HELMHOLTZ MUNICH AIH Institute of AI for Health

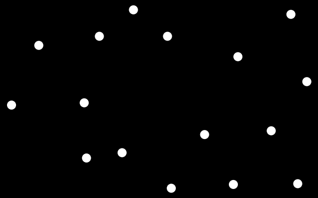
Topological Representation Learning

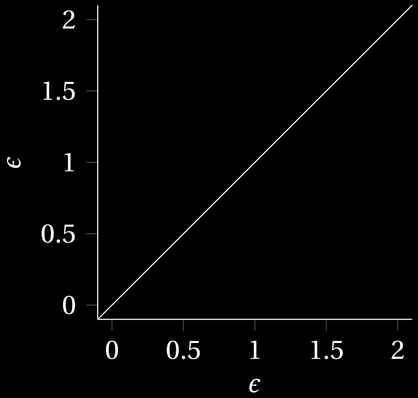
A Differentiable Perspective

Bastian Rieck (@Pseudomanifold)

Persistent homology

Vietoris–Rips complex calculation

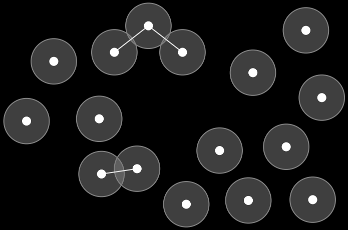


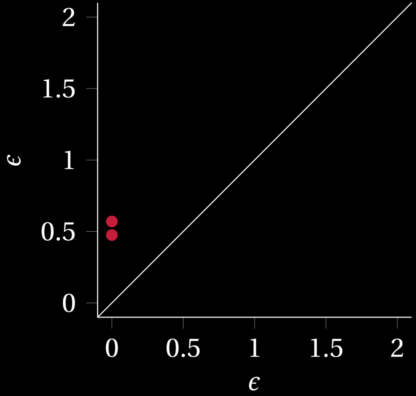


Given $\epsilon \in \mathbb{R}$, the Vietoris–Rips complex contains all simplices whose pairwise distance is less than or equal to ϵ. When using Euclidean balls of radius $r = 0.5\epsilon$, a simplex is created for each pairwise intersection.

Persistent homology

Vietoris–Rips complex calculation

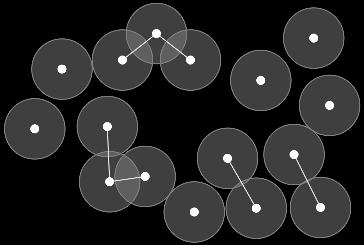


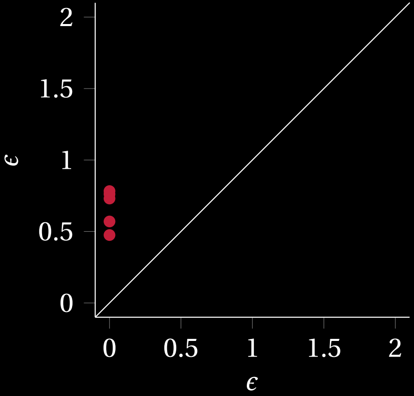


Given $\epsilon \in \mathbb{R}$, the Vietoris–Rips complex contains all simplices whose pairwise distance is less than or equal to ϵ. When using Euclidean balls of radius $r = 0.5\epsilon$, a simplex is created for each pairwise intersection.

Persistent homology

Vietoris–Rips complex calculation

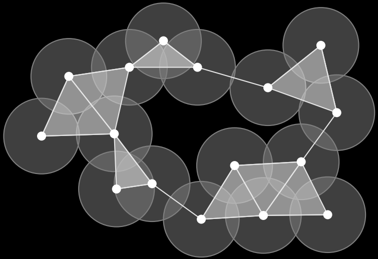


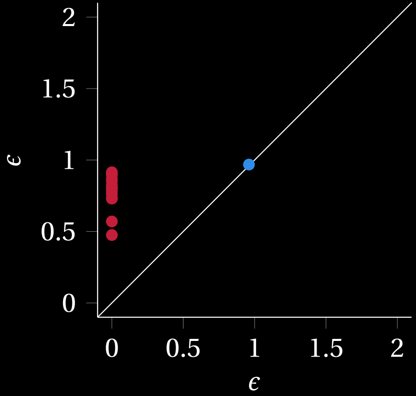


Given $\epsilon \in \mathbb{R}$, the Vietoris–Rips complex contains all simplices whose pairwise distance is less than or equal to ϵ. When using Euclidean balls of radius $r = 0.5\epsilon$, a simplex is created for each pairwise intersection.

Persistent homology

Vietoris–Rips complex calculation

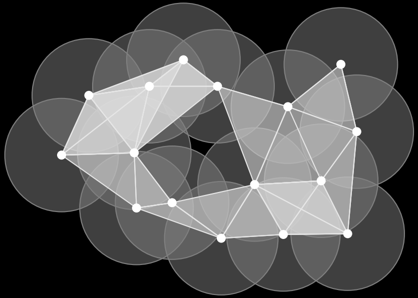


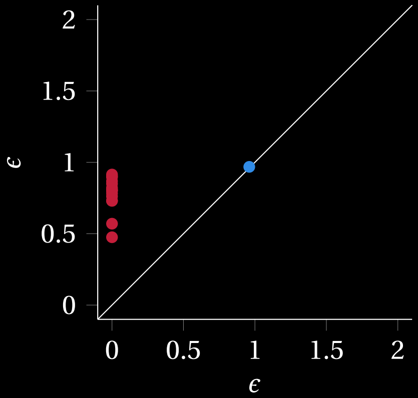


Given $\epsilon \in \mathbb{R}$, the Vietoris–Rips complex contains all simplices whose pairwise distance is less than or equal to ϵ. When using Euclidean balls of radius $r = 0.5\epsilon$, a simplex is created for each pairwise intersection.

Persistent homology

Vietoris–Rips complex calculation

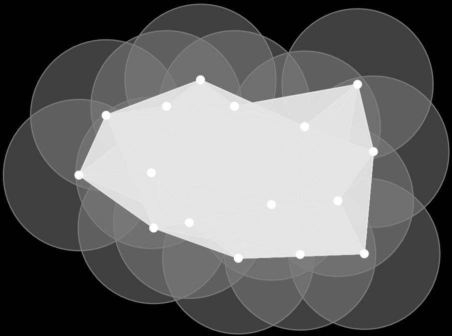


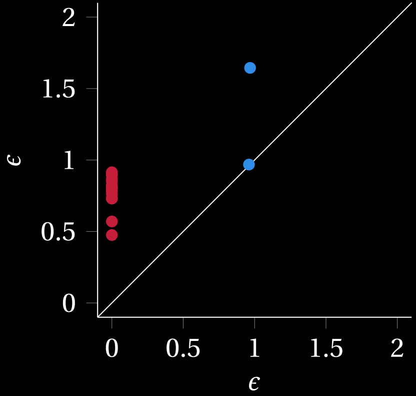


Given $\epsilon \in \mathbb{R}$, the Vietoris–Rips complex contains all simplices whose pairwise distance is less than or equal to ϵ. When using Euclidean balls of radius $r = 0.5\epsilon$, a simplex is created for each pairwise intersection.

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.

Motivation

So far, however, persistent homology is used in a ***passive manner****, meaning that the function* f *mapping simplices to* $\mathbb{R}$ *is* ***fixed and not informed by the learning task****.*[1]

[1] C. D. Hofer, F. Graf, **B. Rieck**, M. Niethammer and R. Kwitt, 'Graph Filtration Learning', *ICML*, ed. by H. Daumé III and A. Singh, Proceedings of Machine Learning Research 119, PMLR, 2020, pp. 4314–4323.

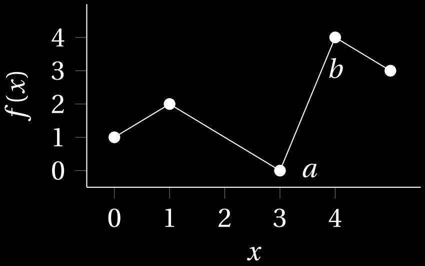
Making persistent homology differentiable

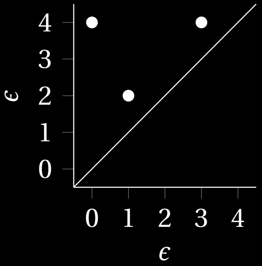
Terminology

- Let $f: \mathbb{M} \to \mathbb{R}$ be a function on a manifold. Persistent homology can be seen as a map from $(\mathbb{M}, f)$ 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

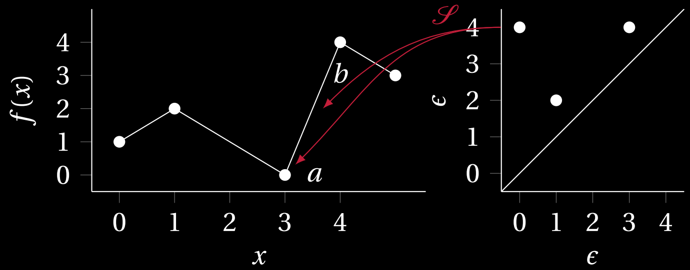
Example





Making persistent homology differentiable

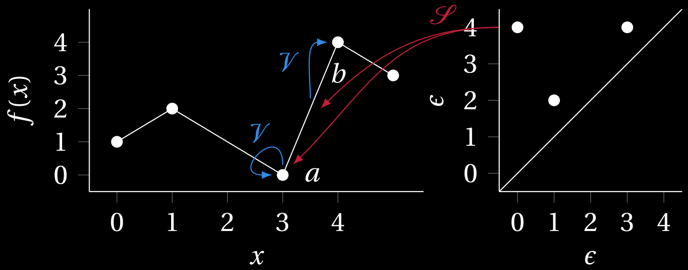
Example



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

Making persistent homology differentiable

Example

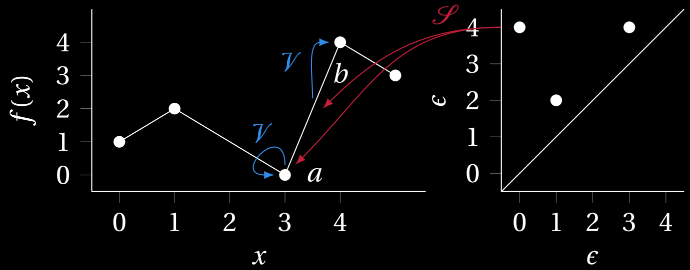


We have $\mathscr{S}(0,4) = (\{a\},\{a,b\})$.

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

Making persistent homology differentiable

Example



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

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

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

Making persistent homology differentiable

Gradient calculation sketch

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

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

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

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

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. Similar ideas occurred first in M. Gameiro, Y. Hiraoka and I. Obayashi, 'Continuation of point clouds via persistence diagrams', *Physica D: Nonlinear Phenomena* 334, 2016, pp. 118–132.

Extensions

Persistent homology calculations can be made differentiable and many general classes of topology-based optimisation schemes can be proven to converge!

M. Carrière, F. Chazal, M. Glisse, Y. Ike, H. Kannan and Y. Umeda, 'Optimizing persistent homology based functions', *ICML*, ed. by M. Meila and T. Zhang, Proceedings of Machine Learning Research 139, PMLR, 2021, pp. 1294–1303

Part I: Unstructured Data

Topological autoencoders

Topological Autoencoders

Michael Moor [†12] Max Horn [†12] Bastian Rieck [‡12] Karsten Borgwardt [‡12]

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 $X := \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^d$ 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 37[th] 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

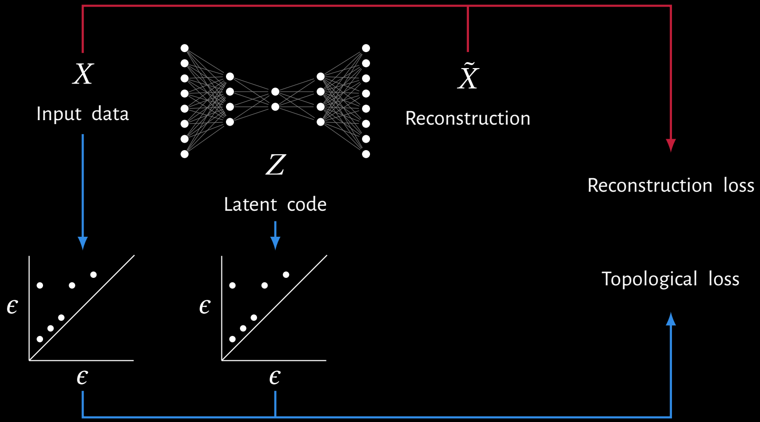
M. Moor*, M. Horn*, **B. Rieck**† and K. Borgwardt†, 'Topological Autoencoders', *ICML*, ed. by H. Daumé III and A. Singh, Proceedings of Machine Learning Research 119, PMLR, 2020, pp. 7045–7054

Topological autoencoders

Motivation

Topological autoencoders

Overview



Topological autoencoders

Gradient calculation intuition

Distance matrix $\mathbf{A}$

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

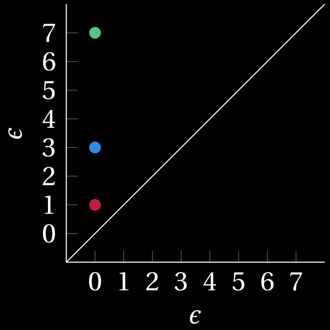
Every point in the persistence diagram can be mapped to *one* entry in the distance matrix! Each entry *is* a distance, so it can be changed during training (at least in the latent space).

Topological autoencoders

Gradient calculation intuition

Distance matrix $\mathbf{A}$

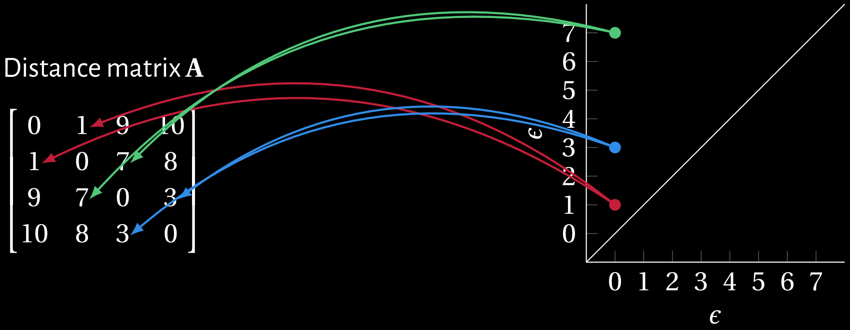
$$\begin{bmatrix} 0 & 1 & 9 & 10 \\ 1 & 0 & 7 & 8 \\ 9 & 7 & 0 & 3 \\ 10 & 8 & 3 & 0 \end{bmatrix}$$



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



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

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

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

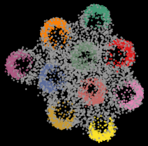
- $\mathcal{X}$: input space
- $\mathcal{Z}$: latent space
- $\mathbf{A}^X$: distances in input mini-batch
- $\mathbf{A}^Z$: distances in latent mini-batch
- π^X: persistence pairing of input mini-batch
- π^Z: persistence pairing of latent mini-batch

The loss is *bi-directional*!

Qualitative evaluation

'Spheres' data set

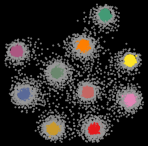
PCA



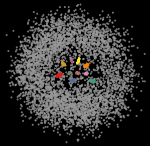
UMAP

Autoencoder

Isomap



t-SNE

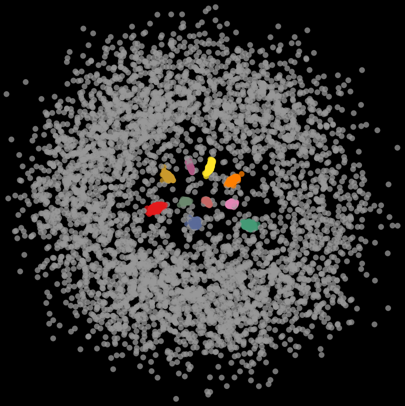


Topological autoencoder

Qualitative evaluation

'Spheres' data set; zooming in...

Autoencoder



Topological autoencoder

A new evaluation metric

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

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

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 $\mathrm{KL}_\sigma := \mathrm{KL}\left(f_\sigma^X \parallel f_\sigma^Z\right)$, which measures the similarity between the two density distributions.

Quantitative evaluation

Method	$KL_{0.01}$	$KL_{0.1}$	KL_{1}	ℓ-MRRE	ℓ-Cont	ℓ-Trust	ℓ-RMSE	MSE (data)
Isomap	0.181	*0.420*	*0.008 81*	*0.246*	*0.790*	*0.676*	10.4	
PCA	0.332	0.651	0.015 30	0.294	0.747	0.626	11.8	0.9610
t-SNE	*0.152*	0.527	0.012 71	**0.217**	0.773	**0.679**	**8.1**	
UMAP	0.157	0.613	0.016 58	0.250	0.752	0.635	*9.3*	
AE	0.566	0.746	0.016 64	0.349	0.607	0.588	13.3	**0.8155**
TopoAE	**0.085**	**0.326**	**0.006 94**	0.272	**0.822**	0.658	13.5	*0.8681*

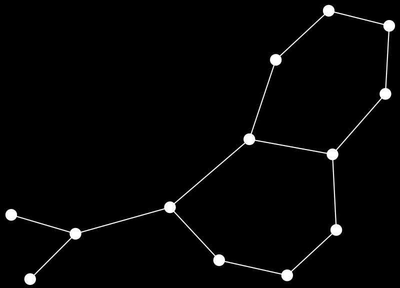
Flexibility of this loss term

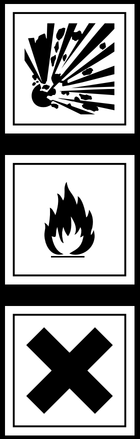
```
class TopologicalAutoencoder(torch.nn.Module):
  def __init__(self, model, lam=1.0):
      super().__init__()

      self.lam = lam
      self.model = model
      self.loss = SignatureLoss(p=2)
      self.vr = VietorisRipsComplex()

  def forward(self, x):
      z = self.model.encode(x)

      pi_x = self.vr(x)
      pi_z = self.vr(z)

      geom_loss = self.model(x)
      topo_loss = self.loss([x, pi_x], [z, pi_z])

      loss = geom_loss + self.lam * topo_loss
      return loss
```

Part II: Structured Data

Graph classification

Example





Potential labels

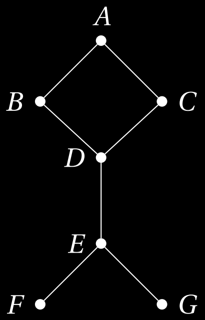
How to represent graphs?

- Two graphs G and G' can have a *different* number of vertices.
- Hence, we require a *vectorised representation* $f\colon \mathcal{G} \to \mathbb{R}^d$ of graphs.
- Such a representation f needs to be *permutation-invariant*.

Message passing

The predominant paradigm in graph machine learning

Neighbouring nodes can exchange *messages*. If this is *iterated*, messages can be 'diffused' to larger parts of the graph.

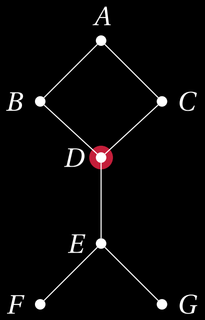


- Operations remain local.
- Message passing can be iterated.
- Need to define aggregation function.
- Representations can be combined.

Message passing

The predominant paradigm in graph machine learning

Neighbouring nodes can exchange *messages*. If this is *iterated*, messages can be 'diffused' to larger parts of the graph.

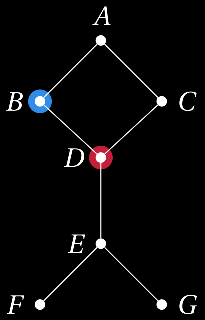


- Operations remain local.
- Message passing can be iterated.
- Need to define aggregation function.
- Representations can be combined.

Message passing

The predominant paradigm in graph machine learning

Neighbouring nodes can exchange *messages*. If this is *iterated*, messages can be 'diffused' to larger parts of the graph.

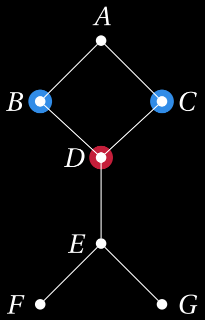


- Operations remain local.
- Message passing can be iterated.
- Need to define aggregation function.
- Representations can be combined.

Message passing

The predominant paradigm in graph machine learning

Neighbouring nodes can exchange *messages*. If this is *iterated*, messages can be 'diffused' to larger parts of the graph.

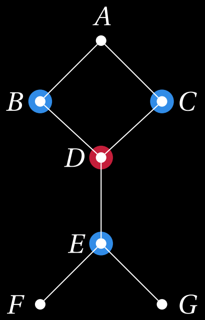


- Operations remain local.
- Message passing can be iterated.
- Need to define aggregation function.
- Representations can be combined.

Message passing

The predominant paradigm in graph machine learning

Neighbouring nodes can exchange *messages*. If this is *iterated*, messages can be ‘diffused’ to larger parts of the graph.

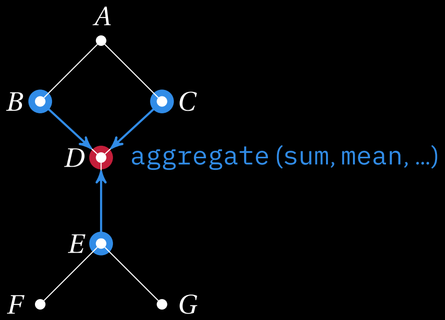


- Operations remain local.
- Message passing can be iterated.
- Need to define aggregation function.
- Representations can be combined.

Message passing

The predominant paradigm in graph machine learning

Neighbouring nodes can exchange *messages*. If this is *iterated*, messages can be 'diffused' to larger parts of the graph.



- Operations remain local.
- Message passing can be iterated.
- Need to define aggregation function.
- Representations can be combined.

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 procedure K times.

$$a_v^{(k)} := \texttt{aggregate}^{(k)}\left(\left\{h_u^{(k-1)} \mid u \in \mathcal{N}_{\mathrm{G}}(v)\right\}\right)$$

$$h_v^{(k)} := \texttt{combine}^{(k)}\left(h_v^{(k-1)}, a_v^{(k)}\right)$$

$$h_{\mathrm{G}} := \texttt{readout}\left(\left\{h_v^{(K)} \mid v \in \mathcal{V}_{\mathrm{G}}\right\}\right)$$

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

A topological layer for graph classification

M. Horn*, E. De Brouwer*, M. Moor, Y. Moreau, **B. Rieck**† and K. Borgwardt†, ‘Topological Graph Neural Networks’, *ICLR*, 2022

arXiv:2102.07835v2 [cs.LG] 3 Jun 2021

Topological Graph Neural Networks

Max Horn[1, 2, *] **Edward De Brouwer**[3, *] **Michael Moor**[1, 2] **Yves Moreau**[3]
Bastian Rieck[1, 2, *, †] **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 for graph and node classification tasks, both on synthetic data sets, which can be classified by humans using their topology 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 [50]. 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 [8]. These structures, however, are relevant for certain applications, such as the analysis of molecular graphs, whose classification necessitates knowledge about connectivity information [29, 46].

By contrast, methods based on topological features, commonly summarised under the 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. Figure 1 provides a motivational example that showcases the potential benefits of using topological information: high predictive performance is reached *earlier* for a *smaller* number of layers.

1

Max Horn
@ExpectationMax

Edward De Brouwer
@EdwardOnBrew

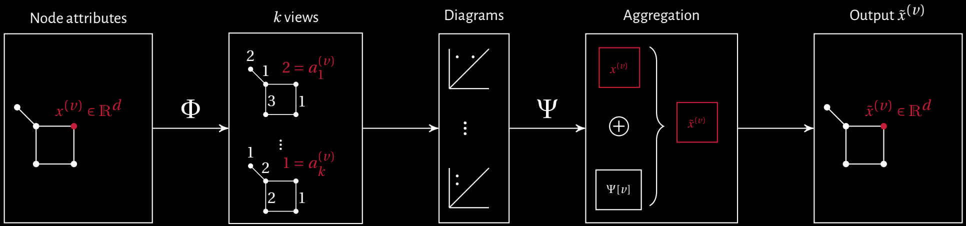
Michael Moor
@Michael_D_Moor

Yves Moreau

Karsten Borgwardt
@kmborgwardt

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.

Choosing Φ and Ψ

- The node map Φ can be realised using a *neural network*.
- The coordinatisation function Ψ can be realised using *any* vectorisation of persistence diagrams (landscapes, images, …), but we found a *differentiable coordinatisation function* to be most effective.[2]

[2] C. D. Hofer, F. Graf, **B. Rieck**, M. Niethammer and R. Kwitt, 'Graph Filtration Learning', *ICML*, ed. by H. Daumé III and A. Singh, Proceedings of Machine Learning Research 119, PMLR, 2020, pp. 4314–4323.

Expressivity of TOGL

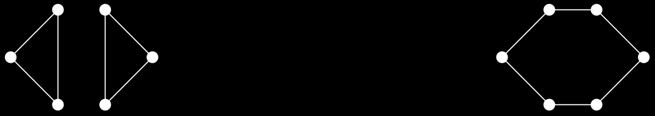
Context

Typical GNN architectures are *no more expressive* than the Weisfeiler–Lehman test for graph isomorphism, commonly abbreviated as WL[1].

Theorem

TOGL (and persistent homology) is ***more expressive*** *than WL[1], i.e. (i) if the WL[1] label sequences for two graphs G and G' diverge, there exists an injective filtration f such that the corresponding persistence diagrams $\mathscr{D}_0$ and $\mathscr{D}_0'$ are not equal, and (ii) there are graphs that WL[1] cannot distinguish but TOGL can!*

Example graphs



G

G′

Experiments

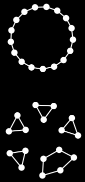
- Take existing GNN architecture.
- Replace one layer by TOGL.
- Measure predictive performance.

This strategy ensures that the number of parameters is approximately the same, thus facilitating a fair comparison!

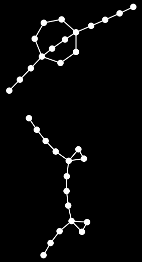
Synthetic data sets

Binary classification problem; generate same number of graphs for each of the classes. Use simple topological structures that are nevertheless challenging to detect with standard GNNs.

Cycles

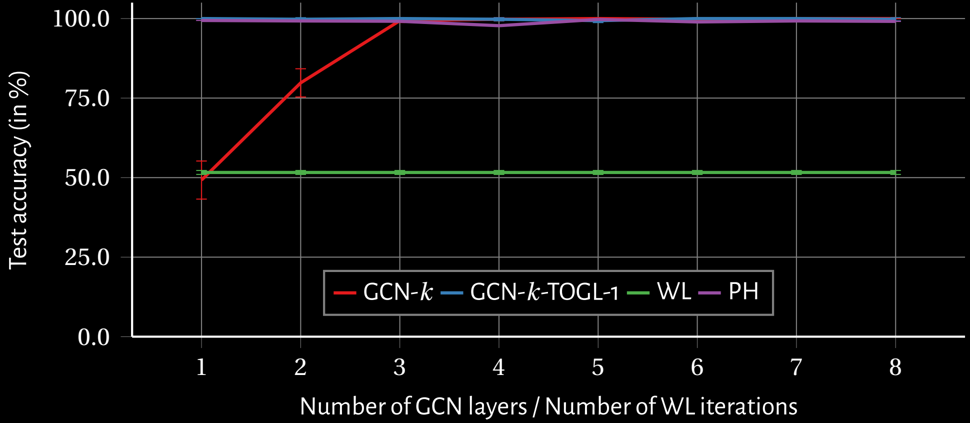


Necklaces



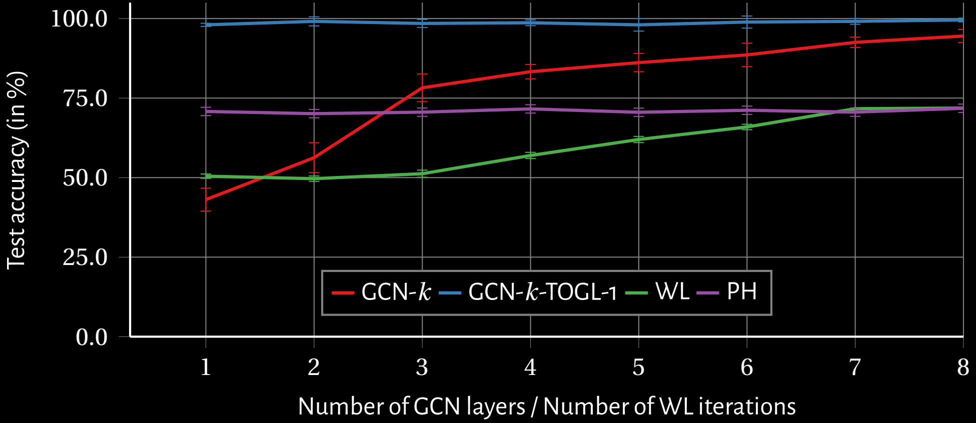
Expressivity

`Cycles` data set



Expressivity

`Necklaces` data set



Classifying graphs/nodes based on structural features alone

Existing data sets tend to 'leak' information into node attributes, thus decreasing the utility of topological features. Hence, we replaced all node features by random ones.

	Graph classification				*Node classification*
Method	DD	ENZYMES	MNIST	PROTEINS	Pattern
GCN-4	68.0±3.6	22.0±3.3	76.2±0.5	68.8±2.8	85.5±0.4
GCN-3-TOGL-1	**75.1±2.1**	**30.3±6.5**	**84.8±0.4**	**73.8±4.3**	**86.6±0.1**
GIN-4	75.6±2.8	21.3±6.5	83.4±0.9	**74.6±3.1**	84.8±0.0
GIN-3-TOGL-1	**76.2±2.4**	**23.7±6.9**	**84.4±1.1**	73.9±4.9	**86.7±0.1**
GAT-4	63.3±3.7	21.7±2.9	63.2±10.4	67.5±2.6	**73.1±1.9**
GAT-3-TOGL-1	**75.7±2.1**	**23.5±6.1**	**77.2±10.5**	**72.4±4.6**	59.6±3.3

Classifying benchmark data sets

While we improve baseline classification performance, the best performance is *not* driven by the availability of topological structures!

	Graph classification							Node classification
METHOD	CIFAR-10	DD	ENZYMES	MNIST	PROTEINS-full	IMDB-B	REDDIT-B	CLUSTER
GATED-GCN-4	**67.3±0.3**	72.9±2.1	65.7±4.9	**97.3±0.1**	**76.4±2.9**	—	—	**60.4±0.4**
WL	—	77.7±2.0	54.3±0.9	—	73.1±0.5	71.2±0.5	78.0±0.6	—
WL-OA	—	**77.8±1.2**	58.9±0.9	—	73.5±0.9	74.0±0.7	87.6±0.3	—
GCN-4	54.2±1.5	72.8±4.1	**65.8±4.6**	90.0±0.3	*76.1±2.4*	68.6±4.9	**92.8±1.7**	57.0±0.9
GCN-3-TOGL-1	*61.7±1.0*	*73.2±4.7*	53.0±9.2	95.5±0.2	76.0±3.9	72.0±2.3	89.4±2.2	60.4±0.2
	7.5	0.4	−12.8	5.5	−0.1	3.4	−3.4	3.4
GIN-4	54.8±1.4	70.8±3.8	*50.0±12.3*	*96.1±0.3*	72.3±3.3	72.8±2.5	81.7±6.9	58.5±0.1
GIN-3-TOGL-1	*61.3±0.4*	*75.2±4.2*	43.8±7.9	96.1±0.1	*73.6±4.8*	**74.2±4.2**	89.7±2.5	*60.4±0.2*
	6.5	4.4	−6.2	0.0	1.3	1.4	8.0	1.9
GAT-4	57.4±0.6	71.1±3.1	26.8±4.1	94.1±0.3	71.3±5.4	*73.2±4.1*	44.2±6.6	56.6±0.4
GAT-3-TOGL-1	*63.9±1.2*	*73.7±2.9*	*51.5±7.3*	*95.9±0.3*	*75.2±3.9*	70.8±8.0	*89.5±8.7*	*58.4±3.7*
	6.5	2.6	24.7	1.8	3.9	−2.4	45.3	1.8

Comparison with other topology-based methods

Using a very simple GCN with TOGL still exhibits favourable performance in comparison to other topology-based methods.

Method	REDDIT-5K	IMDB-MULTI	NCI1	REDDIT-B	IMDB-B
GFL	55.7±2.1	49.7±2.9	71.2±2.1	90.2±2.8	**74.5±4.6**
PersLay	55.6	48.8	73.5	—	71.2
GCN-1-TOGL-1	**56.1±1.8**	**52.0±4.0**	**75.8±1.8**	90.1±0.8	74.3±3.6

Conclusion

- 'If all you have is nails, everything looks like a hammer.'[3] Our data sets may actually *stymie* progress in GNN research because their classification does not necessarily require structural information.
- Nevertheless, higher-order structures (such as cliques) can be crucial in discerning between different graphs or data sets.
- Can we also learn sparse filtrations?
- Large untapped potential in topology-based optimisation methods!

[3] Credit: Mikael Vejdemo-Johannson

Sneak previews & acknowledgements

❤ Acknowledgements

My co-authors Edward, Karsten, Max, Michael, and Yves.

Do you like ML and Topology?

ICLR Workshop on Geometrical and Topological Representation Learning
`https://gt-rl.github.io`; deadline: Feb 25, AoE

Software

`https://github.com/aidos-lab/pytorch-topological`
Looking for additional contributors!