Topological Representation Learning for Structured and Unstructured Data

Bastian Rieck

Pseudomanifold

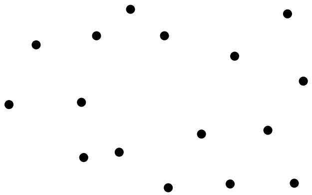
MLCB DBSSE ETH zürich

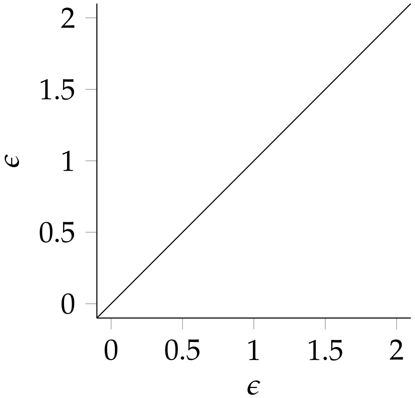
Setting the stage

"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..."

Persistent homology

Vietoris–Rips complex calculation

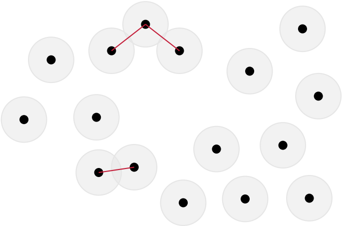


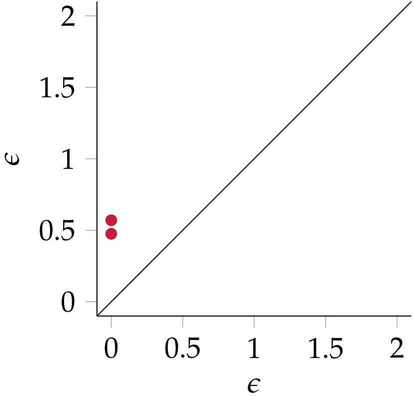


Given $\epsilon \in \mathbb{R}$, the Vietoris–Rips complex contains all simplices whose pairwise distance is less than or equal to ϵ. When using Euclidean balls of radius $r = 0.5\epsilon$, a simplex is created for each pairwise intersection.

Persistent homology

Vietoris–Rips complex calculation

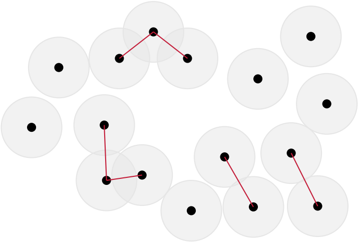


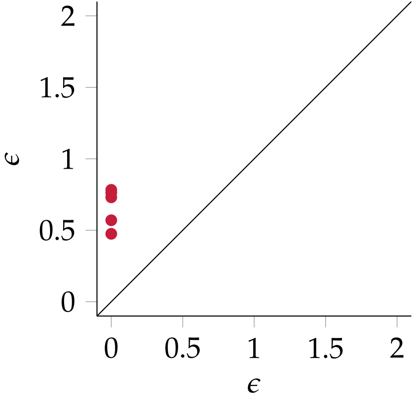


Given $\epsilon \in \mathbb{R}$, the Vietoris–Rips complex contains all simplices whose pairwise distance is less than or equal to ϵ. When using Euclidean balls of radius $r = 0.5\epsilon$, a simplex is created for each pairwise intersection.

Persistent homology

Vietoris–Rips complex calculation

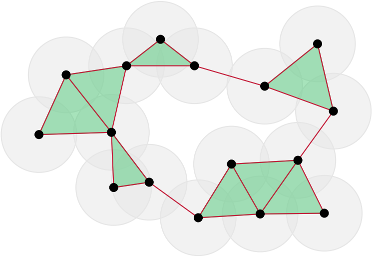


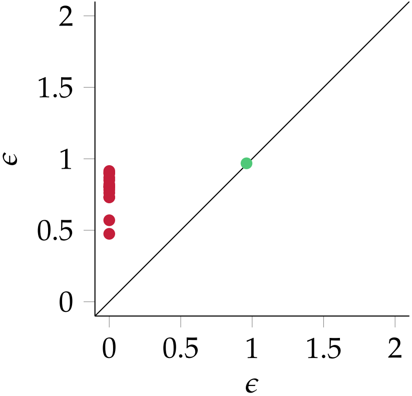


Given $\epsilon \in \mathbb{R}$, the Vietoris–Rips complex contains all simplices whose pairwise distance is less than or equal to ϵ. When using Euclidean balls of radius $r = 0.5\epsilon$, a simplex is created for each pairwise intersection.

Persistent homology

Vietoris–Rips complex calculation

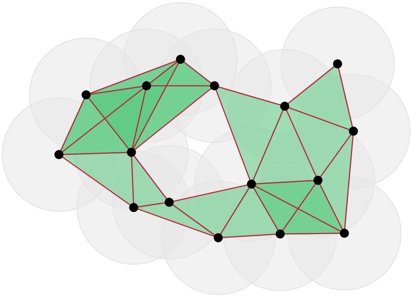


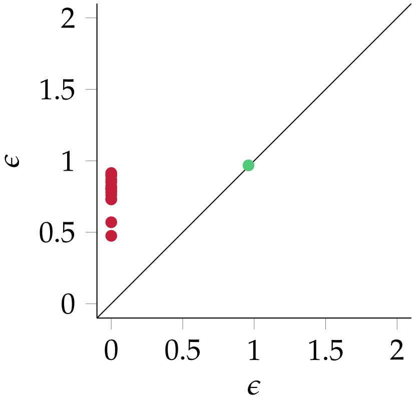


Given $\epsilon \in \mathbb{R}$, the Vietoris–Rips complex contains all simplices whose pairwise distance is less than or equal to ϵ. When using Euclidean balls of radius $r = 0.5\epsilon$, a simplex is created for each pairwise intersection.

Persistent homology

Vietoris–Rips complex calculation

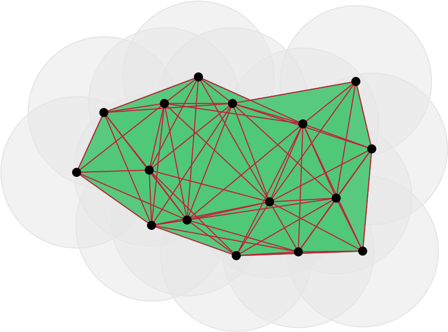


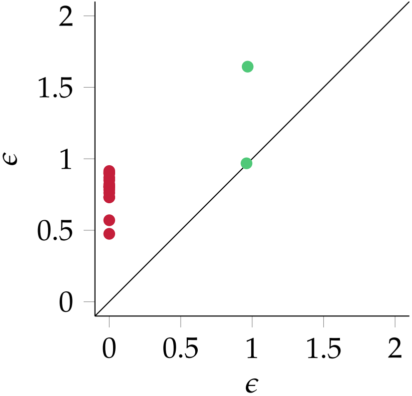


Given $\epsilon \in \mathbb{R}$, the Vietoris–Rips complex contains all simplices whose pairwise distance is less than or equal to ϵ. When using Euclidean balls of radius $r = 0.5\epsilon$, a simplex is created for each pairwise intersection.

Persistent homology

Vietoris–Rips complex calculation



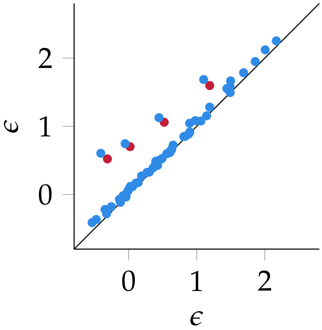


Given $\epsilon \in \mathbb{R}$, the Vietoris–Rips complex contains all simplices whose pairwise distance is less than or equal to ϵ. When using Euclidean balls of radius $r = 0.5\epsilon$, a simplex is created for each pairwise intersection.

Distances between persistence diagrams

Bottleneck distance

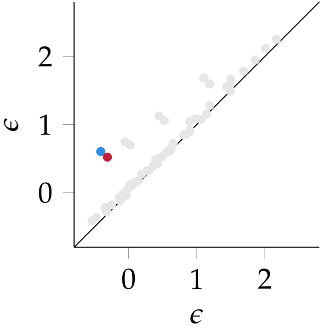
$$W_\infty(\mathcal{D}_1, \mathcal{D}_2) := \inf_{\eta\colon \mathcal{D}_1 \to \mathcal{D}_2} \sup_{x \in \mathcal{D}_1} \|x - \eta(x)\|_\infty$$



Distances between persistence diagrams

Bottleneck distance

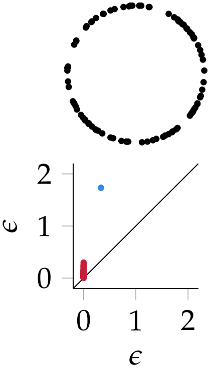
$$W_\infty(\mathcal{D}_1, \mathcal{D}_2) := \inf_{\eta\colon \mathcal{D}_1 \to \mathcal{D}_2} \sup_{x \in \mathcal{D}_1} \|x - \eta(x)\|_\infty$$



Stability theorem

Robustness to *small-scale* perturbations

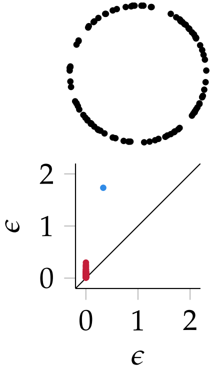
Let $\mathcal{M}$ be a triangulable space with continuous tame functions $f, g \colon \mathcal{M} \to \mathbb{R}$. Then the corresponding persistence diagrams satisfy $\mathrm{W}_\infty(\mathcal{D}_f, \mathcal{D}_g) \leq \|f - g\|_\infty$.

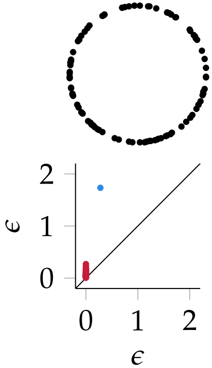


Stability theorem

Robustness to *small-scale* perturbations

Let $\mathcal{M}$ be a triangulable space with continuous tame functions $f, g\colon \mathcal{M} \to \mathbb{R}$. Then the corresponding persistence diagrams satisfy $W_\infty(\mathcal{D}_f, \mathcal{D}_g) \leq \|f - g\|_\infty$.

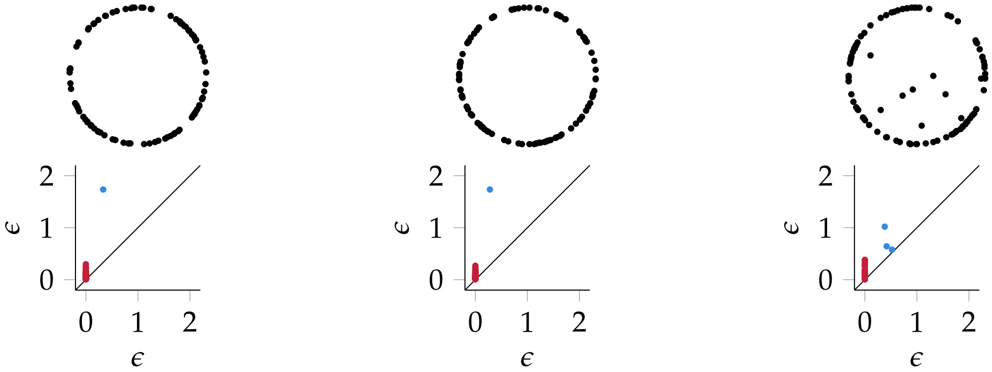




Stability theorem

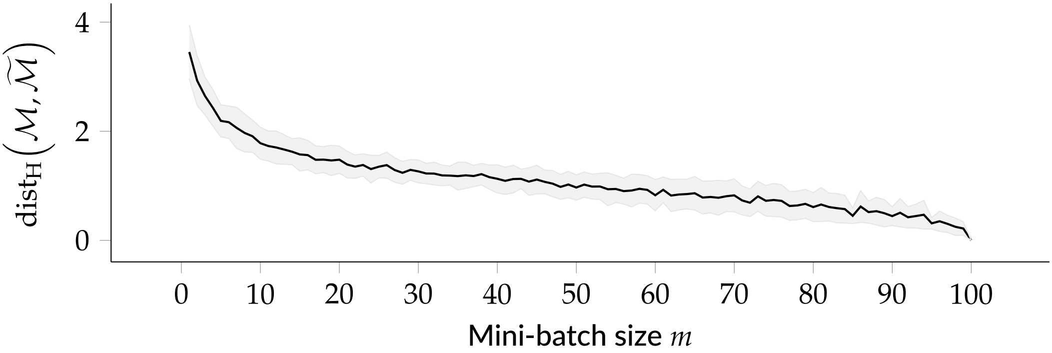
Robustness to *small-scale* perturbations

Let $\mathcal{M}$ be a triangulable space with continuous tame functions $f, g\colon \mathcal{M} \to \mathbb{R}$. Then the corresponding persistence diagrams satisfy $\mathrm{W}_\infty(\mathcal{D}_f, \mathcal{D}_g) \leq \|f - g\|_\infty$.



Implications for machine learning

Need to be careful when working with mini-batches $\widetilde{\mathcal{M}}$ of a point cloud $\mathcal{M}$. As an example, consider a point cloud with 100 points (normally-distributed in $\mathbb{R}^2$) and 50 subsamples of varying size m.



Bridging the chasm

- Persistent homology is inherently *discrete*
- Deep learning is inherently *continuous*

Challenge

Can we make the calculation of a persistence diagram *differentiable*, in particular if we have some control over the input space $\mathcal{M}$?

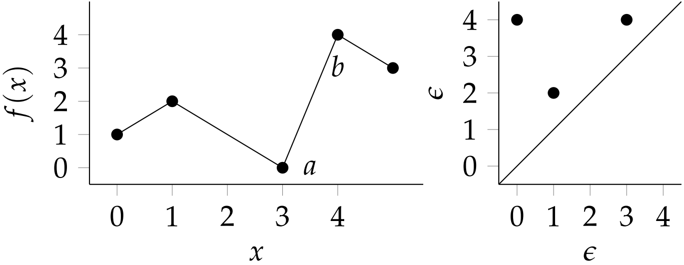
Making persistent homology differentiable

Terminology

- Let $f\colon \mathcal{M} \to \mathbb{R}$ be a function on a manifold. Persistent homology can be seen as a map from $(\mathcal{M}, f)$ to $\{(c_i, d_i)\}_{i \in \mathcal{I}}$.
- Let $\mathcal{S}$ be a map from points in the persistence diagram to simplex pairs (vertices and edges), i.e. $\mathcal{S}(c_i, d_i) = (\sigma_i, \tau_i)$. We write $\mathcal{S}(\cdot)$ to denote the map for a single point.
- Depending on the filtration, we can also map a simplex to one of its vertices. For a sublevel set filtration, we have a map $\mathcal{V}$ with $\mathcal{V}(\sigma) := \arg\max_{v \in \sigma} f(v)$.
- Finally, let $\mathcal{P} := (\mathcal{P}_c, \mathcal{P}_d)$, with $\mathcal{P}_c := \mathcal{V} \circ \mathcal{S}(c_i)$ and $\mathcal{P}_d := \mathcal{V} \circ \mathcal{S}(d_i)$.

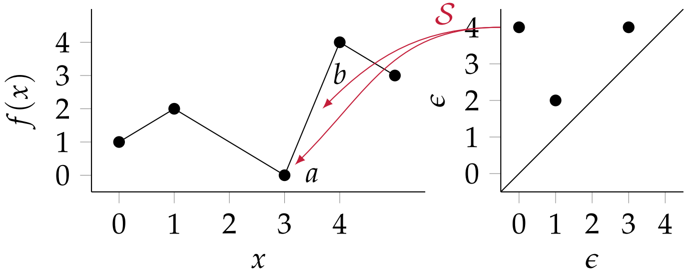
Making persistent homology differentiable

Example



Making persistent homology differentiable

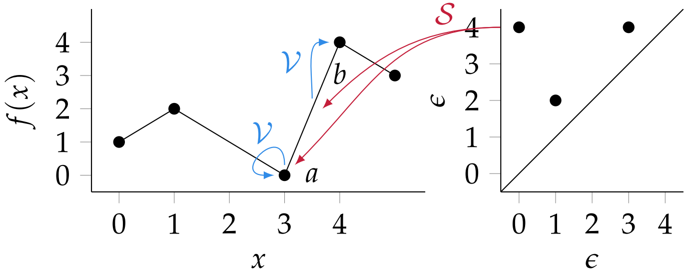
Example



We have $\mathcal{S}(0,4) = (\{a\}, \{a,b\})$.

Making persistent homology differentiable

Example

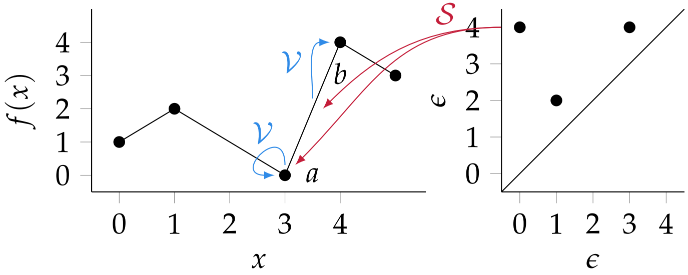


We have $\mathcal{S}(0,4) = (\{a\},\{a,b\})$.

We have $\mathcal{V}(\{a\}) = x_3$ and $\mathcal{V}(\{a,b\}) = x_4$.

Making persistent homology differentiable

Example



We have $\mathcal{S}(0,4) = (\{a\},\{a,b\})$.

We have $\mathcal{V}(\{a\}) = x_3$ and $\mathcal{V}(\{a,b\}) = x_4$.

We have $\mathcal{P}(0,4) = (\mathcal{V} \circ \mathcal{S})(0,4) = (x_3, x_4)$.

Making persistent homology differentiable

Gradient calculation sketch

- If the function values are *distinct*, then $\mathcal{P}$ is *unique*.
- If the function values are *distinct*, then $\mathcal{P}$ is *constant* in some neighbourhood.

Assume that f depends on $\theta = (\theta_1, \theta_2, \dots)$. We then have $f(\mathcal{P}_c(c_i)) = f(v_i) = c_i$, and, since $\mathcal{P}$ is constant,

$$\frac{\partial c_i}{\partial \theta_j} = \frac{\partial f(\mathcal{P}_c(c_i))}{\partial \theta_j} = \frac{\partial f(v_i)}{\partial \theta_j} = \frac{\partial f}{\partial \theta_j}(v_i),$$

i.e. the partial derivative is equivalent to the derivative of the function evaluated at the image of the map $\mathcal{P}_c$.

This formulation is due to A. Poulenard, P. Skraba and M. Ovsjanikov, 'Topological Function Optimization for Continuous Shape Matching', *Computer Graphics Forum* 37.5, 2018, pp. 13–25.

Topological autoencoders

Topological Autoencoders

Michael Moor [†1 2] **Max Horn** [†1 2] **Bastian Rieck** [‡1 2] **Karsten Borgwardt** [‡1 2]

Abstract

We propose a novel approach for preserving topological structures of the input space in latent representations of autoencoders. Using *persistent homology*, a technique from topological data analysis, we calculate topological signatures of both the input and latent space to derive a topological loss term. Under weak theoretical assumptions, we construct this loss in a differentiable manner, such that the encoding learns to retain multi-scale connectivity information. We show that our approach is theoretically well-founded and that it exhibits favourable latent representations on a synthetic manifold as well as on real-world image data sets, while preserving low reconstruction errors.

1. Introduction

While topological features, in particular multi-scale features derived from persistent homology, have seen increasing use in the machine learning community (Carrière et al., 2019, Guss & Salakhutdinov, 2018, Hofer et al., 2017, 2019a,b, Ramamurthy et al., 2019, Reininghaus et al., 2015, Rieck et al., 2019a,b), employing topology *directly* as a constraint for modern deep learning methods remains a challenge. This is due to the inherently discrete nature of these computations, making backpropagation through the computation of topological signatures immensely difficult or only possible in certain special circumstances (Chen et al., 2019, Hofer et al., 2019a, Poulenard et al., 2018).

This work presents a novel approach that permits obtaining gradients during the computation of topological signatures. This makes it possible to employ topological constraints while training deep neural networks, as well as building topology-preserving autoencoders. Specifically, we make the following contributions:

1. We develop a new topological loss term for autoencoders that helps harmonise the topology of the data space with the topology of the latent space.
2. We prove that our approach is stable on the level of mini-batches, resulting in suitable approximations of the persistent homology of a data set.
3. We empirically demonstrate that our loss term aids in dimensionality reduction by preserving topological structures in data sets; in particular, the learned latent representations are useful in that the preservation of topological structures can improve interpretability.

2. Background: Persistent Homology

Persistent homology (Barannikov, 1994, Edelsbrunner & Harer, 2008) is a method from the field of computational topology, which develops tools for analysing topological features (connectivity-based features such as connected components) of data sets. We first introduce the underlying concept of simplicial homology. For a simplicial complex $\mathfrak{K}$, i.e. a generalised graph with higher-order connectivity information such as cliques, simplicial homology employs matrix reduction algorithms to assign $\mathfrak{K}$ a family of groups, the *homology groups*. The d^{th} homology group $\mathrm{H}_d(\mathfrak{K})$ of $\mathfrak{K}$ contains d-dimensional topological features, such as connected components ($d = 0$), cycles/tunnels ($d = 1$), and voids ($d = 2$). Homology groups are typically summarised by their ranks, thereby obtaining a simple invariant "signature" of a manifold. For example, a circle in $\mathbb{R}^2$ has one feature with $d = 1$ (a cycle), and one feature with $d = 0$ (a connected component).

In practice, the underlying manifold $\mathbb{M}$ is unknown and we are working with a point cloud $X := \{x_1, \dots, x_n\} \subseteq \mathbb{R}^d$ and a metric $\mathrm{dist}\colon X \times X \to \mathbb{R}$ such as the Euclidean distance. Persistent homology extends simplicial homology to this setting: instead of approximating $\mathbb{M}$ by means of a *single* simplicial complex, which would be an unstable procedure due to the discrete nature of X, persistent homology tracks changes in the homology groups over *multiple* scales of the metric. This is achieved by constructing a special simplicial complex, the Vietoris–Rips complex (Vietoris, 1927). For $0 \leq \epsilon < \infty$, the Vietoris–Rips complex of X at scale ϵ, denoted by $\mathfrak{R}_\epsilon(X)$, contains all

[†]Equal contribution. [‡]These authors jointly directed this work. [1]Department of Biosystems Science and Engineering, ETH Zurich, 4058 Basel, Switzerland [2]SIB Swiss Institute of Bioinformatics, Switzerland. Correspondence to: Karsten Borgwardt <karsten.borgwardt@bsse.ethz.ch>.

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020. Copyright 2020 by the author(s).

Michael Moor
Michael_D_Moor

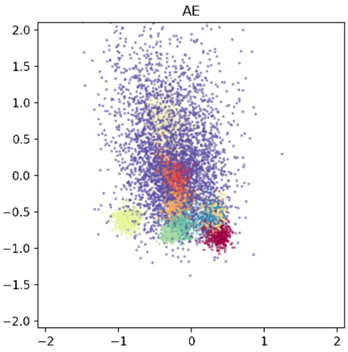
Max Horn
ExpectationMax

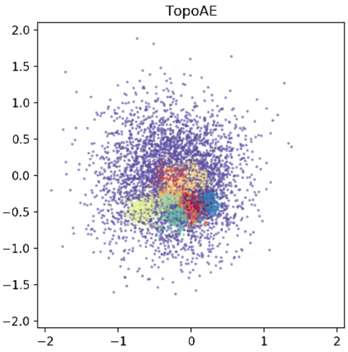
Karsten Borgwardt
kmborgwardt

M. Moor*, M. Horn*, **B. Rieck**† and K. Borgwardt†, 'Topological Autoencoders', *ICML*, 2020, arXiv: `1906.00722 [cs.LG]`

Topological autoencoders

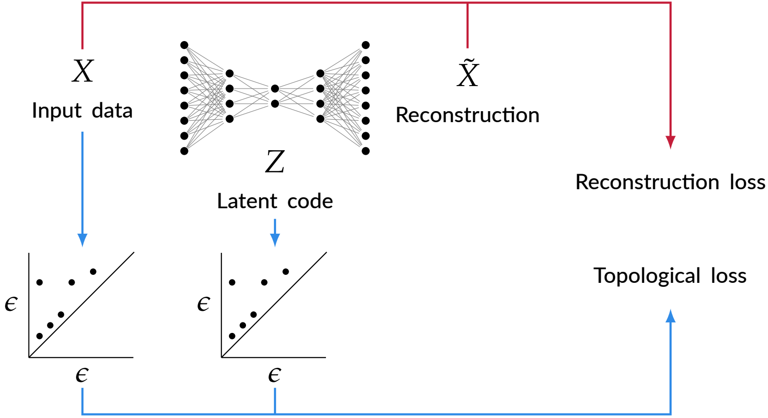
Motivation





Topological autoencoders

Overview



Topological autoencoders

Main intuition

Align persistence diagrams of an *input batch* and of a *latent batch* using a loss function!

Why this works in theory

Let X be a point cloud of cardinality n and $X^{(m)}$ be one subsample of X of cardinality m, i.e. $X^{(m)} \subseteq X$, sampled without replacement. We can bound the probability of the persistence diagrams of $X^{(m)}$ exceeding a threshold in terms of the bottleneck distance as

$$\mathbb{P}\left(\mathrm{W}_\infty\left(\mathcal{D}^X, \mathcal{D}^{X^{(m)}}\right) > \epsilon\right) \leq \mathbb{P}\left(\mathrm{dist}_{\mathrm{H}}\left(X, X^{(m)}\right) > 2\epsilon\right),$$

where $\mathrm{dist}_{\mathrm{H}}$ denotes the Hausdorff distance. In other words: *mini-batches are topologically similar if the subsampling is not too coarse*.

Topological autoencoders

Gradient calculation intuition

Distance matrix $\mathbf{A}$

$$\begin{bmatrix} 0 & 1 & 9 & 10 \\ 1 & 0 & 7 & 8 \\ 9 & 7 & 0 & 3 \\ 10 & 8 & 3 & 0 \end{bmatrix}$$

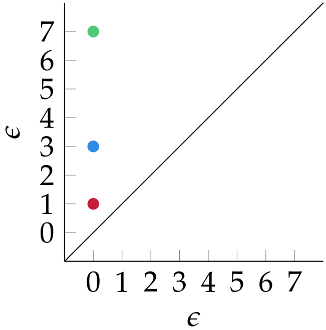
Every point in the persistence diagram can be mapped to *one* entry in the distance matrix! Each entry *is* a distance, so it can be changed during training (at least in the latent space).

Topological autoencoders

Gradient calculation intuition

Distance matrix $\mathbf{A}$

$$\begin{bmatrix} 0 & 1 & 9 & 10 \\ 1 & 0 & 7 & 8 \\ 9 & 7 & 0 & 3 \\ 10 & 8 & 3 & 0 \end{bmatrix}$$



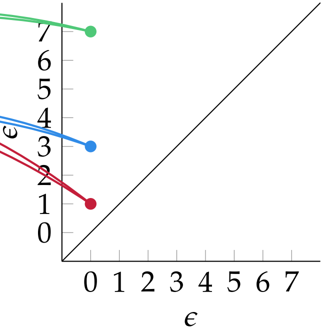
Every point in the persistence diagram can be mapped to *one* entry in the distance matrix! Each entry *is* a distance, so it can be changed during training (at least in the latent space).

Topological autoencoders

Gradient calculation intuition

Distance matrix $\mathbf{A}$

$$\begin{bmatrix} 0 & 1 & 9 & 10 \\ 1 & 0 & 7 & 8 \\ 9 & 7 & 0 & 3 \\ 10 & 8 & 3 & 0 \end{bmatrix}$$



Every point in the persistence diagram can be mapped to *one* entry in the distance matrix! Each entry *is* a distance, so it can be changed during training (at least in the latent space).

Topological autoencoders

Loss term

$$\mathcal{L}_t := \mathcal{L}_{\mathcal{X}\to\mathcal{Z}} + \mathcal{L}_{\mathcal{Z}\to\mathcal{X}}$$

$$\mathcal{L}_{\mathcal{X}\to\mathcal{Z}} := \frac{1}{2}\left\|\mathbf{A}^X\left[\pi^X\right] - \mathbf{A}^Z\left[\pi^X\right]\right\|^2 \qquad \mathcal{L}_{\mathcal{Z}\to\mathcal{X}} := \frac{1}{2}\left\|\mathbf{A}^Z\left[\pi^Z\right] - \mathbf{A}^X\left[\pi^Z\right]\right\|^2$$

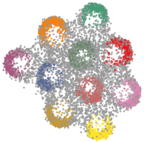
- $\mathcal{X}$: input space
- $\mathcal{Z}$: latent space
- $\mathbf{A}^X$: distances in input mini-batch
- $\mathbf{A}^Z$: distances in latent mini-batch
- π^X: persistence pairing of input mini-batch
- π^Z: persistence pairing of latent mini-batch

The loss is *bi-directional*!

Qualitative evaluation

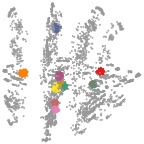
‘Spheres’ data set

PCA

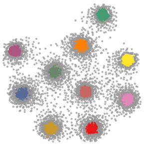


UMAP

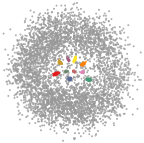
Autoencoder



Isomap



t-SNE

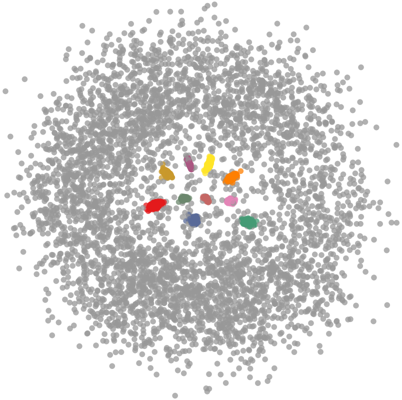


Topological autoencoder

Qualitative evaluation

'Spheres' data set; zooming in...

Autoencoder



Topological autoencoder

A new evaluation metric

Use *distance to a measure* density estimator, i.e.

$$f_\sigma^{\mathcal{X}}(x) := \sum_{y \in \mathcal{X}} \exp\Big(-\sigma^{-1}\operatorname{dist}(x,y)^2\Big),$$

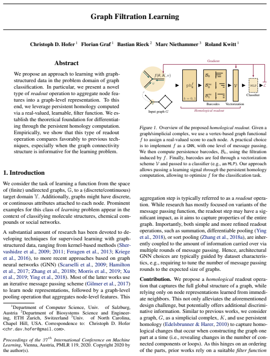
where dist denotes a metric such as the Euclidean distance. This is well-defined on mini-batches and on the full input data set.

Given σ, we evaluate $\mathrm{KL}_\sigma := \mathrm{KL}\big(f_\sigma^X \parallel f_\sigma^Z\big)$, which measures the similarity between the two density distributions.

Quantitative evaluation

Method	$\mathrm{KL}_{0.01}$	$\mathrm{KL}_{0.1}$	KL_{1}	ℓ-MRRE	ℓ-Cont	ℓ-Trust	ℓ-RMSE	MSE (data)
Isomap	0.181	**0.420**	**0.00881**	**0.246**	**0.790**	**0.676**	10.4	
PCA	0.332	0.651	0.01530	0.294	0.747	0.626	11.8	0.9610
t-SNE	**0.152**	0.527	0.01271	<u>**0.217**</u>	0.773	<u>**0.679**</u>	<u>**8.1**</u>	
UMAP	0.157	0.613	0.01658	0.250	0.752	0.635	**9.3**	
AE	0.566	0.746	0.01664	0.349	0.607	0.588	13.3	<u>**0.8155**</u>
TopoAE	<u>**0.085**</u>	<u>**0.326**</u>	<u>**0.00694**</u>	0.272	<u>**0.822**</u>	0.658	13.5	**0.8681**

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.

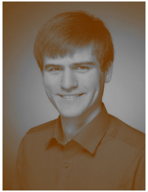
A substantial amount of research has been devoted to developing techniques for supervised learning with graph-structured data, ranging from kernel-based methods (Shervashidze et al., 2009; 2011; Feragen et al., 2013; Kriege et al., 2016), to more recent approaches based on graph neural networks (GNN) (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. Copyright 2020 by the author(s).

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]`

Digression

Graph neural networks in a nutshell

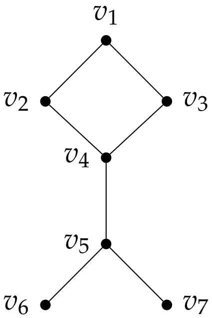
- Learn node representations h_v based on aggregated attributes a_v
- Aggregate them over neighbourhoods
- Iteration k contains information up to k hops away
- Repeat iteration K times

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

This terminology follows K. Xu, W. Hu, J. Leskovec and S. Jegelka, 'How Powerful are Graph Neural Networks?', *International Conference on Learning Representations*, 2019.

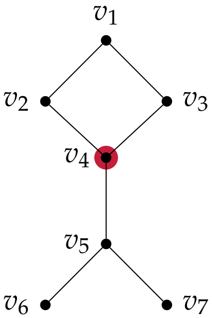
Digression

Message passing in graphs



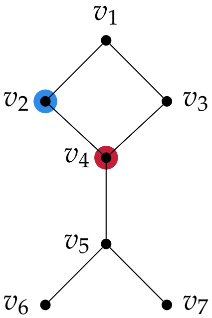
Digression

Message passing in graphs



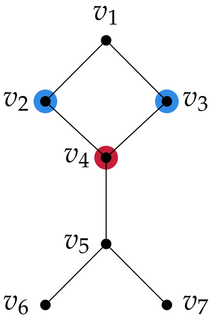
Digression

Message passing in graphs



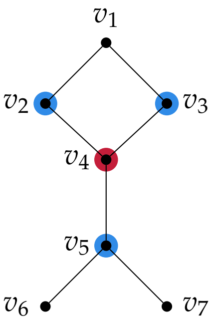
Digression

Message passing in graphs



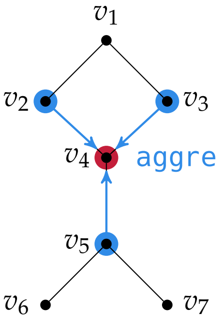
Digression

Message passing in graphs



Digression

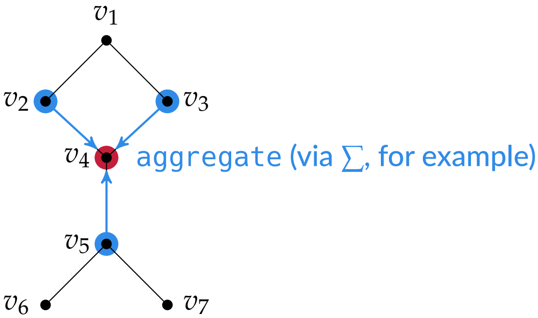
Message passing in graphs



aggregate (via $\sum$, for example)

Digression

Message passing in graphs



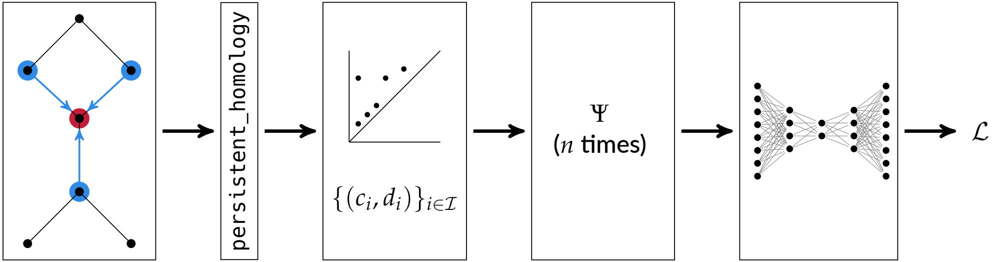
Repeat this process multiple times and update the vertex representations accordingly.
Use a `readout` function to obtain a graph-level representation.

Learning graph filtrations

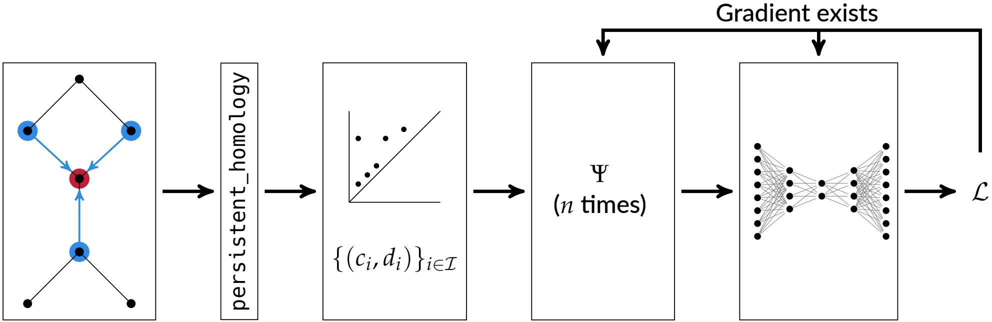
Motivation

- When classifying graphs with TDA, we often employ a *filter function* $f\colon \mathfrak{V} \to \mathbb{R}$. For example, $f(v) := \deg(v)$ is commonly employed.
- We typically extend f to a full graph G by setting $f(\{u, v\} := \max\{f(u), f(v)\}$.
- Can we *learn* f end-to-end?

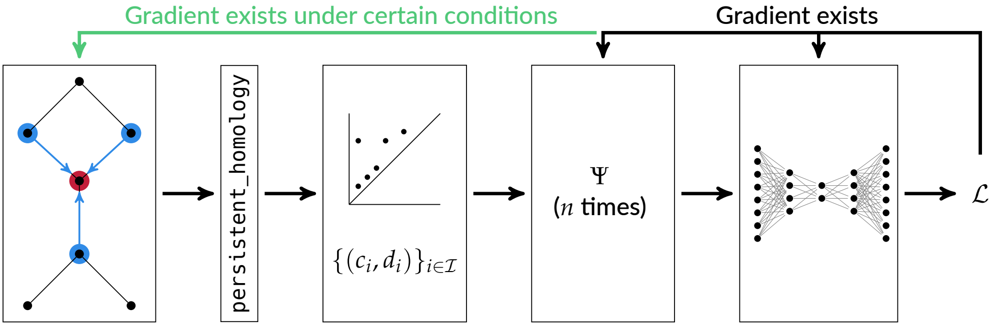
A `readout` function based on persistent homology



A `readout` function based on persistent homology



A `readout` function based on persistent homology



Coordinatisation function

Use a differentiable *coordinatisation* scheme of the form $\Psi\colon \mathcal{D} \to \mathbb{R}$. Letting $p := (a, b)$ denote a tuple in a persistence diagram, we have

$$\Psi(p) := \frac{1}{1 + \|p - c\|_1} - \frac{1}{1 + \mathrm{abs}(r - \|p - c\|_1)},$$

with $c \in \mathbb{R}^2$ and $r \in \mathbb{R}_{>0}$ being *trainable* parameters. The whole diagram is represented as a sum over each individual projections.

Using n different coordinatisations, we obtain a differentiable embedding of a persistence diagram into $\mathbb{R}^n$.

How to initialise f?

Use a single GIN-ϵ layer with one level of message passing (`1-GIN`) with hidden dimensionality 64, followed by a two-layer MLP.

`GIN-1`, $h = 64$ $\longrightarrow$ `MLP`$(64, 64, 1)$ with sigmoid activation

Hence, $f\colon \mathfrak{V} \to [0, 1]$.

We can initialise f using the vertex degree or uniform weights (plus a symbolic perturbation to ensure gradient existence).

Using this 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.

Using this 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
Baseline	72.7±4.6	49.9±4.0
PH	68.9±3.5	46.1±4.2

Topological layers for graph classification

arXiv:2102.07835v1 [cs.LG] 15 Feb 2021

Topological Graph Neural Networks

Max Horn[1,2,*] Edward De Brouwer[3,*] Michael Moor[1,2] Yves Moreau[3]
Bastian Rieck[1,2,*,†], and Karsten Borgwardt[1,2,†]

[1]Department of Biosystems Science and Engineering, ETH Zurich, 4058 Basel, Switzerland
[2]SIB Swiss Institute of Bioinformatics, Switzerland
[3]ESAT-STADIUS, KU LEUVEN, 3001 Leuven, Belgium
[*]These authors contributed equally.
[†]These authors jointly supervised this work.

Abstract

Graph neural networks (GNNs) are a powerful architecture for tackling graph learning tasks, yet have been shown to be oblivious to eminent substructures, such as cycles. We present TOGL, a novel layer that incorporates global topological information of a graph using persistent homology. TOGL can be easily integrated into *any type* of GNN and is strictly more expressive in terms of the Weisfeiler–Lehman test of isomorphism. Augmenting GNNs with our layer leads to beneficial predictive performance, both on synthetic data sets, which can be trivially classified by humans but not by ordinary GNNs, and on real-world data.

1. Introduction

Graphs are a natural description of structured data sets in many domains, including bioinformatics, image processing, and social network analysis. Numerous methods address graph learning problems such as graph classification or node classification. *Graph neural networks* (GNNs) describe a flexible set of architectures for graph learning tasks and have seen many successful applications over recent years. At their core, many GNNs are based on iterative message passing schemes. Since these schemes are collating information over the neighbours of every node, GNNs cannot necessarily capture certain simple topological structures in graphs, such as cycles.

By contrast, methods based on topological features, commonly summarised under the umbrella term of *topological data analysis* (TDA), have shown promising results in machine learning tasks. Focusing on coarse structures—such as the presence or absence of cycles—they can be used to provide multi-scale representations that capture the shape of complex structured and unstructured data sets. In this paper, we propose a Topological Graph Layer (TOGL) that can be easily integrated into any GNN to make it 'topology-aware'. We thus obtain a generic way to augment existing GNNs and increase their expressivity in graph learning tasks.

Our contributions. We describe a layer based on TDA concepts that can be integrated into any GNN. Our layer is differentiable and capable of learning different topological representations of a graph. We prove that even by itself, our layer is strictly more expressive than any GNN since it incorporates the ability to work with multi-scale topological information in a graph. Moreover, we show that TOGL can improve predictive performance of a GNN when topological information is relevant for the task.

2. Background: Computational Topology

This paper considers undirected graphs, i.e. of the form $G = (V, E)$ with a set of vertices V and a set of edges $E \subseteq V \times V$. A simple set of topological features of is given by their number of connected components β_0 and their number of cycles β_1. These counts are also known as the *Betti numbers* in dimension 0 and dimension 1, respectively, and can be computed efficiently (we will discuss this below). Betti numbers are invariant under graph isomorphism; this is a consequence of a more general theorem in algebraic topology that incorporates higher-dimensional topological features as well (Hatcher, 2002, pp. 103–133). Thus, given two graphs G and G' with $G \simeq G'$, their Betti numbers will coincide. As with many invariants that are computationally efficient to compute, the converse relation does not hold, limiting their ability to discriminate between graphs. This is exacerbated by the fact that Betti numbers constitute mere feature counts. It is possible to increase their expressivity by assuming the existence of a *graph filtration*, i.e. a sequence of nested subgraphs of G such that

$$\emptyset = G^{(0)} \subseteq G^{(1)} \subseteq G^{(2)} \subseteq \cdots \subseteq G^{(n-1)} \subseteq G^{(n)} = G. \quad (1)$$

A filtration makes it possible to obtain more insights into the graph by 'monitoring' topological features of *each* $G^{(i)}$ and calculating their topological relevance, which is referred to as their *persistence*. If a topological feature appears for the first time in $G^{(i)}$ and disappears in $G^{(j)}$, we assign this feature a persistence of $j - i$. Equivalently, we can represent each of these features as a tuple (i, j), which we collect in a *persistence diagram* $\mathcal{D}$. If a feature never disappears, we represent it by a tuple (i, ∞); such features are the ones that are counted for the respective Betti numbers. This process was formalised and extended to a wider class of structured data sets, namely simplicial complexes, and is known under the name of *persistent*

1

Max Horn
ExpectationMax

Edward De Brouwer
EdwardOnBrew

Michael Moor
Michael_D_Moor

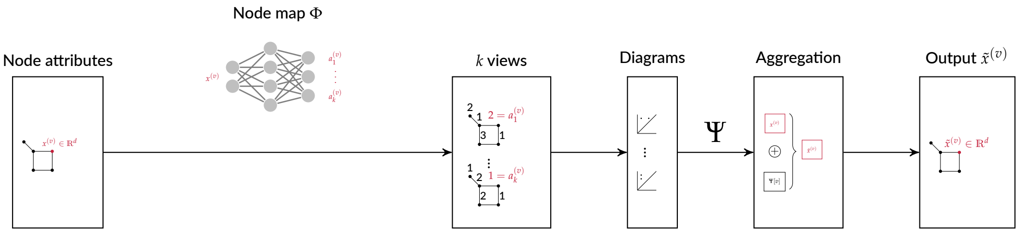
Yves Moreau

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]`

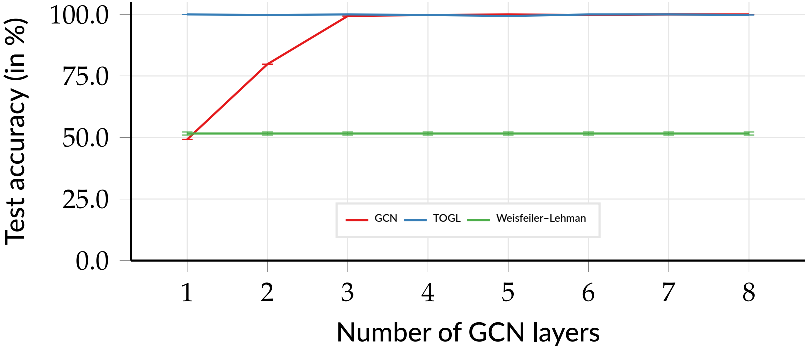
Topological graph neural networks

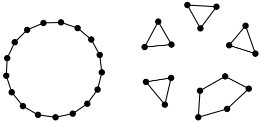
Overview



Expressivity

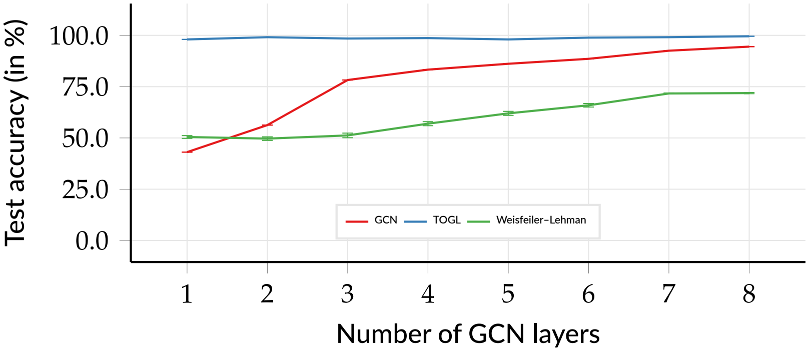
Cycles data set

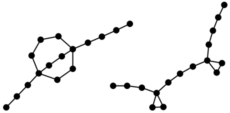




Expressivity

Necklaces data set





Empirical results

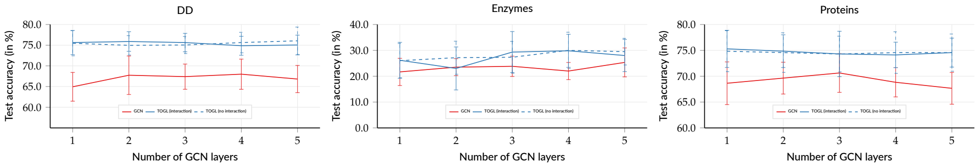
Method	PROTEINS-full	ENZYMES	DD	IMDB-BINARY	REDDIT-BINARY
GAT-4	76.3 ± 2.4	68.5 ± 5.2	75.9 ± 3.8	—	—
GATED-GCN-4	76.4 ± 2.9	65.7 ± 4.9	72.9 ± 2.1	—	—
GCN-4	76.1 ± 2.4	65.8 ± 4.6	72.8 ± 4.1	68.6 ± 4.9	92.8 ± 1.7
GIN-4	74.1 ± 3.4	65.3 ± 6.8	71.9 ± 3.9	72.9 ± 4.7	89.8 ± 2.2
TopoGNN-3-1	76.0 ± 3.9	53.0 ± 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

Empirical results

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

Empirical results

Only random node features



Summary

- Persistent homology *can* be made differentiable!
- Topological features improve representation learning tasks.
- Often, the main performance drive is unclear; we need *ablation studies* that disentangle performance.
- *Hybrid* models show particular promise for graph classification.

♥ My main co-authors Christian, Christoph, Edward, Karsten, Max, Michael, Roland.