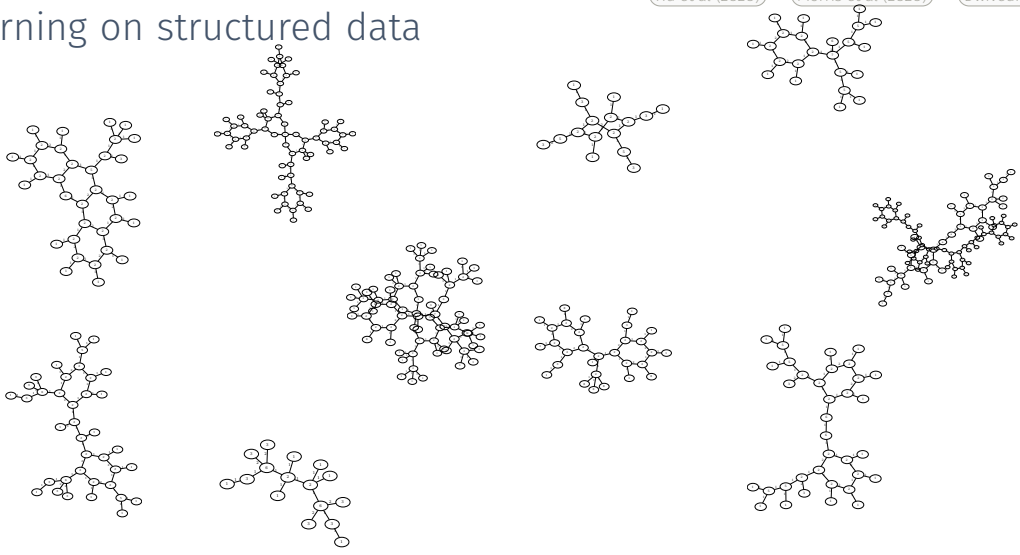
A Tale of Graph Representation Learning

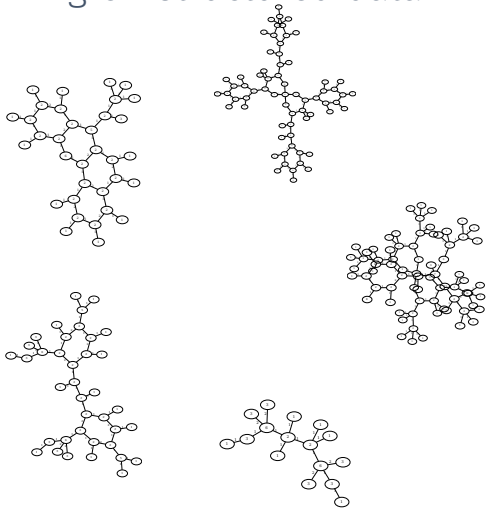
Pascal Welke

LOGML Summer School on 13. July 2026

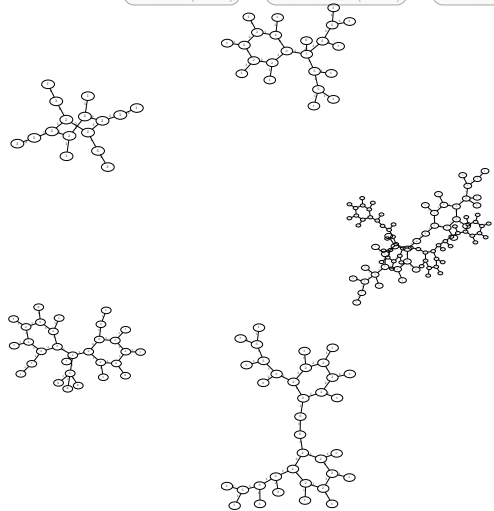
Learning on structured data



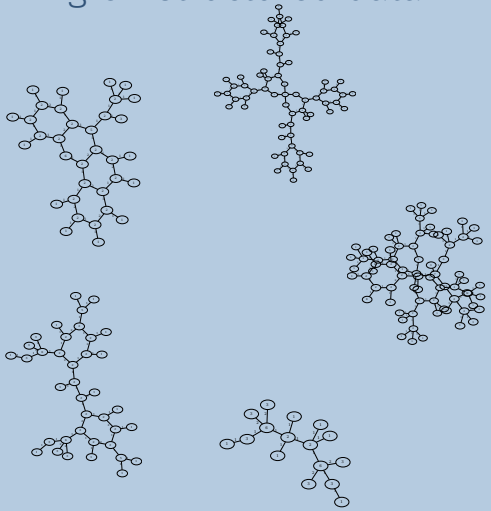
Learning on structured data



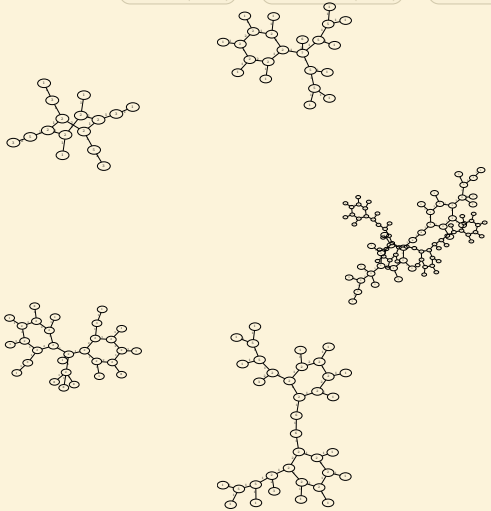
Hu et al (2020) Morris et al (2020) Dwivedi et al (2022)



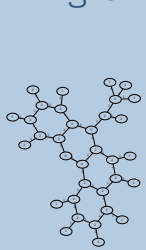
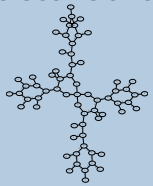
Learning on structured data

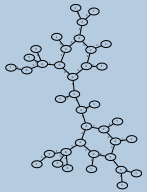


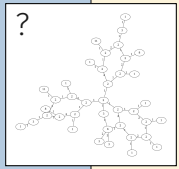
Hu et al (2020) Morris et al (2020) Dwivedi et al (2022)



Learning on structured data

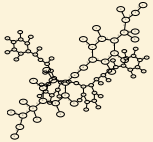
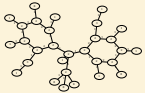
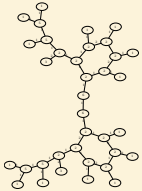







Hu et al (2020) Morris et al (2020) Dwivedi et al (2022)

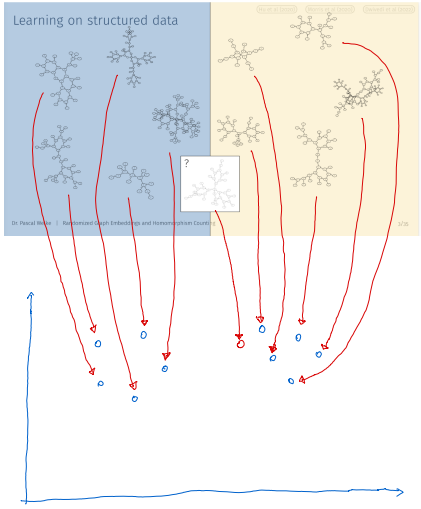


Graph Representation Learning

The goal

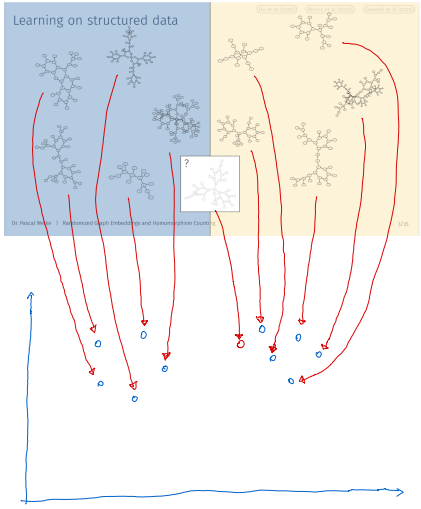
Vectorial graph representations that



The goal

Vectorial graph representations that

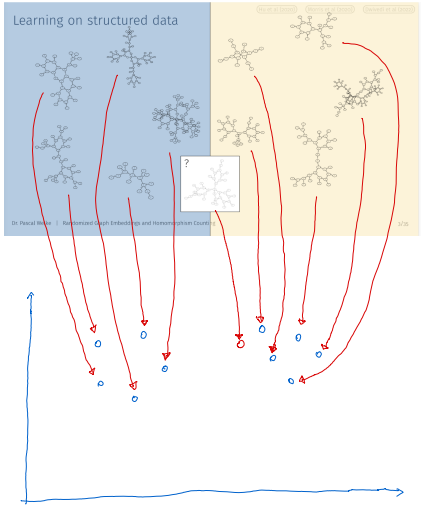
- help you to solve a learning task



The goal

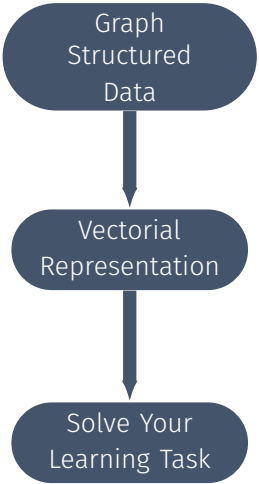
Vectorial graph representations that

- help you to solve a learning task
- yield semantically and structurally meaningful distances
- are interpretable
- are adaptable to given data

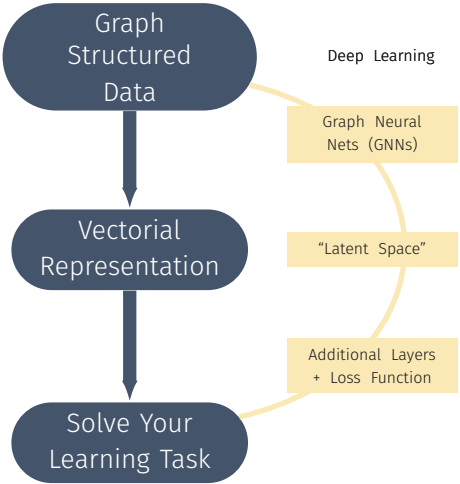


The three pillars of graph learning

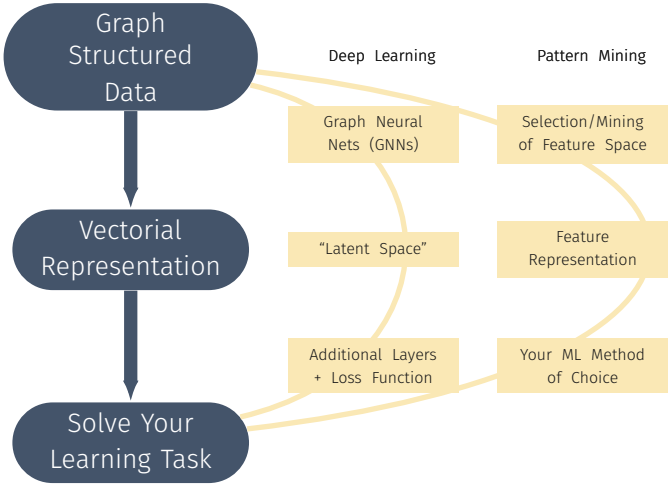
Graph representation learning is older than you may think



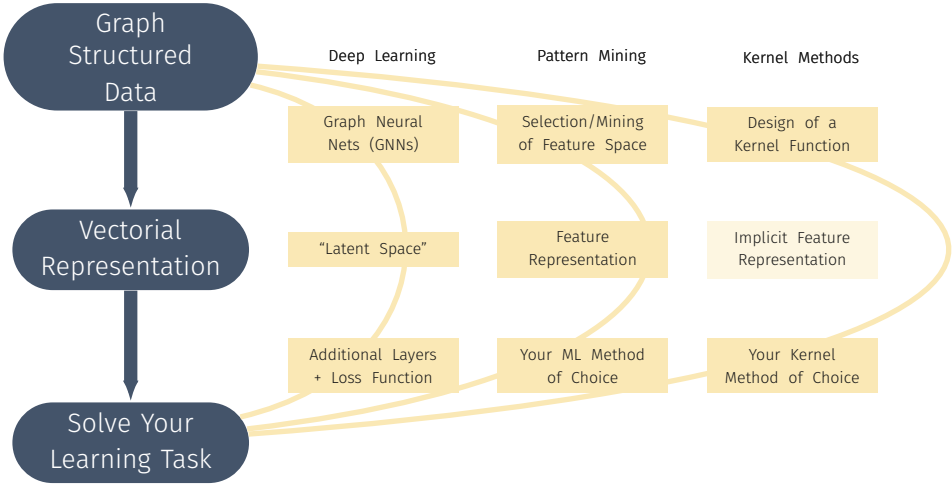
Graph representation learning is older than you may think



Graph representation learning is older than you may think

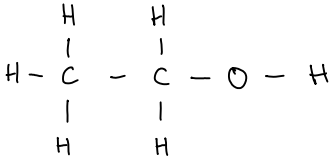


Graph representation learning is older than you may think

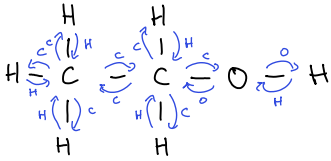


Message Passing in a nutshell

Deep Learning  Pattern Mining

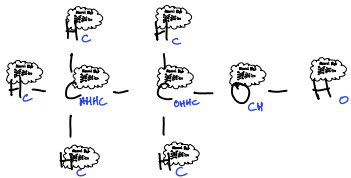


Message Passing in a nutshell



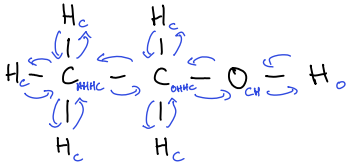
- Nodes can initially tweet their types

Message Passing in a nutshell



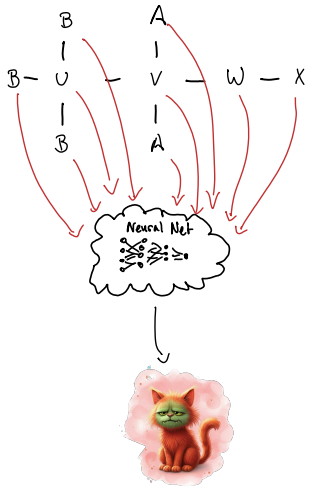
- Nodes can initially tweet their types
- ...then learn how to react on that

Message Passing in a nutshell



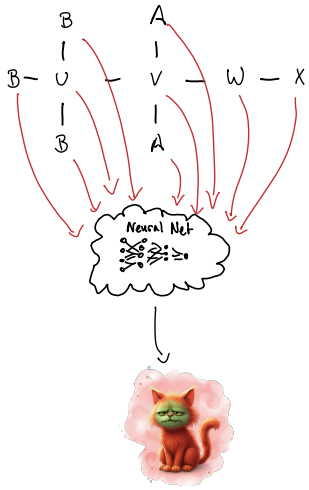
- Nodes can initially tweet their types
- ...then learn how to react on that
- We can repeat this for a few rounds

Message Passing in a nutshell



- Nodes can initially tweet their types
- ...then learn how to react on that
- We can repeat this for a few rounds
- ...and finally use a neural network to learn to interpret the result

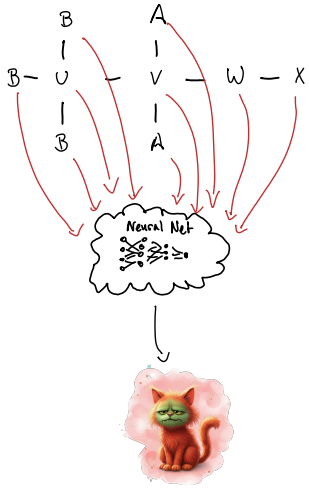
Message Passing in a nutshell



- Nodes can initially tweet their types
- ...then learn how to react on that
- We can repeat this for a few rounds
- ...and finally use a neural network to learn to interpret the result

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

Message Passing in a nutshell



- Nodes can initially tweet their types
- ...then learn how to react on that
- We can repeat this for a few rounds
- ...and finally use a neural network to learn to interpret the result

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

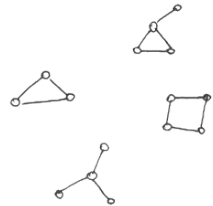
$$r(G) = \mathrm{agg}(\{\{r_l(v) | v \in V(G)\}\})$$

$$p(G) = \mathrm{f}(r(G))$$

Pattern Mining in a nutshell

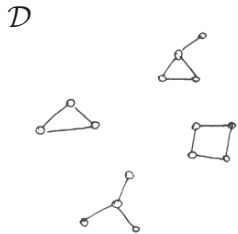
- We would like to describe a graph dataset $\mathcal{D}$

$\mathcal{D}$



Pattern Mining in a nutshell

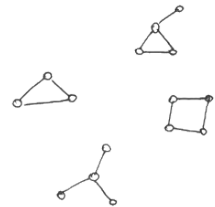
- We would like to describe a graph dataset $\mathcal{D}$
- We pick a set of patterns in advance and search for their appearance in the graphs of the database



Pattern Mining in a nutshell

- We would like to describe a graph dataset $\mathcal{D}$
- We pick a set of patterns in advance and search for their appearance in the graphs of the database
- These patterns can be almost anything you can properly define (but it is usually computationally hard to find all of them)

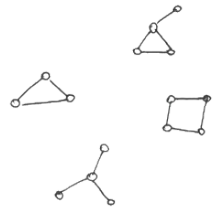
$\mathcal{D}$



Frequent Subgraph Mining

- *Frequent subgraphs* are one reasonable choice (Deshpande et al (2005)) to define similarities in a domain of graphs

$\mathcal{D}$

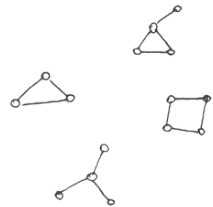


Frequent Subgraph Mining

- *Frequent subgraphs* are one reasonable choice (Deshpande et al (2005)) to define similarities in a domain of graphs

Frequent Connected Subgraph Mining (FCSM)

$\mathcal{D}$



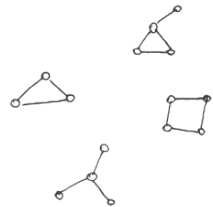
Frequent Subgraph Mining

- *Frequent subgraphs* are one reasonable choice (Deshpande et al (2005)) to define similarities in a domain of graphs

Frequent Connected Subgraph Mining (FCSM)

Given a dataset of graphs $\mathcal{D}$ and an integer threshold $t \leq |\mathcal{D}|$

$\mathcal{D}$



Frequent Subgraph Mining

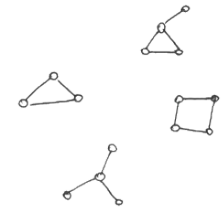
- *Frequent subgraphs* are one reasonable choice (Deshpande et al (2005)) to define similarities in a domain of graphs

Frequent Connected Subgraph Mining (FCSM)

Given a dataset of graphs $\mathcal{D}$ and an integer threshold $t \leq |\mathcal{D}|$

List all connected graphs $P \in \mathcal{P}$ that are subgraph isomorphic to at least t graphs in $\mathcal{D}$ (and where they appear).

$\mathcal{D}$



2-frequent subgraphs of $\mathcal{D}$

How is this useful for learning?

We obtain two things:

How is this useful for learning?

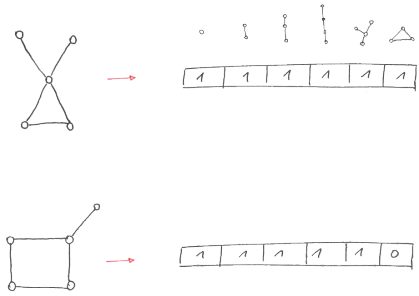
We obtain two things:

- a set of features (e.g all frequent subgraphs)
  - dataset dependent
  - can sometimes help to explore the dataset

How is this useful for learning?

We obtain two things:

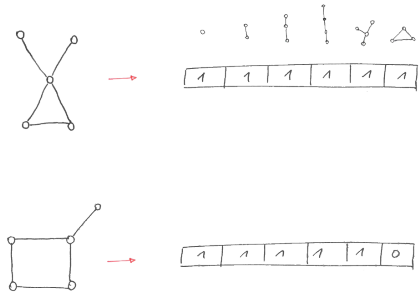
- a set of features (e.g all frequent subgraphs)
  - dataset dependent
  - can sometimes help to explore the dataset
- an interpretable representation of each graph (node, ...) in terms of these features
  - which graph contains which pattern



How is this useful for learning?

We obtain two things:

- a set of features (e.g all frequent subgraphs)
  - dataset dependent
  - can sometimes help to explore the dataset
- an interpretable representation of each graph (node, ...) in terms of these features
  - which graph contains which pattern



Use with “classical” ML techniques to build learners, compare different ways of describing your graphs, ...

Graph Kernels in a nutshell

Definition

A *graph kernel* on a graph class $\mathcal{G}$ is a function $k : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ such that

$$k(G_1, G_2) = \langle \Phi(G_1), \Phi(G_2) \rangle$$

for all $G_1, G_2 \in \mathcal{G}$, where $\Phi : \mathcal{G} \to \mathbb{F}$ maps graphs to some (possibly infinite dimensional) real vector space $\mathbb{F}$ equipped with a dot product.

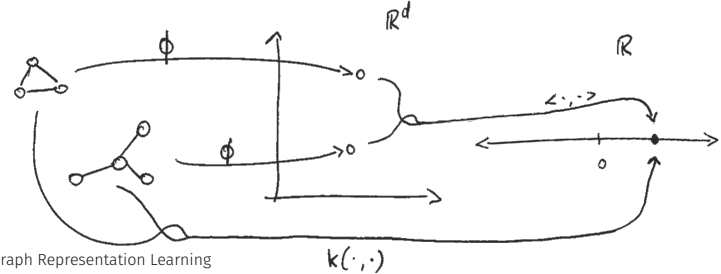
Graph Kernels in a nutshell

Definition

A *graph kernel* on a graph class $\mathcal{G}$ is a function $k : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ such that

$$k(G_1, G_2) = \langle \Phi(G_1), \Phi(G_2) \rangle$$

for all $G_1, G_2 \in \mathcal{G}$, where $\Phi : \mathcal{G} \to \mathbb{F}$ maps graphs to some (possibly infinite dimensional) real vector space $\mathbb{F}$ equipped with a dot product.



Graph Kernels: Properties

Given a graph kernel $k : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ we can define for all $G_1, G_2 \in \mathcal{G}$

- a *norm* $||G_1|| := \sqrt{k(G_1, G_1)}$

Graph Kernels: Properties

Given a graph kernel $k : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ we can define for all $G_1, G_2 \in \mathcal{G}$

- a *norm* $||G_1|| := \sqrt{k(G_1, G_1)}$
- a *distance* $d_k(G_1, G_2) := \sqrt{k(G_1, G_1) + k(G_2, G_2) - 2k(G_1, G_2)}$

Graph Kernels: Properties

Given a graph kernel $k : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ we can define for all $G_1, G_2 \in \mathcal{G}$

- a *norm* $||G_1|| := \sqrt{k(G_1, G_1)}$
- a *distance* $d_k(G_1, G_2) := \sqrt{k(G_1, G_1) + k(G_2, G_2) - 2k(G_1, G_2)}$
  $= \sqrt{\langle \Phi(G_1) - \Phi(G_2), \Phi(G_1) - \Phi(G_2) \rangle} = ||\Phi(G_1) - \Phi(G_2)||$

Graph Kernels: Properties

Given a graph kernel $k : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ we can define for all $G_1, G_2 \in \mathcal{G}$

- a *norm* $||G_1|| := \sqrt{k(G_1, G_1)}$
- a *distance* $d_k(G_1, G_2) := \sqrt{k(G_1, G_1) + k(G_2, G_2) - 2k(G_1, G_2)}$
  $= \sqrt{\langle \Phi(G_1) - \Phi(G_2), \Phi(G_1) - \Phi(G_2) \rangle} = ||\Phi(G_1) - \Phi(G_2)||$
- ...and hence work with graphs (almost) as if they were vectors in a vector space over the real numbers

Graph Kernels: Properties

Given a graph kernel $k : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ we can define for all $G_1, G_2 \in \mathcal{G}$

- a *norm* $||G_1|| := \sqrt{k(G_1, G_1)}$
- a *distance* $d_k(G_1, G_2) := \sqrt{k(G_1, G_1) + k(G_2, G_2) - 2k(G_1, G_2)}$
  $= \sqrt{\langle \Phi(G_1) - \Phi(G_2), \Phi(G_1) - \Phi(G_2) \rangle} = ||\Phi(G_1) - \Phi(G_2)||$
- ...and hence work with graphs (almost) as if they were vectors in a vector space over the real numbers
- ...even if we don't know how to compute the representation $\Phi(G)$

Benefits of (Graph) Kernels

- Many machine learning algorithms can be “kernelized”

Benefits of (Graph) Kernels

- Many machine learning algorithms can be “kernelized”
- That is, optimization problems can be rewritten such that they *never* use (graph) representations *directly*, but *only within dot products*

Benefits of (Graph) Kernels

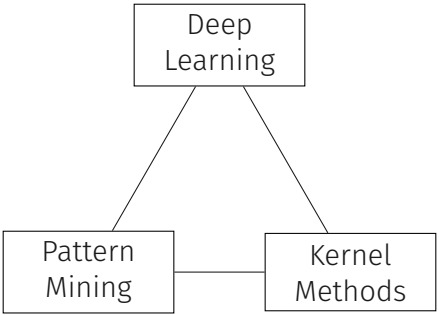
- Many machine learning algorithms can be “kernelized”
- That is, optimization problems can be rewritten such that they *never* use (graph) representations *directly*, but *only within dot products*
- Examples:
  - kernel perceptron,
  - support-vector machines (SVM),
  - Gaussian processes (GP),
  - principal components analysis (PCA),
  - canonical correlation analysis,
  - ridge regression,
  - spectral clustering, or
  - linear adaptive filters

Benefits of (Graph) Kernels

- Many machine learning algorithms can be “kernelized”
- That is, optimization problems can be rewritten such that they *never* use (graph) representations *directly*, but *only within dot products*
- Examples:
  - kernel perceptron,
  - support-vector machines (SVM),
  - Gaussian processes (GP),
  - principal components analysis (PCA),
  - canonical correlation analysis,
  - ridge regression,
  - spectral clustering, or
  - linear adaptive filters
- We can directly apply all these methods, once we have a graph kernel

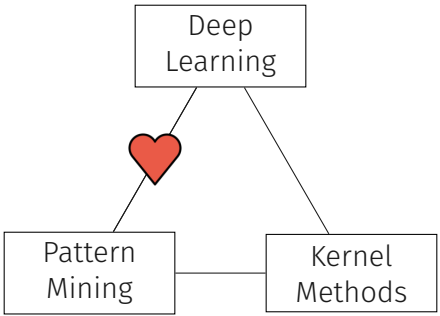
Fun with Combinations

Combinations of these techniques yield progress in all fields



These three approaches are not exclusive.

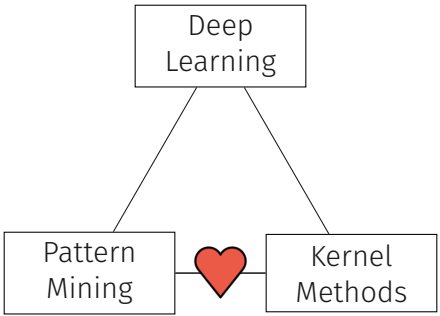
Combinations of these techniques yield progress in all fields



These three approaches are not exclusive.

- Many (maybe most) higher order GNNs are designed combining Deep Learning and Pattern Mining techniques

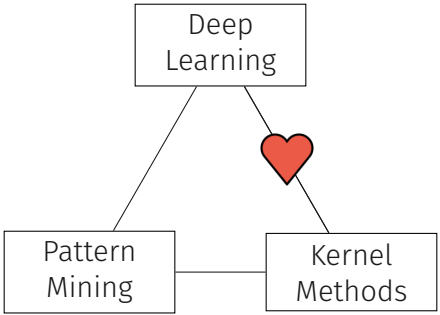
Combinations of these techniques yield progress in all fields



These three approaches are not exclusive.

- Many (maybe most) higher order GNNs are designed combining Deep Learning and Pattern Mining techniques
- Convolutional graph kernels use common patterns in graphs to define similarities

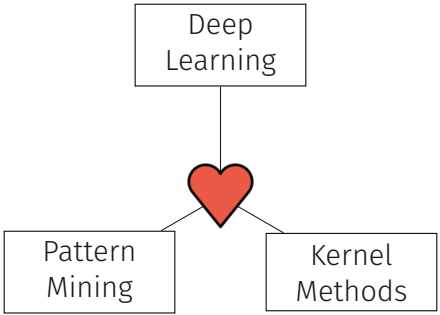
Combinations of these techniques yield progress in all fields



These three approaches are not exclusive.

- Many (maybe most) higher order GNNs are designed combining Deep Learning and Pattern Mining techniques
- Convolutional graph kernels use common patterns in graphs to define similarities
- GNTK, Kernels based on learned representations, analysis of DL approaches with kernel machinery

Combinations of these techniques yield progress in all fields



These three approaches are not exclusive.

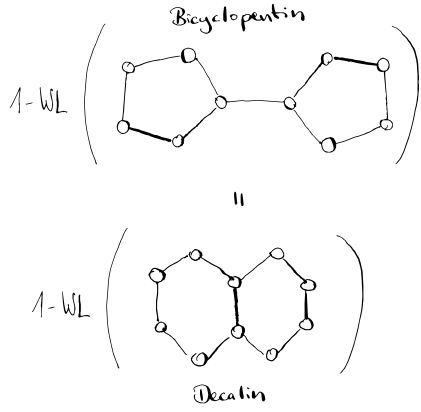
- Many (maybe most) higher order GNNs are designed combining Deep Learning and Pattern Mining techniques
- Convolutional graph kernels use common patterns in graphs to define similarities
- GNTK, Kernels based on learned representations, analysis of DL approaches with kernel machinery

Deep Learning

Pattern Mining

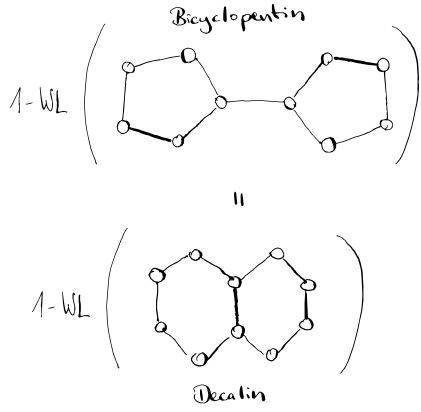
Practical problem

- Message passing/1-WL is sometimes not expressive enough



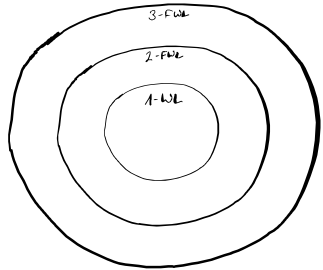
Practical problem

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



Practical problem

- Message passing/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

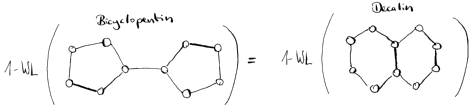
- Property prediction for small molecules is one main application area of GNNs

At a glance

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

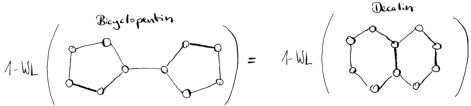
At a glance

- 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

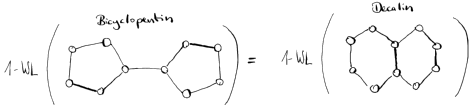
- Property prediction for small molecules is one main application area of GNNs
- Number and type of cycles in molecules is important
- But



- we propose a generalized message passing architecture

At a glance

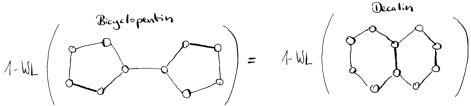
- Property prediction for small molecules is one main application area of GNNs
- Number and type of cycles in molecules is important
- But



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

At a glance

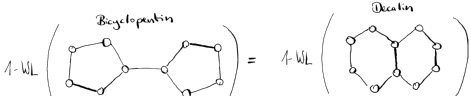
- Property prediction for small molecules is one main application area of GNNs
- Number and type of cycles in molecules is important
- But



- 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)

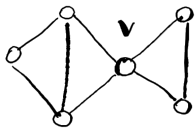
At a glance

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



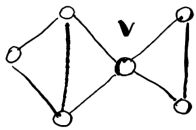
- we propose a generalized message passing architecture
- it can distinguish graphs with different ***r***-cycle counts
- it can homomorphism-count all ***r***-cactus graphs (strictly more expressive than 1-WL)
- fast in practice, s.o.t.a. results

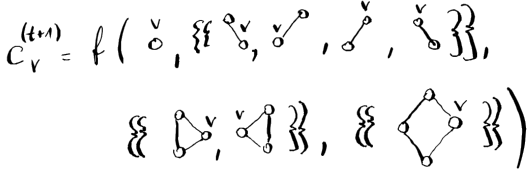
A glimpse at the implementation



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

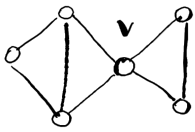
A glimpse at the implementation

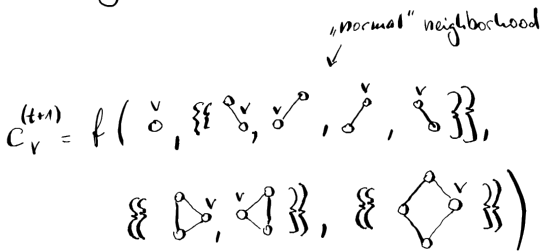




- Generalized message passing over multiple sets of local “neighborhoods”

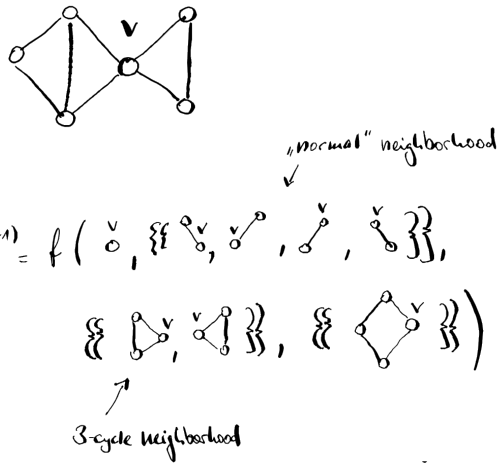
A glimpse at the implementation





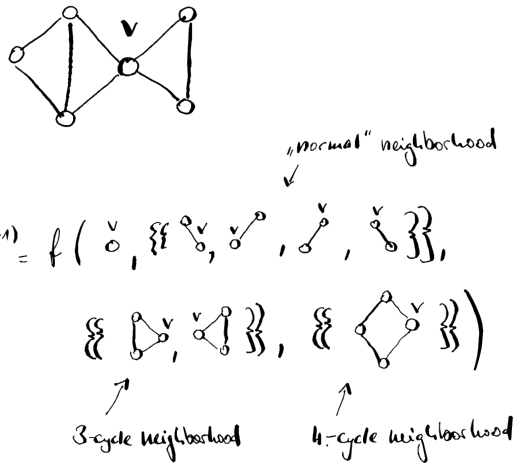
- Generalized message passing over multiple sets of local "neighborhoods"

A glimpse at the implementation



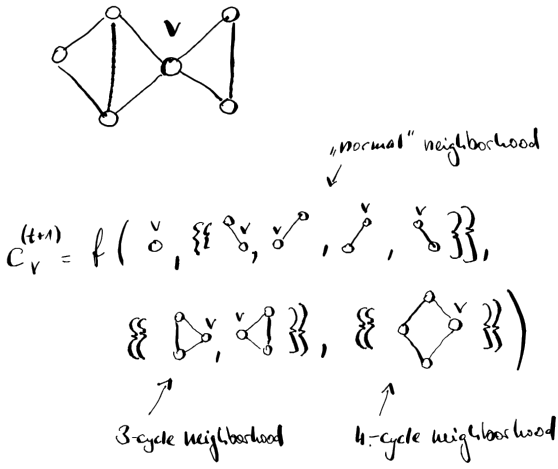
- Generalized message passing over multiple sets of local "neighborhoods"

A glimpse at the implementation



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

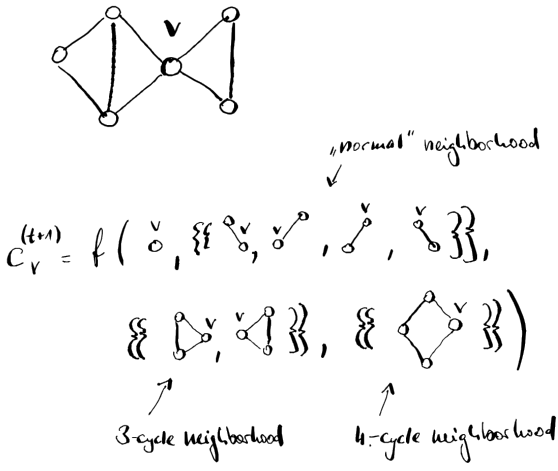
A glimpse at the implementation



- Generalized message passing over multiple sets of local "neighborhoods"
- Cycles can be enumerated quickly on many sparse graphs

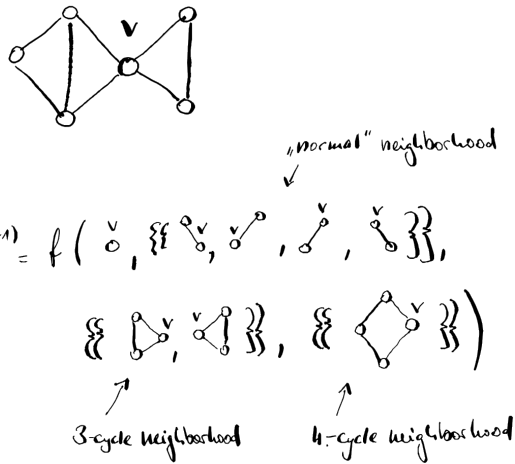
Horváth et al (2004)

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:

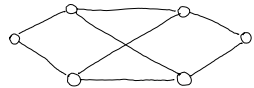
Contributions

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

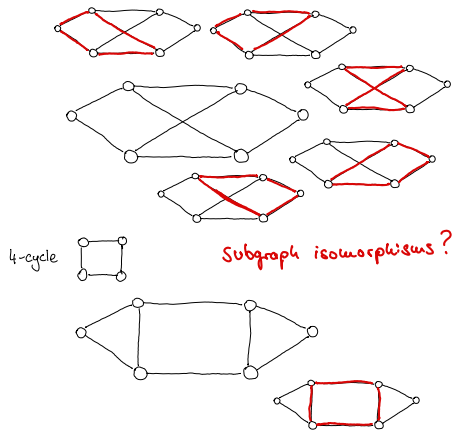
- is efficient on sparse graphs



Contributions

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

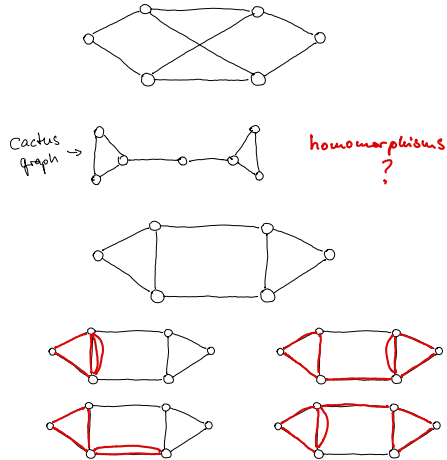
- is efficient on sparse graphs
- can subgraph count all cycles of length up to ***r***



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



This construction is actually a general recipe

- Select some graph patterns that you can locate more or less efficiently

This construction is actually a general recipe

- Select some graph patterns that you can locate more or less efficiently
  - Nodes
  - Edges
  - Cycles
  - Frequent subgraphs
  - ...

This construction is actually a general recipe

- Select some graph patterns that you can locate more or less efficiently
  - Nodes
  - Edges
  - Cycles
  - Frequent subgraphs
  - …
- Make them nodes in a new graph

This construction is actually a general recipe

- Select some graph patterns that you can locate more or less efficiently
  - Nodes
  - Edges
  - Cycles
  - Frequent subgraphs
  - ...
- Make them nodes in a new graph
- Define which objects are neighbors

This construction is actually a general recipe

- Select some graph patterns that you can locate more or less efficiently
  - Nodes
  - Edges
  - Cycles
  - Frequent subgraphs
  - ...
- Make them nodes in a new graph
- Define which objects are neighbors
- Define how to aggregate neighbor representations (using some neural net)

This construction is actually a general recipe

- Select some graph patterns that you can locate more or less efficiently
  - Nodes
  - Edges
  - Cycles
  - Frequent subgraphs
  - ...
- Make them nodes in a new graph
- Define which objects are neighbors
- Define how to aggregate neighbor representations (using some neural net)
- You have a (higher order) graph neural network

Examples

- Simplicial complexes: CWNetworks Bodnar et al (2021)

Examples

- Simplicial complexes: CWNetworks (Bodnar et al (2021))
- k-tuples (k-WL) (Morris et al (2019)) (Maron et al (2019))

Examples

- Simplicial complexes: CWNetworks (Bodnar et al (2021))
- k-tuples (k-WL) (Morris et al (2019)) (Maron et al (2019))
- paths (Graziani et al (2024)) (Michel et al (2023))

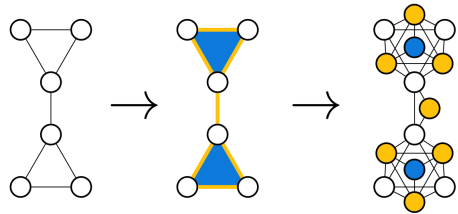
Examples

- Simplicial complexes: CWNetworks (Bodnar et al (2021))
- k-tuples (k-WL) (Morris et al (2019)) (Maron et al (2019))
- paths (Graziani et al (2024)) (Michel et al (2023))
- CAT for outerplanar graphs (Bause* et al (2025))

Examples

- Simplicial complexes: CWNetworks (Bodnar et al (2021))
- k-tuples (k-WL) (Morris et al (2019)) (Maron et al (2019))
- paths (Graziani et al (2024)) (Michel et al (2023))
- CAT for outerplanar graphs (Bause* et al (2025))

Most of these approaches can be explicitly written as simple message passing on transformed graphs, where all objects become nodes (Jogl et al (2023))



(Jogl et al (2023))

Deep Learning

Kernel Methods

Graph Kernels: Desired properties

We would like our graph kernels to be

- *tractable*, i.e., that $k(G_1, G_2)$ can be computed in polynomial time in the size of G_1 and G_2

Graph Kernels: Desired properties

We would like our graph kernels to be

- *tractable*, i.e., that $k(G_1, G_2)$ can be computed in polynomial time in the size of G_1 and G_2
- *invariant under isomorphism*, i.e. if $G_1, G_2 \in \mathcal{G}$ are isomorphic then

Graph Kernels: Desired properties

We would like our graph kernels to be

- *tractable*, i.e., that $k(G_1, G_2)$ can be computed in polynomial time in the size of G_1 and G_2
- *invariant under isomorphism*, i.e. if $G_1, G_2 \in \mathcal{G}$ are isomorphic then
  - $k(G_1, G_2) = k(G_1, G_1) = k(G_2, G_2)$

Graph Kernels: Desired properties

We would like our graph kernels to be

- *tractable*, i.e., that $k(G_1, G_2)$ can be computed in polynomial time in the size of G_1 and G_2
- *invariant under isomorphism*, i.e. if $G_1, G_2 \in \mathcal{G}$ are isomorphic then
  - $k(G_1, G_2) = k(G_1, G_1) = k(G_2, G_2)$
  - $k(G_1, H) = k(G_2, H)$ for any $H \in \mathcal{G}$

Graph Kernels: Desired properties

We would like our graph kernels to be

- *tractable*, i.e., that $k(G_1, G_2)$ can be computed in polynomial time in the size of G_1 and G_2
- *invariant under isomorphism*, i.e. if $G_1, G_2 \in \mathcal{G}$ are isomorphic then
  - $k(G_1, G_2) = k(G_1, G_1) = k(G_2, G_2)$
  - $k(G_1, H) = k(G_2, H)$ for any $H \in \mathcal{G}$
- *complete* i.e.,

$$\Phi(G_1) = \Phi(G_2) \text{ if and only if } G_1 \text{ is isomorphic to } G_2$$

Do tractable complete Graph Kernels exist?

Theorem (Gärtner et al (2003))

Computing a complete graph kernel on some graph class $\mathcal{G}$ is Isomorphism-hard.

Do tractable complete Graph Kernels exist?

Theorem (Gärtner et al (2003))

Computing a complete graph kernel on some graph class $\mathcal{G}$ is Isomorphism-hard.

- Any deep learning based graph learning method computes vectorial (or tensorial) representations for graphs

Do tractable complete Graph Kernels exist?

Theorem (Gärtner et al (2003))

Computing a complete graph kernel on some graph class $\mathcal{G}$ is Isomorphism-hard.

- Any deep learning based graph learning method computes vectorial (or tensorial) representations for graphs
- The proof (from 2003!) holds for all graph representations, whether learned or not.

Do tractable complete Graph Kernels exist?

Theorem (Gärtner et al (2003))

Computing a complete graph kernel on some graph class $\mathcal{G}$ is Isomorphism-hard.

- Any deep learning based graph learning method computes vectorial (or tensorial) representations for graphs
- The proof (from 2003!) holds for all graph representations, whether learned or not.
- It follows that it is highly unlikely that we will ever find an efficient maximally expressive GNN architecture for all graphs in the uniform sense

The issue with the datasets

- Recently we've been discussing a lot about datasets and benchmarks in the GNN community

The issue with the datasets

- Recently we've been discussing a lot about datasets and benchmarks in the GNN community
- Bechler-Speicher et al (2025) Coupette et al (2025) Bechler-Speicher et al (2024) Stoll et al (2025)

The issue with the datasets

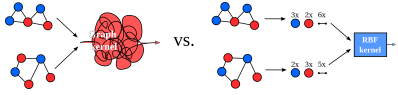
- Recently we've been discussing a lot about datasets and benchmarks in the GNN community
- Bechler-Speicher et al (2025) Coupette et al (2025) Bechler-Speicher et al (2024) Stoll et al (2025)
- this "reproduces" results from the graph kernel community

  Schulz and Welke (2018)

The issue with the datasets

- Recently we've been discussing a lot about datasets and benchmarks in the GNN community
- Bechler-Speicher et al (2025) Coupette et al (2025) Bechler-Speicher et al (2024) Stoll et al (2025)
- this "reproduces" results from the graph kernel community

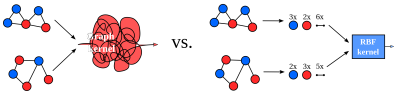
  Schulz and Welke (2018)



The issue with the datasets

- Recently we've been discussing a lot about datasets and benchmarks in the GNN community
- Bechler-Speicher et al (2025) Coupette et al (2025) Bechler-Speicher et al (2024) Stoll et al (2025)
- this "reproduces" results from the graph kernel community

Schulz and Welke (2018)



	NoG	psf	bpsf	fsg	cp	gs	sp	rw	wl
AIDS	99.65	98.25	98.45	97.85	98.60				98.70
BZR	86.16				78.78	78.53		73.82	
BZR_MD	70.19		58.14			52.34		50.66	60.75
COIL-DEL	14.13	7.83	7.68			9.74			
COIL-RAG	7.22	3.38	3.36	5.69		3.16	6.61		
COX2	81.37	78.16	78.16	69.39	77.95	78.16		77.95	
COX2_MD	65.26					51.81		51.15	
DD	76.67								
DHFR	74.06				54.77	67.05		60.98	82.02
DHFR_MD	64.88						69.20		
ENZYMES	43.33	28.33	32.00			30.50		17.33	50.67
ER_MD	74.46					59.42	59.21	59.42	
IMDB-BINARY	70.70	59.00	60.10	61.50		63.90	47.90		
IMDB-MULTI	46.73	38.93	40.13			39.53	34.13		
Letter-high	34.58	29.42	28.89			18.70	30.47		
Letter-low	43.02	27.29	27.07	27.29		14.93	48.60	47.08	39.91
Letter-med	38.71	26.80	27.07	26.80		14.51	44.67		
MSRC_21	86.46	51.84	52.22	46.79		14.60		5.06	
MSRC_21C	81.59			64.63		16.43		6.43	
MSRC_9	88.35					25.09		11.73	
MUTAG	87.31								
Mutagenicity	75.56				79.06	64.61			83.56
NCI1	69.93		74.33	76.28		62.68			84.72
NCI109	68.48	73.56	72.64	75.67	73.56	64.67			85.20
PROTEINS	74.58								
PROTEINS_full	74.58								
PTC_FM	63.63	58.17				57.08	57.85		
PTC_FR	67.25					65.53		65.25	
PTC_MM	66.70				61.93	61.31		61.62	
PTC_MR	57.60								
REDDIT-BINARY	83.50	55.75	57.70						72.80
REDDIT-MULTI-12K	36.99	23.18	23.12						
REDDIT-MULTI-5K	49.81	22.46	22.90						
SYNTHETIC	50.00								
SYNTHETICnew	64.33	51.33						54.00	99.33
Synthie	50.69		42.28			41.32		16.05	
Tox21_AHR	90.89	88.47	88.36	88.37		88.38	88.41		93.36
Tox21_AR-LBD	98.05					96.50	97.50		98.49
Tox21_ARE	86.42	84.79	84.74	84.69		84.67	84.75		88.94

Conclusion

Conclusion

- There are at least three pillars of graph learning

Conclusion

- There are at least three pillars of graph learning
- Combining techniques from different areas yields new methods and new insights

Conclusion

- There are at least three pillars of graph learning
- Combining techniques from different areas yields new methods and new insights
- Seeing these things may help to gain perspective and to be slightly less overwhelmed with the amount of new work being published every month in our field

References

Franka Bause*, Fabian Jogl*, Patrick Indri, Tamara Drucks, David Penz, Nils Morten Kriege, Thomas Gärtner, Pascal Welke, Maximilian Thiessen (2025) Maximally expressive gnns for outerplanar graphs. Transactions on Machine Learning Research (TMLR)

Maya Bechler-Speicher, Ido Amos, Ran Gilad-Bachrach, Amir Globerson (2024) Graph neural networks use graphs when they shouldn't. In: Ruslan Salakhutdinov, Zico Kolter, Katherine A Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, Felix Berkenkamp (eds) Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024, PMLR / OpenReview.net, Proceedings of Machine Learning Research, vol 235, pp 3284–3304, URL `https://proceedings.mlr.press/v235/bechler-speicher24a.html`

Maya Bechler-Speicher, Ben Finkelshtein, Fabrizio Frasca, Luis Müller, Jan Tönshoff, Antoine Siraudin, Viktor Zaverkin, Michael M Bronstein, Mathias Niepert, Bryan Perozzi, Mikhail Galkin, Christopher Morris (2025) Position: Graph learning will lose relevance due to poor benchmarks. In: Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025 - Position Paper Track, OpenReview.net, URL `https://proceedings.mlr.press/v267/bechler-speicher25a.html`

Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yuguang Wang, Pietro Liò, Guido F Montúfar, Michael M Bronstein (2021) Weisfeiler and lehman go cellular: CW networks. 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 2625–2640, URL `https://proceedings.neurips.cc/paper/2021/hash/157792e4abb490f99dbd738483e0d2d4-Abstract.html`

Corinna Coupette, Jeremy Wayland, Emily Simons, Bastian Rieck (2025) No metric to rule them all: Toward principled evaluations of graph-learning datasets. In: Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, Jerry Zhu (eds) Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025, PMLR / OpenReview.net, Proceedings of Machine Learning Research, vol 267, URL `https://proceedings.mlr.press/v267/coupette25a.html`

M Deshpande, M Kuramochi, N Wale, G Karypis (2005) Frequent substructure-based approaches for classifying chemical compounds. TKDE 17(8):1036–1050, DOI `10.1109/tkde.2005.127`

References

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

Thomas Gärtner, Peter A Flach, Stefan Wrobel (2003) On graph kernels: Hardness results and efficient alternatives. In: Bernhard Schölkopf, Manfred K Warmuth (eds) Computational Learning Theory and Kernel Machines, 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003, Proceedings, Springer, Lecture Notes in Computer Science, vol 2777, pp 129–143, DOI 10.1007/978-3-540-45167-9_11

Caterina Graziani, Tamara Drucks, Fabian Jogl, Monica Bianchini, Franco Scarselli, Thomas Gärtner (2024) The expressive power of path-based graph neural networks. In: Ruslan Salakhutdinov, Zico Kolter, Katherine A Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, Felix Berkenkamp (eds) Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024, PMLR / OpenReview.net, Proceedings of Machine Learning Research, vol 235, pp 16,226–16,249, URL https://proceedings.mlr.press/v235/graziani24a.html

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

References

Fabian Jogl, Maximilian Thiessen, Thomas Gärtner (2023) Expressivity-preserving GNN simulation. In: Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, Sergey Levine (eds) Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023, URL `http://papers.nips.cc/paper_files/paper/2023/hash/ebf95a6f3c575322da15d4fd0fc2b3c8-Abstract-Conference.html`

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, Yaron Lipman (2019) Provably powerful graph networks. In: Hanna M Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B Fox, Roman Garnett (eds) Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pp 2153–2164, URL `https://proceedings.neurips.cc/paper/2019/hash/bb04af0f7ecaee4aae62035497da1387-Abstract.html`

Gaspard Michel, Giannis Nikolentzos, Johannes F Lutzeyer, Michalis Vazirgiannis (2023) Path neural networks: Expressive and accurate graph neural networks. In: Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, Jonathan Scarlett (eds) Proceedings of the 40th International Conference on Machine Learning, PMLR, Proceedings of Machine Learning Research, vol 202, pp 24,737–24,755, URL `https://proceedings.mlr.press/v202/michel23a.html`

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, Martin Grohe (2019) Weisfeiler and leman go neural: Higher-order graph neural networks. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, AAAI Press, pp 4602–4609, DOI `10.1609/aaai.v33i01.33014602`, URL `https://doi.org/10.1609/aaai.v33i01.33014602`

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

Till Hendrik Schulz, Pascal Welke (2018) On the necessity of graph kernel baselines. Graph Embedding and Mining Workshop, (GEM@ECMLPKDD)

Timo Stoll, Chendi Qian, Ben Finkelshtein, Ali Parviz, Darius Weber, Fabrizio Frasca, Hadar Shavit, Antoine Siraudin, Arman Mielke, Marie Anastacio, et al (2025) Graphbench: Next-generation graph learning benchmarking. arXiv preprint arXiv:251204475