Recent Advances in Topology-Based Graph Classification

Bastian Rieck

Pseudomanifold

MLCB D BSSE ETH zürich

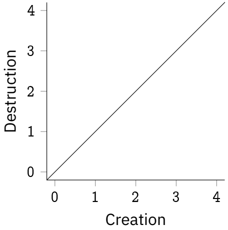
Setting the stage

"Nerzhin, his lips tightly drawn, was inattentive to the point of rudeness; he did not even bother to ask what exactly Verenyov had written about this arid branch of mathematics in which he himself had done a little work for one of his courses. [...] Topology belonged to the stratosphere of human thought. It might conceivably turn out to be of some use in the twenty-fourth century, but for the time being..."

The persistent homology of unlabelled & unweighted graphs

Use a filtration based on *vertex degrees*. Set $f(v) := \mathtt{deg}(v)$ for every vertex v and $f(u, v) := \mathtt{max}(\mathtt{deg}(u), \mathtt{deg}(v))$ for every edge (u, v) to obtain $f \colon \mathcal{G} \to \mathbb{R}$.

$f^{-1}((-\infty, 0])$



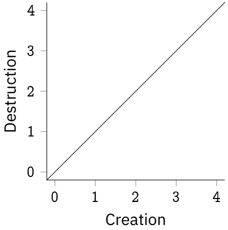
Classification

Vectorise persistence diagrams using persistence images, for instance, and add them to your favourite machine learning pipeline.

The persistent homology of unlabelled & unweighted graphs

Use a filtration based on *vertex degrees*. Set $f(v) := \mathtt{deg}(v)$ for every vertex v and $f(u,v) := \mathtt{max}(\mathtt{deg}(u), \mathtt{deg}(v))$ for every edge (u,v) to obtain $f\colon \mathcal{G} \to \mathbb{R}$.

$f^{-1}((-\infty, 1])$

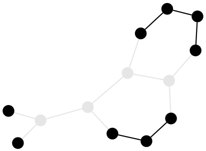


Classification

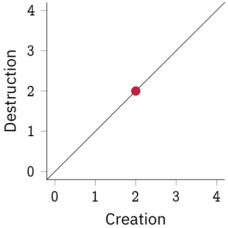
Vectorise persistence diagrams using persistence images, for instance, and add them to your favourite machine learning pipeline.

The persistent homology of unlabelled & unweighted graphs

Use a filtration based on *vertex degrees*. Set $f(v) := \mathrm{deg}(v)$ for every vertex v and $f(u, v) := \max(\mathrm{deg}(u), \mathrm{deg}(v))$ for every edge (u, v) to obtain $f\colon \mathcal{G} \to \mathbb{R}$.



$f^{-1}((-\infty, 2])$

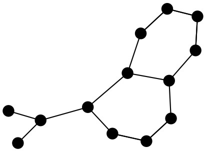


Classification

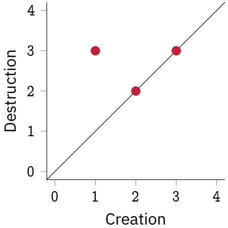
Vectorise persistence diagrams using persistence images, for instance, and add them to your favourite machine learning pipeline.

The persistent homology of unlabelled & unweighted graphs

Use a filtration based on *vertex degrees*. Set $f(v) := \mathrm{deg}(v)$ for every vertex v and $f(u, v) := \max(\mathrm{deg}(u), \mathrm{deg}(v))$ for every edge (u, v) to obtain $f\colon \mathcal{G} \to \mathbb{R}$.



$f^{-1}((-\infty, 3])$

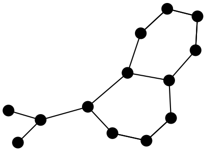


Classification

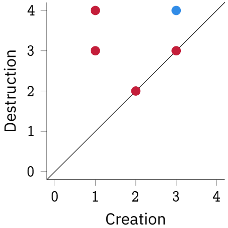
Vectorise persistence diagrams using persistence images, for instance, and add them to your favourite machine learning pipeline.

The persistent homology of unlabelled & unweighted graphs

Use a filtration based on *vertex degrees*. Set $f(v) := \deg(v)$ for every vertex v and $f(u,v) := \max(\deg(u), \deg(v))$ for every edge (u,v) to obtain $f\colon \mathcal{G} \to \mathbb{R}$.



$f^{-1}((-\infty, 3])$



Classification

Vectorise persistence diagrams using persistence images, for instance, and add them to your favourite machine learning pipeline.

Interlude

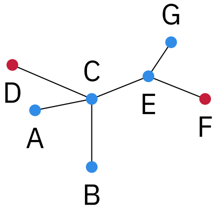
The Weisfeiler–Lehman test for graph isomorphism

1. *Create* a colour for each node in the graph (based on its label or its degree).
2. *Collect* colours of adjacent nodes in a multiset.
3. *Compress* the colours in the multiset and the node's colour to form a new one.
4. *Continue* this relabelling scheme until the colours are stable.

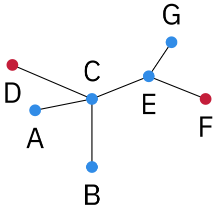
If the compressed labels of two graphs *diverge*, the graphs are *not* isomorphic!

The other direction is not valid! Non-isomorphic graphs can give rise to coinciding compressed labels.

Weisfeiler–Lehman iteration & feature vector

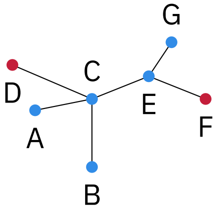


Weisfeiler–Lehman iteration & feature vector



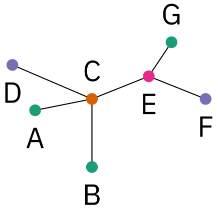
Node	Own label	Adjacent labels
A	🔵	🔵
B	🔵	🔵
C	🔵	🔵 🔵 🔵 🔴
D	🔴	🔵
E	🔵	🔵 🔵 🔴
F	🔴	🔵
G	🔵	🔵

Weisfeiler–Lehman iteration & feature vector



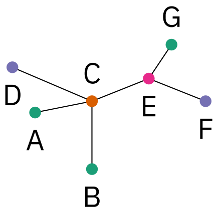
Node	Own label	Adjacent labels	Hashed label
A	●	●	●
B	●	●	●
C	●	● ● ● ●	●
D	●	●	●
E	●	● ● ●	●
F	●	●	●
G	●	●	●

Weisfeiler–Lehman iteration & feature vector



Node	Own label	Adjacent labels	Hashed label
A	●	●	●
B	●	●	●
C	●	● ● ● ●	●
D	●	●	●
E	●	● ● ●	●
F	●	●	●
G	●	●	●

Weisfeiler–Lehman iteration & feature vector



Label				
Count	3	1	2	1

$$\Phi(\mathcal{G}) := (3, 1, 2, 1)$$

Compare $\mathcal{G}$ and $\mathcal{G}'$ using a *kernel function*.

Kernels

$$\mathtt{k}(\mathcal{G}, \mathcal{G}') := \langle \Phi(\mathcal{G}), \Phi(\mathcal{G}') \rangle$$

$$\mathtt{k}(\mathcal{G}, \mathcal{G}') := \exp\left(-\|\Phi(\mathcal{G}) - \Phi(\mathcal{G}')\|^2 / \left(2\sigma^2\right)\right)$$

Making Weisfeiler–Lehman features persistent

A Persistent Weisfeiler–Lehman Procedure for Graph Classification

Bastian Rieck [*1] Christian Bock [*1] Karsten Borgwardt [1]

Abstract

The Weisfeiler–Lehman graph kernel exhibits competitive performance in many graph classification tasks. However, its subtree features are not able to capture connected components and cycles, topological features known for characterising graphs. To extract such features, we leverage propagated node label information and transform unweighted graphs into metric ones. This permits us to augment the subtree features with topological information obtained using *persistent homology*, a concept from topological data analysis. Our method, which we formalise as a generalisation of Weisfeiler–Lehman subtree features, exhibits favourable classification accuracy and its improvements in predictive performance are mainly driven by including cycle information.

1. Introduction

Graph-structured data sets are ubiquitous in a variety of different application domains, each of them posing a separate challenge while also requiring different tasks to be solved. A common task involves *graph classification*, for which a variety of methods exists. These methods comprise convolutional neural networks (Duvenaud et al. 2015), recurrent neural networks (Lei et al. 2017), or Hilbert space methods (Vishwanathan et al. 2010), the latter also being referred to as *graph kernels*. While several approaches for defining graph kernels exist, the most common one uses the $\mathcal{R}$-convolution framework (Haussler 1999), which makes it possible to define the similarity between two graphs as a function of the similarity of their substructures.

Substructures that have been used for graph classification range from graphlets (Shervashidze et al. 2009), i.e. small non-isomorphic graphs of fixed size, over shortest paths (Borgwardt & Kriegel 2005), to random walks (Gärtner et al. 2003, Kashima et al. 2003, Sugiyama & Borgwardt 2015). One of the most powerful substructures is the set of *subtree patterns* (Ramon & Gärtner 2003), i.e. patterns based on rooted subgraphs of a graph. Their computational complexity made their applicability somewhat limited, until the Weisfeiler–Lehman (WL) graph kernel framework (Shervashidze & Borgwardt 2009, Shervashidze et al. 2011) was developed. Properly trained, it still constitutes the state-of-the-art method for many graph classification tasks. The framework is based on the idea of iteratively propagating (node) label information through a graph, leading to a feature vector representation that can be used to assess the dissimilarity of two graphs.

One of the disadvantages of this framework is that its relabelling step, i.e. the step in which subtree patterns are being *compressed*, is somewhat "brittle": labels are only compared with a Dirac kernel, making their dissimilarity a coarse function. Moreover, the subtree feature vector only contains counts of compressed labels and can neither account for their relevance with respect to the topology of the graph nor capture connected components and cycles, both of which are important and interpretable features for characterising graphs (Rieck et al. 2018, Sizemore et al. 2017). We thus propose an enhancement of the original WL stabilisation procedure that uses recent advances in topological data analysis (Munch 2017) to alleviate these issues. Our contributions are as follows:

- We measure the relevance of topological features (connected components and cycles) in graphs and use them to define a novel set of WL subtree features, which we show to be a generalised version of the original ones.
- We develop a topology-based kernel that uses an iterative variant of the WL stabilisation procedure to classify non-attributed graphs.
- We demonstrate that our proposed features perform favourably on a range of graph classification benchmark data sets. In particular, we empirically show that the inclusion of *cycle* information yields classification accuracy improvements over state-of-the-art methods.

[*]Equal contribution [1]Department of Biosystems Science and Engineering, ETH Zurich, 4058 Basel, Switzerland. Correspondence to: Bastian Rieck <bastian.rieck@bsse.ethz.ch>, Karsten Borgwardt <karsten.borgwardt@bsse.ethz.ch>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

Christian Bock
chrs_bock

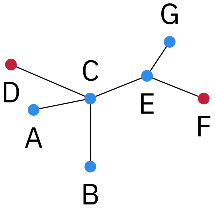
Karsten Borgwardt
kmborgwardt

- The Weisfeiler–Lehman algorithm vectorises labelled graphs
- Persistent homology captures relevant topological features
- We can *combine* them to obtain a *generalised* formulation

B. Rieck*, C. Bock* and K. Borgwardt, 'A Persistent Weisfeiler–Lehman Procedure for Graph Classification', *ICML*, 2019, pp. 5448–5458

A distance between label multisets

Example for $p = 1$



$$
\begin{aligned}
\mathrm{dist}_{\mathfrak{M}}(C, E) &= \mathrm{dist}_{\mathfrak{M}}\left(\{{\color{blue}\bullet}^3, {\color{red}\bullet}^1\}, \{{\color{blue}\bullet}^2, {\color{red}\bullet}^1\}\right) \\
&= \mathrm{dist}_{\mathfrak{M}}([3, 1], [2, 1]) \\
&= 1
\end{aligned}
$$

Use vertex label from *previous* Weisfeiler–Lehman iteration, i.e. $\mathrm{l}_{v_i}^{(h-1)}$, as well as $\mathrm{l}_{v_i}^{(h)}$, the one from the *current* iteration:

$$
\mathrm{dist}_V(v_i, v_j) := \left[\mathrm{l}_{v_i}^{(h-1)} \neq \mathrm{l}_{v_j}^{(h-1)}\right] + \mathrm{dist}_{\mathfrak{M}}\left(\mathrm{l}_{v_i}^{(h)}, \mathrm{l}_{v_j}^{(h)}\right) + \tau
$$

This turns *any* labelled graph into a weighted graph whose persistent homology we can calculate!

Persistence-based Weisfeiler–Lehman algorithm

1. Perform Weisfeiler–Lehman iteration for a number of steps.
2. In each step, obtain a filtration using the vertex distance.
3. Store persistence-based features.

Connected components

$$\Phi^{(h)}_{\text{P-WL}} := \left[\mathfrak{p}^{(h)}(l_0), \mathfrak{p}^{(h)}(l_1), \ldots\right]$$

$$\mathfrak{p}^{(h)}(l_i) := \sum_{\mathfrak{l}(v)=l_i} \operatorname{pers}(v)^p,$$

Cycles

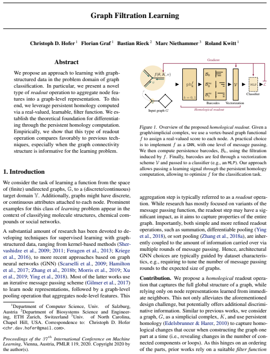
$$\Phi^{(h)}_{\text{P-WL-C}} := \left[\mathfrak{z}^{(h)}(l_0), \mathfrak{z}^{(h)}(l_1), \ldots\right]$$

$$\mathfrak{z}^{(h)}(l_i) := \sum_{l_i \in \mathfrak{l}(u,v)} \operatorname{pers}(u,v)^p,$$

Classification results & summary

	D & D	MUTAG	NCI1	NCI109	PROTEINS	PTC-MR	PTC-FR	PTC-MM	PTC-FM
V-Hist	78.32 ± 0.35	85.96 ± 0.27	64.40 ± 0.07	63.25 ± 0.12	72.33 ± 0.32	58.31 ± 0.27	68.13 ± 0.23	66.96 ± 0.51	57.91 ± 0.83
E-Hist	72.90 ± 0.48	85.69 ± 0.46	63.66 ± 0.11	63.27 ± 0.07	72.14 ± 0.39	55.82 ± 0.00	65.53 ± 0.00	61.61 ± 0.00	59.03 ± 0.00
RetGK*	81.60 ± 0.30	90.30 ± 1.10	84.50 ± 0.20		75.80 ± 0.60	62.15 ± 1.60	67.80 ± 1.10	67.90 ± 1.40	63.90 ± 1.30
WL	79.45 ± 0.38	87.26 ± 1.42	85.58 ± 0.15	84.85 ± 0.19	76.11 ± 0.64	63.12 ± 1.44	67.64 ± 0.74	67.28 ± 0.97	64.80 ± 0.85
Deep-WL*		82.94 ± 2.68	80.31 ± 0.46	80.32 ± 0.33	75.68 ± 0.54	60.08 ± 2.55			
P-WL	79.34 ± 0.46	86.10 ± 1.37	85.34 ± 0.14	84.78 ± 0.15	75.31 ± 0.73	63.07 ± 1.68	67.30 ± 1.50	68.40 ± 1.17	64.47 ± 1.84
P-WL-C	78.66 ± 0.32	90.51 ± 1.34	85.46 ± 0.16	84.96 ± 0.34	75.27 ± 0.38	64.02 ± 0.82	67.15 ± 1.09	68.57 ± 1.76	65.78 ± 1.22
P-WL-UC	78.50 ± 0.41	85.17 ± 0.29	85.62 ± 0.27	85.11 ± 0.30	75.86 ± 0.78	63.46 ± 1.58	67.02 ± 1.29	68.01 ± 1.04	65.44 ± 1.18

Learning graph filtrations



Graph Filtration Learning

Christoph D. Hofer[1] Florian Graf[1] Bastian Rieck[2] Marc Niethammer[3] Roland Kwitt[1]

Abstract

We propose an approach to learning with graph-structured data in the problem domain of graph classification. In particular, we present a novel type of *readout* operation to aggregate node features into a graph-level representation. To this end, we leverage persistent homology computed via a real-valued, learnable, filter function. We establish the theoretical foundation for differentiating through the persistent homology computation. Empirically, we show that this type of readout operation compares favorably to previous techniques, especially when the graph connectivity structure is informative for the learning problem.

Figure 1. Overview of the proposed *homological readout*. Given a graph/simplicial complex, we use a vertex-based graph functional f to assign a real-valued score to each node. A practical choice is to implement f as a `GNN`, with one level of message passing. We then compute persistence barcodes, $\mathcal{B}_k$, using the filtration induced by f. Finally, barcodes are fed through a vectorization scheme $\mathcal{V}$ and passed to a classifier (e.g., an `MLP`). Our approach allows passing a learning signal *through* the persistent homology computation, allowing to optimize f for the classification task.

1. Introduction

We consider the task of learning a function from the space of (finite) undirected graphs, $\mathbb{G}$, to a (discrete/continuous) target domain $\mathbb{Y}$. Additionally, graphs might have discrete, or continuous attributes attached to each node. Prominent examples for this class of *learning* problem appear in the context of classifying molecule structures, chemical compounds or social networks.

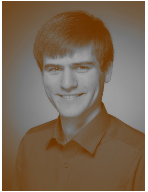
A substantial amount of research has been devoted to developing techniques for supervised learning with graph-structured data, ranging from kernel-based methods (Shervashidze et al., 2009; 2011; Feragen et al., 2013; Kriege et al., 2016), to more recent approaches based on graph neural networks (GNN) (Scarselli et al., 2009; Hamilton et al., 2017; Zhang et al., 2018b; Morris et al., 2019; Xu et al., 2019; Ying et al., 2018). Most of the latter works use an iterative message passing scheme (Gilmer et al., 2017) to learn node representations, followed by a graph-level pooling operation that aggregates node-level features. This aggregation step is typically referred to as a *readout* operation. While research has mostly focused on variants of the message passing function, the readout step may have a significant impact, as it aims to capture properties of the entire graph. Importantly, both simple and more refined readout operations, such as summation, differentiable pooling (Ying et al., 2018), or sort pooling (Zhang et al., 2018a), are inherently coupled to the amount of information carried over via multiple rounds of message passing. Hence, architectural GNN choices are typically guided by dataset characteristics, e.g., requiring to tune the number of message passing rounds to the expected size of graphs.

Contribution. We propose a *homological* readout operation that captures the full global structure of a graph, while relying only on node representations learned from immediate neighbors. This not only alleviates the aforementioned design challenge, but potentially offers additional discriminative information. Similar to previous works, we consider a graph, G, as a simplicial complex, K, and use persistent homology (Edelsbrunner & Harer, 2010) to capture homological changes that occur when constructing the graph one part at a time (i.e., revealing changes in the number of connected components or loops). As this hinges on an ordering of the parts, prior works rely on a suitable *filter function*

[1]Department of Computer Science, Univ. of Salzburg, Austria [2]Department of Biosystems Science and Engineering, ETH Zurich, Switzerland [3]Univ. of North Carolina, Chapel Hill, USA. Correspondence to: Christoph D. Hofer <chr.dav.hofer@gmail.com>.

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

Christoph Hofer



Florian Graf

Marc Niethammer
MarcNiethammer

Roland Kwitt
rkwitt1982

C. D. Hofer, F. Graf, **B. Rieck**, M. Niethammer and R. Kwitt, 'Graph Filtration Learning', *ICML*, 2020, arXiv: `1905.10996 [cs.LG]`

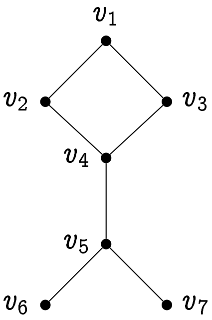
Graph neural networks in a nutshell

- Learn node representations h_v based on aggregated attributes a_v
- Aggregate them over neighbourhoods
- Iteration k contains information up to k hops away
- Repeat iteration K times

$$
\begin{aligned}
a_v^{(k)} &:= \texttt{aggregate}^{(k)}\left(\left\{h_u^{(k-1)} \mid u \in \mathcal{N}_{\mathcal{G}}(v)\right\}\right)\\
h_v^{(k)} &:= \texttt{combine}^{(k)}\left(h_v^{(k-1)}, a_v^{(k)}\right)\\
h_{\mathcal{G}} &:= \texttt{readout}\left(\left\{h_v^{(K)} \mid v \in V_{\mathcal{G}}\right\}\right)
\end{aligned}
$$

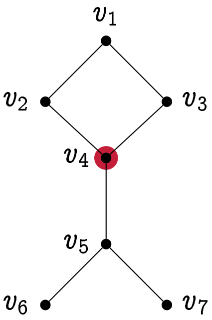
This terminology follows K. Xu, W. Hu, J. Leskovec and S. Jegelka, ‘How Powerful are Graph Neural Networks?’, *ICLR*, 2019.

Message passing in graphs



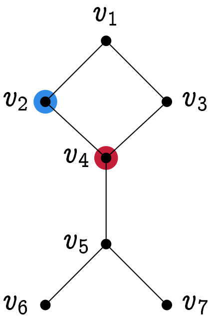
Here, $v_i \in \mathbb{R}^d$ is a d-dimensional attribute vector (use one-hot encoding for labels).

Message passing in graphs



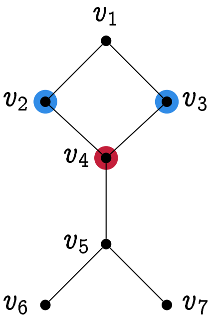
Here, $v_i \in \mathbb{R}^d$ is a d-dimensional attribute vector (use one-hot encoding for labels).

Message passing in graphs



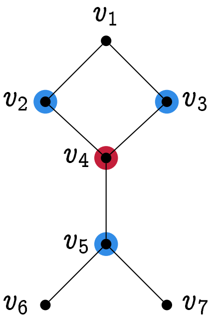
Here, $v_i \in \mathbb{R}^d$ is a d-dimensional attribute vector (use one-hot encoding for labels).

Message passing in graphs



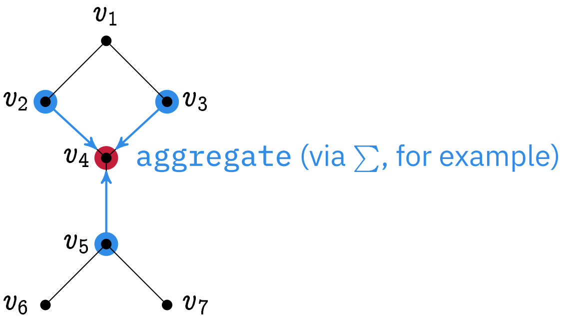
Here, $v_i \in \mathbb{R}^d$ is a d-dimensional attribute vector (use one-hot encoding for labels).

Message passing in graphs



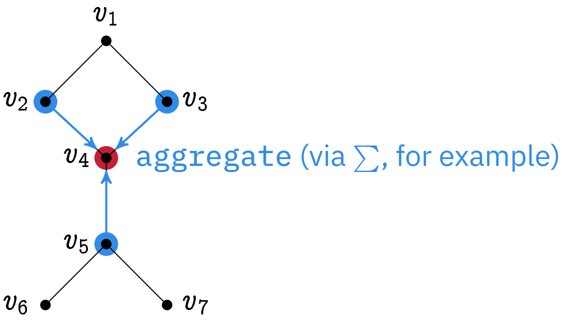
Here, $v_i \in \mathbb{R}^d$ is a d-dimensional attribute vector (use one-hot encoding for labels).

Message passing in graphs



Here, $v_i \in \mathbb{R}^d$ is a d-dimensional attribute vector (use one-hot encoding for labels).

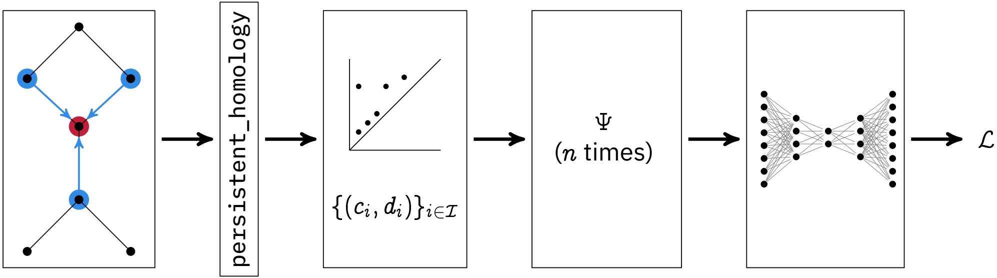
Message passing in graphs



Here, $v_i \in \mathbb{R}^d$ is a d-dimensional attribute vector (use one-hot encoding for labels).

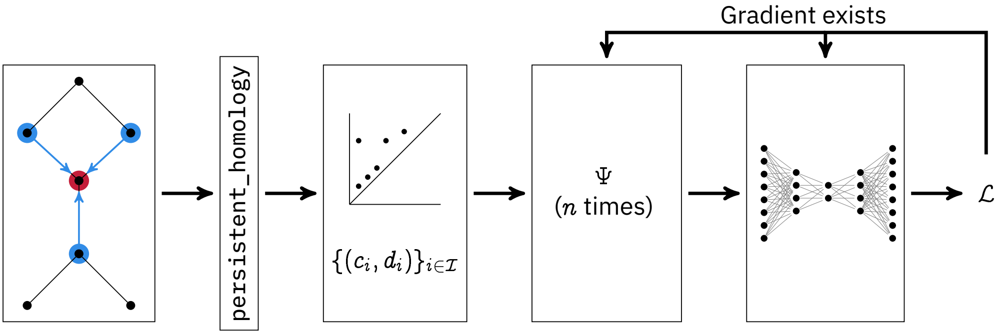
Repeat this process multiple times and update the vertex representations accordingly. Use a `readout` function to obtain a graph-level representation.

A `readout` function based on persistent homology



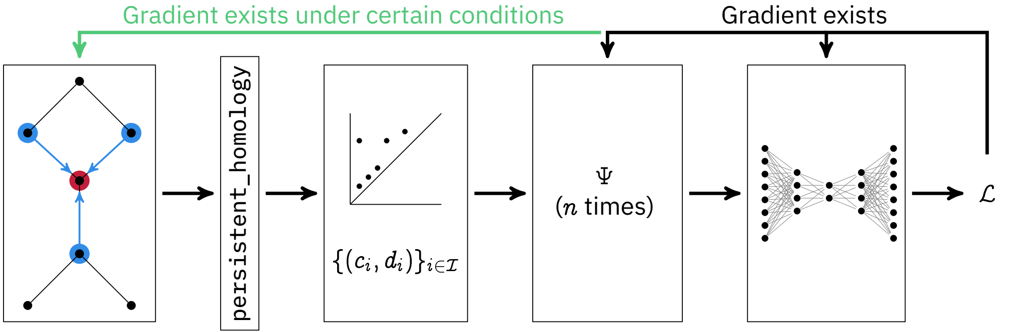
$\Psi\colon \mathcal{D} \to \mathbb{R}$ is a *differentiable* coordinatisation (embedding) function. By stacking n copies of Ψ, with different embedding parameters, we obtain an embedding into $\mathbb{R}^n$.

A `readout` function based on persistent homology



$\Psi \colon \mathcal{D} \to \mathbb{R}$ is a *differentiable* coordinatisation (embedding) function. By stacking n copies of Ψ, with different embedding parameters, we obtain an embedding into $\mathbb{R}^n$.

A `readout` function based on persistent homology



$\Psi \colon \mathcal{D} \to \mathbb{R}$ is a *differentiable* coordinatisation (embedding) function. By stacking n copies of Ψ, with different embedding parameters, we obtain an embedding into $\mathbb{R}^n$.

Graph filtration learning in practice

- If f is *injective* on the graph vertices, the gradient exists.
- We can initialise f using the vertex degree or uniform weights (plus a symbolic perturbation to ensure gradient existence).
- Simple integration into existing architectures.

Method	IMDB-BINARY	IMDB-MULTI
1-GIN (GFL)	74.5 ± 4.6	49.7 ± 2.9
1-GIN (SUM)	73.5 ± 3.8	50.3 ± 2.6
1-GIN (SP)	73.0 ± 4.0	50.5 ± 2.1
Only node features	72.7 ± 4.6	49.9 ± 4.0
PH	68.9 ± 3.5	46.1 ± 4.2

Topological layers for graph classification

Max Horn
ExpectationMax

Edward De Brouwer
EdwardOnBrew

Michael Moor
Michael_D_Moor

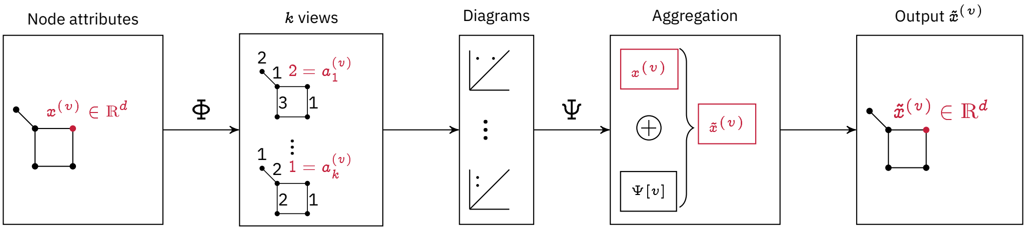
Yves Moreau

Karsten Borgwardt
kmborgwardt

M. Horn*, E. De Brouwer*, M. Moor, Y. Moreau, **B. Rieck***† and K. Borgwardt†,
Topological Graph Neural Networks, 2021, arXiv: `2102.07835 [cs.LG]`

Topological graph neural networks

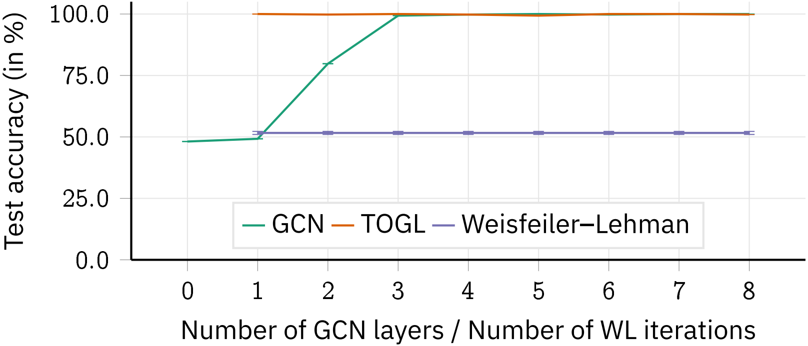
Overview

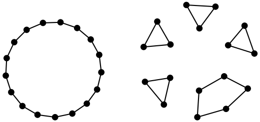


- Use a node map $\Phi \colon \mathbb{R}^d \to \mathbb{R}^k$ to create k different filtrations of the graph.
- Use a coordinatisation function Ψ to create *compatible* representations of the node attributes.

Expressivity

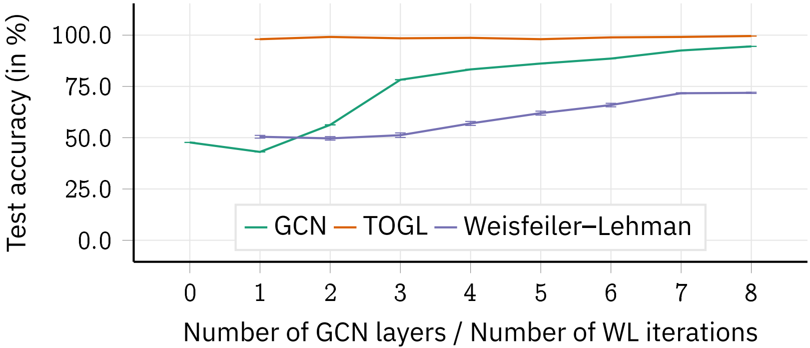
Cycles data set





Expressivity

Necklaces data set



Empirical results

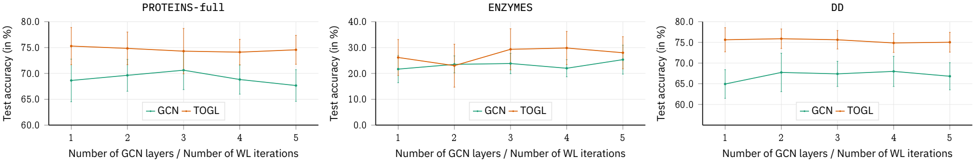
Replacing one GCN layer with TOGL

Method	PROTEINS-full	ENZYMES	DD	IMDB-BINARY	REDDIT-BINARY
GAT-4	76.3 ± 2.4	68.5 ± 5.2	75.9 ± 3.8	—	—
GATED-GCN-4	76.4 ± 2.9	65.7 ± 4.9	72.9 ± 2.1	—	—
GCN-4	76.1 ± 2.4	65.8 ± 4.6	72.8 ± 4.1	68.6 ± 4.9	92.8 ± 1.7
GIN-4	74.1 ± 3.4	65.3 ± 6.8	71.9 ± 3.9	72.9 ± 4.7	89.8 ± 2.2
TopoGNN-3-1	76.0 ± 3.9	53.0 ± 9.2	73.2 ± 4.7	72.0 ± 2.3	89.4 ± 2.2
WL	73.1 ± 0.5	54.3 ± 0.9	77.7 ± 2.0	71.2 ± 0.5	78.0 ± 0.6
WL-OA	73.5 ± 0.9	58.9 ± 0.9	77.8 ± 1.2	74.0 ± 0.7	87.6 ± 0.3

The inclusion of global topological features has a slightly detrimental effect for some of the data sets! Why?

Empirical results

What if we *drop* existing node features?



Summary

- Topological features improve graph classification tasks.
- Recent advances result in *differentiable* representations.
- Often, the main performance drive is unclear; we need *ablation studies* that disentangle performance.
- *Hybrid* models show particular promise for graph classification.

Acknowledgements


MLCB @ ETH Zurich


TU Graz


KU Leuven