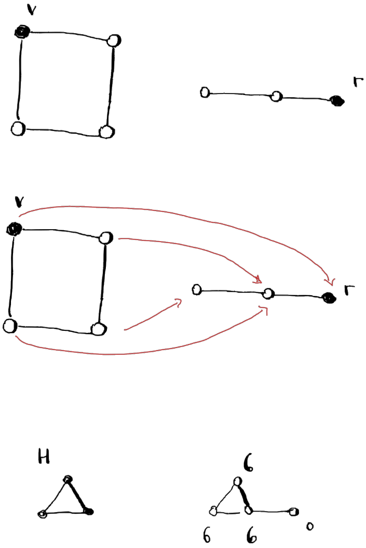


Rooted homomorphism counting

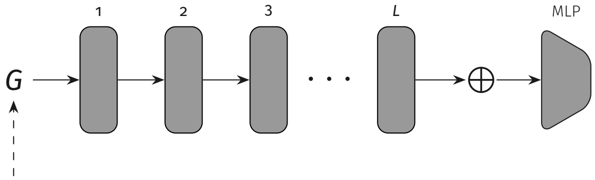
A *rooted graph* (G, v) is a graph G with a special root $v \in V(G)$

A rooted homomorphism h from (H, r) to (G, v) is a homomorphism h with $h(r) = v$

- We can now count rooted homomorphisms for any node v in G

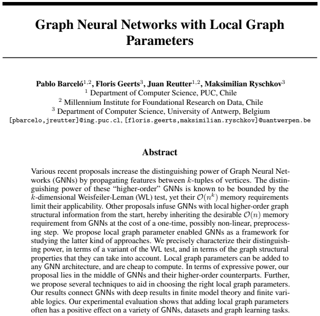


Graph Homomorphism Convolution ($\mathcal{F}$-MPNNs) Barceló et al (2021)



add hom-counts here

- This architecture is more expressive than WL



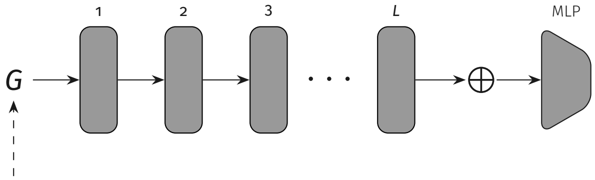
Graph Neural Networks with Local Graph Parameters

Pablo Barceló[1,2], Floris Geerts[3], Juan Reutter[1,2], Maksimilian Ryschkov[3]
[1] Department of Computer Science, PUC, Chile
[2] Millennium Institute for Foundational Research on Data, Chile
[3] Department of Computer Science, University of Antwerp, Belgium
[pbarcelo,jreutter]@ing.puc.cl, [floris.geerts,maksimilian.ryschkov]@uantwerpen.be

Abstract

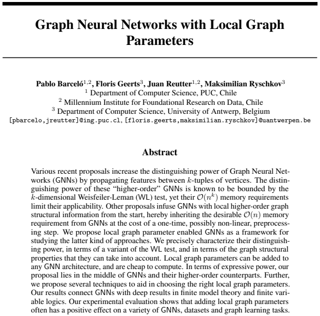
Various recent proposals increase the distinguishing power of Graph Neural Networks (GNNs) by propagating features between k-tuples of vertices. The distinguishing power of these "higher-order" GNNs is known to be bounded by the k-dimensional Weisfeiler-Leman (WL) test, yet their $\mathcal{O}(n^k)$ memory requirements limit their applicability. Other proposals infuse GNNs with local higher-order graph structural information from the start, hereby inheriting the desirable $\mathcal{O}(n)$ memory requirement from GNNs at the cost of a one-time, possibly non-linear, preprocessing step. We propose local graph parameter enabled GNNs as a framework for studying the latter kind of approaches. We precisely characterize their distinguishing power, in terms of a variant of the WL test, and in terms of the graph structural properties that they can take into account. Local graph parameters can be added to any GNN architecture, and are cheap to compute. In terms of expressive power, our proposal lies in the middle of GNNs and their higher-order counterparts. Further, we propose several techniques to aid in choosing the right local graph parameters. Our results connect GNNs with deep results in finite model theory and finite variable logics. Our experimental evaluation shows that adding local graph parameters often has a positive effect on a variety of GNNs, datasets and graph learning tasks.

Graph Homomorphism Convolution ($\mathcal{F}$-MPNNs) Barceló et al (2021)



add hom-counts here

- This architecture is more expressive than WL
- It is incomparable to 2-WL



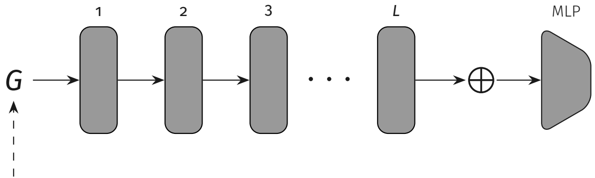
Graph Neural Networks with Local Graph Parameters

Pablo Barceló[1,2], **Floris Geerts**[3], **Juan Reutter**[1,2], **Maksimilian Ryschkov**[3]
[1] Department of Computer Science, PUC, Chile
[2] Millennium Institute for Foundational Research on Data, Chile
[3] Department of Computer Science, University of Antwerp, Belgium
[pbarcelo,jreutter]@ing.puc.cl, [floris.geerts,maksimilian.ryschkov]@uantwerpen.be

Abstract

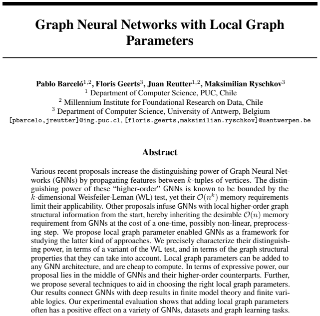
Various recent proposals increase the distinguishing power of Graph Neural Networks (GNNs) by propagating features between k-tuples of vertices. The distinguishing power of these "higher-order" GNNs is known to be bounded by the k-dimensional Weisfeiler-Leman (WL) test, yet their $\mathcal{O}(n^k)$ memory requirements limit their applicability. Other proposals infuse GNNs with local higher-order graph structural information from the start, hereby inheriting the desirable $\mathcal{O}(n)$ memory requirement from GNNs at the cost of a one-time, possibly non-linear, preprocessing step. We propose local graph parameter enabled GNNs as a framework for studying the latter kind of approaches. We precisely characterize their distinguishing power, in terms of a variant of the WL test, and in terms of the graph structural properties that they can take into account. Local graph parameters can be added to any GNN architecture, and are cheap to compute. In terms of expressive power, our proposal lies in the middle of GNNs and their higher-order counterparts. Further, we propose several techniques to aid in choosing the right local graph parameters. Our results connect GNNs with deep results in finite model theory and finite variable logics. Our experimental evaluation shows that adding local graph parameters often has a positive effect on a variety of GNNs, datasets and graph learning tasks.

Graph Homomorphism Convolution ($\mathcal{F}$-MPNNs) (Barceló et al (2021))



add hom-counts here

- This architecture is more expressive than WL
- It is incomparable to 2-WL
- Can be bounded by $\mathcal{F}$-WL (!)



Graph Neural Networks with Local Graph Parameters

Pablo Barceló[1,2], Floris Geerts[3], Juan Reutter[1,2], Maksimilian Ryschkov[3]
[1] Department of Computer Science, PUC, Chile
[2] Millennium Institute for Foundational Research on Data, Chile
[3] Department of Computer Science, University of Antwerp, Belgium
[pbarcelo,jreutter]@ing.puc.cl, [floris.geerts,maksimilian.ryschkov]@uantwerpen.be

Abstract

Various recent proposals increase the distinguishing power of Graph Neural Networks (GNNs) by propagating features between k-tuples of vertices. The distinguishing power of these "higher-order" GNNs is known to be bounded by the k-dimensional Weisfeiler-Leman (WL) test, yet their $\mathcal{O}(n^k)$ memory requirements limit their applicability. Other proposals infuse GNNs with local higher-order graph structural information from the start, hereby inheriting the desirable $\mathcal{O}(n)$ memory requirement from GNNs at the cost of a one-time, possibly non-linear, preprocessing step. We propose local graph parameter enabled GNNs as a framework for studying the latter kind of approaches. We precisely characterize their distinguishing power, in terms of a variant of the WL test, and in terms of the graph structural properties that they can take into account. Local graph parameters can be added to any GNN architecture, and are cheap to compute. In terms of expressive power, our proposal lies in the middle of GNNs and their higher-order counterparts. Further, we propose several techniques to aid in choosing the right local graph parameters. Our results connect GNNs with deep results in finite model theory and finite variable logics. Our experimental evaluation shows that adding local graph parameters often has a positive effect on a variety of GNNs, datasets and graph learning tasks.

Experimental Results

(a) Results for the ZINC dataset show that homomorphism (hom) counts of cycles improve every model. We compare the mean absolute error (MAE) of each model without any homomorphism count (baseline), against the model augmented with the hom count, and with subgraph isomorphism (iso) counts of C_3–C_{10}.

MODEL	MAE (BASE)	MAE (HOM)	MAE (ISO)
GAT	0.47±0.02	**0.22±0.01**	0.24±0.01
GCN	0.35±0.01	**0.20±0.01**	0.22±0.01
GraphSage	0.44±0.01	**0.24±0.01**	0.24±0.01
MoNet	0.25±0.01	0.19±0.01	**0.16±0.01**
GatedGCN	0.34±0.05	**0.1353±0.01**	0.1357±0.01

(b) The effect of different cycles for the GAT model over the ZINC dataset, using mean absolute error.

SET ($\mathcal{F}$)	MAE
NONE	0.47±0.02
$\{C_3\}$	0.45±0.01
$\{C_4\}$	0.34±0.02
$\{C_6\}$	0.31±0.01
$\{C_5, C_6\}$	0.28±0.01
$\{C_3, \ldots, C_6\}$	0.23±0.01
$\{C_3, \ldots, C_{10}\}$	**0.22±0.01**

Table 2: Results for the PATTERN dataset show that homomorphism counts improve all models except GatedGCN. We compare weighted accuracy of each model without any homomorphism count (baseline) against the model augmented with the counts of the set $\mathcal{F}$ that showed best performance (best $\mathcal{F}$).

MODEL + BEST $\mathcal{F}$	ACCURACY BASELINE	ACCURACY BEST
GAT $\{K_3, K_4, K_5\}$	78.83 ± 0.60	85.50 ± 0.23
GCN $\{K_3, K_4, K_5\}$	71.42 ± 1,38	82.49 ± 0.48
GraphSage $\{K_3, K_4, K_5\}$	70.78 ± 0,19	85,85 ± 0.15
MoNet $\{K_3, K_4, K_5\}$	85.90 ± 0,03	**86.63 ± 0.03**
GatedGCN $\{\emptyset\}$	86.15 ± 0.08	86.15 ± 0.08

Table 3: All models improve the Hits@50 metric over the COLLAB dataset. We compare each model without any homomorphism count (baseline) against the model augmented with the counts of the set of patterns that showed best performance (best $\mathcal{F}$).

MODEL + BEST $\mathcal{F}$	HITS@50 BASELINE	HITS@50 BEST
GAT $\{K_3\}$	50.32±0.55	52.87±0.87
GCN $\{K_3, K_4, K_5\}$	51.35±1.30	**54.60±1.01**
GraphSage $\{K_5\}$	50.33±0.68	51.39±1.23
MoNet $\{K_4\}$	49.81±1.56	51.76±1.38
GatedGCN $\{K_3\}$	51.00±2.54	51.57±0.68

Insights?

- By adding homcounts to the node labels before message passing, we get an architecture that is at least as expressive as message passing

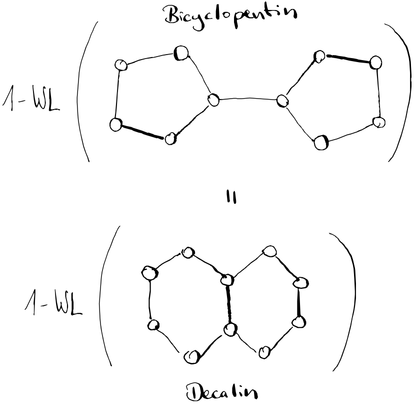
Insights?

- By adding homcounts to the node labels before message passing, we get an architecture that is at least as expressive as message passing
- Cycle counting seems to be important ;)

GNNs can Count Homomorphisms – Implicitly

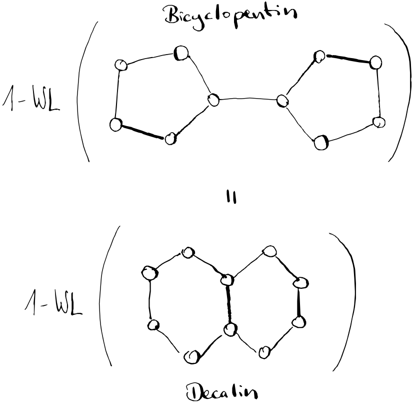
Practical problem

- 1-WL is sometimes not expressive enough



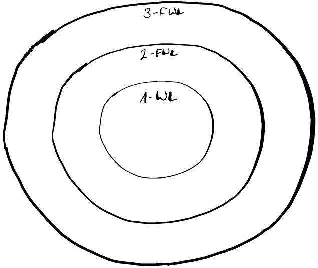
Practical problem

- 1-WL is sometimes not expressive enough
- In particular, it is insensitive to the number of cycles



Practical problem

- 1-WL is sometimes not expressive enough
- In particular, it is insensitive to the number of cycles
- 2-FWL is already impractical



Weisfeiler and Leman Go Loopy: A New Hierarchy for Graph Representational Learning




NeurIPS 2024 (oral)

Raffaele Paolino*, Sohir Maskey*, Pascal Welke, and Gitta Kutyniok

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

TU WIEN

At a glance

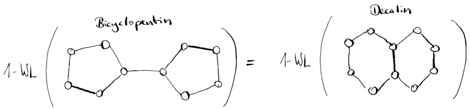
- Property prediction for small molecules is one main application area of GNNs

At a glance

- Property prediction for small molecules is one main application area of GNNs
- Number and type of cycles in molecules is important

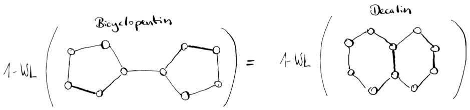
At a glance

- Property prediction for small molecules is one main application area of GNNs
- Number and type of cycles in molecules is important
- But



At a glance

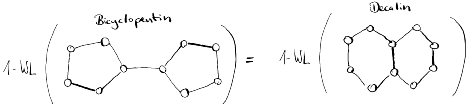
- Property prediction for small molecules is one main application area of GNNs
- Number and type of cycles in molecules is important
- But



- we propose a generalized message passing architecture

At a glance

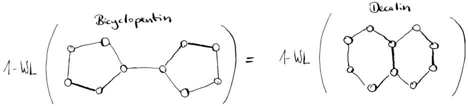
- Property prediction for small molecules is one main application area of GNNs
- Number and type of cycles in molecules is important
- But



- we propose a generalized message passing architecture
- it can distinguish graphs with different r-cycle counts

At a glance

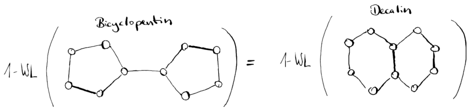
- Property prediction for small molecules is one main application area of GNNs
- Number and type of cycles in molecules is important
- But



- we propose a generalized message passing architecture
- it can distinguish graphs with different r-cycle counts
- it can homomorphism-count all r-cactus graphs (strictly more expressive than 1-WL)

At a glance

- Property prediction for small molecules is one main application area of GNNs
- Number and type of cycles in molecules is important
- But



- we propose a generalized message passing architecture
- it can distinguish graphs with different r-cycle counts
- it can homomorphism-count all r-cactus graphs (strictly more expressive than 1-WL)
- fast in practice, s.o.t.a. results

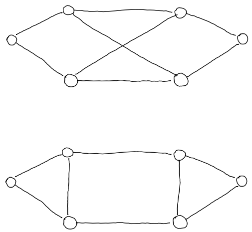
Contributions

A novel GNN architecture that is parametrized by cycle length r that

Contributions

A novel GNN architecture that is parametrized by cycle length r that

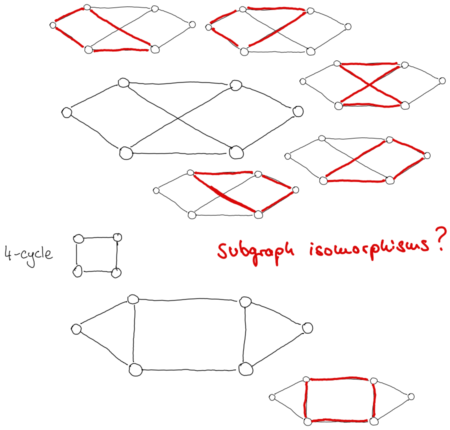
- is efficient on sparse graphs



Contributions

A novel GNN architecture that is parametrized by cycle length ***r*** that

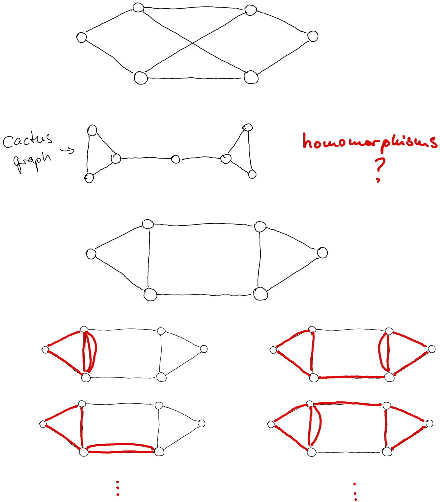
- is efficient on sparse graphs
- can subgraph count all cycles of length up to ***r***



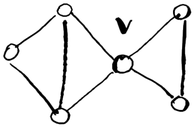
Contributions

A novel GNN architecture that is parametrized by cycle length ***r*** that

- is efficient on sparse graphs
- can subgraph count all cycles of length up to ***r***
- can homomorphism count all ***r***-cactus graphs

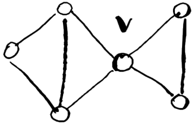


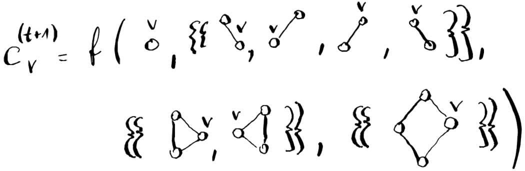
A glimpse at the implementation



- Generalized message passing over multiple sets of local "neighborhoods"

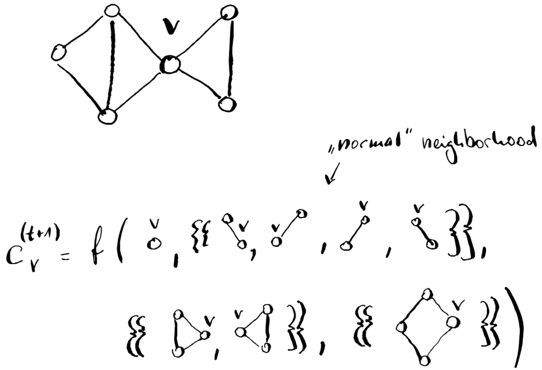
A glimpse at the implementation





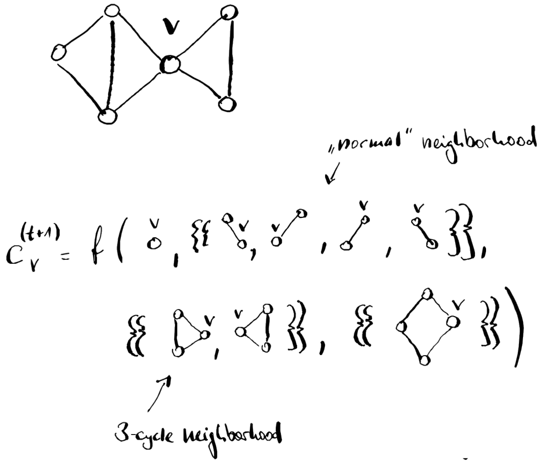
- Generalized message passing over multiple sets of local “neighborhoods”

A glimpse at the implementation



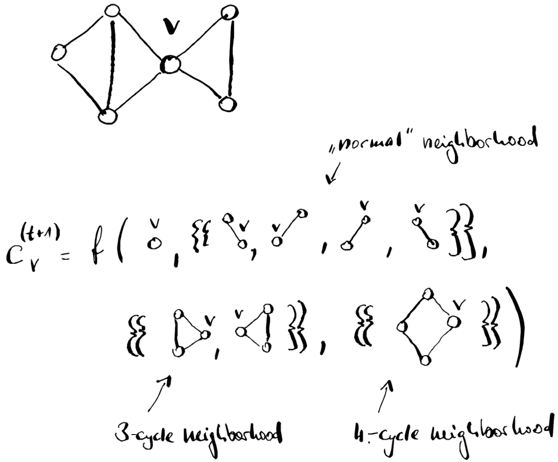
- Generalized message passing over multiple sets of local "neighborhoods"

A glimpse at the implementation



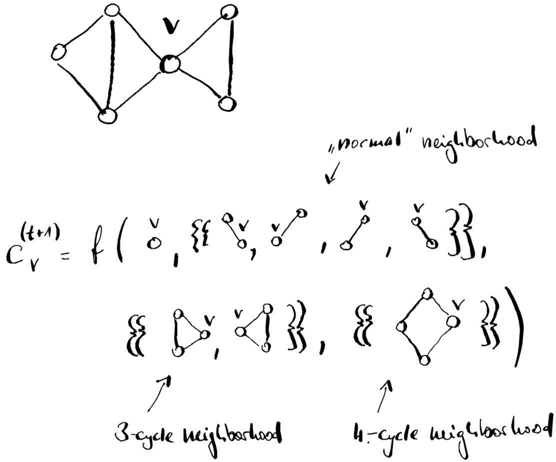
- Generalized message passing over multiple sets of local “neighborhoods”

A glimpse at the implementation



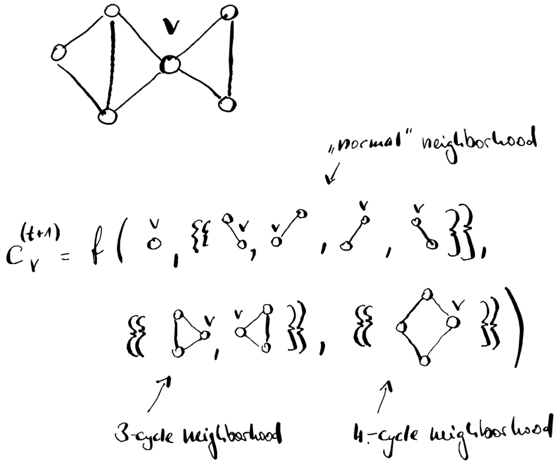
- Generalized message passing over multiple sets of local “neighborhoods”

A glimpse at the implementation



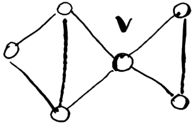
- Generalized message passing over multiple sets of local "neighborhoods"
- Cycles can be enumerated quickly on many sparse graphs
  Horváth et al (2004)

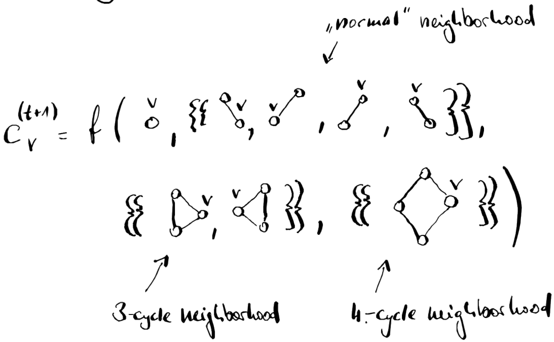
A glimpse at the implementation



- Generalized message passing over multiple sets of local "neighborhoods"
- Cycles can be enumerated quickly on many sparse graphs
  Horváth et al (2004)
- Cycle representations can be computed with GINs

A glimpse at the implementation





- Generalized message passing over multiple sets of local "neighborhoods"
- Cycles can be enumerated quickly on many sparse graphs
  Horváth et al (2004)
- Cycle representations can be computed with GINs

A complete representation for cycles :

Empirical results

Table 4: Normalized test MAE (↓) on graph regression, QM9 dataset. Top three models as $\boxed{1^{\text{st}}}$, $\boxed{2^{\text{nd}}}$, $\underline{3^{\text{rd}}}$.

Model	μ	α	$\varepsilon_{\text{homo}}$
1-GNN	0.493	0.78	0.00321
1-2-3-GNN	0.476	0.27	0.00337
DTNN	$\boxed{0.244}$	0.95	0.00388
Deep LRP	0.364	0.298	$\underline{0.00254}$
PPGN	$\boxed{0.231}$	0.382	0.00276
NestedGNN	0.428	0.290	0.00265
I2-GNN	0.428	$\underline{0.230}$	0.00261
DRFWL GNN	$\underline{0.346}$	$\boxed{0.222}$	$\boxed{0.00226}$
5-ℓGIN	0.350 ± 0.011	$\boxed{0.217}$ ± 0.025	$\boxed{0.00205}$ ± 0.00005

Empirical results

Table 4: Normalized test MAE (↓) on graph regression, QM9 dataset. Top three models as 1[st], 2[nd], 3[rd].

Model	μ	α	$\varepsilon_{\text{homo}}$
1-GNN	0.493	0.78	0.00321
1-2-3-GNN	0.476	0.27	0.00337
DTNN	0.244	0.95	0.00388
Deep LRP	0.364	0.298	0.00254
PPGN	0.231	0.382	0.00276
NestedGNN	0.428	0.290	0.00265
I2-GNN	0.428	0.230	0.00261
DRFWL GNN	0.346	0.222	0.00226
5-ℓGIN	0.350 ± 0.011	0.217 ± 0.025	0.00205 ± 0.00005

Table 3: Test MAE (↓) on graph regression, ZINC dataset. Top three models as 1[st], 2[nd], 3[rd].

Model	ZINC12K	ZINC250K
GIN	0.163 ± 0.004	0.088 ± 0.002
GCN	0.321 ± 0.009	-
GAT	0.384 ± 0.007	-
GSN	0.115 ± 0.012	-
CIN	0.079 ± 0.006	0.022 ± 0.002
NestedGNN	0.111 ± 0.003	0.029 ± 0.001
SUN	0.083 ± 0.003	-
GNNAK+	0.080 ± 0.001	-
I2-GNN	0.083 ± 0.001	0.023 ± 0.001
DRFWL GNN	0.077 ± 0.002	0.025 ± 0.003
SignNet	0.084 ± 0.004	0.024 ± 0.003
HIMP	0.151 ± 0.006	0.036 ± 0.002
PathNN	0.090 ± 0.004	-
5-ℓGIN	0.072 ± 0.002	0.022 ± 0.001

Open questions

We have seen different hierarchies of expressiveness

Open questions

We have seen different hierarchies of expressiveness

- increasing the size of $\mathcal{F}$ in (NT and Maehara (2020))

Open questions

We have seen different hierarchies of expressiveness

- increasing the size of $\mathcal{F}$ in (NT and Maehara (2020))
- (Barceló et al (2021)) s $\mathcal{F}$-WL hierarchy

Open questions

We have seen different hierarchies of expressiveness

- increasing the size of $\mathcal{F}$ in (NT and Maehara (2020))
- (Barceló et al (2021)) s $\mathcal{F}$-WL hierarchy
- the $\boldsymbol{r}$-loopy WL test of (NeurIPS 2024)

Open questions

We have seen different hierarchies of expressiveness

- increasing the size of $\mathcal{F}$ in NT and Maehara (2020)
- Barceló et al (2021) s $\mathcal{F}$-WL hierarchy
- the r-loopy WL test of NeurIPS 2024

How are they connected?

Open questions

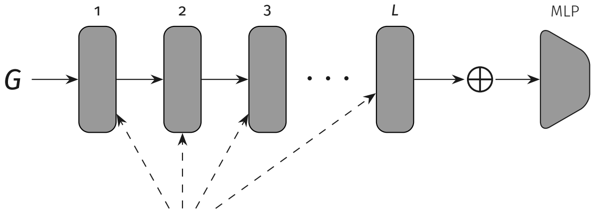
We have seen different hierarchies of expressiveness

- increasing the size of $\mathcal{F}$ in (NT and Maehara (2020))
- (Barceló et al (2021)) s $\mathcal{F}$-WL hierarchy
- the $\boldsymbol{r}$-loopy WL test of (NeurIPS 2024)

How are they connected?

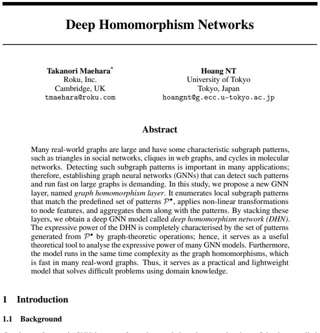
Can we collect most of our results in one architecture?

Deep Homomorphism Networks



change the layers

- Message passing can be generalized to homomorphism counting



Deep Homomorphism Networks

Takanori Maehara[*]
Roku, Inc.
Cambridge, UK
tmaehara@roku.com

Hoang NT
University of Tokyo
Tokyo, Japan
hoangnt@g.ecc.u-tokyo.ac.jp

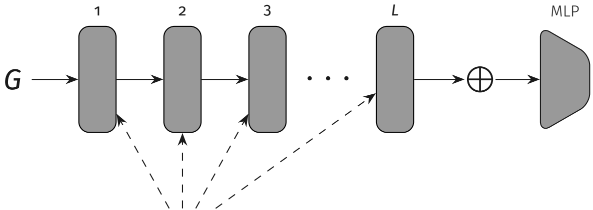
Abstract

Many real-world graphs are large and have some characteristic subgraph patterns, such as triangles in social networks, cliques in web graphs, and cycles in molecular networks. Detecting such subgraph patterns is important in many applications; therefore, establishing graph neural networks (GNNs) that can detect such patterns and run fast on large graphs is demanding. In this study, we propose a new GNN layer, named *graph homomorphism layer*. It enumerates local subgraph patterns that match the predefined set of patterns $\mathcal{P}^\bullet$, applies non-linear transformations to node features, and aggregates them along with the patterns. By stacking these layers, we obtain a deep GNN model called *deep homomorphism network (DHN)*. The expressive power of the DHN is completely characterised by the set of patterns generated from $\mathcal{P}^\bullet$ by graph-theoretic operations; hence, it serves as a useful theoretical tool to analyse the expressive power of many GNN models. Furthermore, the model runs in the same time complexity as the graph homomorphisms, which is fast in many real-word graphs. Thus, it serves as a practical and lightweight model that solves difficult problems using domain knowledge.

1 Introduction

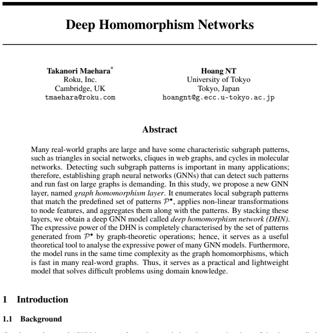
1.1 Background

Deep Homomorphism Networks



change the layers

- Message passing can be generalized to homomorphism counting
- We have to use a node-weighted variant of homomorphisms, though



Deep Homomorphism Networks

Takanori Maehara[*]
Roku, Inc.
Cambridge, UK
tmaehara@roku.com

Hoang NT
University of Tokyo
Tokyo, Japan
hoangnt@g.ecc.u-tokyo.ac.jp

Abstract

Many real-world graphs are large and have some characteristic subgraph patterns, such as triangles in social networks, cliques in web graphs, and cycles in molecular networks. Detecting such subgraph patterns is important in many applications; therefore, establishing graph neural networks (GNNs) that can detect such patterns and run fast on large graphs is demanding. In this study, we propose a new GNN layer, named *graph homomorphism layer*. It enumerates local subgraph patterns that match the predefined set of patterns $\mathcal{P}^\bullet$, applies non-linear transformations to node features, and aggregates them along with the patterns. By stacking these layers, we obtain a deep GNN model called *deep homomorphism network (DHN)*. The expressive power of the DHN is completely characterised by the set of patterns generated from $\mathcal{P}^\bullet$ by graph-theoretic operations; hence, it serves as a useful theoretical tool to analyse the expressive power of many GNN models. Furthermore, the model runs in the same time complexity as the graph homomorphisms, which is fast in many real-word graphs. Thus, it serves as a practical and lightweight model that solves difficult problems using domain knowledge.

1 Introduction

1.1 Background

Deep Homomorphism Network Architecture

- Homomorphism counts can be weighted by the node weights

$$\hom\left((F^\circ, \mu), (G^\circ, x)\right) = \sum_{\pi \in \mathrm{Hom}(F^\circ, G^\circ)} \prod_{p \in V(F^\circ)} \mu_p\left(x_{\pi(p)}\right)$$

elementwise

Deep Homomorphism Network Architecture

- Homomorphism counts can be weighted by the node weights
- Node weights can be computed by learnable functions

$$\hom\left((F^\circ, \mu), (G^\circ, x)\right) = \sum_{\pi \in \mathrm{Hom}(F^\circ, G^\circ)} \prod_{p \in V(F^\circ)} \mu_p\left(x_{\pi(p)}\right)$$

elementwise

Deep Homomorphism Network Architecture

- Homomorphism counts can be weighted by the node weights
- Node weights can be computed by learnable functions
- Suitable pattern sets $\mathcal{P}$ allow to obtain architectures as powerful as our previous examples

$$\hom((F^\circ, \mu), (G^\circ, x)) = \sum_{\pi \in \mathrm{Hom}(F^\circ, G^\circ)} \prod_{p \in V(F^\circ)} \mu_p(x_{\pi(p)})$$

elementwise

$$\mathrm{GHL}_{\mathcal{P}^\circ}\left((G^\circ, x); \rho, \{\mu_{P^\circ} : P^\circ \in \mathcal{P}^\circ\}\right) :=$$

$$\rho\left(\hom((P^\circ, \mu_{P^\circ}), (G^\circ, x)) : P^\circ \in \mathcal{P}^\circ\right)$$

Concluding Remarks

Concluding Remarks

- Homomorphism-based methods work well in theory and practice
  ICML 2023 NeurIPS 2024 ECML/PKDD 2018

Concluding Remarks

- Homomorphism-based methods work well in theory and practice
  ICML 2023 NeurIPS 2024 ECML/PKDD 2018
- Randomization yields expressive graph representations
  ICML 2023 KDD 2020 PhD thesis 2019

Concluding Remarks

- Homomorphism-based methods work well in theory and practice
  ICML 2023 NeurIPS 2024 ECML/PKDD 2018
- Randomization yields expressive graph representations
  ICML 2023 KDD 2020 PhD thesis 2019
- There is much more...

Concluding Remarks

- Homomorphism-based methods work well in theory and practice
  ICML 2023 NeurIPS 2024 ECML/PKDD 2018
- Randomization yields expressive graph representations
  ICML 2023 KDD 2020 PhD thesis 2019
- There is much more...
  - Generalization bounds of GNNs using homomorphism counts Li et al (2024)

Concluding Remarks

- Homomorphism-based methods work well in theory and practice
  ICML 2023 NeurIPS 2024 ECML/PKDD 2018
- Randomization yields expressive graph representations
  ICML 2023 KDD 2020 PhD thesis 2019
- There is much more...
  - Generalization bounds of GNNs using homomorphism counts Li et al (2024)
  - Intricate results linking homomorphism counting and the k-WL test Neuen (2024) Lanzinger and Barceló (2024)

Concluding Remarks

- Homomorphism-based methods work well in theory and practice
  ICML 2023 NeurIPS 2024 ECML/PKDD 2018
- Randomization yields expressive graph representations
  ICML 2023 KDD 2020 PhD thesis 2019
- There is much more...
  - Generalization bounds of GNNs using homomorphism counts Li et al (2024)
  - Intricate results linking homomorphism counting and the $\boldsymbol{k}$-WL test Neuen (2024) Lanzinger and Barceló (2024)
  - Homomorphism bases (aka spasms) of patterns allow to compute and learn(!) very powerful graph invariants Jin et al (2024) Dell et al (2018) Curticapean et al (2017)

References

Pablo Barceló, Floris Geerts, Juan L Reutter, Maksimilian Ryschkov (2021) Graph neural networks with local graph parameters. In: Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N Dauphin, Percy Liang, Jennifer Wortman Vaughan (eds) Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, pp 25,280–25,293, URL https://proceedings.neurips.cc/paper/2021/hash/d4d8d1ac7e00e9105775a6b660dd3cbb-Abstract.html

Paul Beaujean, Florian Sikora, Florian Yger (2021) Graph homomorphism features: Why not sample? In: Machine Learning and Principles and Practice of Knowledge Discovery in Databases - International Workshops of ECML PKDD 2021, Virtual Event, September 13-17, 2021, Proceedings, Part I, Springer, Communications in Computer and Information Science, vol 1524, pp 216–222, DOI 10.1007/978-3-030-93736-2_17, URL https://doi.org/10.1007/978-3-030-93736-2_17

Andrei Dragos Brasoveanu, Fabian Jogl, Pascal Welke, Maximilian Thiessen (2023) Extending graph neural networks with global features. In: Learning on Graphs Conference (LoG), URL https://openreview.net/forum?id=aisVQy6R2k

Radu Curticapean, Holger Dell, Dániel Marx (2017) Homomorphisms are a good basis for counting small subgraphs. In: Hamed Hatami, Pierre McKenzie, Valerie King (eds) Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017, ACM, pp 210–223, DOI 10.1145/3055399.3055502, URL https://doi.org/10.1145/3055399.3055502

Holger Dell, Martin Grohe, Gaurav Rattan (2018) Lovász meets weisfeiler and leman. In: Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, Donald Sannella (eds) 45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, LIPIcs, vol 107, pp 40:1–40:14, DOI 10.4230/LIPICS.ICALP.2018.40, URL https://doi.org/10.4230/LIPIcs.ICALP.2018.40

Vijay Prakash Dwivedi, Ladislav Rampášek, Mikhail Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, Dominique Beaini (2022) Long range graph benchmark. In: Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track, URL https://openreview.net/forum?id=in7XC5RcjEn

References

Tamás Horváth, Thomas Gärtner, Stefan Wrobel (2004) Cyclic pattern kernels for predictive graph mining. In: Won Kim, Ron Kohavi, Johannes Gehrke, William DuMouchel (eds) Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004, ACM, pp 158–167, DOI 10.1145/1014052.1014072

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, Jure Leskovec (2020) Open graph benchmark: Datasets for machine learning on graphs. In: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, URL https://proceedings.neurips.cc/paper/2020/hash/fb60d411a5c5b72b2e7d3527cfc84fd0-Abstract.html

Emily Jin, Michael M Bronstein, Ismail Ilkan Ceylan, Matthias Lanzinger (2024) Homomorphism counts for graph neural networks: All about that basis. In: Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024, OpenReview.net, URL https://openreview.net/forum?id=zRrzSLwNHQ

Matthias Lanzinger, Pablo Barceló (2024) On the power of the weisfeiler-leman test for graph motif parameters. In: The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024, OpenReview.net, URL https://openreview.net/forum?id=FddFxi08J3

Shouheng Li, Floris Geerts, Dongwoo Kim, Qing Wang (2024) Towards bridging generalization and expressivity of graph neural networks. URL https://arxiv.org/abs/2410.10051, 2410.10051

Tobias Mette (2023) Hops for homomorphism count estimation. BSc Thesis, University of Bonn

Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, Marion Neumann (2020) Tudataset: A collection of benchmark datasets for learning with graphs. CoRR abs/2007.08663, URL https://arxiv.org/abs/2007.08663, 2007.08663

References

Daniel Neuen (2024) Homomorphism-distinguishing closedness for graphs of bounded tree-width. In: Olaf Beyersdorff, Mamadou Moustapha Kanté, Orna Kupferman, Daniel Lokshtanov (eds) 41st International Symposium on Theoretical Aspects of Computer Science, STACS 2024, March 12-14, 2024, Clermont-Ferrand, France, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, LIPIcs, vol 289, pp 53:1–53:12, DOI 10.4230/LIPICS.STACS.2024.53, URL https://doi.org/10.4230/LIPIcs.STACS.2024.53

Hoang NT, Takanori Maehara (2020) Graph homomorphism convolution. In: Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, PMLR, Proceedings of Machine Learning Research, vol 119, pp 7306–7316, URL http://proceedings.mlr.press/v119/nguyen20c.html

Raffaele Paolino*, Sohir Maskey*, Pascal Welke, Gitta Kutyniok (2024) Weisfeiler and leman go loopy: A new hierarchy for graph representational learning. 2403.13749

Till Hendrik Schulz, Tamás Horváth, Pascal Welke, Stefan Wrobel (2018) Mining tree patterns with partially injective homomorphisms. In: Michele Berlingerio, Francesco Bonchi, Thomas Gärtner, Neil Hurley, Georgiana Ifrim (eds) European Conference on Machine Learning and Knowledge Discovery in Databases (ECMLPKDD), Springer, Lecture Notes in Computer Science, vol 11052, pp 585–601, DOI 10.1007/978-3-030-10928-8_35, URL https://doi.org/10.1007/978-3-030-10928-8_35

Pascal Welke (2019) Efficient frequent subtree mining beyond forests. Dissertations in Artificial Intelligence 348, URL https://hdl.handle.net/20.500.11811/7893

Pascal Welke, Florian Seiffarth, Michael Kamp, Stefan Wrobel (2020) HOPS: probabilistic subtree mining for small and large graphs. In: Rajesh Gupta, Yan Liu, Jiliang Tang, B Aditya Prakash (eds) SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), ACM, pp 1275–1284, DOI 10.1145/3394486.3403180, URL https://doi.org/10.1145/3394486.3403180

Pascal Welke*, Maximilian Thiessen*, Fabian Jogl, Thomas Gärtner (2023) Expectation-complete graph representations with homomorphisms. In: International Conference on Machine Learning (ICML), URL https://proceedings.mlr.press/v202/welke23a.html, 2306.05838