DSC 214
Topological Data Analysis
Topic 8: TDA + Machine Learning

Instructor: Zhengchao Wan

- We have seen different topological objects that can be potentially used for data analysis
- They can be used to potentially augment / strengthen machine learning approaches

- Today
  - Examples of how they connect or can be combined with ML pipelines.

- Graph Classification
  - Kernel methods
  - Weisfeiler-Lehman + persistent homology
- Deep Learning + TDA
  - Neural networks for handling PD
- Topological constraints / priors
  - Optimizing topological loss function

Section 1:
Graph classification

- Weighted persistence image kernel
- Persistent Weisfeiler–Lehman procedure
- …

Persistence-based Framework

- Persistence-based feature representation

Input data:
A collection of point sets, graphs, 2D/3D images, time-series , etc

→

Persistence feature space:
A collection of persistence-based feature representations (e.g, vectors)

→

Further data analysis:
Feed to data analysis pipelines, say SVM, CNN, etc

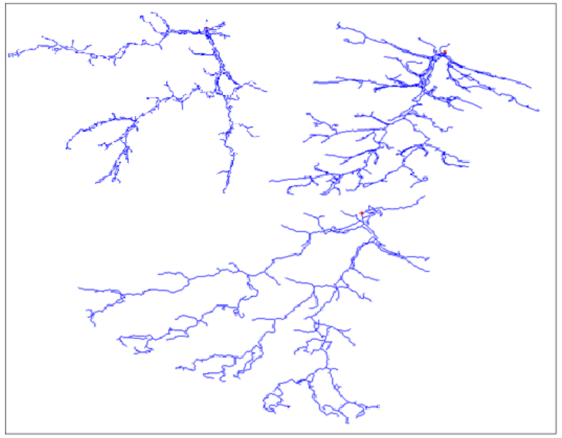
Persistence-based Framework

- Persistence-based feature representation

Input data:
A collection of point sets, graphs, 2D/3D images, time-series , etc

Persistence feature space:
A collection of persistence-based feature representations (e.g, vectors)

Further data analysis:
Feed to data analysis pipelines, say SVM, CNN, etc

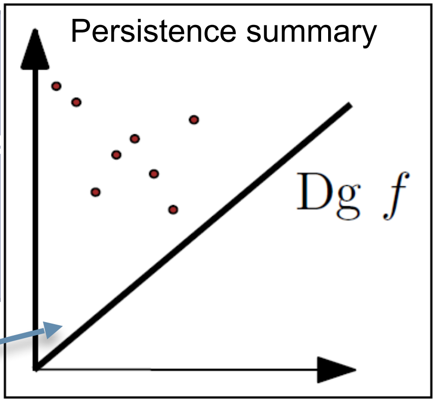


[Li, et al, W., PLOS One 2017]

Persistence-based Framework

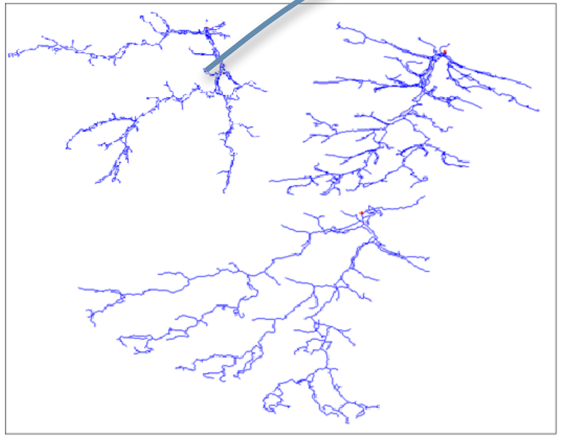
- Persistence-based feature representation

Input data:
A collection of point se
graphs, 2D/3D image
time-series , etc



nce feature
ace:
of persistence-
l feature
tations (e.g,
ctors)

Further data analysis:
Feed to data analysis pipelines, say SVM, CNN, etc

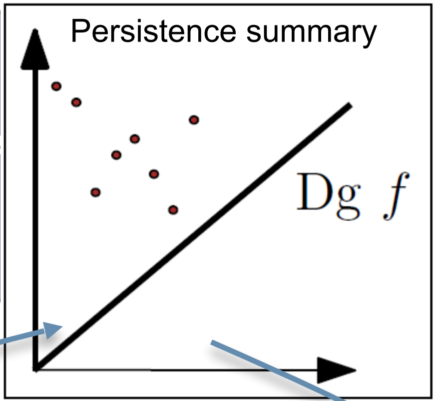


[Li, et al, W., PLOS One 2017]

Persistence-based Framework

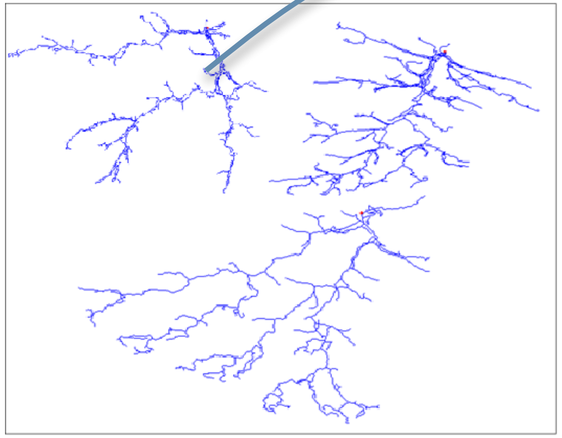
- Persistence-based feature representation

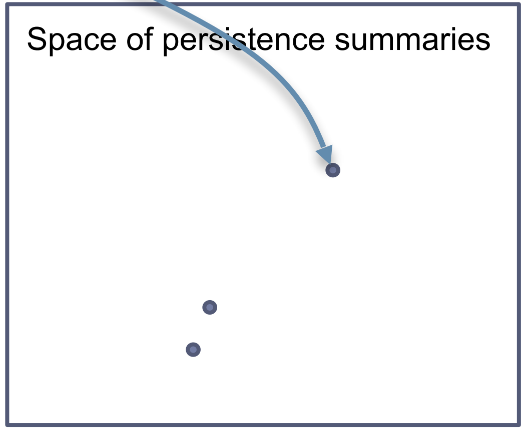
Input data:
A collection of point se
graphs, 2D/3D image
time-series , etc



nce feature
ace:
of persistence-
l feature
tations (e.g,
ctors)

Further data analysis:
Feed to data analysis pipelines, say SVM, CNN, etc

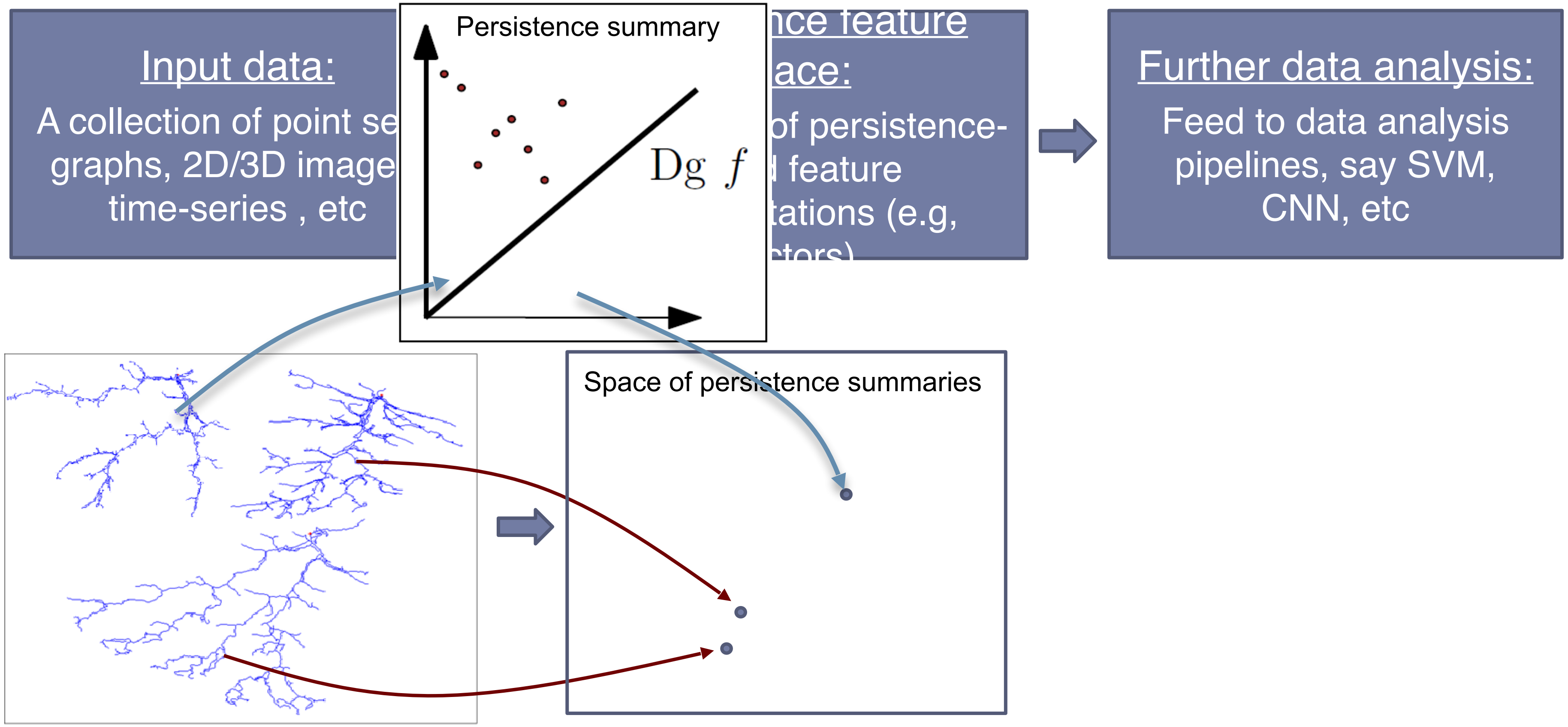




[Li, et al, W., PLOS One 2017]

Persistence-based Framework

- Persistence-based feature representation

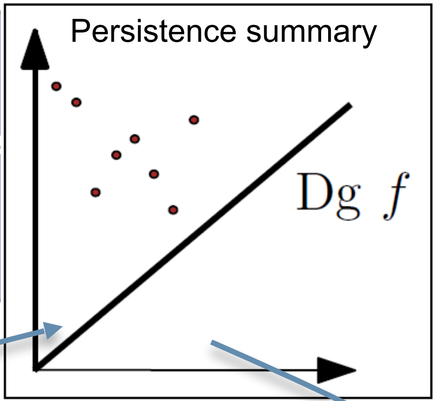


[Li, et al, W., PLOS One 2017]

Persistence-based Framework

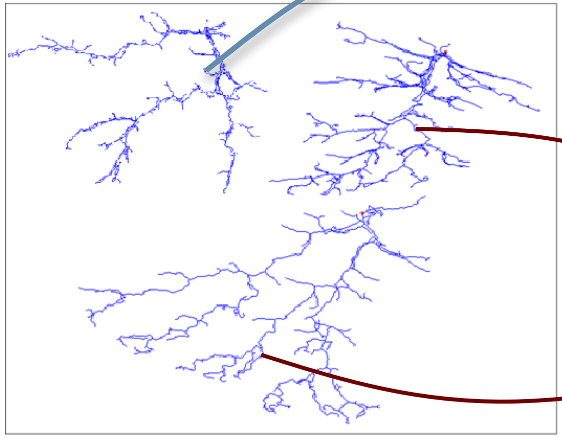
- Persistence-based feature representation

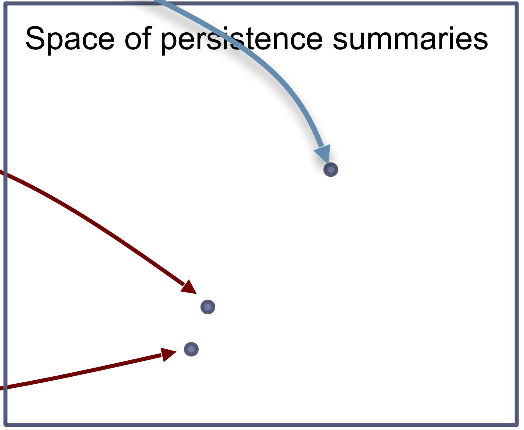
Input data:
A collection of point se
graphs, 2D/3D image
time-series , etc

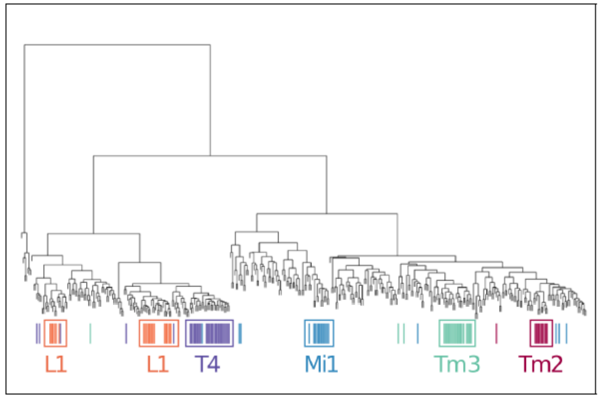


nce feature
ace:
of persistence-
l feature
tations (e.g,
ctors)

Further data analysis:
Feed to data analysis pipelines, say SVM, CNN, etc







[Li, et al, W., PLOS One 2017]

Persistence feature representation

- Persistence landscapes
  - [Bubenik 2012]
- Persistence scale space kernel
  - [Reininghause et al., 2014]
- Persistence images
  - [Adams et al., 2015, 2017]
- Persistence weighted Gaussian kernel
  - [Kusano et al., 2017]
- Sliced Wasserstein kernel
  - [Carriere et al., 2017]
- Persistence Fisher kernel
  - [Le and Yamada 2018]
- ....

Persistence feature representation

- Persistence landscapes
  - [Bubenik 2012]
- Persistence scale space kernel
  - [Reininghause et al., 2014]
- Persistence images
  - [Adams et al., 2015, 2017]
- Persistence weighted Gaussian kernel
  - [Kusano et al., 2017]
- Sliced Wasserstein kernel
  - [Carriere et al., 2017]
- Persistence Fisher kernel
  - [Le and Yamada 2018]
- ....

We can then feed such persistence summaries into kernel-SVM etc.

Learning metrics for persistence-based summaries and applications for graph classification

Qi Zhao
zhao.2017@osu.edu

Yusu Wang
yusu@cse.ohio-state.edu

Computer Science and Engineering Department
The Ohio State University
Columbus, OH 43221

Abstract

Recently a new feature representation framework based on a topological tool called persistent homology (and its persistence diagram summary) has gained much momentum. A series of methods have been developed to map a persistence diagram to a vector representation so as to facilitate the downstream use of machine learning tools. In these approaches, the importance (weight) of different persistence features are usually *pre-set*. However often in practice, the choice of the weight-function should depend on the nature of the specific data at hand. It is thus highly desirable to *learn* a best weight-function (and thus metric for persistence diagrams) from labelled data. We study this problem and develop a new weighted kernel, called *WKPI*, for persistence summaries, as well as an optimization framework to learn the weight (and thus kernel). We apply the learned kernel to the challenging task of graph classification, and show that our WKPI-based classification framework obtains similar or (sometimes significantly) better results than **the best results** from a range of previous graph classification frameworks on benchmark datasets.

1 Introduction

In recent years a new data analysis methodology based on a topological tool called persistent homology has started to attract momentum. The persistent homology is one of the most important developments in the field of topological data analysis, and there have been fundamental developments both on the theoretical front (e.g, [23, 10, 13, 8, 14, 5]), and on algorithms / implementations (e.g, [43, 4, 15, 20, 29, 3]). On the high level, given a domain X with a function $f : X \to \mathbb{R}$ on it, the persistent homology summarizes "features" of X across multiple scales simultaneously in a single summary called the *persistence diagram* (see the second picture in Figure 1). A persistence diagram consists of a multiset of points in the plane, where each point $p = (b, d)$ intuitively corresponds to the birth-time (b) and death-time (d) of some (topological) features of X w.r.t. f. Hence it provides a concise representation of X, capturing *multi-scale features* of it simultaneously. Furthermore, the persistent homology framework can be applied to complex data (e.g, 3D shapes, or graphs), and different summaries could be constructed by putting different descriptor functions on input data.

Due to these reasons, a new persistence-based feature vectorization and data analysis framework (Figure 1) has become popular. Specifically, given a collection of objects, say a set of graphs modeling chemical compounds, one can first convert each shape to a persistence-based representation. The input data can now be viewed as a set of points in a persistence-based feature space. Equipping this space with appropriate distance or kernel, one can then perform downstream data analysis tasks (e.g, clustering).

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

- An extension of persistence image
- A learnable weight function
- Suitable for graph classification

Recall: Persistence Images

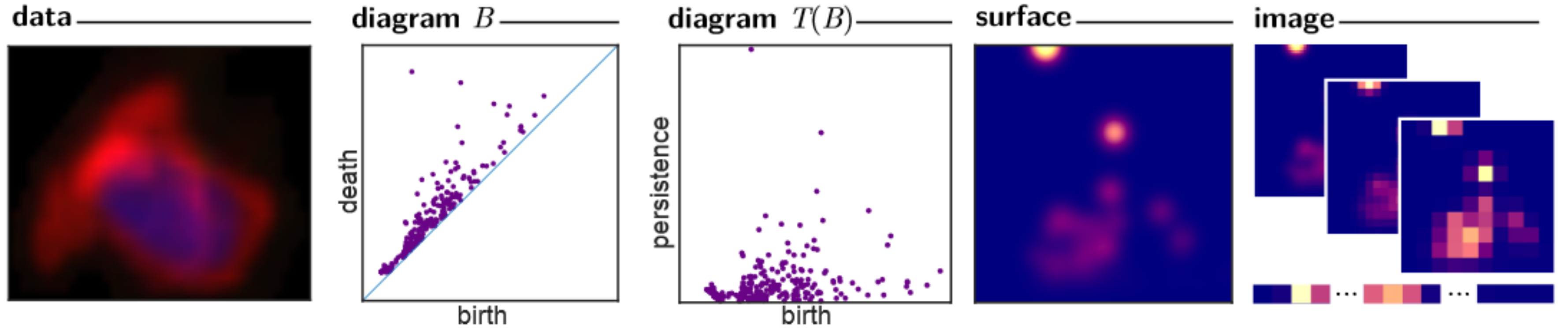


Figure 1: Algorithm pipeline to transform data into a persistence image.

Image taken from [Adams et al., 2017]

- Given a persistence diagram B
  - $T : \mathbb{R}^2 \to \mathbb{R}^2$ such that $T(x, y) = (x, y - x)$
  - Persistence surface $\rho_B : R^2 \to R$ where
    - $\rho_B(z) = \sum_{u \in T(B)} f(u) \cdot \phi_u(z)$
  - Persistence image:
    - I_B: a discretization of ρ_B: $I_B[p] = \int_p \rho_B dxdy$

Weighted persistence image kernel

Definition 3.1 *Let* $\omega : \mathbb{R}^2 \rightarrow \mathbb{R}$ *be a weight-function. Given two persistence images* PI *and* PI$'$, *the* (ω-)weighted persistence image kernel (WKPI) *is defined as:* $k_w(\mathrm{PI}, \mathrm{PI}') := \sum_{s=1}^{N} \omega(p_s) e^{-\frac{(\mathrm{PI}(s) - \mathrm{PI}'(s))^2}{2\sigma^2}}$.

- This is positive semi-definite and induces a WKPI distance

$$\mathrm{D}_\omega(A, B) := \sqrt{k_w(\mathrm{PI}_A, \mathrm{PI}_A) + k_w(\mathrm{PI}_B, \mathrm{PI}_B) - 2k_w(\mathrm{PI}_A, \mathrm{PI}_B)}.$$

- The weight ω can be learned through a metric learning approach
- The learned weighted persistence image kernel is combined with SVM for classification

Optimization problem for metric learning

- Given a collection of objects $\{X_1, \ldots, X_n\}$ already classified to k classes $C_1, \ldots, C_k$, let $\{A_1, \ldots, A_n\}$ denote the corresponding persistence diagrams
- For any $t \in \{1, 2, \ldots, k\}$

$$cost_\omega(t,t) = \sum_{i,j \in \mathcal{C}_t} \mathrm{D}_\omega{}^2(A_i, A_j); \quad \text{and} \quad cost_\omega(t,\cdot) = \sum_{i \in \mathcal{C}_t, j \in \{1,2,\cdots,n\}} \mathrm{D}_\omega{}^2(A_i, A_j).$$

Definition 3.5 (Optimization problem) *Given a weight-function $\omega : \mathbb{R}^2 \to \mathbb{R}$, the* total-cost *of its induced WKPI-distance over Ξ is defined as: $TC(\omega) := \sum_{t=1}^{k} \frac{cost(t,t)}{cost(t,\cdot)}$. The* optimal distance problem *aims to find the best weight-function ω^* from a certain function class $\mathcal{F}$ so that the total-cost is minimized; that is: $TC^* = \min_{\omega \in \mathcal{F}} TC(\omega)$; and $\omega^* = \mathrm{argmin}_{\omega \in \mathcal{F}} TC(\omega)$.*

WKPI framework

- Given labeled training data, learn optimal ω^* through the metric learning framework

- Perform classification task using ω^*-WKPI + SVM

Graph classification

- Benchmark datasets:

Datasets	graph #	class #	average_nodes #	average edges #	label #
MUTAG	188	2	17.93	19.79	7
PTC	344	2	14.29	14.69	19
ENZYME	600	6	32.63	64.14	3
PROTEIN	1113	2	39.06	72.82	3
DD	1178	2	284.32	715.66	81
NCI1	4110	2	29.87	32.30	37
IMDB BINARY	1000	2	19.77	96.53	-
IMDB MULTI	1500	3	13.00	65.94	-
REDDIT BINARY	2000	2	429.63	497.75	-
REDDIT 5K	4999	5	508.82	594.87	-
REDDIT 12K	12929	11	391.41	456.89	-

Statistics of benchmark graph data sets

Notes, some are attributed graphs.

Graph classification, cont.

- Comparison methods:
  - Weisfeiler-Lehman Kernel (WL) [Shervashidze et al., 2011]
  - Graphlet kernel (GK) [Shervashidze et al, 2009]
  - Deep Graph Kernel (DGK) [Yanardag and Vishwanathan, 2015]
  - FGSD [Verma and Zhang, 2017]
  - RetGK [Zhang et al., 2018]

  - Graph isomorphism network (GIN) [Xu et al., 2018]
  - PATCHYSAN (PSCN) [Niepert et al., 2016]

Graph classification, cont.

- Classification accuracy + variance
  - by SVM + a metric-learning approach by [Zhao, W., 2019]
  - 10 times 10-fold cross validation

Dataset	RetGK	WL	GK	DGK	PSCN	GIN	WKPI-kM	WKPI-kC
NCI1	84.5±0.2	85.4±0.3	62.3±0.3	80.3±0.5	76.3±1.7	82.7±1.6	**87.2±0.4**	84.7±0.4
NCI109	-	84.5±0.2	66.6±0.2	80.3±0.3	-	-	85.6±0.3	**87.3±0.3**
PTC	62.5±1.6	55.4±1.5	57.3±1.1	60.1±2.5	62.3±5.7	66.6±6.9	63.1±2.4	**67.1±2.2**
PROTEIN	75.8±0.6	71.2±0.8	71.7±0.6	75.7±0.5	75.0±2.5	76.2±2.6	**78.8±0.4**	74.9±0.3
DD	81.6±0.3	78.6±0.4	78.5±0.3	-	76.2±2.6	-	**82.0±0.5**	80.3±0.4
MUTAG	**90.3±1.1**	84.4±1.5	81.6±2.1	87.4±2.7	89.0±4.4	90.0±8.8	86.9±2.5	87.5±2.6
IMDB-BINARY	71.9±1.0	70.8±0.5	65.9±1.0	67.0±0.6	71.0±2.3	75.1±5.1	70.7±1.1	**75.4±1.1**
IMDB-MULTI	47.7±0.3	49.8±0.5	43.9±0.4	44.6±0.4	45.2±2.8	**52.3 ±2.8**	46.4±0.5	49.5±0.4
REDDIT-5K	56.1±0.5	51.2±0.3	41.0±0.2	41.3±0.2	49.1±0.7	57.5±1.5	58.5±0.4	**60.2±0.6**
REDDIT-12K	**48.7±0.2**	32.6±0.3	31.8±0.1	32.2±0.1	41.3±0.4	-	47.7±0.5	48.6±0.5

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:

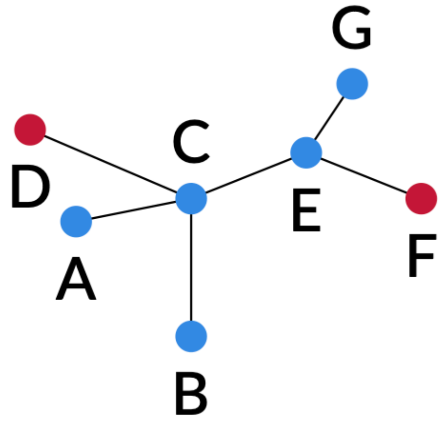
- We measure the relevance of topological features (connected components and cycles) in graphs and use them to define a novel set of WL subtree features, which we show to be a generalised version of the original ones.
- We develop a topology-based kernel that uses an iterative variant of the WL stabilisation 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.

- Classic WL test provides a procedure to vectorize (labeled) graphs
- Persistent homology encodes topological info into the above procedure

Weisfeiler-Lehman iteration



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

Weisfeiler-Lehman graph isomorphism test

- Example

- Bad example

Weisfeiler-Lehman subtree kernel

- [Shervashidze et al. 2011]
- Given two labeled graphs G and G', the WL iteration generates a sequence of labeled graphs $(G_0, G'_0), \ldots, (G_h, G'_h)$
- Let l_u^h denote the label of vertex u in iteration h
- List and sort all distinct l_u^h into a list $\{ l_0, \ldots, \}$
- Let c^h denote the counting function

$$\phi_{\mathrm{WL}}^{(h)} := \left[\mathrm{c}^{(h)}(l_0), \mathrm{c}^{(h)}(l_1), \ldots \right]$$

$$k_{WLsubtree}^{(h)}(G, G') = \langle \phi_{WLsubtree}^{(h)}(G), \phi_{WLsubtree}^{(h)}(G') \rangle,$$

Filtration via WL iteration

- At iteration h
- $dist(v_i, v_j) := [l_{v_i}^{h-1} \neq l_{v_j}^{h-1}] + dist(\{\{l_w^{h-1} : w \sim v_i\}\}, \{\{l_w^{h-1} : w \sim v_j\}\}) + \tau$
- This turns any labeled graph into a weighted graph! Then one can compute persistent homology
- 0-dim PD: $(0{,}w_1), \ldots, (0{,}w_k)$
- 1-dim PD: $(w_1', \infty), \ldots, (w_l', \infty)$
- $Pers(a_i, a_j) := |a_j - a_i|$ or a_i if $a_j = \infty$

Persistence features based on WL iteration

Connected components

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

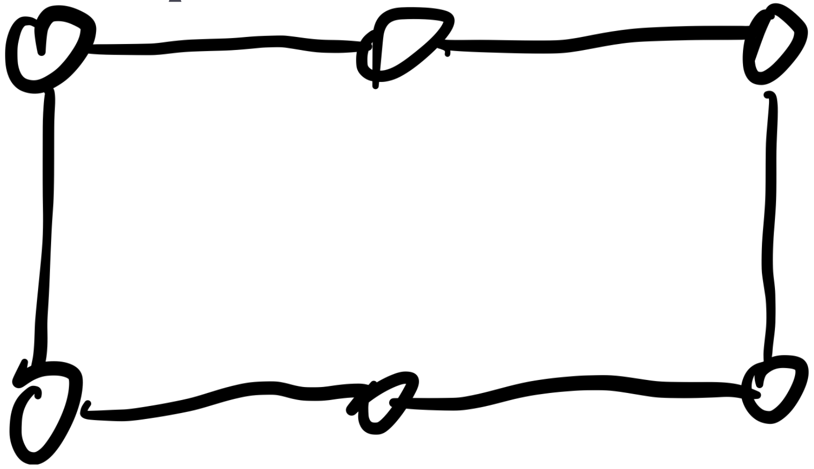
$$\mathfrak{p}^{(h)}(l_i) := \sum_{\mathrm{l}(v)=l_i} \operatorname{pers}(v)^p,$$

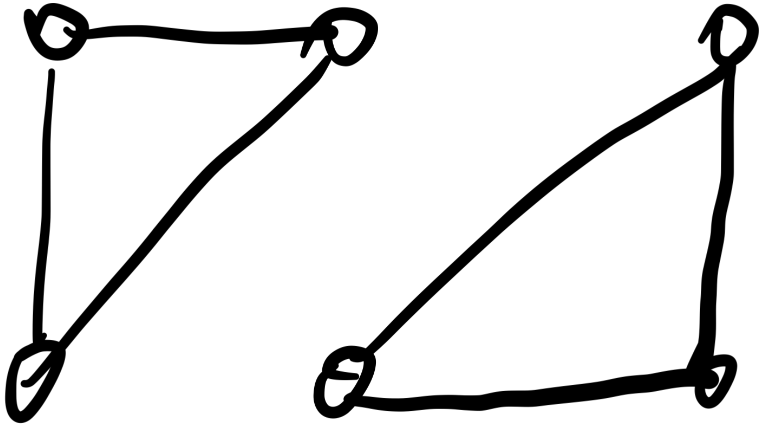
Cycles

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

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

Example





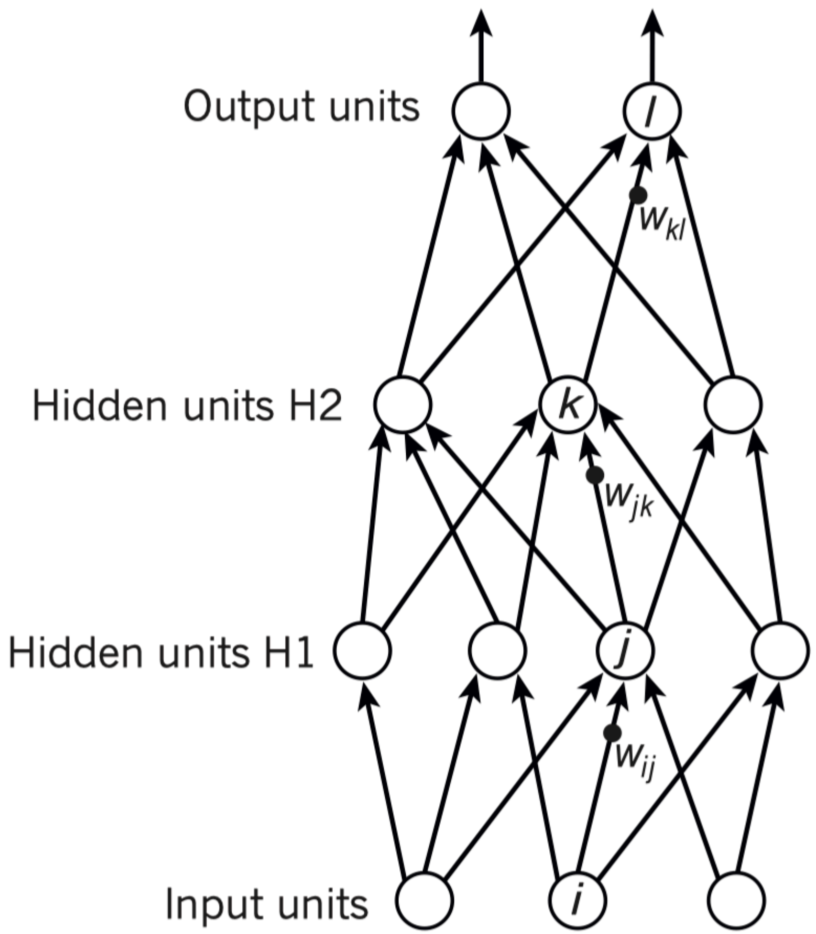
- $dist(v_i, v_j) = \tau$
- 0-dim PD for G: 5* $(0,\tau)$ and 1 $(0,\infty)$; for G':4* $(0,\tau)$ and 2* $(0,\infty)$
- 1-dim PD for G: (τ, ∞); for G': 2* (τ, ∞)
- $\Phi_{P-WL}(G) = 5\tau^p$ and $\Phi_{P-WL}(G') = 4\tau^p$
- $\Phi_{P-WL-C}(G) = \tau^p$ and $\Phi_{P-WL-C}(G') = 2\tau^p$

Classification results

	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

Section 2:
TDA + Deep Learning

Neural network



$$y_l = f(z_l)$$

$$z_l = \sum_{k \,\varepsilon\, \mathrm{H2}} w_{kl} y_k$$

$$y_k = f(z_k)$$

$$z_k = \sum_{j \,\varepsilon\, \mathrm{H1}} w_{jk} y_j$$

$$y_j = f(z_j)$$

$$z_j = \sum_{i \,\varepsilon\, \mathrm{Input}} w_{ij} x_i$$

LeCun et al. 2015

[Hofer et al., 2017]

Deep Learning with Topological Signatures

Christoph Hofer
Department of Computer Science
University of Salzburg, Austria
chofer@cosy.sbg.ac.at

Roland Kwitt
Department of Computer Science
University of Salzburg, Austria
Roland.Kwitt@sbg.ac.at

Marc Niethammer
UNC Chapel Hill, NC, USA
mn@cs.unc.edu

Andreas Uhl
Department of Computer Science
University of Salzburg, Austria
uhl@cosy.sbg.ac.at

Abstract

Inferring topological and geometrical information from data can offer an alternative perspective on machine learning problems. Methods from topological data analysis, e.g., persistent homology, enable us to obtain such information, typically in the form of summary representations of topological features. However, such topological signatures often come with an unusual structure (e.g., multisets of intervals) that is highly impractical for most machine learning techniques. While many strategies have been proposed to map these topological signatures into machine learning compatible representations, they suffer from being agnostic to the target learning task. In contrast, we propose a technique that enables us to input topological signatures to deep neural networks and *learn* a task-optimal representation during training. Our approach is realized as a novel input layer with favorable theoretical properties. Classification experiments on 2D object shapes and social network graphs demonstrate the versatility of the approach and, in case of the latter, we even outperform the state-of-the-art by a large margin.

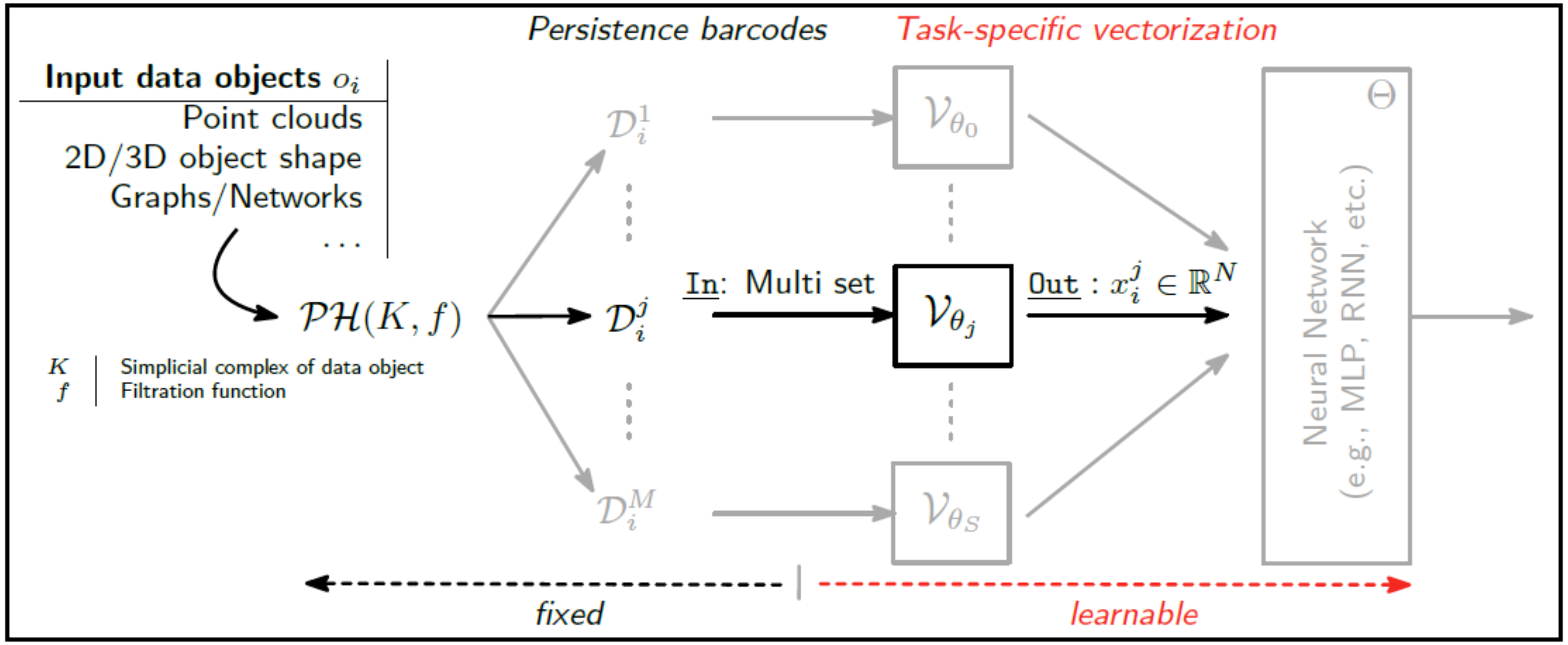
1 Introduction

Methods from algebraic topology have only recently emerged in the machine learning community, most prominently under the term *topological data analysis (TDA)* [7]. Since TDA enables us to infer relevant topological and geometrical information from data, it can offer a novel and potentially beneficial perspective on various machine learning problems. Two compelling benefits of TDA are (1) its versatility, i.e., we are not restricted to any particular kind of data (such as images, sensor measurements, time-series, graphs, etc.) and (2) its robustness to noise. Several works have demonstrated that TDA can be beneficial in a diverse set of problems, such as studying the manifold of natural image patches [8], analyzing activity patterns of the visual cortex [28], classification of 3D surface meshes [27, 22], clustering [11], or recognition of 2D object shapes [29].

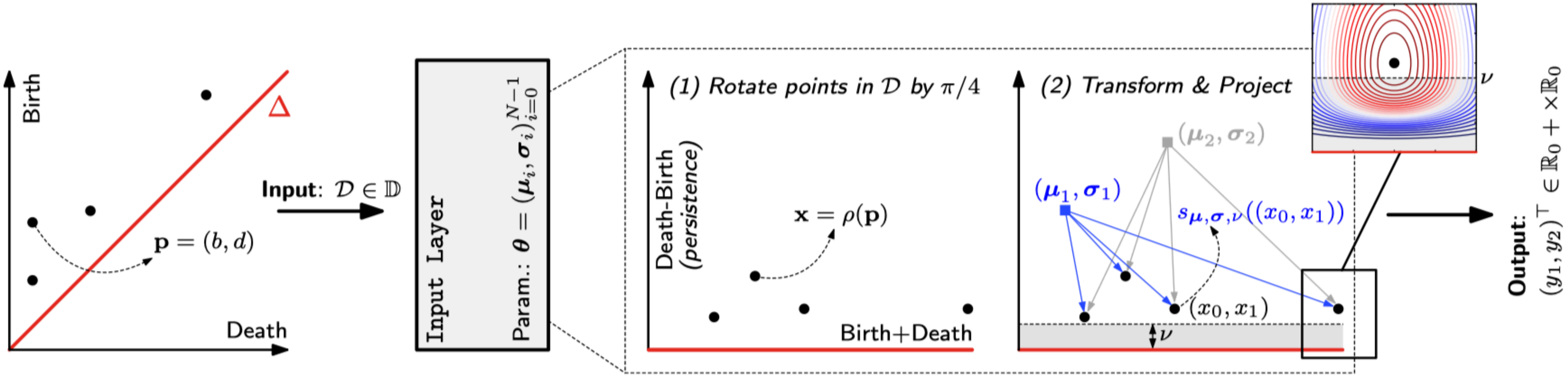
Currently, the most widely-used tool from TDA is *persistent homology* [15, 14]. Essentially[1], persistent homology allows us to track topological changes as we analyze data at multiple "scales". As the scale changes, topological features (such as connected components, holes, etc.) appear and disappear. Persistent homology associates a *lifespan* to these features in the form of a *birth* and a *death* time. The collection of (birth, death) tuples forms a multiset that can be visualized as a persistence diagram or a barcode, also referred to as a *topological signature* of the data. However, leveraging these signatures for learning purposes poses considerable challenges, mostly due to their

[1] We will make these concepts more concrete in Sec. 2.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.



Zoom in to the topological layer



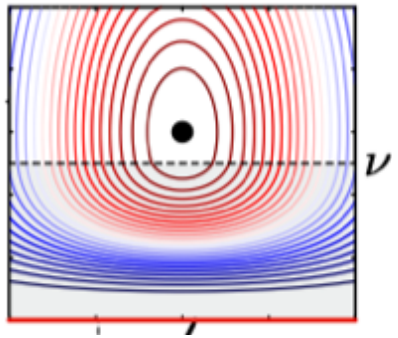
A layer for topological features

Definition 3. *Let* $\boldsymbol{\mu} = (\mu_0, \mu_1)^\top \in \mathbb{R} \times \mathbb{R}^+, \boldsymbol{\sigma} = (\sigma_0, \sigma_1) \in \mathbb{R}^+ \times \mathbb{R}^+$ *and* $\nu \in \mathbb{R}^+$. *We define*

$$s_{\boldsymbol{\mu},\boldsymbol{\sigma},\nu} : \mathbb{R} \times \mathbb{R}_0^+ \to \mathbb{R}$$

as follows:

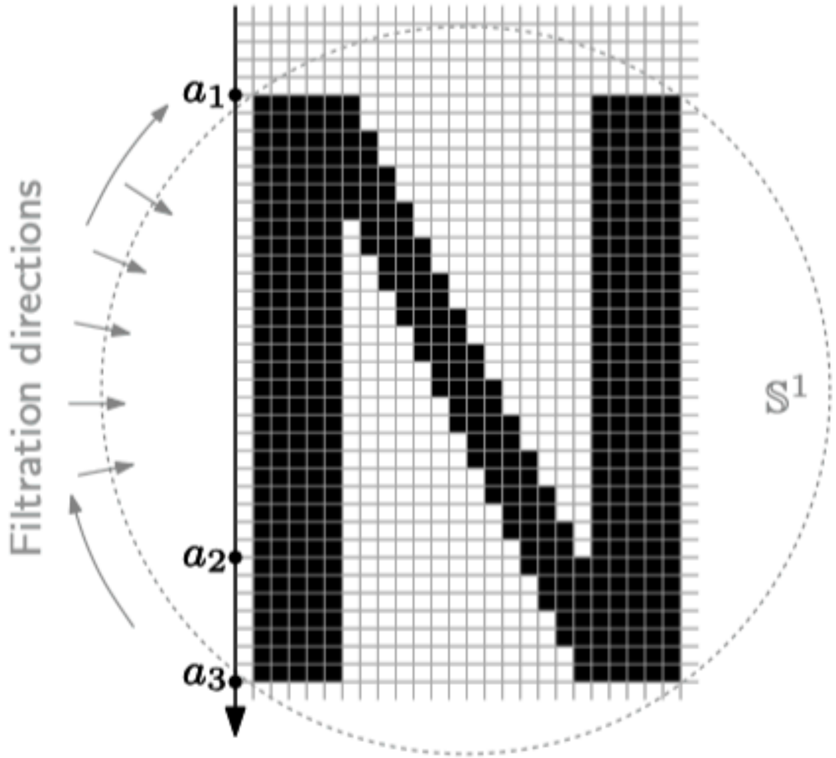
$$s_{\boldsymbol{\mu},\boldsymbol{\sigma},\nu}\big((x_0, x_1)\big) = \begin{cases} e^{-\sigma_0^2(x_0-\mu_0)^2-\sigma_1^2(x_1-\mu_1)^2}, & x_1 \in [\nu, \infty) \\ e^{-\sigma_0^2(x_0-\mu_0)^2-\sigma_1^2(\ln(\frac{x_1}{\nu})+\nu-\mu_1)^2}, & x_1 \in (0, \nu) \\ 0, & x_1 = 0 \end{cases}$$

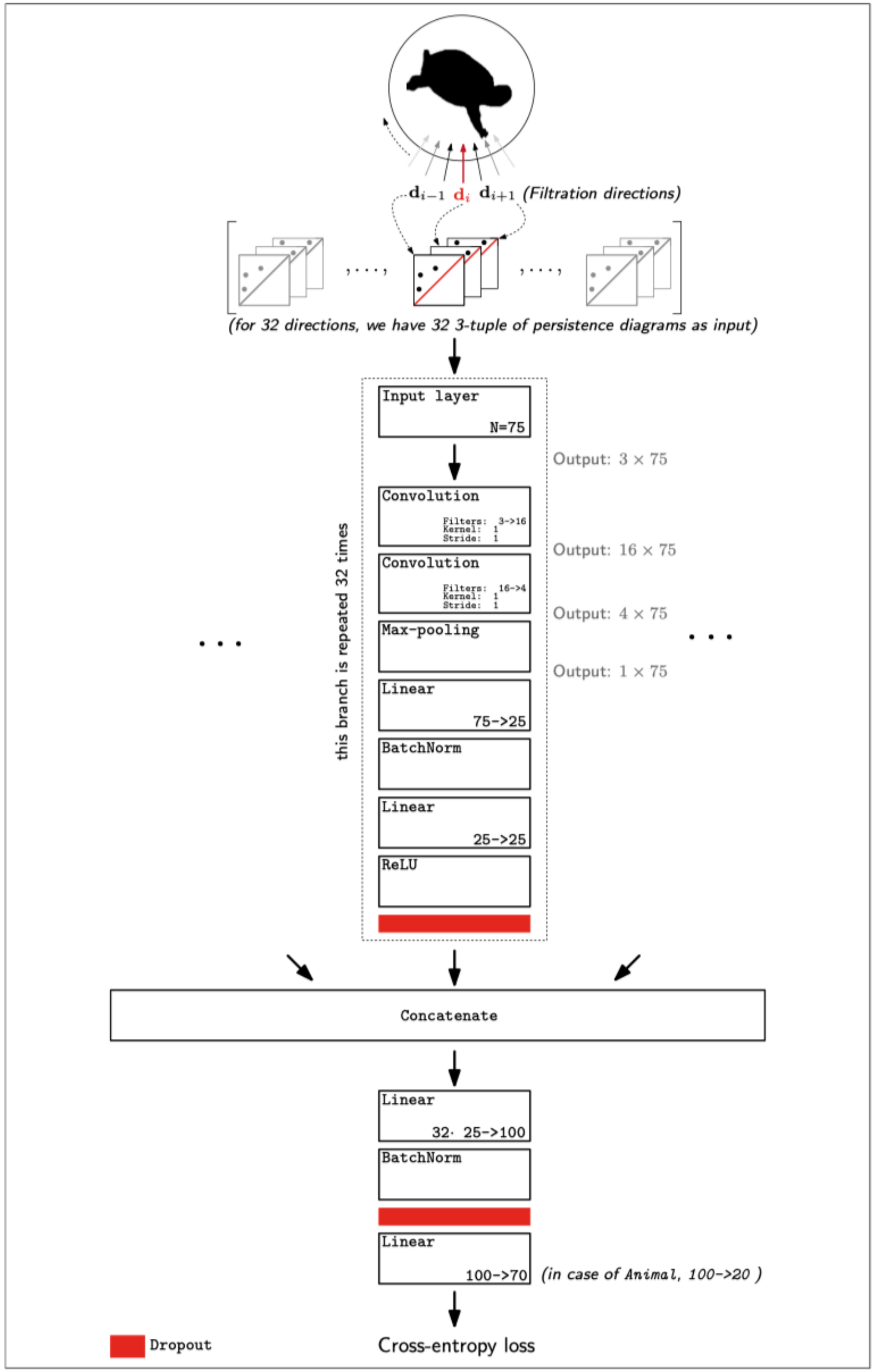


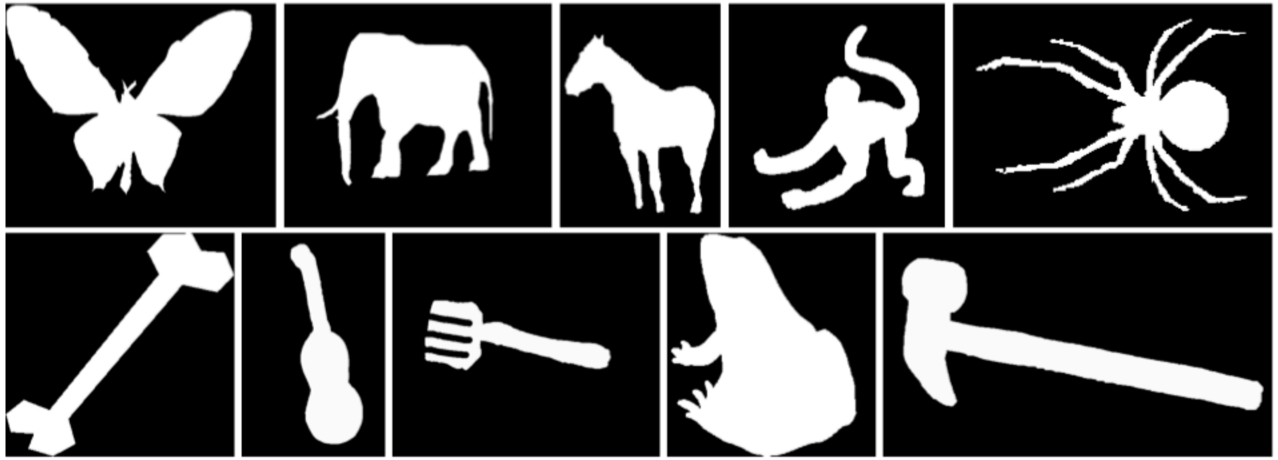
A persistence diagram $\mathcal{D}$ *is then projected w.r.t.* $s_{\boldsymbol{\mu},\boldsymbol{\sigma},\nu}$ *via*

$$S_{\boldsymbol{\mu},\boldsymbol{\sigma},\nu} : \mathbb{D} \to \mathbb{R}, \qquad \mathcal{D} \mapsto \sum_{\mathbf{x} \in \mathcal{D}} s_{\boldsymbol{\mu},\boldsymbol{\sigma},\nu}(\rho(\mathbf{x})) \ . \tag{4}$$

A classification scheme





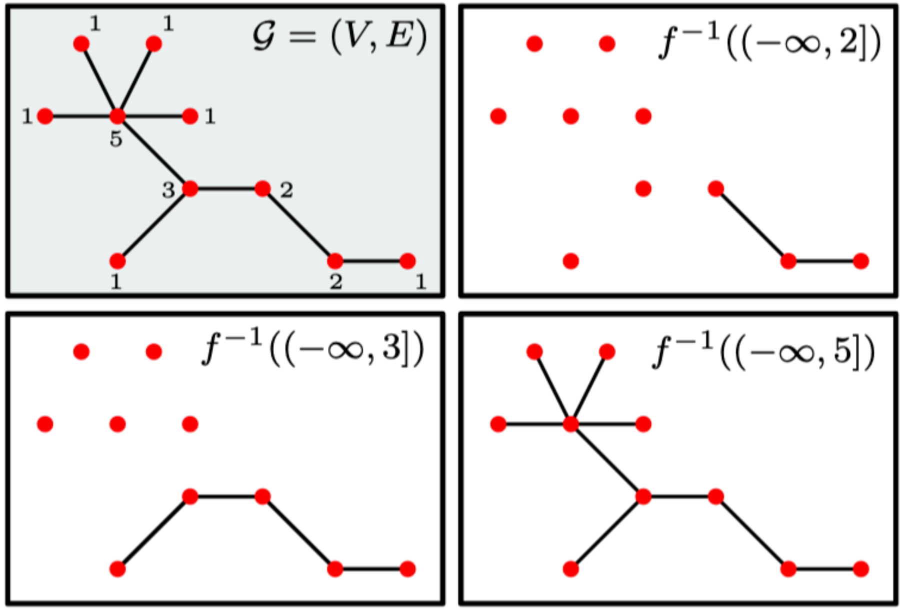


	MPEG-7	Animal
‡Skeleton paths	86.7	67.9
‡Class segment sets	90.9	69.7
†ICS	96.6	78.4
†BCF	97.2	83.4
Ours	91.8	69.5

Figure 3: *Left*: some examples from the `MPEG-7` (*bottom*) and `Animal` (*top*) datasets. *Right*: Classification results, compared to the two best (†) and two worst (‡) results reported in [30].

- Graph classification on social network data

	reddit-5k	reddit-12k
GK [31]	41.0	31.8
DGK [31]	41.3	32.2
PSCN [24]	49.1	41.3
RF [4]	50.9	42.7
Ours (w/o essential)	49.1	38.5
Ours (w/ essential)	**54.5**	**44.5**



PersLay

- [Carriere et al., 2020]
- Given a persistence diagram $D \in \mathfrak{D}$,
  - $Perslay(D) := \text{op}\left(\left\{ \omega(p) \cdot \phi(p) \right\}_{p \in D} \right)$
    - where $\omega: R^2 \to R$ is a weight function for points in persistence diagram, and
    - $\phi: R^2 \to R^k$ is a point transformation function to map each persistent point to a k-vector,
    - and op is a permutation invariant set function
      - e.g, max, sum, k-th largest value, etc
- Essentially is a modified *DeepSet* architecture

PersLay: A Neural Network Layer for Persistence Diagrams and New Graph Topological Signatures

Mathieu Carrière
Columbia University

Frédéric Chazal
Inria Saclay

Yuichi Ike
Fujitsu Laboratories

Théo Lacombe
Inria Saclay

Martin Royer
Inria Saclay

Yuhei Umeda
Fujitsu Laboratories

Abstract

Persistence diagrams, the most common descriptors of Topological Data Analysis, encode topological properties of data and have already proved pivotal in many different applications of data science. However, since the metric space of persistence diagrams is not Hilbert, they end up being difficult inputs for most Machine Learning techniques. To address this concern, several vectorization methods have been put forward that embed persistence diagrams into either finite-dimensional Euclidean space or implicit infinite dimensional Hilbert space with kernels.

In this work, we focus on persistence diagrams built on top of graphs. Relying on extended persistence theory and the so-called heat kernel signature, we show how graphs can be encoded by (extended) persistence diagrams in a provably stable way. We then propose a general and versatile framework for learning vectorizations of persistence diagrams, which encompasses most of the vectorization techniques used in the literature. We finally showcase the experimental strength of our setup by achieving competitive scores on classification tasks on real-life graph datasets.

1 INTRODUCTION

Topological Data Analysis (TDA) is a field of data science whose goal is to detect and encode topological features (such as connected components, loops, cavities...) that are present in datasets in order to improve inference and prediction. Its main descriptor is the so-called *persistence diagram*, which takes the form of a set of points in the Euclidean plane $\mathbb{R}^2$, each point corresponding to a topological feature of the data, with its coordinates encoding the feature size. This descriptor has been successfully used in many different applications of data science, such as signal analysis (Perea & Harer, 2015), material science (Buchet et al., 2018), cellular data (Cámara, 2017), or shape recognition (Li et al., 2014) to name a few. This wide range of applications is mainly due to the fact that persistence diagrams encode information based on topology, and as such this information is very often complementary to the one retrieved by more classical descriptors.

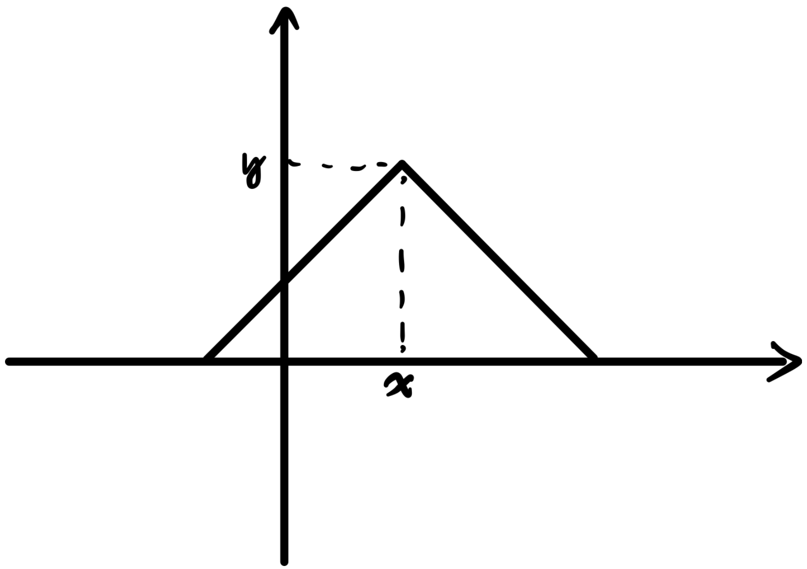
However, the space of persistence diagrams heavily lacks structure: different persistence diagrams may have different number of points, and several basic operations are not well-defined, such as addition and scalar multiplication, which unfortunately dramatically impedes their use in machine learning applications. To handle this issue, a lot of attention has been devoted to *vectorizations* of persistence diagrams through the construction of either *finite-dimensional embeddings* (Adams et al., 2017; Carrière et al., 2015; Chazal et al., 2015; Kališnik, 2018), i.e., embeddings turning persistence diagrams into vectors in Euclidean space $\mathbb{R}^d$, or *kernels* (Bubenik, 2015; Carrière et al., 2017; Kusano et al., 2016; Le & Yamada, 2018; Reininghaus et al., 2015), i.e., generalized scalar products that implicitly turn persistence diagrams into elements of infinite-dimensional Hilbert spaces.

Even though these methods improved the use of persistence diagrams in machine learning tremendously, several issues remain. For instance, most of these vectorizations only have a few trainable parameters, which may prevent them from fitting well to specific appli-

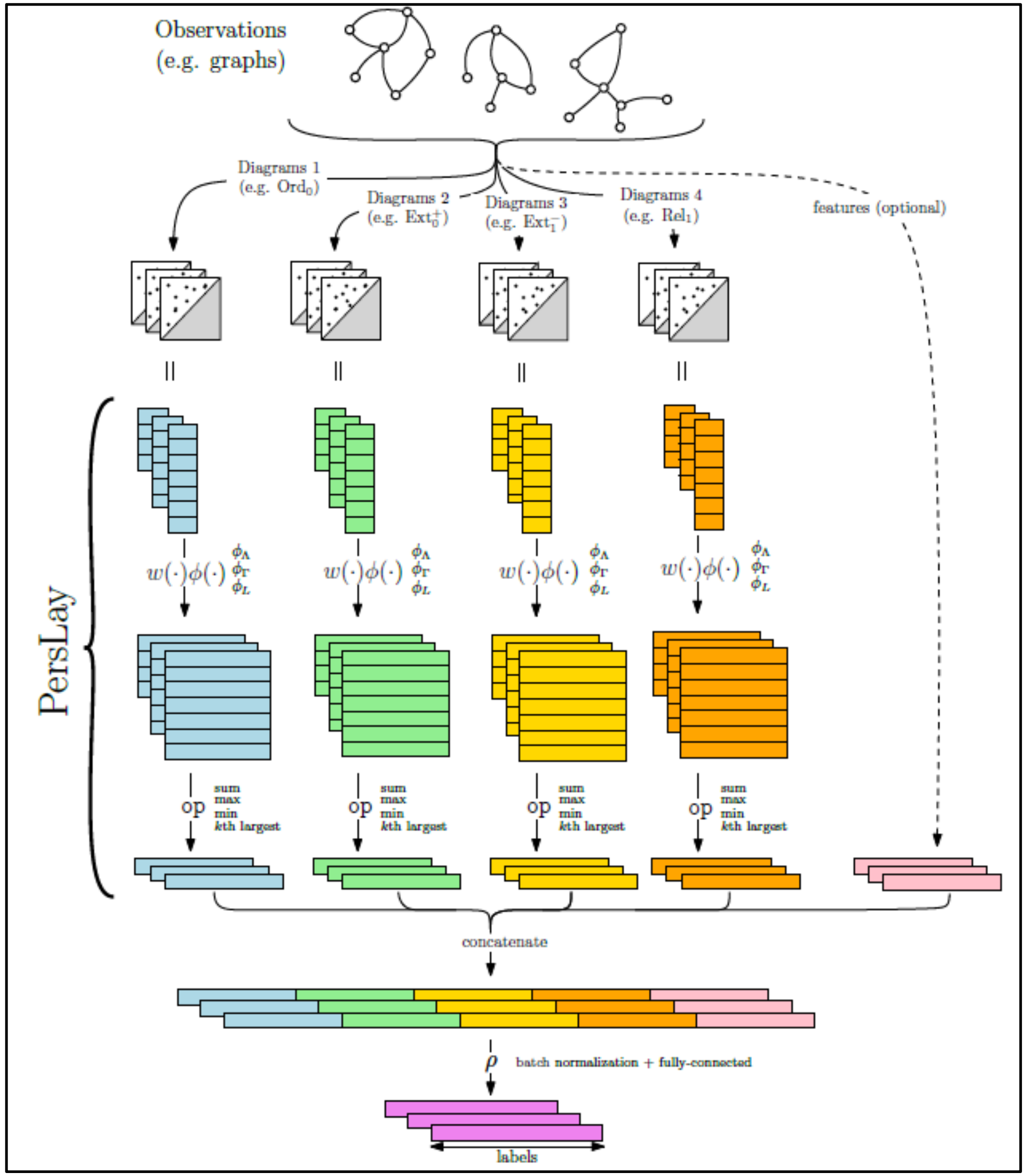
Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108. Copyright 2020 by the author(s).

Examples of ϕ

- The *triangle point transformation* $\phi_\Lambda : \mathbb{R}^2 \to \mathbb{R}^q, p \mapsto \left[\Lambda_p(t_1), \Lambda_p(t_2), \ldots, \Lambda_p(t_q)\right]^T$ where the triangle function Λ_p associated to a point $p = (x, y) \in \mathbb{R}^2$ is $\Lambda_p \colon t \mapsto \max\{0, y - |t - x|\}$, with $q \in \mathbb{N}$ and $t_1, \ldots, t_q \in \mathbb{R}$.
- The *Gaussian point transformation* $\phi_\Gamma : \mathbb{R}^2 \to \mathbb{R}^q, p \mapsto \left[\Gamma_p(t_1), \Gamma_p(t_2), \ldots, \Gamma_p(t_q)\right]^T$, where the Gaussian function Γ_p associated to a point $p = (x, y) \in \mathbb{R}^2$ is $\Gamma_p \colon t \mapsto \exp\left(-\|p - t\|_2^2/(2\sigma^2)\right)$ for a given $\sigma > 0$, $q \in \mathbb{N}$ and $t_1, \ldots, t_q \in \mathbb{R}^2$.
- The *line point transformation* $\phi_L \colon \mathbb{R}^2 \to \mathbb{R}^q, p \mapsto \left[L_{\Delta_1}(p), L_{\Delta_2}(p), \ldots, L_{\Delta_q}(p)\right]^T$, where the line function L_Δ associated to a line Δ with direction vector $e_\Delta \in \mathbb{R}^2$ and bias $b_\Delta \in \mathbb{R}$ is $L_\Delta : p \mapsto \langle p, e_\Delta \rangle + b_\Delta$, with $q \in \mathbb{N}$ and $\Delta_1, \ldots, \Delta_q$ are q lines in the plane.

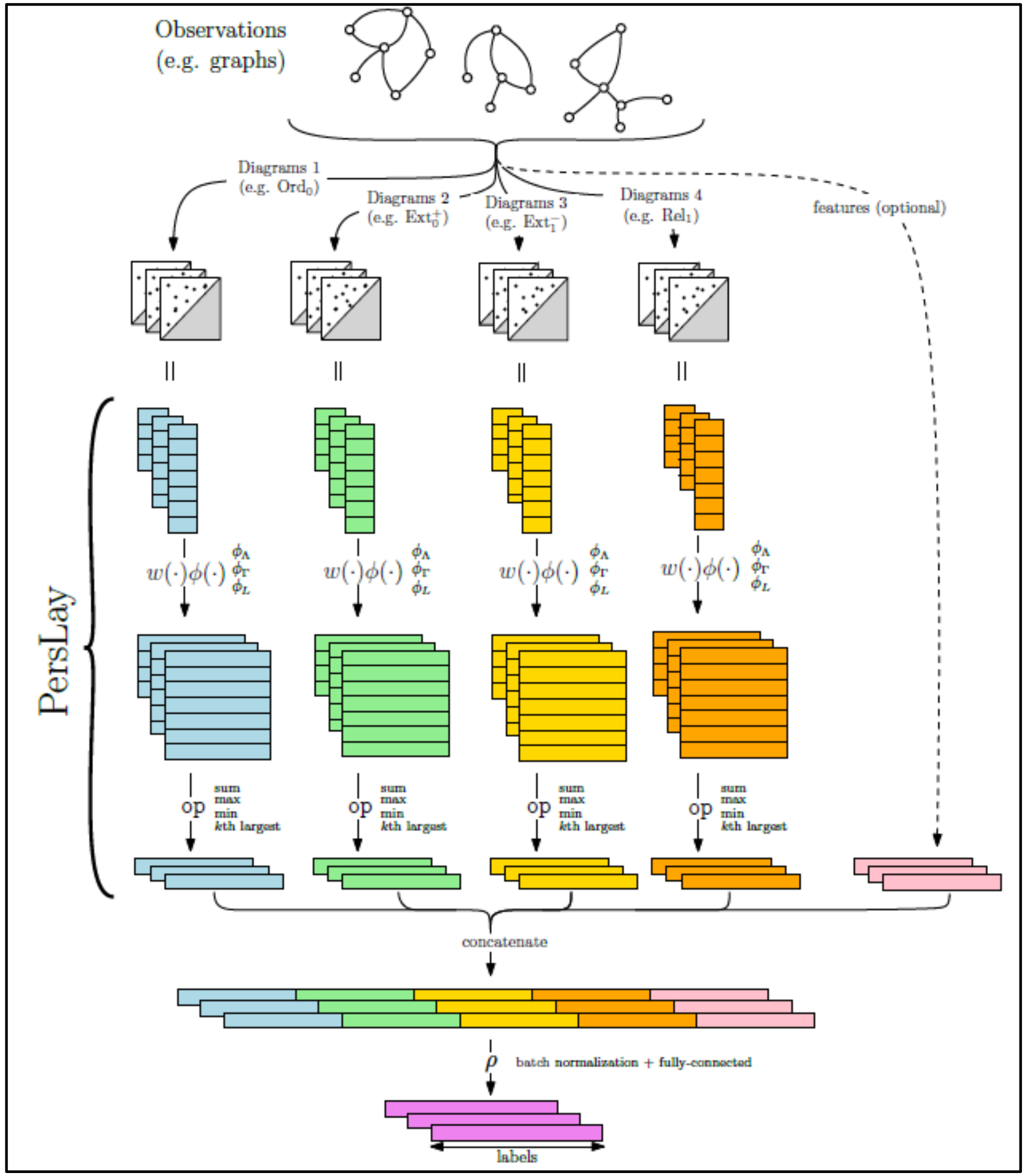


- PersLay is a general framework
- By choosing specific ω and ϕ, can recover most other feature vectorization,
  - persistence landscapes
  - persistence images
  - Hofer's network structure



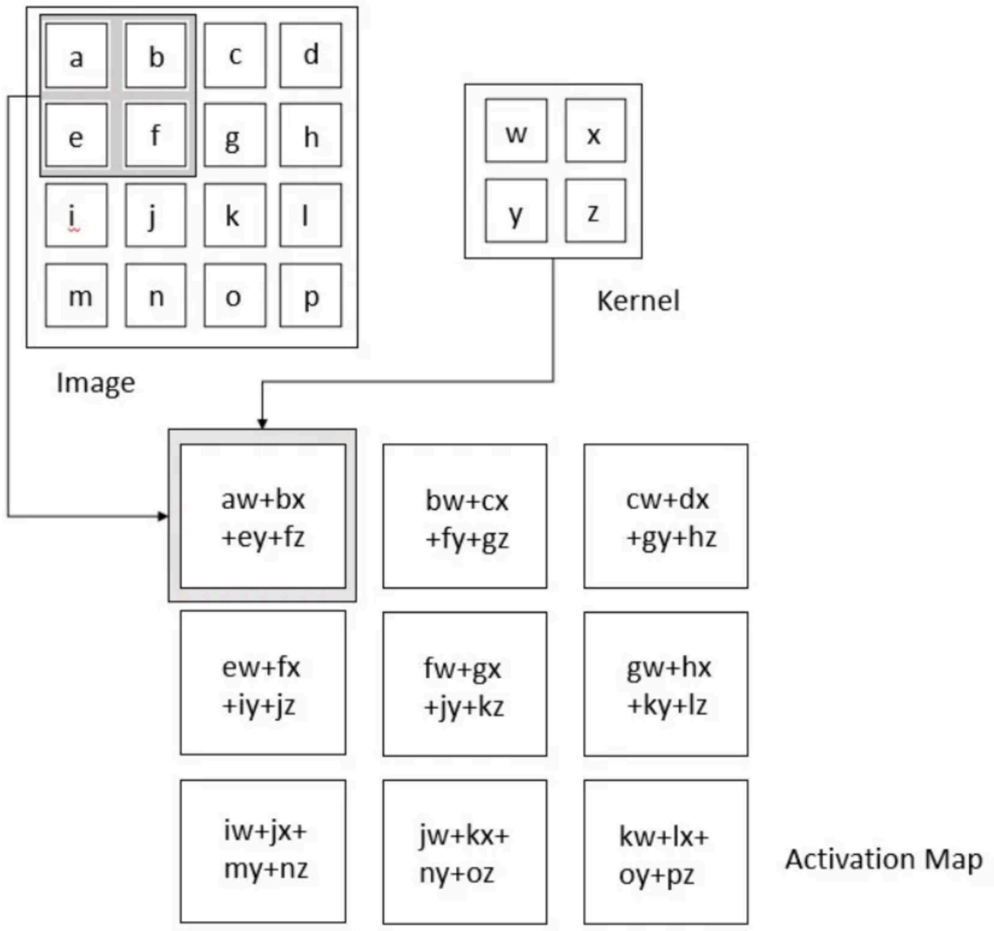
- PersLay is a general framework
- By choosing specific ω and ϕ, can recover most other feature vectorization,
  - persistence landscapes

- Using: $\phi = \phi_\Lambda$ with samples $t_1, \ldots, t_q \in \mathbb{R}$, $\texttt{op} = k$th largest value, $w = 1$ (a constant weight function), amounts to evaluating the kth *persistence landscape* (Bubenik, 2015) on $t_1, \ldots, t_q \in \mathbb{R}$.

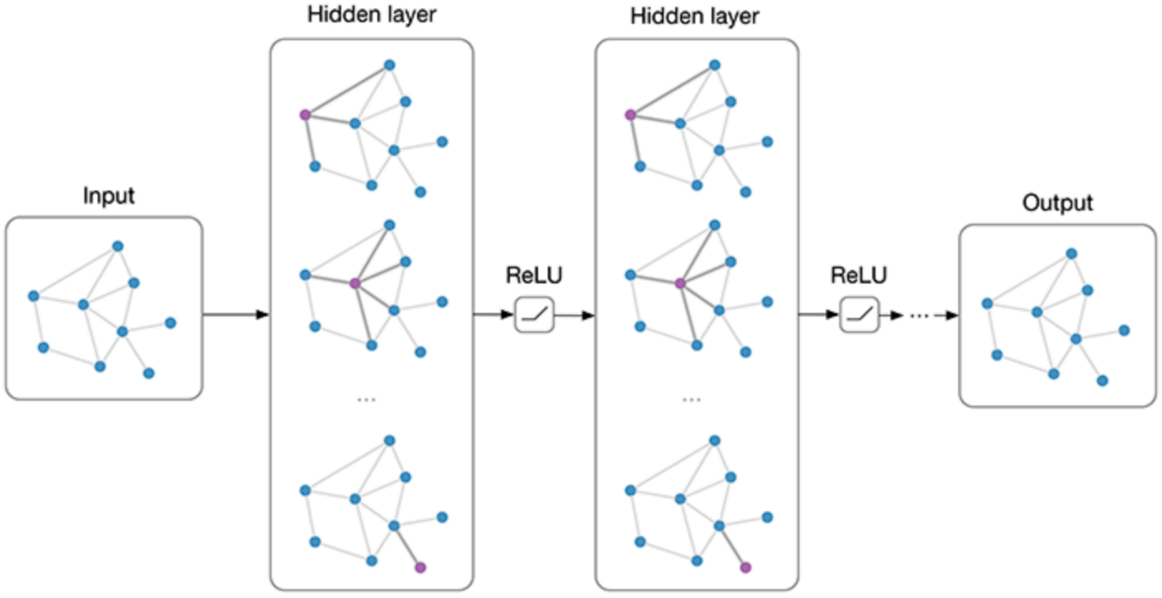


Graph neural networks

Graph neural network



Courtesy of Goodfellow et al.



Courtesy of Thomas Kipf

Graph neural network

- A convolutional layer

$$\vec{h}_u^{\ell} = \sigma\left(\sum_{v \in \overline{\mathcal{N}}(u)} W^{\ell} \vec{h}_v^{\ell-1}\right).$$

- Discriminative power controlled by the WL test (Xu et al. 2019)

Persistence Enhanced Graph Neural Network

Qi Zhao* Ze Ye† Chao Chen† Yusu Wang*
* Department of Computer Science and Engineering, The Ohio State University
†Department of Biomedical Informatics, Stony Brook University

Abstract

Local structural information can increase the adaptability of graph convolutional networks to large graphs with heterogeneous topology. Existing methods only use relatively simple topological information, such as node degrees. We present a novel approach leveraging advanced topological information, i.e., persistent homology, which measures the information flow efficiency at different parts of the graph. To fully exploit such structural information in real world graphs, we propose a new network architecture which learns to use persistent homology information to reweight messages passed between graph nodes during convolution. For node classification tasks, our network outperforms existing ones on a broad spectrum of graph benchmarks.

1 INTRODUCTION

Deep learning methods have achieved immense success in different domains such as computer vision and natural language processing (Goodfellow et al., 2016). While deep neural networks have shown strong performance on image or text data, their learning power is yet to be fully exploited on graph-structured data. At the same time, data with a latent graph structure is ubiquitous in modern data science. It is highly desirable to develop deep learning techniques that best suite graph structures, such as social network, knowledge network, brain connectivity network, etc.

Earlier works on Graph Neural Networks (GNNs) (Gori et al., 2005; Scarselli et al., 2009) use recursive networks. Those GNNs process the graph using a set of neurons, each corresponding to a node in the graph. The neurons update nodes representation and exchange information from linked neighbor nodes iteratively until reaching equilibrium.

Inspired by the power of convolutional networks on image and text data, different ideas have been proposed to implement the "convolution" on graph structures. There are two main directions, spectral convolutions and spatial convolutions. Spectral convolutional networks (Bruna et al., 2014; Defferrard et al., 2016) apply convolutions to the spectral domain or the frequency domain of the input graph. These methods tend to be efficient, but are highly graph-dependent.

In a more explicit manner, spatial convolutional networks implement convolutions on graphs. The feature representation of each node is iteratively updated by aggregating information from immediate neighbors (Hamilton et al., 2017; Xu et al., 2019) or a receptive field determined by special methods (Niepert et al., 2016). A transformation of the information from neighbors - either linear or non-linear - can be learned through training. The node representation information transferred between vertices are called *messages*. Veličković et al. (2018) used self-attention mechanism to further refine the messages based on local information, namely, features of source and target nodes.

In spatial convolutions, it is essential to have shared filter parameters across different parts of the graph. However, it has been observed that the filters should be adaptive to different local graph structures. In particular, node degrees have been used to reweight the messages or as additional features of node representations (Kipf and Welling, 2017; Monti et al., 2017). This way, messages relevant to hub nodes with high degrees will be different from messages between normal nodes. However, node degree is only the simplest graph structural property. There are much richer and advanced structural information that should be exploited in order to develop structure-adaptive convolutional filters.

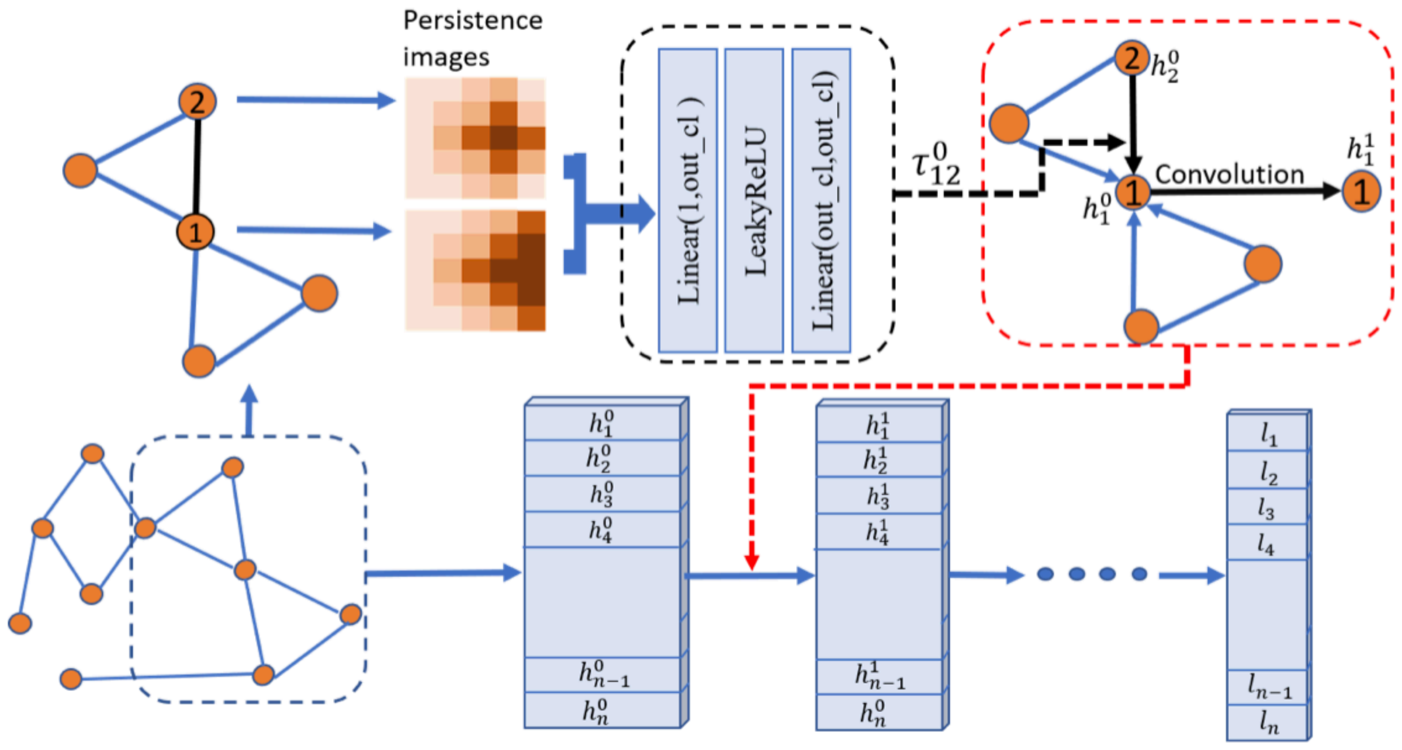
In this paper, we propose a novel and principled approach to maximally leverage the structural information in spatial graph convolution. In particular, for

Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

- Using persistence image to enhance graph neural networks
- Beat many GNN models on a broad spectrum of graph benchmarks

Reweighting via topological information

$$\vec{h}_u^{\ell} = \sigma\left(\sum\nolimits_{v \in \overline{\mathcal{N}}(u)} \mathrm{diag}(\tau_{v\to u}^{\ell}) W^{\ell} \vec{h}_v^{\ell-1}\right).$$



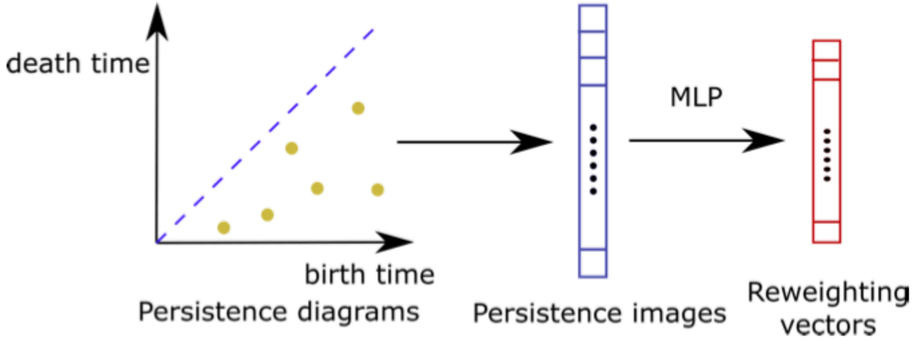


Table 1: Classification Accuracies on Benchmark Datasets

Method	Cora	Citeseer	PubMed	Coauthor CS	Coauthor Physics	Amazon Computer	Amazon Photo
MLP	58.2	59.1	70.0±2.1	88.3±0.7	88.9±1.1	44.9±5.8	69.6±3.8
MoNet	81.7	71.2	78.6±2.3	90.8±0.6	92.5±0.9	83.5±2.2	91.2±1.3
GraphSAGE	79.2	71.2	77.4±2.2	91.3±2.8	93.0±0.8	82.4±1.8	91.4±1.3
U-Net	82.5	72.0	78.9	92.7	94.0	86.0	91.9
WLCN	78.9	67.4	78.1	89.1	90.7	67.6	82.1
GCN	81.5±0.5	70.9±0.5	79.0±0.3	91.1±0.5	92.8±1.0	82.6±2.4	91.2±1.2
GAT	**83.0**±0.7	**72.5**±0.7	79.0±0.3	90.5±0.6	92.5±0.9	78.0±19.0	85.1±20.3
PEGN-RC-2	82.7±0.5	71.9±0.6	**79.4**±0.7	**92.9**±0.3	94.1±0.3	84.2±1	91.7±0.5
PEGN-RC-1	82.6±0.6	71.7±0.6	78.8±0.5	92.7±0.3	**94.2**±0.2	**86.3**±0.6	**92.5**±0.4

Section 3:
Topological loss, constraints / priors etc

Motivation

- Many learning / inference problems can be thought of as optimization problems

Motivation

- Many learning / inference problems can be thought of as optimization problems

- Typical neural network architectures could be "structure-oblivious"
  - e.g, a CNN is not effective at capture a global loop feature in the image (which could be important in cell segmentation)

Motivation

- Many learning / inference problems can be thought of as optimization problems

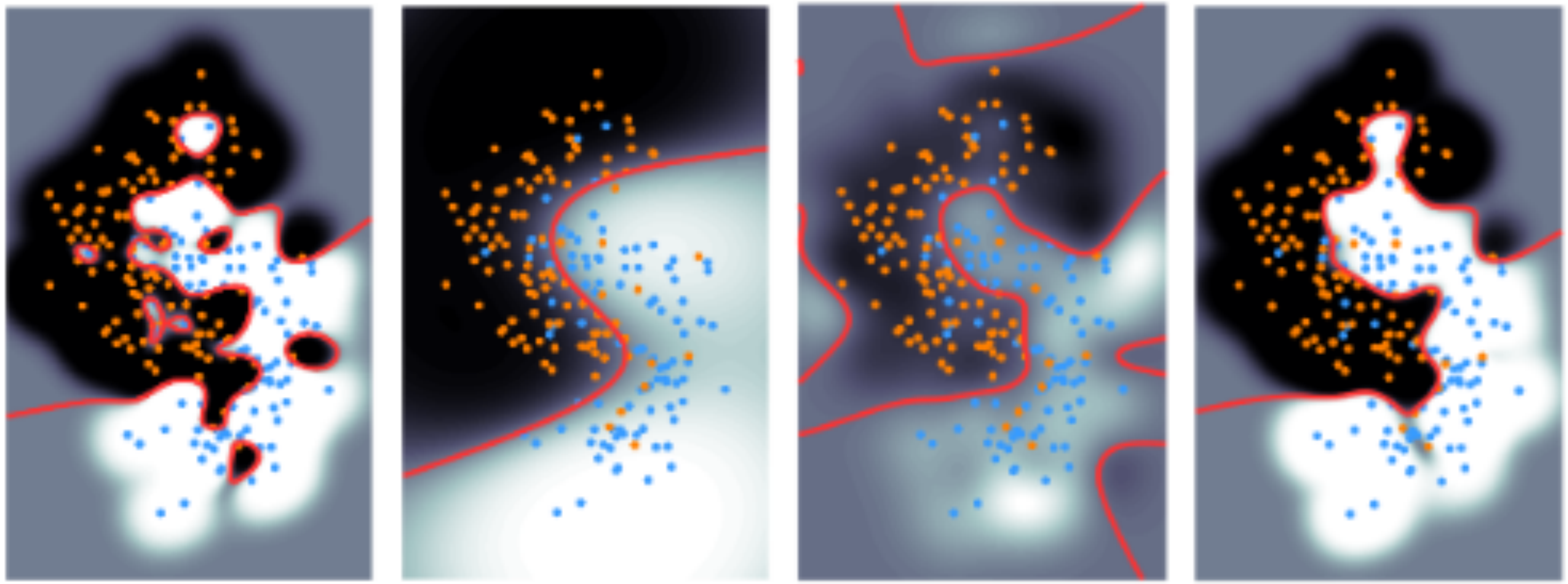
- Typical neural network architectures could be “structure-oblivious”
  - e.g, a CNN is not effective at capture a global loop feature in the image (which could be important in cell segmentation)

- Hence it makes sense to be able to optimization a function concerning topological properties of domain or data

- Motivating examples of where topological functions arise

Example 1: Regularizing Classifier Function

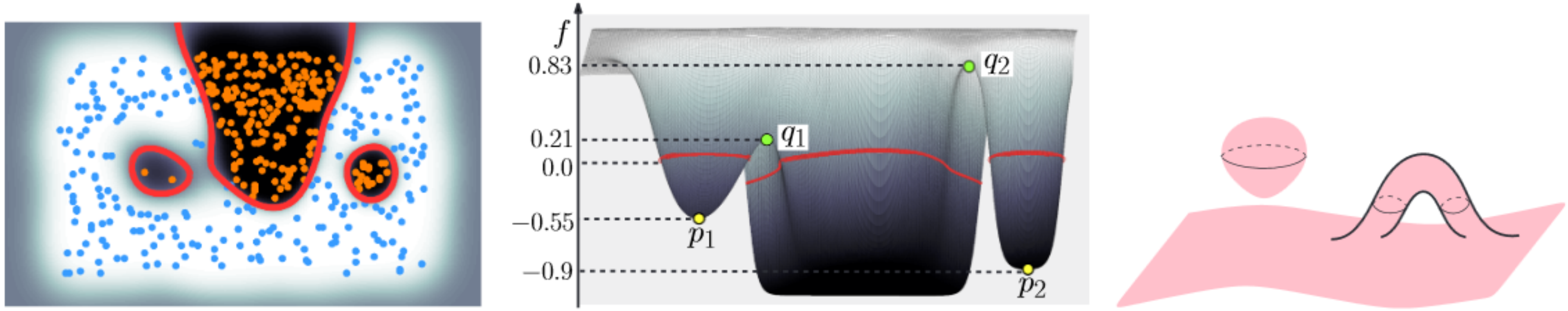
- Classifier function $f: X \to R$ defined on feature space X
  - e.g, in binary classification case, $S_f = f^{-1}(0)$ is classification boundary
- Loss function: $L(f)$
  - Quality of f in making predictions + regularization
- One issue:
  - Structure agnostic



Classical kernel method; small σ	Classical kernel method; large σ	DGR by [Bai et al., ICML 2016]	Topological penalty based loss

Main Idea

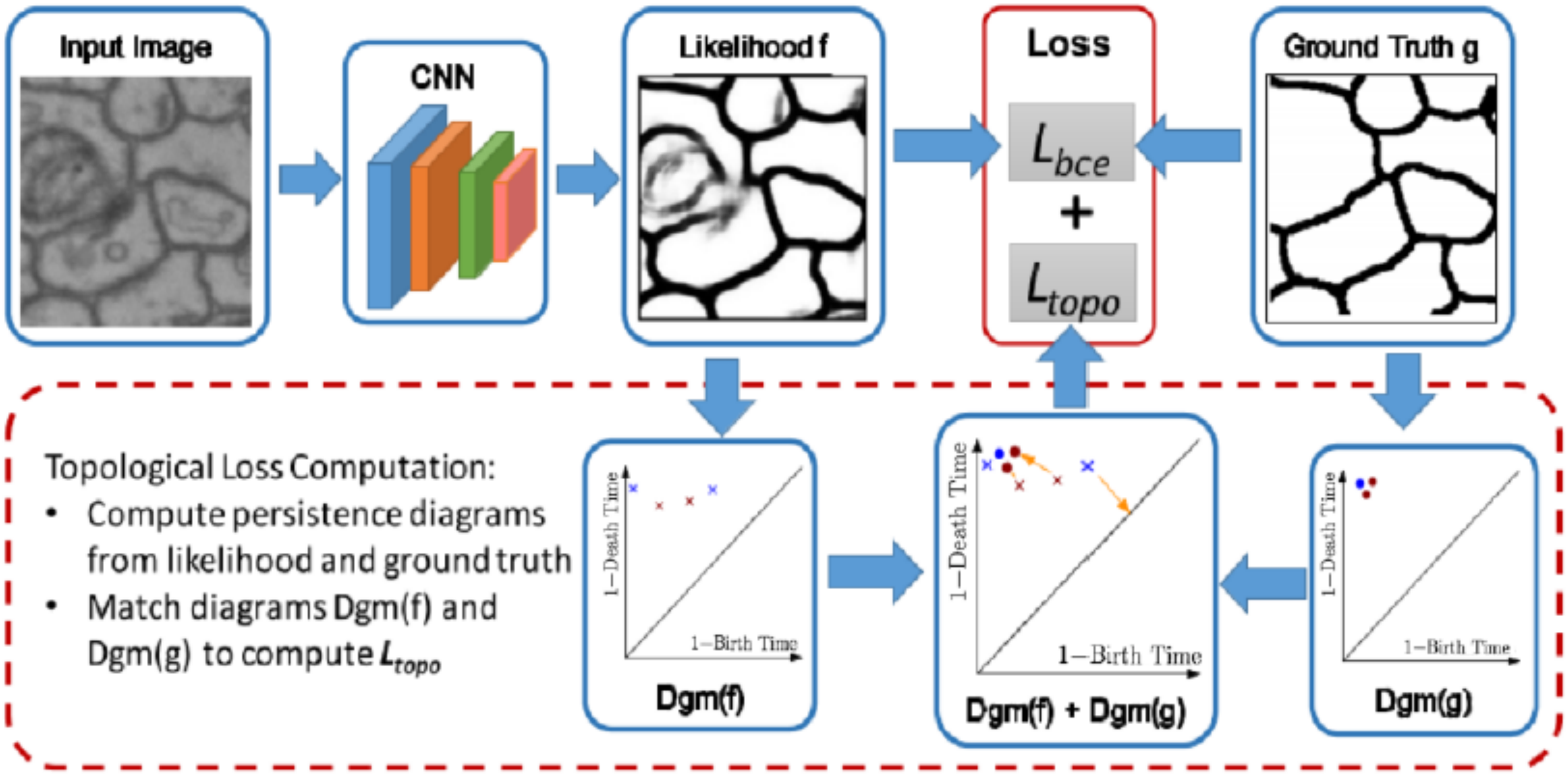
- [Chen, Ni, Bai, Wang, 2019]
- A topological loss $L_T(f)$
  - to regularize ``*topological simplicity*''
    -



(a) (b) (c)

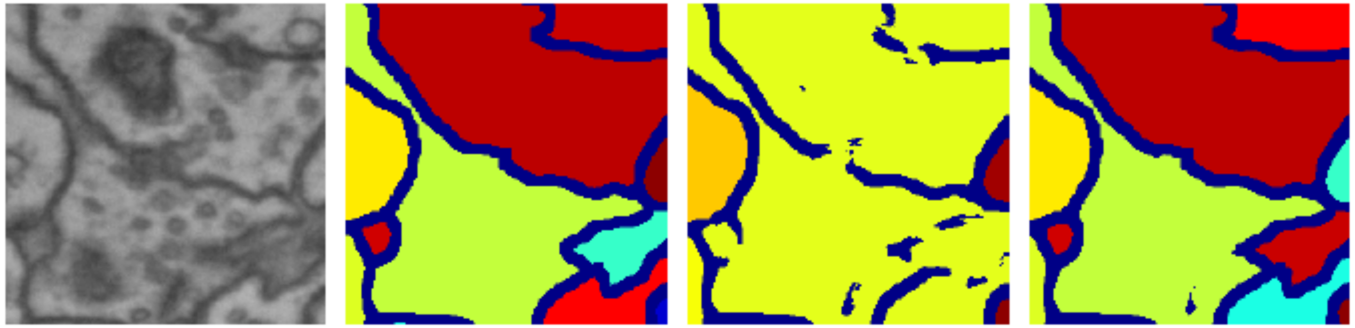
Example 2: Topological faithfulness of segmentation

- [Hu et al., 2019]



Result:

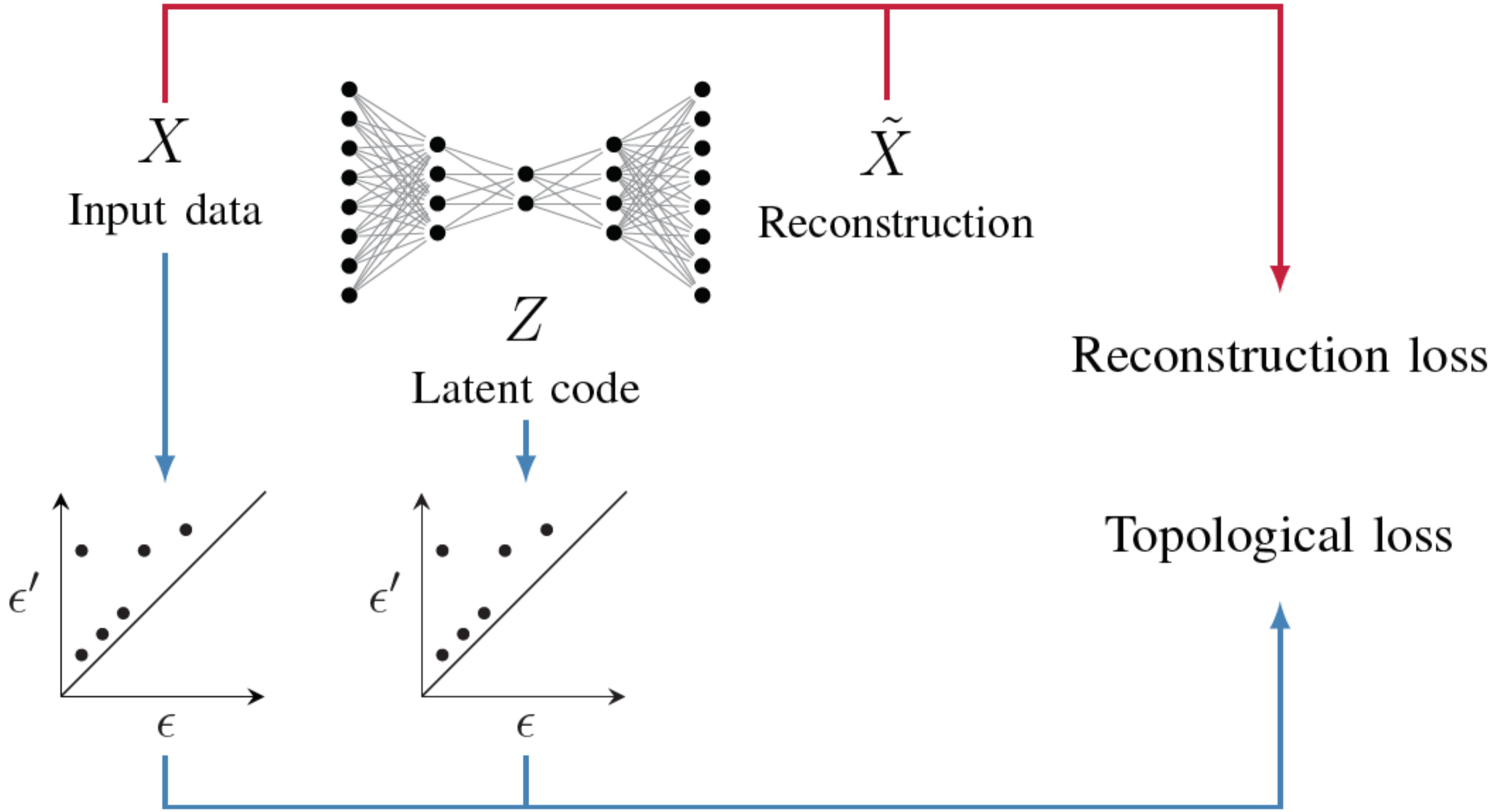
- Topology-preserving image segmentation



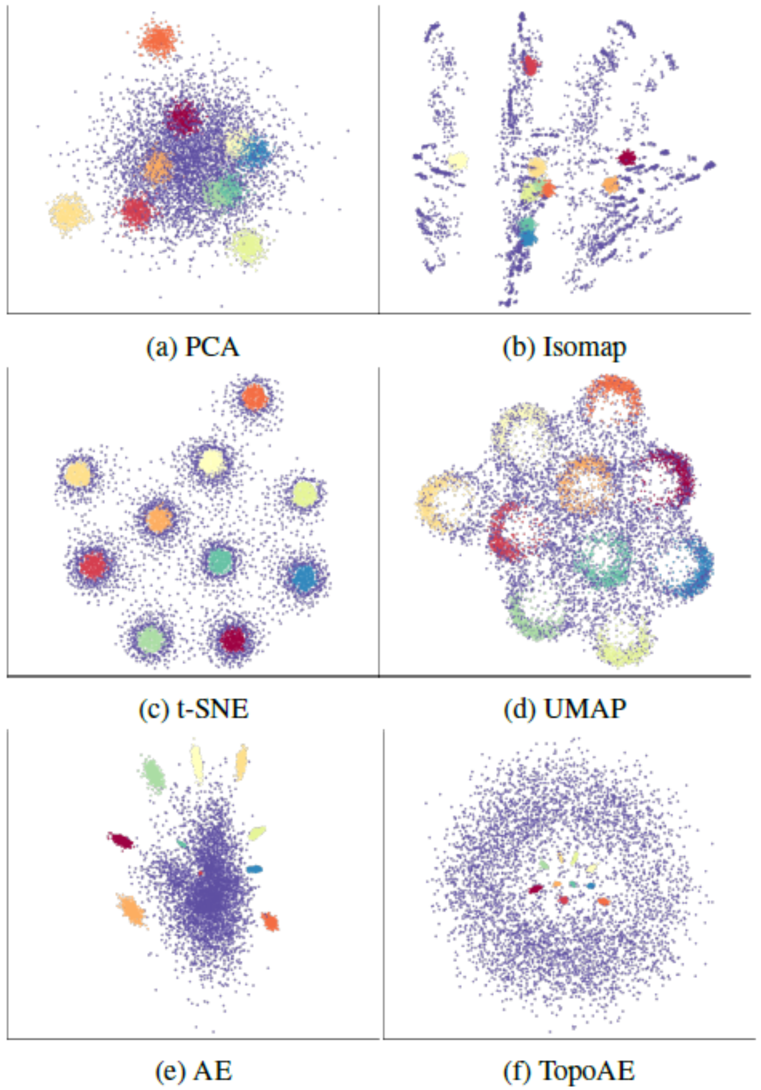
Standard architecture such as CNN (and variants) are less aware of explicit structure behind

Example 3: Topological autoencoder

- [Moor et al., 2020]



- An embedding of a collection of 10 spheres enclosed in a bigger 11-th sphere in very high dimensional space



- Challenges:
  - How do we encode topological information ?
  - Can we “differentiate” a function encoding topological information ?
- Turns out that we can compute gradients for topological function based on persistence homology (PH)
- In what follows:
  - a more detailed example where topological function comes into picture
  - gradients of a PH-based topological function

Recall: classifier example

- A topological loss
  - to regularize ``*topological simplicity*''
  - $$L_T(f) = \sum_{c \in C(S_f)} \rho^2(c)$$
    - $C(S_f)$ is the set of topological features of classification boundary S_f
    - $\rho(c)$ measures ``*cost*'' of feature $c \in C(S_f)$
    - for simplicity: $C(S_f)$ is the 0-th dimensional features (components)

A Topological Regularizer for Classifiers via Persistent Homology

Chao Chen[1] **Xiuyan Ni**[2] **Qinxun Bai**[3] **Yusu Wang**[4]

[1]Stony Brook University, Stony Brook, NY [2]City University of New York, New York, NY

[3]Hikvision Research America, Santa Clara, CA [4]Ohio State University, Columbus, OH

Abstract

Regularization plays a crucial role in supervised learning. Most existing methods enforce a global regularization in a structure agnostic manner. In this paper, we initiate a new direction and propose to enforce the structural simplicity of the classification boundary by regularizing over its *topological complexity*. In particular, our measurement of topological complexity incorporates the *importance* of topological features (e.g., connected components, handles, and so on) in a meaningful manner, and provides a direct control over spurious topological structures. We incorporate the new measurement as a topological penalty in training classifiers. We also propose an efficient algorithm to compute the gradient of such penalty. Our method provides a novel way to topologically simplify the global structure of the model, without having to sacrifice too much of the flexibility of the model. We demonstrate the effectiveness of our new topological regularizer on a range of synthetic and real-world datasets.

1 Introduction

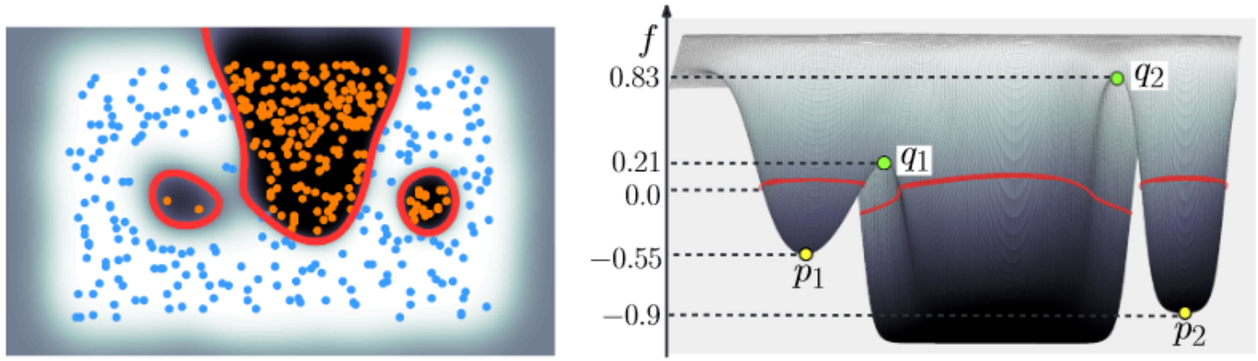
Regularization plays a crucial role in supervised learning. A successfully regularized model strikes a balance between a perfect description of the training data and the ability to generalize to unseen data. A common intuition for the design of regularzers is the Occam's razor principle, where a regularizer enforces certain simplicity of the model in order to avoid overfitting. Classic regularization techniques include functional norms such as L_1 (Krishnapuram et al., 2005), L_2 (Tikhonov) (Ng, 2004) and RKHS norms (Schölkopf and Smola, 2002). Such norms produce a model with relatively less flexibility and thus is less likely to overfit.

A particularly interesting category of methods is inspired by the geometry. These methods design new penalty terms to enforce a geometric simplicity of the classifier. Some methods stipulate that similar data should have similar score according to the classifier, and enforce the smoothness of the classifier function (Belkin et al., 2006; Zhou and Schölkopf, 2005; Bai et al., 2016). Others directly pursue a simple geometry of the classifier boundary, i.e., the submanifold separating different classes (Cai and Sowmya, 2007; Varshney and Willsky, 2010; Lin et al., 2012, 2015). These geometry-based regularizers are intuitive and have been shown to be useful in many supervised and semi-supervised learning settings. However, regularizing total smoothness of the classifier (or that of the classification boundary) is not always flexible enough to balance the tug of war between overfitting and overall accuracy. The key issue is that these measurement are usually structure agnostic. For example, in Figure 1, a classifier may either overfit (as in (b)), or becomes too smooth and lose overall accuracy (as in (c)).

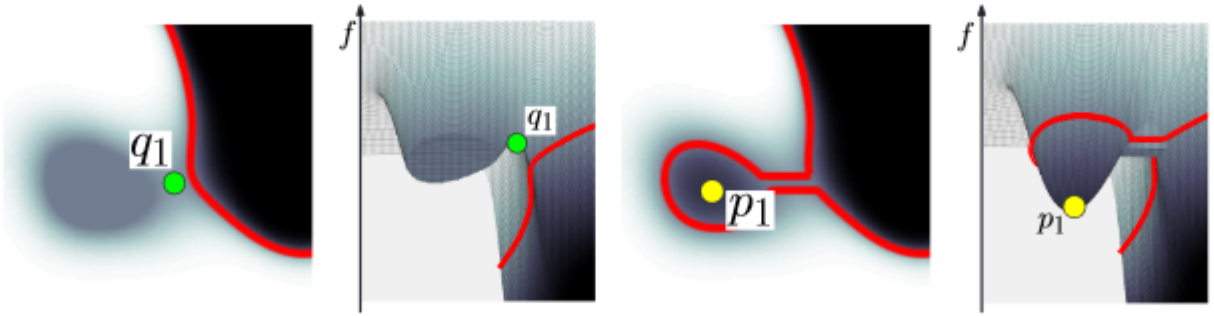
In this paper, we propose a new direction to regularize the "simplicity" of a classifier – Instead of using geometry such as total curvature, we directly enforce the "simplicity" of the classification boundary, by regularizing over its *topological complexity*. (Here, we take a similar functional view as Bai et al. (2016) and consider the classifier boundary as the 0-valued level set of the classifier function $f(x)$; see e.g., Figure 2.) Our measurement of topological complexity incorporates the *importance* of topological structures, e.g., connected components, handles, in a meaningful manner, and provides a direct control over spurious topological structures. This new structural simplicity can be combined with other regularizing terms (say geometry-based ones or functional norms) to train a better classifier. See Figure 1 (a) for an example, where the classifier computed with topological regularization achieves a better balance between overfitting and classification accuracy.

Proceedings of the 22[nd] International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89. Copyright 2019 by the author(s).

Topological loss



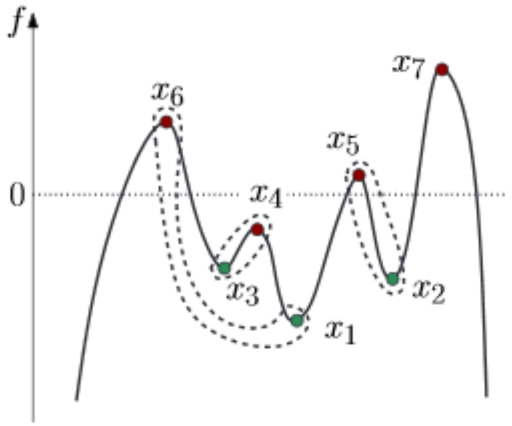
- Two options to remove the left component:

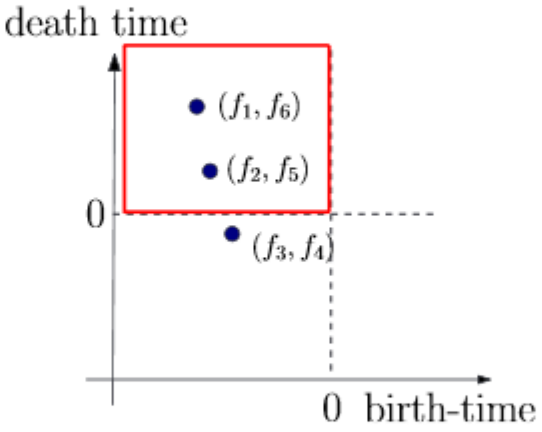


(a) (b) (c) (d)

Topological loss

- Given classifier function $f : X \to R$
  - compute its 0-th persistence diagram $Dg\ f$
  - set $C\left(S_f\right) = \{(b, d) \in Dg\ f \mid b \leq 0, d \geq 0\}$





Topological loss

- Given classifier function $f: X \to R$
  - compute its 0-th persistence diagram $Dg\ f$
  - set $C(S_f) = \{(b, d) \in Dg\ f \mid b \leq 0, d \geq 0\ \}$
  - set $\rho(c) = \min\{\ |b|,\ |d|\ \}$ for each $c \in \Pi_S$

- Topological loss
  - $L_T(f) = \sum_{c \in C(S_f)} \rho^2(c)$

- Total loss
  - $L_N(f) = L(f) + L_T(f)$

Key challenge:
Compute gradient $\nabla_w L_T(\mathrm{f_w})$ for topological loss

Differentiation of a PH-based topological function

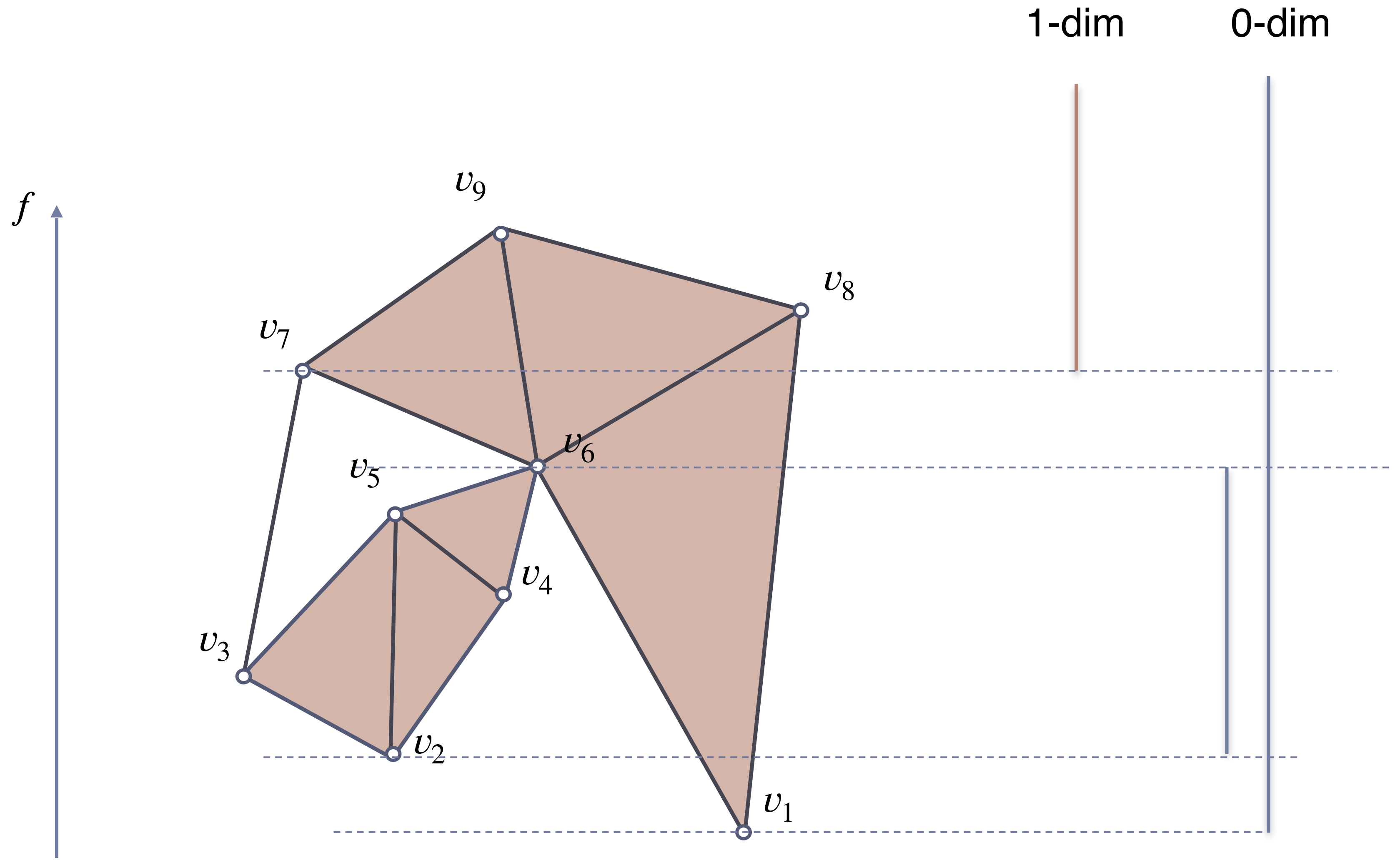
- Suppose we are now given a topological function
  - $L_T(f) := L_T(D_f)$ that depends on persistent points in a persistence diagram D_f induced by a function f
  - Now assume the function $f = f_\omega$ is parameterized by parameters ω
    - Say, f_ω could be modeled by a neural network

- Theorem ([Chen, Ni, Bai, Wang, 2019], [Gameiro, Hiraoka, Obayashi, 2016], [Poulenard, Skraba, Ovsjanikov, 2018])
  - Using piecewise linear approximation $\hat{f}_\omega$ of f_ω, the topological loss $L_T(\hat{f}_\omega)$ is differentiable almost everywhere w.r.t. parameters ω.

How?

- Suppose we consider $f: X \to R$,
  - its persistent diagram $D_f = \left\{ \left(b_i,\ d_i\right),\ i \in \left[1,\ s\right] \right\}$
- Assume $\omega \in R^m$ is the parameter that parametrize $f_\omega: X \to R$
  - In order to compute $\nabla_w L_T(f_w)$, one needs to compute $\dfrac{\partial b_i}{\partial w}$ and $\dfrac{\partial d_i}{\partial w}$
  - Assume that f_w is a PL function

Example



How?

- Suppose we consider $f : X \to R$,
  - its persistent diagram $D_f = \left\{ \left(b_i,\ d_i\right),\ i \in \left[1,\ s\right] \right\}$
- Recall, each of $b_i / \ d_i$ in fact is the function value of some critical point.
  - Say for each pair $p_i = (b_i,\ d_i)$, let $(v_{\ell_i},\ v_{r_i}) \in V \times V$ be the corresponding pair of critical points.
- Assume $\omega \in R^m$ is the parameter that parametrize $f_\omega : X \to R$
  - Let $\xi : R^m \to 2^{V \times V}$ be the map $\xi(\omega) = \left\{ \left(v_{\ell_i},\ v_{r_i} \right) \right\}_{\left(b_i,\ d_i\right) \in D_{f_\omega}}$

How?

- Suppose we consider $f_\omega : X \to R$, with parameter $\omega \in R^m$
  - its persistent diagram $D_{f_\omega} = \{(b_i,\ d_i),\ i \in [1,\ s]\}$
  - let $\xi : R^m \to 2^{V \times V}$ be the map $\xi(\omega) = \left\{\left(v_{\ell_i},\ v_{r_i}\right)\right\}_{(b_i,\ d_i) \in D_{f_\omega}}$

Proposition 13.9. *Suppose $f_\omega : |K| \to \mathbb{R}$ is a PL-function with distinct values on all vertices V of K, and K is a finite simplicial complex. Then there exists a neighborhood of ω in the parameter space such that ξ remains constant within this neighborhood; that is, the image set $\xi(\omega) = \rho_\omega(\mathrm{Dgm} f_\omega)$ remains the same for all parameters within this neighborhood.*

- Now given a topological loss function $L_T(f_\omega) := L_T(D_{f_\omega})$
  - computing $\frac{\partial L_T}{\partial \omega}$ essentially boils down to computing $\frac{\partial b_i}{\partial \omega}$ and $\frac{\partial d_i}{\partial \omega}$ for persistent points $(b_i,\ d_i) \in D_{f_\omega}$ via chain rule
  - $\frac{\partial b_i}{\partial w} = \frac{\partial f_w(\rho_w(b_i))}{\partial w} = \frac{\partial f_w(v_{\ell_i})}{\partial w}$ which can be computed; and similarly for $\frac{\partial d_i}{\partial \omega}$

Gradients computation

- For the classifier example:
  - $$\nabla_\omega L_T\left(f_\omega\right) = \sum_{c \in C\left(S_f\right)} \nabla_\omega\left(\rho(c)^2\right) = \sum_{c \in C(S_f)} 2 f_\omega\left(v_c\right) \frac{\partial f_\omega(v_c)}{\partial \omega}$$

Synthetic							
	KNN	LG	SVM	EE	DGR	KLR	TopoReg
Blob-2 (500,5)	7.61	8.20	7.61	8.41	7.41	7.80	7.20
Moons (500,2)	20.62	20.00	19.80	19.00	19.01	18.83	18.63
Moons (1000,2,Noise 0%)	19.30	19.59	19.89	17.90	19.20	17.80	17.60
Moons (1000,2,Noise 5%)	21.60	19.29	19.59	22.00	22.30	19.00	19.00
Moons (1000,2,Noise 10%)	21.10	19.19	19.89	24.40	26.30	20.00	19.70
Moons (1000,2,Noise 20%)	23.00	19.79	19.40	30.60	30.20	19.50	19.40
AVERAGE	18.87	17.68	17.70	20.39	20.74	21.63	16.92
UCI							
	KNN	LG	SVM	EE	DGR	KLR	TopoReg
SPECT (267,22)	17.57	17.20	18.68	16.38	23.92	18.31	17.54
Congress (435,16)	5.04	4.13	4.59	4.59	4.80	4.12	4.58
Molec. (106,57)	24.54	19.10	19.79	17.25	16.32	19.10	12.62
Cancer (286,9)	29.36	28.65	28.64	28.68	31.42	29.00	28.31
Vertebral (310,6)	15.47	15.46	23.23	17.15	13.56	12.56	12.24
Energy (768,8)	0.78	0.65	0.65	0.91	0.78	0.52	0.52
AVERAGE	15.46	14.20	15.93	14.16	15.13	13.94	11.80
Biomedicine							
	KNN	LG	SVM	EE	DGR	KLR	TopoReg
KIRC (243,166)	30.12	28.87	32.56	31.38	35.50	31.38	26.81
fMRI (1092,19)	46.70	74.91	74.08	82.51	31.32	34.07	33.24

Other TDA+ML examples

- Using PH to study activation network
- Topological complexity of neural network?
- Using Mapper to understand a trained neural network
- …

FIN