UNIVERSITY OF SOUTHAMPTON

Faculty of Engineering and Physical Sciences
School of Electronics and Computer Science

Persistence-Based Summaries for Data Analysis with Applications to Cyber Security

by

Thomas Davies

A thesis submitted for the degree of
Doctor of Philosophy

April 2023

University of Southampton

Abstract

Faculty of Engineering and Physical Sciences
School of Electronics and Computer Science

Doctor of Philosophy

Persistence-Based Summaries for Data Analysis with Applications to Cyber Security

by Thomas Davies

First formalised by Poincaré in his seminal text *Analysis Situs*, meaning the geometry of position, topology is the mathematical study of structure that remains invariant under continuous deformation. For over a hundred years this understanding of shape was confined to pure mathematics, but the advent of persistence-based summaries which enable practitioners to compute concise representations of the topology of data with strong theoretical guarantees has led to applications of the topological notion of shape to data analysis and machine learning.

The first part of this thesis is concerned with understanding and extending the application of persistence-based summaries to machine learning. Motivated by an investigation into the utility of topological loss terms through the lens of statistical learning theory, we adapt a recent extension of the higher-order Laplacian to the persistent case for machine learning, suggesting a vectorisation scheme and baselining its efficacy on the MNIST and MoleculeNet datasets. We find that it outperforms persistent homology across all of our baseline tasks. We also extend the ubiquitous fuzzy c-means clustering algorithm to the space of persistence diagrams, proving the same convergence guarantees as the Euclidean case. We apply the fuzzy clustering algorithm to model selection, matching pre-trained deep learning models to datasets via the topology of their decision boundaries.

In the second part of this thesis we consider applications of persistence-based summaries to cyber security. Cyber security is a critical application domain, with the annual cost of cyber crime to the UK economy estimated to be in excess of £27 billion and cyber attacks considered a tier 1 national security risk by the UK government. We investigate the utility of persistence-based summaries when detecting malicious behaviour in host-based computer logs, which are intrinsically extremely structured. We find that our methods can rival a standard baseline from the literature.

Contents

List of Figures

List of Tables

Declaration of Authorship

I declare that this thesis and the work presented in it is my own and has been generated by me as the result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. Parts of this work have been published as:

   (i) N. Bishop, T. Davies, and L. Tran-Thanh. Hypothesis classes with a unique persistence diagram are not nonuniformly learnable. *NeurIPS 2020 Workshop on Topological Data Analysis and Beyond*, 2020.

   (ii) N. Miolane et al. ICLR 2021 challenge for computational geometry & topology: Design and results, arXiv:2108.09810, 2021.

   (iii) T. Davies. A Review of Topological Data Analysis for Cybersecurity, *AAAI Workshop on Artificial Intelligence for Cyber Security*, 2022a.

   (iv) T. Davies. Topological Data Analysis for Anomaly Detection in Host-Based Logs, *SDM Workshop on Applications of Topological Data Analysis to Data Science, Artificial Intelligence, and Machine Learning*, 2022b.

   (v) T. Davies et al. Fuzzy c-means clustering in persistence diagram space for deep learning model selection. *NeurIPS workshop on symmetry and geometry in neural representations*, PMLR 197:137-157, 2023.

   (vi) T. Davies et al. The Persistent Laplacian for Data Science: Evaluating Higher-Order Persistent Spectral Representations of Data, *International Conference on Machine Learning*, PMLR 202:7249-7263, 2023.

Signed:... Date:..................

Acknowledgements

First and foremost, I am thankful for my supervisors Ruben Sanchez-Garcia, Long Tran-Thanh, and Corina Cirstea. Each of them has provided me with guidance and support at points throughout my PhD. I am particularly grateful to Ruben for his unwavering support, both academic and pastoral, from the moment he joined my supervisory team. He has consistently been there to provide insight, advice, and reassurance, and I am exceptionally grateful to him.

I have been lucky to collaborate with many other impressive people from a broad array of backgrounds throughout the course of my PhD. The breadth of content I have been exposed to has been a real highlight of my graduate studies, and it would not have been possible without experts from other fields willing to work with me. In particular, Nick Bishop, Sunil Manghani, and Zhengchao Wan have been brilliant collaborators. A special thanks must also go to Jack Aspinall, who went from my first friend at a new primary school to a co-author on my first paper.

I am truly thankful for my friends. To those from Stroud (JD, JC, ED, and the LB), Birmingham (especially 'egg', RM, and two generations of 32T), Southampton (CM, JE, LB, and LP), and London (SHF, AH, AM, TS, and the Lipari group), I enormously appreciate all of your friendships. My deepest thanks go especially to FS, who has been my closest friend and confidant for many years.

Finally, none of this would have been possible without the unwavering support of my parents, Deanna and Michael Davies, who have been there every step of the way. I am forever indebted to them both. It definitely would have been possible without my siblings Alice and James, but I am certain the experience would have been far worse. Anytime I have had imposter syndrome a round of golf with Uncle Nicholas has shown me what it truly means to be an imposter. My grandparents Goronwy Ogwen Davies, Wolodymyr Szypylawyj Griffiths, and Sheila Griffiths played a huge part in making me who I am today. This thesis is very much written in their memory.

Dedicated to the memory of my grandmother, Sheila Griffiths,
winner of the 1949 Stourport County School Mathematics Prize.

Chapter 1

Introduction

Topology is the study of shape, characterising spaces by structure that remains constant under continuous deformation. Leibniz called this *Analysis Situs*, or the geometry of position (Leibniz and Gerhardt, 1863). In the introduction to his seminal paper of the same name, Poincaré (1895) described it as

> "... the art of reasoning well from badly drawn figures; however, these figures, if they are not to deceive us, must satisfy certain conditions; the proportions may be grossly altered, but the relative positions of the different parts must not be upset."

Later in the paper he formalised for the first time what such an altering may look like by defining a *homeomorphism*: a function between (topological) spaces which is continuous, bijective, and has a continuous inverse (Poincaré and Stillwell, 2010). This function may deform and stretch the underlying space, but by definition the resultant space is topologically the same; the structure is preserved. The archetypal example is the homeomorphism between a mug and a doughnut: a doughnut may be continuously and bijectively deformed into a mug, and vice versa, so they are topologically the same. A depiction of homeomorphisms is shown in Figure 1.1, which shows an artistic interpretation of such continuous deformations.

In order to better understand structures with the same topology, mathematicians looked to find *topological invariants*: properties of spaces that do not change under homeomorphism. First Brouwer (1911) showed that dimension is a topological invariant. Shortly after the Betti number, introduced by Betti (1871) to study the p-dimensional connectivity of spaces, was shown to be a topological invariant by Alexander (1915). Informally, the Betti number β_0 corresponds to the number of connected pieces, β_1 corresponds to the number of holes, and β_2 corresponds to the number of voids. It was later that Noether (1926) realised that the natural way to view Betti numbers was as an invariant of *homology groups*; algebraic groups H_p that encode aspects of the topological structure.

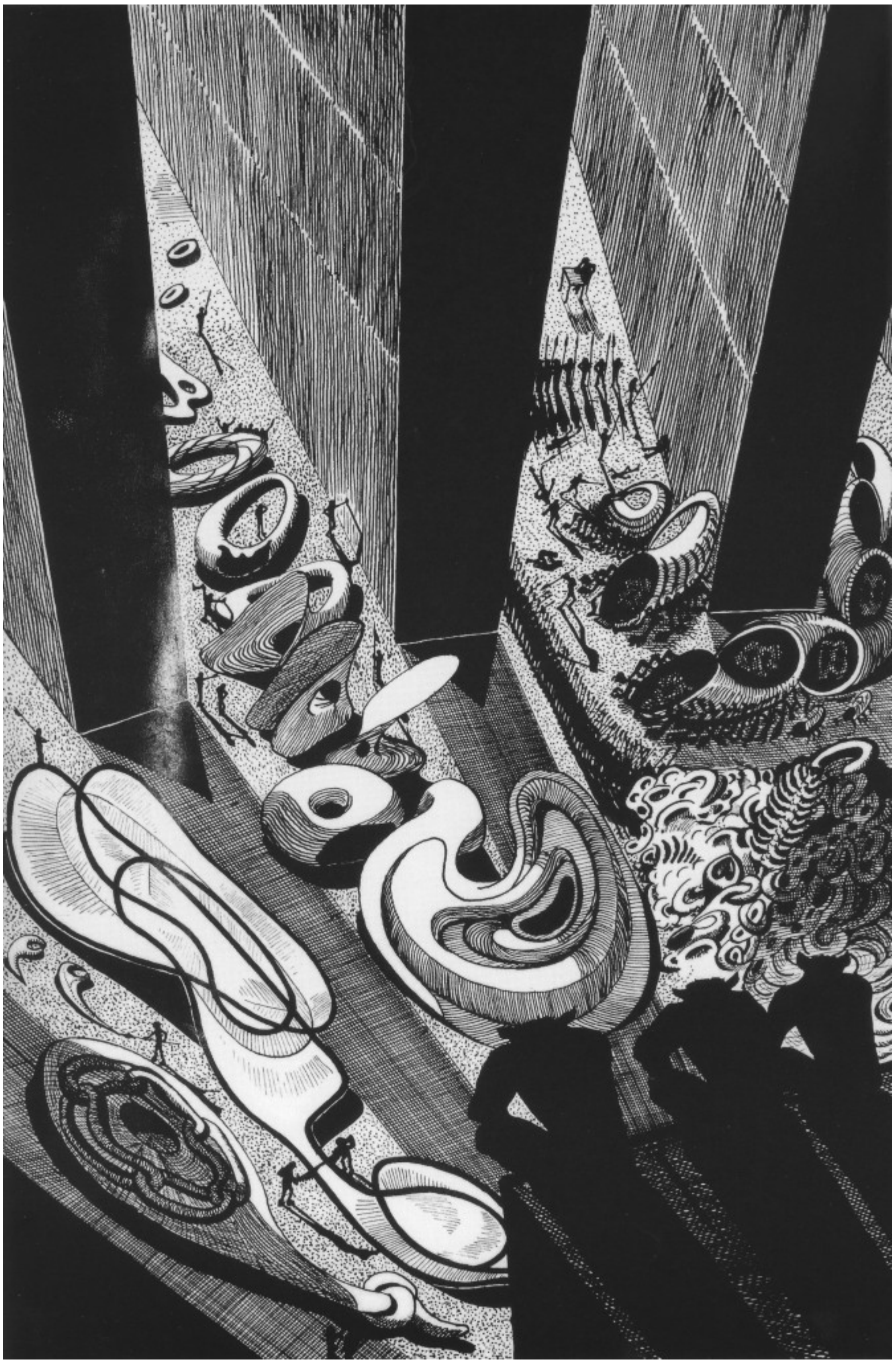


FIGURE 1.1: Anatoly Fomenko's 'Topological Zoo' depicts the continuous deformations which preserve topological structure. Reproduced from Fomenko and Fuchs (2016).

These are topological invariants: if two spaces are homeomorphic (in fact, *homotopic*), their homology groups are *isomorphic*, where an isomorphism is a bijective function between groups $f : G \to H$ such that $f(gh) = f(g)f(h)$ for all $g \in G$ and $h \in H$. An equivalence class of the homology group corresponds to a topological feature, and the rank of H_p is the Betti number β_p. In the words of topologist and Bletchley Park codebreaker Hilton (1988), Noether's work meant that the mathematical place of these topological invariants was properly understood for the first time.

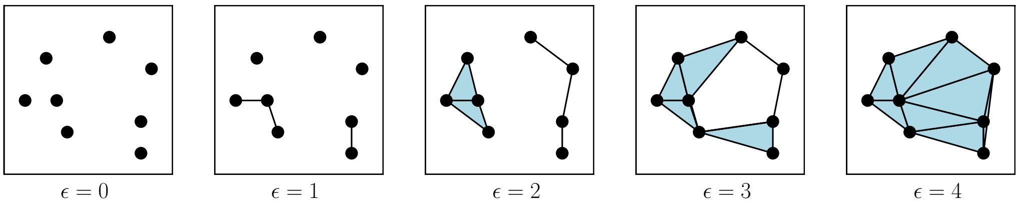
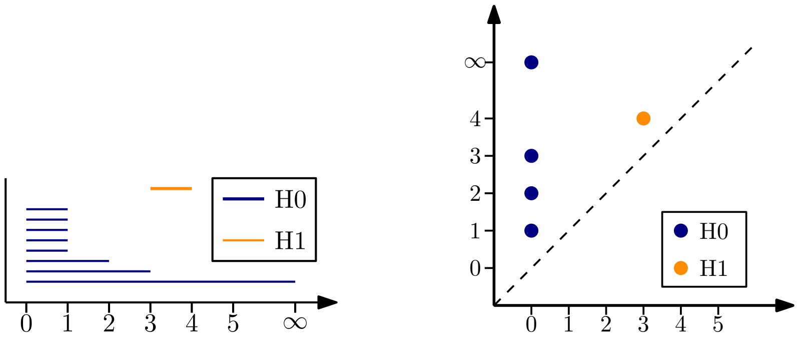


FIGURE 1.2: A filtration of simplicial complexes. Note that $K_\epsilon \subseteq K_{\epsilon'}$ whenever $\epsilon \leq \epsilon'$. In this case ϵ is a threshold with respect to the Euclidean distanc between points.

Over a century after Poincare's *Analyis Situs*, topologists laid out the case for the application of topological methods to data analysis. One such paper was 'Topology and Data' by Carlsson (2009), in which he posited that theoretically motivated topological summaries are capable of efficiently representing high-dimensional noisy datasets. One of the tools he mentions is *persistent homology*. *Persistence* refers to the idea of computing homology groups over a growing sequence of topological spaces called a *filtration*, which traces its roots back to work by Frosini (1992) on size functions and Robins (1999) on the topology of attractors in dynamical systems (Perea, 2018). Persistent homology groups are typically computed over a filtration of simplicial complexes. We can think of an (abstract) simplicial complex is a higher-order graph, consisting of p-dimensional triangles called *simplices*. A 0-simplex is a vertex, and a p-simplex ($p > 0$) is a collection of $p+1$ vertices. Formally, a simplicial complex K is a collection of subsets, called simplices, σ such that every subsimplex $\sigma' \subseteq \sigma$ (called a *face* of σ) is also in K. A filtration of simplicial complexes is a collection of simplicial complexes $\{K_i\}_{i \in I}$ indexed by an ordered set I such that $K_i \subseteq K_j$ whenever $i \leq j$. An example of a filtration of simplicial complexes is shown in Figure 1.2.

Given a filtration $\{K_i\}_{i \in I}$, the persistent Betti number $\beta_p^{i,j}$ tells you the number of p-dimensional homology classes that are present in K_i and *persist to* (are still present in) K_j (we formally define this in Section 2.2.4). Edelsbrunner et al. (2002) give a first algorithm for the efficient computation of the persistent Betti numbers, and introduces the persistence barcode as a representation of the persistent homology. This depicts the *birth and deaths* of the homology classes, i.e., the points in the filtration $\{K_i\}_{i \in I}$ such that a persistent homology class enters the filtration at K_i and vanishes (becomes zero) in K_j; an example is shown in Figure 1.3i. Zomorodian and Carlsson (2005) then characterised the persistent homology as a graded module, which meant that decomposing this module using the structure theorem for finitely generated modules enabled a decomposition of the *persistence module* into a basis which directly corresponded to the birth/death coordinates of persistent homology classes. This representation was shown to be an isometry by Chazal et al. (2016), building on earlier work by Cohen-Steiner et al. (2007) which showed it was a stable representation of functional filtrations. An equivalent representation called the *persistence diagram*, a planar representation of the persistent



(i) A persistence barcode

(ii) A persistence diagram

FIGURE 1.3: The persistence barcode (i) and persistence diagram (ii) corresponding to the filtration in Figure 1.2. We formally define these summaries in Section 2.2.4.

homology which is equivalent to persistence barcodes with birth values on the x axis and death values on the y axis, has become the prevalent visual depiction of the persistent homology. An example is shown in Figure 1.3ii. This is one example of a *persistence-based summary*, which are the representations of data that underpin this thesis.

With the theoretical foundations laid the stage was set for the field of *Topological Data Analysis* (TDA) to take off. Initially persistence diagrams and their feature vectors were investigated via the development of statistics on the space of persistence diagrams, which we review in Section 3.1. However, persistence diagrams are not a convenient representation to use with downstream data analysis tools: it is expensive to compute distances between them, their means are not unique, and the number of points depends on the topology of the underlying data. Because of this, a significant number of featurisation methods that map persistence diagrams into a vector in $\mathbb{R}^n$ have been proposed. We discuss common methods in Section 3.2.2, but we are inclined to agree with Bubenik (2020), who believes that "perhaps since the persistence diagram is such a rich invariant, it seems that any reasonable way of encoding it in a vector works fairly well".

Each of these vectorisations of the persistence diagram is also a persistence-based summary; in this thesis we are particularly interested in their application to *machine learning*. Machine learning studies computational algorithms that 'learn' from data. In fact, the development of a feature vector which summarises the topology of an underlying dataset with strong theoretical guarantees makes it a natural candidate for usage in machine learning methodologies. Machine learning algorithms are either *supervised*, in which an algorithm learns a function f_ω with parameters ω which maps data X to labels y from a training dataset with ground truth labels, or *unsupervised*, in which a function f is learnt to infer labels from a dataset X with no ground truth. Typically, they are either for *classification*, in which the learner aims to map input data $x \in X$ into a discrete category, or *regression*, in which input data $x \in X$ is mapped to a continuous variable (Bishop and Nasrabadi, 2006). In this thesis, we often use random forests (Ho, 1995), a supervised learning method that can be used for either

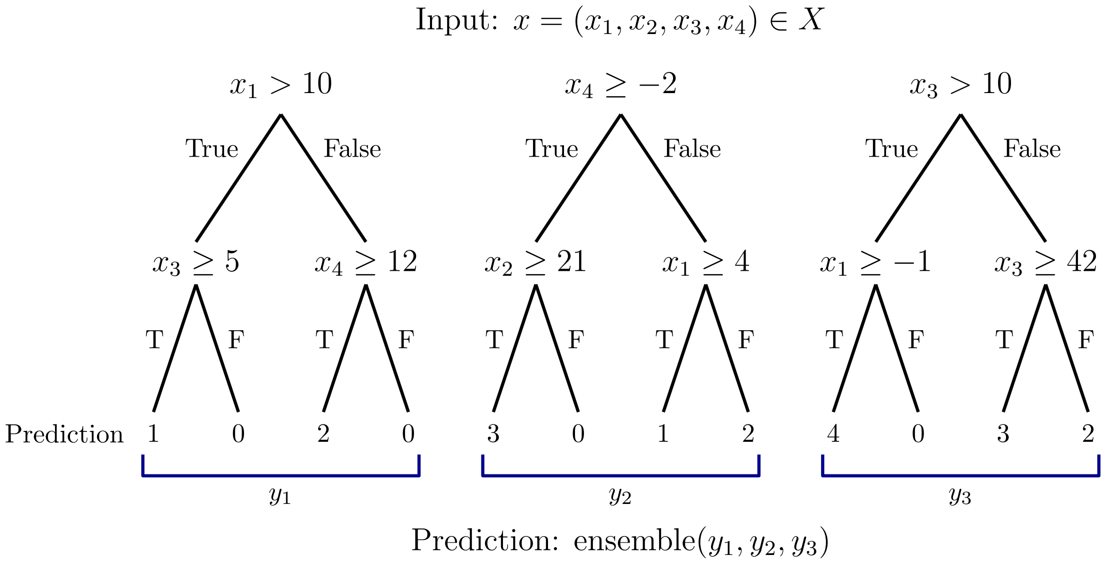


FIGURE 1.4: An example random forest with three decision trees. Each decision tree takes the data as an input at the top, which passes down through learnt thresholds to make a prediction. The predictions of each tree are ensembled by averaging for regression or majority voting for classification.

classification or regression, and works by aggregating the predictions of individual decision trees (Breiman, 2017). A random forest comprising of three decision trees is shown in Figure 1.4. Each node in a decision tree learns a parameter and threshold when it is being trained, which it can then use to make predictions. Individual predictions are typically ensembled by majority voting for classification tasks and computing the mean prediction for regression tasks. Although an individual tree is prone to overfitting, a random forest will significantly increase the performance over an individual tree, at the cost of being less able to inspect the tree to interpret its prediction (Hastie et al., 2009). We typically use random forests as our downstream models when applying persistence-based summaries, specifically in Chapters 4 and 7. We extend an unsupervised learning algorithm to persistence diagrams in Chapter 5.

Techniques like random forests are sometimes referred to as *shallow learning*, as they are learning relatively few parameters. In comparison, work pioneered by LeCun, Bengio, and Hinton (2015) has led to a *deep learning* revolution in which models with sometimes hundreds of billions of parameters have performed extremely well at a broad array of tasks. Topological tools have been integrated into deep learning models, providing knowledge of the topology of the data to the learner. In particular, deep learning models typically use gradient descent optimisers that rely on a loss function $\mathcal{L}(y, y')$ that computes the error between the ground truth labels y and the models current predictions y'. By computing the error of the learner $f_\omega : \mathbf{X} \to \mathbf{y}$ in terms of the parameters ω, deep learning can update the parameters to decrease the error of the model. This is of course a simplification, but it means that as long as we can differentiate through the computation of persistence diagrams we can integrate topological priors into the training of a deep learning model by adding a topological term to the loss function $\mathcal{L}$. We review this process in more detail in Section 3.2.3, before investigating the efficacy of such methods through the lens of *statistical learning theory* in Chapter 4.

1.1 Research Contributions

The specific research contributions of this thesis are as follows. Citations correspond to our peer-reviewed work.

- In the first part of Chapter 4 (Section 4.1), we provide the first investigation of topological loss terms in deep learning through the lens of statistical learning theory. A topological loss term implicitly restricts the hypothesis class of the learner to the preimage of a persistence diagram. We show that one cannot provide learning theoretic guarantees about such hypothesis classes (Bishop et al., 2020).
- In the remainder of Chapter 4, we build on recent work introducing the persistent Laplacian. We propose a vectorisaton scheme to enable its use its machine learning and evaluate its efficacy on the MNIST and MoleculeNet datasets, demonstrating that it outperforms persistent homology (Davies et al., 2023b).
- In Chapter 5 we extend the fuzzy c-means (FCM) clustering algorithm to the space of persistence diagrams, proving the same convergence guarantee as Euclidean FCM. We apply the algorithm to match pre-trained deep learning models to datasets (Davies et al., 2023a). We also implemented this algorithm for Riemannian manifolds (Miolane et al., 2021).
- In Chapter 6 we review the literature applying Topological Data Analysis to cyber security (Davies, 2022a).
- In Chapter 7 we apply TDA to host-based computer logs, proposing a method to represent host-based logs as a filtration of simplicial complexes in order to identify malicous activity. We evaluate the performance of our method against three baselines in two different scenarios and at two levels of data acuity (Davies, 2022b).
- We have released our implementations of fuzzy c-means clustering for persistence diagrams and the persistent Laplacian for machine learning, enabling practitioners to utilise our methodologies.

I have also contributed to the following publications on topics outside the main focus of this thesis:

- Fernanda Ribeiro, Valentina Shumovskaia, Thomas Davies and Ira Ktena. How fair is your graph? Exploring fairness concerns in neuroimaging studies, *Machine Learning for Healthcare Conference*, PMLR 182:459-478, 2022.
- Fernanda Ribeiro, Valentina Shumovskaia, Thomas Davies and Ira Ktena. Evaluating graph fairness in transductive learning, *Medical Imaging with Deep Learning Conference*, 2022.
- Sunil Manghani and Thomas Davies. Structuralism, Large Language Models and Electronic Life, *under review*.

1.2 Thesis Structure

The main body of this thesis is presented in three parts.

- In Part I we introduce the necessary preliminaries, giving the required theoretical background in Chapter 2 and reviewing the literature on Topological Data Analysis and Topological Machine Learning in Chapter 3.
- In Part II we present our research into persistence-based summaries, evaluating the efficacy of the recently introduced persistent Laplacian as a feature vector for machine learning in Chapter 4 and extending the fuzzy c-means clustering algorithm to persistence diagram space in Chapter 5.
- In Part III we consider applications of Topological Data Analysis to cyber security, reviewing the relevant literature in Chapter 6 and applying persistence-based summaries to host-based computer logs in Chapter 7.

We end by concluding the thesis in Chapter 8, listing our references and providing revelant additional details in Appendices A, B, and C.

Part I

Preliminaries

Chapter 2

Theoretical Background

In this chapter we expand on the theoretical background behind persistence-based summaries. We introduced the persistence diagram at a high level in the introduction, but in this chapter we give more specific details on the construction of persistent homology. The focus of this thesis is on the application of these tools and the adaptation of recent theory to the field of data analysis. As such this section is scoped to provide the theoretical background underpinning many of these tools, without proving the main results. Further reading on Topological Data Analysis can be found in textbooks such as Carlsson and Vejdemo-Johansson (2021) or Dey and Wang (2022), or there are many excellent lectures notes available, for example those by Nanda (2021). For an introduction to homology theory, see perhaps Munkres (1984) or Hatcher (2000).

In this thesis we also consider the *Laplacian* as a feature vector. The (graph) Laplacian is a linear operator on graphs that was originally introduced by Kirchhoff (1958) to study electrical circuits. It is the basis of Spectral Graph Theory, a field that uses the spectrum of the Laplacian, and other matrices associated to graphs, to study the structure of graphs (Nica, 2016). It has been extended to simplicial complexes in the form of the combinatorial Laplacian (Horak and Jost, 2013), and recently a persistent version has been defined for pairs of simplicial complexes, the so-called persistent Laplacian (Mémoli et al., 2022). In fact, the nullity of the p-Laplacian is the p-Betti number β_p, and the non-zero eigenvalues are known to contain information about the structure of graphs, so the Laplacian contains significant topological and other structural detail. For this reason we use it throughout this thesis as an alternative, although closely related, way to analyse the structure of data. In fact, Chapter 4 evaluates the persistent Laplacian as a feature vector for machine learning. We introduce the relevant theory in Section 2.3.

2.1 Complexes and Boundaries

We start by introducing data structures which can represent interactions (pairwise, and higher-order), and the *boundary operator* defined on those representations.

2.1.1 Graphs and simplicial complexes

A graph is a discrete structure consisting of vertices V connected by edges $E \subseteq V \times V$ that represent pairwise interactions between vertices. In order to study higher-order topological features like holes and voids, we also need to consider higher-order interactions. A *simplicial complex* is a collection of p-dimensional triangles, *p-simplices*, that represents higher-order structure; 0-simplices and 1-simplices are nodes and edges, just like graphs, but k-simplices are collections of $k+1$ nodes, corresponding to $k+1$ cliques in a graph. This is shown in Figure 2.1. In particular, an abstract simplicial complex is a collection of sets K which is closed under taking subsets, i.e., given any set $\sigma \in K$, for any subset $\tau \subseteq \sigma$ we have that $\tau \in K$.

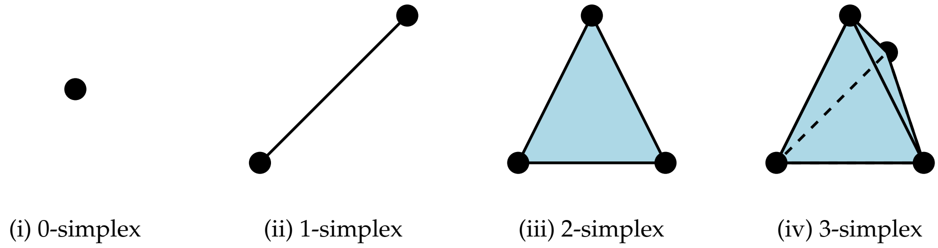

(i) 0-simplex (ii) 1-simplex (iii) 2-simplex (iv) 3-simplex

FIGURE 2.1: Examples of p-simplices.

We can also think about a simplicial complex as a geometric object. In this case each simplex is *realised* in a real space $\mathbb{R}^d$, for some $d \in \mathbb{N}^+$. We call a set of points $x_0, \ldots, x_p \in \mathbb{R}^d$ for some $d \in \mathbb{N}^+$ *affinely independent* if the vectors $\{x_1 - x_0, \ldots, x_p - x_0\}$ are linearly independent. Intuitively, this means that the $p+1$ points do not lie on a $p-1$ dimensional space. For example, three points cannot be colinear. Given a set of affinely independent points $\{x_0, \ldots, x_p\} \subset \mathbb{R}^d$, the *geometric p-simplex generated by* $x_0, \ldots, x_p$ is the convex hull of those points. Specifically,

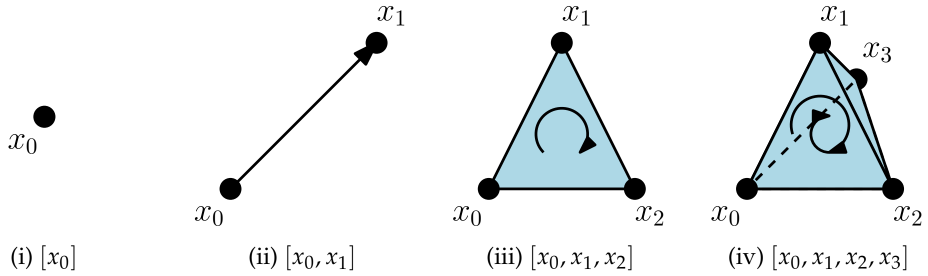
$$\sigma = \left\{ \sum_{i=0}^{p} t_i x_i : \sum_{i=0}^{p} t_i = 1, t_i \geq 0 \; \forall \, i \right\}.$$

A *face* of a simplex σ is a simplex generated by a subset of $x_0, \ldots, x_p$. This definition of a face aligns with intuition; for example, a face of a tetrahedron (3-simplex) is one of its constituent triangles (2-simplices). With this in mind, we have the following definition.

Definition 2.1. A *geometric simplicial complex* is a collection of geometric simplices $|K|$ such that every face of a simplex in $|K|$ is also in $|K|$, and the intersection of any two non-disjoint simplices in $|K|$ is a face of both simplices.

In practice, data we are working with may naturally include pairwise or higher-order interactions, in which case it can naturally be represented as a simplicial complex. If we are just given point data, then we can choose to add higher-order simplices in several ways. We may take the 0-simplices to be the points, and add higher-order simplices via a distance between the points or another property of the data. Carefully tailoring the construction of the simplicial complex from the input data is often a key step in the Topological Data Analysis pipeline.

Oriented simplices. We identify a simplex with its set of vertices, and call it *oriented* if we fix an *orientation*, that is, an ordering of its vertices. Given points $\{x_0, \ldots, x_p\}$ we write the oriented simplex generated by those points and with the vertex ordering $(x_0, \ldots, x_p)$ as $[x_0, \ldots, x_p]$. We say that two orderings of $\{x_0, \ldots, x_p\}$ are equivalent if the permutation between them is even (i.e., can be expressed as an even number of two-element swaps). An ordering on a simplex can be induced onto its faces. For example, the ordered 2-simplex $[x_0, x_1, x_2]$ has 1-faces $[x_0, x_1], [x_1, x_2], [x_2, x_0]$ with the induced ordering. Given an ordered simplex σ we say that the simplex with opposite orientation is one that is not equivalent, i.e., the permutation required to attain it from σ is odd. We denote a simplex with the opposite orientation as σ'. An example of oriented simplices is shown in Figure 2.2.



(i) $[x_0]$ (ii) $[x_0, x_1]$ (iii) $[x_0, x_1, x_2]$ (iv) $[x_0, x_1, x_2, x_3]$

FIGURE 2.2: Examples of oriented p-simplices.

Filtrations. A *simplicial pair* is a pair of simplicial complexes K, L such that $K \subseteq L$. We are often concerned with a growing series of simplicial complexes called a *filtration*. In particular, a *filtration of simplicial complexes* is a collection of simplicial complexes $\mathcal{F} = \{K_i\}_{i \in \mathbb{R}}$ such that $K_i \subseteq K_j$ is a simplicial pair whenever $i \leq j$. A filtration of simplicial complexes, which we often refer to as simply a *filtration*, is a key way to represent data that we consider throughout this thesis. An example of a filtration is shown in Figure 2.3.

Building complexes. To study the topology of a set of points $X \subset \mathbb{R}^d$, we map them into a simplicial complex; two closely related techniques are the Cêch (Čech et al., 1966) and the Vietoris-Rips (Vietoris, 1927) complexes. The *ϵ-Cêch complex* $C_\epsilon(X)$ adds the

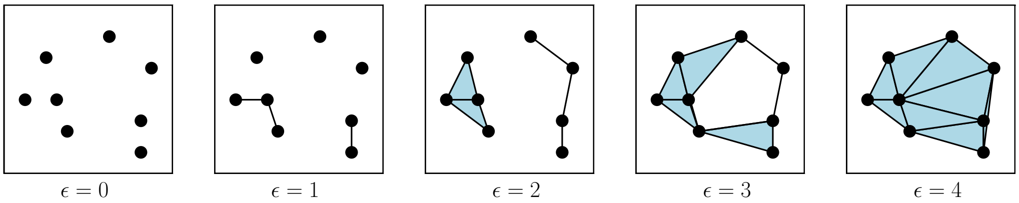


FIGURE 2.3: An example of a filtration of simplicial complexes $K_\epsilon, \epsilon \in \mathbb{R}$. A filtration requires that $K_\epsilon \subseteq K_{\epsilon'}$ for every $0 \leq \epsilon \leq \epsilon'$.

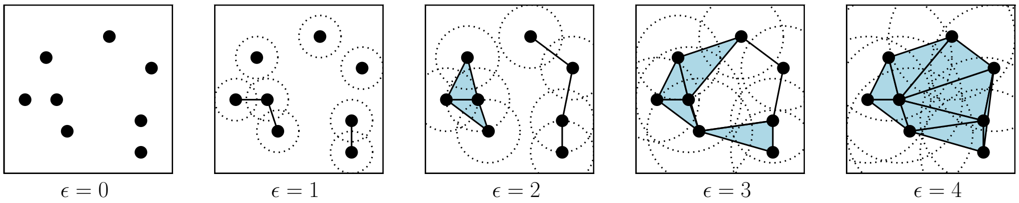


FIGURE 2.4: In fact, the filtration in Figure 2.3 is a Čech filtration. When the ϵ-balls around $p+1$ points intersect their convex hull is added as a p-simplex.

simplex generated by n points to the complex K whenever the Euclidean balls of radius ϵ centered around those n points have non-empty intersection. The *ϵ-Rips complex*, $R_\epsilon(X)$, adds the simplex generated by n points whenever the n points are pairwise at most ϵ distance apart. The Nerve Theorem (Hatcher, 2000, Corollary 4G.3) tells us that the Cêch complex is homotopy-equivalent to the union of the balls that we build it from: roughly, it represents the topology of the union of balls. This gives us a theoretical guarantee that we are correctly representing the topology of the space. Note that as increasing ϵ will only add new simplices to the complex for both the Vietoris-Rips and Cêch complexes, the collection of either complexes for either construction $(K_\epsilon)_{\epsilon \in \mathbb{R}_0^+}$ is a filtration of simplicial complexes, as required. See Figure 2.4 for an example of a Cêch filtration.

2.1.2 Chains and boundaries

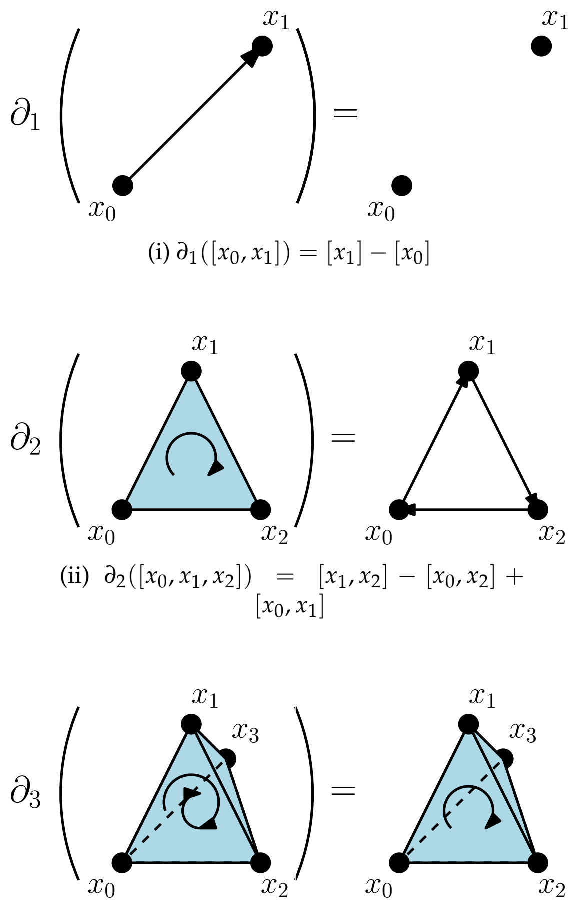
Let K be a simplicial complex with (potentially arbitrarily) ordered simplices. The *chain group* is the free Abelian group $C_p(K, G)$ generated by the p-simplices taking coefficients from an Abelian group G. A *p-chain* of K is an element of $C_p(K, G)$, and therefore a (formal) linear combination of p-simplices $\sum_{i=1}^n a_i \sigma_i$ where $\sigma_1, \ldots, \sigma_n$ are the p-simplices in K and $a_1, \ldots a_n \in G$ are the coefficients. More generally the coefficients can be taken from an R-module over a ring R. Typically $R = \mathbb{Z}$, making $C_p(K, R)$ a free Abelian group, or R is a field such as $\mathbb{F}_2$ or $\mathbb{R}$, making $C_p(K, R)$ a vector space over the field. Taking coefficients from $\mathbb{R}$ additionally makes the chain group an inner product space;

we will see why this is useful in Section 2.1.3 when we define the dual of the chain group.

Let $\sigma = [x_0, \dots, x_p]$ be an ordered p-simplex. The *p-boundary operator* is a linear map $\partial_p : C_p(K) \to C_{p-1}(K)$ for $p > 0$ taking a p-simplex to its $p-1$ dimensional boundary given by

$$\partial_p[\sigma] = \partial_p[x_0, \dots, x_p] = \sum_{i=0}^{p} (-1)^i [x_0, \dots, x_{i-1}, x_{i+1}, \dots, x_p].$$

For example, it sends a triangle to its constituent edges: $\partial_2([a,b,c]) = [b,c] - [a,c] + [a,b]$, or an edge to its vertices: $\partial_1([a,b]) = b - a$, as shown in Figure 2.5.



(i) $\partial_1([x_0, x_1]) = [x_1] - [x_0]$

(ii) $\partial_2([x_0, x_1, x_2]) = [x_1, x_2] - [x_0, x_2] + [x_0, x_1]$

(iii) $\partial_3([x_0, x_1, x_2, x_3]) = [x_1, x_2, x_3] - [x_0, x_2, x_3] + [x_0, x_1, x_3] - [x_0, x_1, x_2]$

FIGURE 2.5: Examples of the boundary operator on p-simplices.

The chain groups and boundary operators for all $p \in \mathbb{Z}_0$ give a sequence of chain groups connected with linear maps called the *chain complex*

$$\cdots \xrightarrow{\partial_{p+2}} C_{p+1}(K) \xrightarrow{\partial_{p+1}} C_p(K) \xrightarrow{\partial_p} C_{p-1}(K) \xrightarrow{\partial_{p-1}} \cdots .$$

In fact, the boundary of a boundary is always zero, as stated in the following lemma.

Lemma 2.2. *Let L be a chain complex. Given a chain complex consisting of chain groups $C_p(K, \mathbb{Z})$, $p = 0, 1, \dots$, and boundary operators $\partial_1, \partial_2, \dots$, the boundary of a boundary is always zero, that is*

$$\partial_{p-1}\partial_p = 0 \text{ for all } p > 0.$$

We will see why this lemma is useful when we formally define the homology groups in Section 2.2, but first we introduce the duals of the chain group and boundary operator.

2.1.3 Cochains and coboundaries

Given an oriented simplicial complex K and chain group $C_p(K, G)$ with coefficients in an Abelian group G, we define the cochain group

$$C^p(K, G) = \hom(C_p(K, G), G)$$

where $\hom(A, B)$ is the Abelian group of homomorphisms from A to B. For coefficients in a field F, $C^p(K, F) = \hom(C_p(K, F), F)$ is the dual vector space of $C_p(K, F)$. The basis of $C^p(K, G)$ is given by elementary cochains, where the elementary cochain associated with a p-simplex σ is given by $e_\sigma(\sigma) = 1$ and $e_\sigma(\tau) = 0$ for all $\tau \neq \sigma$.

We can then define the coboundary operator $\delta_p : C^p(K, \mathbb{R}) \to C^{p+1}(K, \mathbb{R})$ between cochain groups. Letting $f \in C^p(K, \mathbb{R})$ be a cochain, we define its boundary as the linear map $\delta_p f \in C^{p+1}$ given by

$$(\delta_p f)\sigma = (\delta_p f)[x_0, \dots, x_{p+1}] = \sum_{i=0}^{p+1} (-1)^i f([x_0, \dots, x_{i-1}, x_{i+1}, \dots, x_{p+1}]).$$

Note that the coboundary operator moves up a dimension,

$$\cdots \xleftarrow{\delta_{p+1}} C^{p+1}(K, \mathbb{R}) \xleftarrow{\delta_p} C^p(K, \mathbb{R}) \xleftarrow{\delta_p} C^p(K, \mathbb{R}) \xleftarrow{\delta_{p-2}} \dots.$$

in contrast to the boundary operator, which maps down through dimensions.

Recall that if we take coefficients from $\mathbb{R}$ then $C^p(K, \mathbb{R})$ is an inner product space. If we choose inner products $\langle \cdot, \cdot\cdot \rangle_{C^p}$ and $\langle \cdot, \cdot \rangle_{C^{p+1}}$, then we can define the adjoint of the coboundary operator as $\delta_p^* : C^{p+1}(K, \mathbb{R}) \to C^p(K, \mathbb{R})$ as the unique function δ_p^* that satisfies

$$\langle \delta_p f_1, f_2 \rangle_{C^{p+1}} = \langle f_1, \delta_p^* f_2 \rangle_{C^p}$$

for every $f_1 \in C^p$ and $f_2 \in C^{p+1}$. In fact, $\delta_p^* = \partial_p$ and $\partial_p^* = \delta_p$ (Hatcher, 2000, p.198). This means that when we compute the matrix of the boundary operator (which we do in Section 2.4.1), we can obtain the matrix of the coboundary operator by taking the transpose.

2.2 Homology and Persistence

We have introduced graphs and simplicial complexes to represent higher-order interactions within data, and a collection of free Abelian groups linked by (boundary, and coboundary) homomorphisms that represent that structure algebraically. Now we introduce homology groups: a notion going back to Poincaré (1895) that enables us to use these algebraic representations of simplicial complexes to study their topology. When we build our simlicial complexes from datasets, we are therefore able to study the topology of data.

2.2.1 Topology

In this section we briefly formalise the notion of a topological space and homeomorphism. In particular, following Munkres (1974), a *topology* on a set X is a collection $\mathcal{T}$ of subsets of X such that

(i) $\emptyset, X \in X$,

(ii) for any subcollection $\mathcal{T}' \in \mathcal{T}$, $\cup_{T \in \mathcal{T}'} T \in \mathcal{T}$, and

(iii) for any finite subcollection $\mathcal{T}' \in \mathcal{T}$, $\cap_{T \in \mathcal{T}'} T \in \mathcal{T}$.

In other words, a topology on X is a collection of subsets of X that includes both the empty set and X, and is closed under unions and finite intersections. A set X with a topology $\mathcal{T}$ is said to be a *topological space*. A *homeomorphism* between two topological spaces X, Y is a continuous bijective function $f : X \to Y$ with a continuous inverse $f^{-1} : Y \to X$. If such a function exists then we say that X and Y are *homeomorphic*. We can investigate homeomorphic spaces by proxy via *topological invariants*, properties of topological spaces that stay the same under homeomorphism.

2.2.2 Homology

We are now in a position to define homology groups: topological invariants that form the basis of much of Topological Data Analysis. Given the chain complex associated with a simplicial complex K we define the group of *p-cycles* as $Z_p(K) = \ker \partial_p$ and the group of *p-boundaries* as $B_p(K) = \text{im}\,\partial_{p+1}$. Note that by Lemma 2.2, $\delta_{p-1}\delta_p = 0$, so $B_p(K) \subseteq Z_p(K)$ and we can define the *p-homology group* as the quotient

$$H_p(K) = Z_p(K)/B_p(K).$$

We define the *pth Betti number* as $\beta_p = \text{rank}(H_p)$. Although it is not obvious from the definition, the homology groups and their ranks capture important topological information. In fact, homology groups and β_p are topologically invariants: β_0 counts the number of connected components, β_1 the number of holes, β_2 the number of voids (Edelsbrunner et al., 2002). This means homology groups extract useful topological information about simplicial complexes that we can use to understand their structure.

As with the boundary operator, we have $\delta_p\delta_{p+1} = 0$ for the coboundary operator so we can define the *cohomology groups* as

$$H^p(K, \mathbb{Z}) = \ker \delta_p / \text{im}\ \delta_{p-1}.$$

Having defined homology groups, we now briefly provide some intuition about why they are topologically invariant.

2.2.3 The invariance of homology groups

Let X and Y be topological spaces and $f, g : X \to Y$ be continuous functions. *Homotopy theory* lets us study a much looser definition of equivalence than homeomorphism. In particular, we say that f is *homotopic* to g if there exists a continuous function $H : X \times [0,1] \to Y$ such that for all $x \in X$, $H(x,0) = f(x)$ and $H(x,1) = g(x)$. The function H is called a *homotopy* between f and g, and we write that $f \simeq g$. We call two topological spaces A, B *homotopy equivalent* if there exists continuous maps $f : A \to B$ and $g : B \to A$ such that $f \cdot g \simeq \text{id}_A$ and $g \cdot f \simeq \text{id}_B$. Note that this is a much weaker definition of equivalence than homeomorphism; a homeomorphism would require the compositions $f \cdot g$ and $g \cdot f$ to be equal to the identity functions, whereas homotopy equivalence simply requires they can be deformed into them.

If K and L are simplicial complexes then a map $f : K \to L$ is a *simplicial map* if it sends each simplex in K to a simplex in L by a linear map taking vertices to vertices. Such a simplicial map $f : K \to L$ induces homomorphisms $f_{*,p} : H_p(K) \to H_p(L)$. In particular, if $\sigma = \sum_{i=1}^n a_i\sigma_i$ is a cycle in $H_p(K)$, then $f(\sigma)$ will be a (potentially trivial) cycle in $H_p(L)$ under the induced homomorphism given by

$$f(\sigma) = f\left(\sum_{i=1}^{n} a_i\sigma_i\right) = \sum_{i=1}^{n} a_i f(\sigma_i).$$

It can be shown that if $f : K \to L$ is a homeomorphism, then the induced homomorphism $f_{*,p} : H_p(K) \to H_p(L)$ is in fact an isomorphism for each dimension p (Hatcher, 2000, §2.1). In other words, if two simplicial complexes are homeomorphic (in fact, if they are homotopy equivalent), then their homology groups are isomorphic. This result confirms that we are capturing topological information by studying the homology groups.

2.2.4 Persistent homology

We have defined homology groups and seen that they capture the topological structure of a simplicial complex. We also now know that we can build simplicial complexes from point data using e.g. the Vietoris-Rips complex, but this construction requires a parameter determining how close the points must be to connect them/form a simplex. How should we choose that parameter? If ϵ is too small then the majority of points will remain disconnected, or if its too large every point will be connected and any potentially interesting structure will be obscured. The *persistence trick* answers this question by enabling homology groups to be computed over an entire filtration (i.e., for all choices of a parameter). We start by introducing persistence for a pair of simplicial complexes, and then introduce it over a filtration.

For complex pairs. We define a *complex pair*, written $K \subseteq L$, as two simplicial complexes K and L with the same set of vertices and such that all simplices of K are simplices of L. Let $K \subseteq L$ be a complex pair, then the boundary maps of K and of L can be written as a commutative diagram where vertical arrows are inclusion maps

$$\begin{tikzcd}
\cdots \arrow[r] & C^K_{p+1} \arrow[r, "\partial^K_{p+1}"] \arrow[d, hook, dashed] & C^K_p \arrow[r, "\partial^K_p"] \arrow[d, hook, dashed] & C^K_{p-1} \arrow[r] \arrow[d, hook, dashed] & \cdots \\
\cdots \arrow[r] & C^L_{p+1} \arrow[r, "\partial^L_{p+1}"] & C^L_p \arrow[r, "\partial^L_p"] & C^L_{p-1} \arrow[r] & \cdots
\end{tikzcd}$$

Now we can define *persistent homology* on a simplicial complex pair $K \hookrightarrow L$ by $H_p^{K,L} = Z_p^K/(B_p^L \cap Z_p^K)$. This represents homology classes present in $H_p(K)$ that are also present in $H_p(L)$, i.e., topological features in K that are still present in L. We can similarly define the *p-persistent Betti number* $\beta_p^{K,L} = \text{rank } H_p^{K,L}$, which counts these persistent topological features from K to L.

For a filtration. Let $\mathcal{F} = \{K_i\}_{i \in I}$ be a filtration indexed by a set I. Then we can define the *pth persistent homology group* from K_i to K_j as

$$H_p^{i,j} = Z_p^{K_i}/(B_p^{K_j} \cap Z_p^{K_i}),$$

and the pth persistent Betti numbers as $\beta_p^{i,j} = \text{rank } H_p^{i,j}$. Then $\beta_p^{i,j}$ counts the pth dimensional homology classes which persist from $H(K_i)$ to $H(K_j)$. Zomorodian and

Carlsson (2005) showed that the persistent homology can be represented via a basis $\{(b_k, d_k)\}_{k=1}^n$ with $b_k \in \mathbb{R}$ and $d_k \in \mathbb{R} \cup \infty$. This basis admits a topological interpretation: each b_k and d_k give the birth and death respectively of a topological feature (namely, a homology class in dimension p) through the evolving filtration. Note that some topological features can last forever (for example, in any non-trivial filtration there will always be at least one connected component). We say that the death time of such a topological feature is ∞. The birth and death times give rise to a convenient representation of the persistent homology that we now introduce.

2.2.5 Persistence diagrams

We can represent the points $\{(b_k, d_k)\}_{k=1}^n$ as a multiset in the extended plane $\mathbb{R} \times (\mathbb{R} \cup \{\infty\})$, called a *persistence diagram*. We can see in Figure 2.6i how the births (x-axis) and deaths (y-axis) are represented in the persistence diagram. Comparing to the filtration in Figure 2.4 which corresponds to the persistence diagram, we can see for example that there is one (one-dimensional) hole, which is born at $\epsilon = 3$ and dies (is filled in) at $\epsilon = 4$, detected by H_1. This is correspondingly represented in the persistence diagram by a point at $(3, 4)$. Similarly, the connected components correspond to points in H_0. Note as we move from $\epsilon = 2$ to $\epsilon = 3$ the addition of more simplices to the filtration means that we go from two connected components to one connected component. Therefore one connected component, which was born at $\epsilon = 0$, dies at $\epsilon = 3$, and so there is a point at $(0, 3)$ in the persistence diagram.. Defining $x_k = (b_k, d_k)$, persistence diagrams are often denoted as a discrete measure $\mu = \sum_{k=1}^n \delta_{x_k}$ in statistical applications where δ_x is the Dirac delta function. Persistence diagrams can be used interchangeably with the equivalent *persistence barcode*. The barcode is an equivalent representation of birth-death pairs consisting of n horizontal lines in the plane, which start at $x = b_k$ and end at $x = d_k$ for each k. A barcode is shown in Figure 2.6ii.

Given a persistence diagram D, we add all points (a, a), $a \in \mathbb{R}_0^+$, to D with infinite multiplicity. This allows the distance between diagrams with differing numbers of points to be well defined. Specifically the *p-Wasserstein distance* between two diagrams D_1, D_2 is given by

$$W_p(D_1, D_2) = \left(\inf_{\phi: D_1 \to D_2} \sum_{x \in D_1} ||x - \phi(x)||^p \right)^{1/p}, \tag{2.1}$$

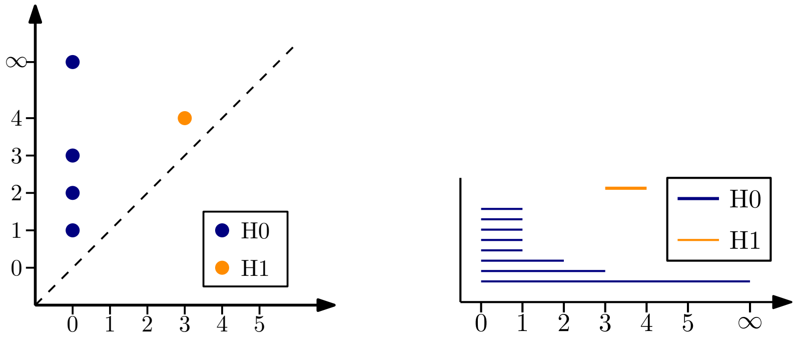
where the infimum is taken over all bijections ϕ. That is, each point in D_1 is matched to either a point in D_2 or the diagonal, and vice versa. The Wasserstein distance minimises the total distance between each matching. This matching distance is commonly used in many areas of research, and is sometimes referred to as the Earth mover's distance. We will use it in Chapter 5 to compare persistence diagrams. Note that the p-Wasserstein distance is a metric on the space of persistence diagrams, but not on isometry classes of point clouds - this is relevant in the clustering of context as the triangle inequality is important to the quality of the clustering (Rass et al., 2022). The p-Wasserstein distance can be extended to $p = \infty$ as the *bottleneck distance* given by

$$W_\infty(D_1, D_2) = \inf_{\phi : D_1 \to D_2} \sup_{x \in D_1} ||x - \phi(x)||.$$

The stability of persistence diagrams as representations of the underlying data has been studied with respect to the this distance. Given $f, g : X \to \mathbb{R}$, the first stability result for persistence diagrams characterised the stability of the persistence diagrams of the filtrations on X induced by f, g (where the filtration induced by f on X at ϵ is $K_\epsilon = \{x \in X : f(x) < \epsilon\}$) as

$$W_\infty(\mathcal{D}(f), \mathcal{D}(g)) \leq ||f - g||_\infty. \tag{2.2}$$

We will see in Section 3.1.1 that this stability result means that if we are sampling data from a distribution, then the persistence diagram of the sampled data is guaranteed to be close to the persistence diagram of the distribution.



(i) Persistence diagram

(ii) Persistence barcode

FIGURE 2.6: The persistence diagram (i) and persistence barcode (ii) corresponding to the filtration in Figure 2.3.

2.3 The Laplacian

We use the Laplacian as a feature vector throughout this thesis. We introduce it in this section, as well as recent work on the persistent Laplacian which we consider as a feature vector for machine learning for the first time in Chapter 4.

2.3.1 The graph Laplacian

The graph Laplacian, first introduced by Kirchhoff (1847) to study electrical circuits, is a linear operated associated to a graph $G = (V, E)$. It is given by $L = D - A$, where

D is the $|V| \times |V|$ *degree matrix* given by $D(v, v) = \text{degree}(v)$, and 0 otherwise, and A is the $|V| \times |V|$ *adjacency matrix* given by $A(v, v') = 1$ if $vv' \in E$, and 0 otherwise. We can also express it in terms of the boundary and coboundary operators. Note that we can do that as a graph is just a simplicial complex, where the nodes are 0-simplices and the edges are 1-simplices. Therefore, given a graph $G = (V, E)$ the boundary operator $\partial_1 : E \to V$ takes us down a dimension (from edges to vertices), and the couboundary operator δ_1, which is the adjoint of the boundary operator (so that $\delta_1 = \partial_1^*$) will take us up a dimension (from vertices to edges). Then the graph Laplacian is defined as the composition of the coboundary and boundary operators, $\Delta_0 \colon V \to E \to V$, $\Delta_0 = \partial_1 \circ (\partial_1)^*$. This coincides with the interpretation of the graph Laplacian as a discrete update step for the heat equation: it can be viewed as moving energy from nodes, onto connected edges, then sharing it amongst neighbouring vertices, i.e., it is aggregating energy in the neighbourhoods of each node.

The graph Laplacian, and in particular its eigenvalues, are deeply tied to the structure of the graph G. For example, Kirchoff's work introducing the graph Laplacian was generalised to show that the number of spanning trees in a connected graph is equal to $\frac{1}{n}\lambda_1, \ldots, \lambda_{n-1}$, where n is the number of vertices and $\lambda_1, \ldots, \lambda_{n-1}$ are the non-zero eigenvalues of the Laplacian (Biggs, 1993). The Cheeger inequality for graphs (Dodziuk, 1984) relates the second smallest eigenvalue of the Laplacian to the sparsest cut, a measure of how well connected a graph is. In particular, the first n eigenvectors of the Laplacian of a graph G (ordered by ascending size of their respective eigenvalues) gives us an optimal mapping of G into $\mathbb{R}^n$ (Belkin and Niyogi, 2001). Applying this to a graph built from points using pairwise distances as the edge weights (i.e. if A is a dissimilarity matrix) gives rise to the widely used spectral clustering (Pothen et al., 1990; Demmel, 1999). Convolutions in the eigenbasis of the graph Laplacian are (approximately) used in Graph Convolutional Networks (Kipf and Welling, 2017).

To illustrate how the spectrum of the Laplacian arises in practice, we demonstrate how its eigenvectors correspond to low dimensional representations of graphs (Belkin and Niyogi, 2001). Suppose you have a weighted graph $G = (V, E)$, where each edge $(i, j) \in E$ has a corresponding weight $w_{(i,j)}$. Then the (i, j)th entry in the weighted adjacency matrix A is given by $w_{(i,j)}$ if $(i, j) \in E$ or 0 otherwise. Suppose we wish to map G onto $\mathbb{R}$ in a way that best preserves structure. In particular, so that the higher the weight between connected nodes, the closer together the resulting points are. Specifically, for each node $i \in V$ we wish to find $y_i \in \mathbb{R}$ so that

$$\sum_{i,j \in V} (y_i - y_j)^2 a_{ij}$$

is minimised. Let $y = (y_1, \ldots, y_n)^T$ and observe that the non-zero entries of the degree matrix D are $d_{ii} = \sum_j a_{ij}$. Then since

$$\begin{aligned}
\sum_{i,j}(y_i - y_j)^2 a_{ij} &= \sum_{i,j}(y_i^2 + y_j^2 - 2y_i y_j)a_{ij} \\
&= \sum_i y_i^2 a_{ij} + \sum_i y_j^2 a_{ij} - 2\sum_{i,j} y_i y_j a_{ij} \\
&= 2\left(\sum_i y_i^2 d_{ii} - \sum_{i,j} y_i y_j a_{ij}\right) \\
&= 2\left(y^T D y - y^T A y\right) \\
&= 2y^T(D - A)y \\
&= 2y^T L y,
\end{aligned}$$

we get that

$$\underset{y^T Dy=1}{\arg\min} \frac{1}{2}\sum_{i,j}(y_i - y_j)^2 a_{ij} = \underset{y^T Dy=1}{\arg\min}\, y^T Ly, \tag{2.3}$$

where the constraint $y^T Dy = 1$ has been added to remove an arbitrary scaling factor. Then Equation 2.3 is minimised by the minimal eigenvalue λ from the generalised eigenvalue problem $Ly = \lambda Dy$, as

$$y^T Ly = \lambda \iff y^T Ly = \lambda y^T Dy \iff Ly = \lambda Dy.$$

If the graph is connected, then the first eigenvalue of L is 0 with eigenvector **1** constant one (we will see why later). To prevent this trivial solution we add the additional constraint that our solution must be orthogonal to **1**, to give the solution

$$y_{\text{opt}} = \underset{y^T Dy=1, y^T D\mathbf{1}=0}{\arg\min} \quad y^T Ly.$$

2.3.2 The combinatorial Laplacian

The graph Laplacian can be extended to simplicial complexes. This 'higher-order' Laplacian is referred to in the literature as the *combinatorial Laplacian, the higher Laplacian, or the discrete Hodge Laplacian*. Consider a simplicial complex K, and the chain complex along with the coboundary and boundary (dual) operators at each dimension,

$$\begin{tikzcd}
\cdots \arrow[r] & C^K_{p+1} \arrow[r, shift right, "\partial^K_{p+1}"'] & C^K_p \arrow[l, shift right, "(\partial^K_{p+1})^*"'] \arrow[r, shift right, "\partial^K_p"'] & C^K_{p-1} \arrow[l, shift right, "(\partial^K_p)^*"'] \arrow[r] & \cdots
\end{tikzcd}$$

The *up* and *down combinatorial Laplacians* are defined as the linear maps

$$\Delta^K_{p,\text{up}} = (\partial^K_{p+1})^* \circ \partial^K_{p+1} \text{ and } \Delta^K_{p,\text{down}} = \partial^K_p \circ (\partial^K_p)^*$$

respectively, and the *combinatorial Laplacian* is defined as

$$\Delta_p^K = \Delta_{p,\text{up}}^K + \Delta_{p,\text{down}}^K.$$

This generalises the graph Laplacian, as the up 0-Laplacian coincides with the graph Laplacian (see above), and the down 0-Laplacian is zero. Moreover, the combinatorial Laplacian captures the (non-persistent) homology at each dimension.

It turns out the p-Laplacian is connected to the p-homology group. In fact, we know that the kernel of the p-Laplacian is in direct correspondence with the p-cohomology group via the following theorem, due to Eckmann (1944/45).

Theorem 2.3. *Let K be an abstract simplicial complex. Then*

$$\ker \Delta_p(K) \cong H^p(K, \mathbb{R}).$$

This theorem means that the combinatorial Laplacian captures topological information, as well as the structural information contained in the non-zero eigenvalues. This makes it a natural candidate for application to data analysis, particularly given the recent introduction of the persistent Laplacian, which we now introduce.

2.3.3 The persistent Laplacian

The combinatorial Laplacian, like homology, has an extension to pairs of simplicial complexes. Lieutier (2014) first introduced the persistent Laplacian in a presentation given at Inria. Wang et al. (2019) later reintroduced the persistent Laplacian, and Mémoli et al. (2022) formalised and further studied the persistent Laplacian, characterising its theoretical properties and introducing an efficient algorithm for its computation. We follow their notation in our introduction.

Let $K \subseteq L$ be a simplicial pair and consider the subspace of C_p^L given by

$$C_p^{L,K} = \left\{ c \in C_p^L \ : \ \partial_p^L(c) \in C_{p-1}^K \right\} \subseteq C_p^L. \tag{2.4}$$

That is, $C_p^{L,K}$ consists of all simplices in C_p^K that have their boundary in $C_{p-1}^K \subseteq C_{p-1}^L$. Write $\partial_p^{L,K}$ for the restriction of the boundary operator to this subspace, i.e., $\partial_p^L|_{C_p^{L,K}}$. Then, the *persistent p-Laplacian* is defined by Mémoli et al. (2022) as

$$\Delta_p^{K,L} = \left(\partial_{p+1}^{L,K}\right)^* \circ \partial_{p+1}^{L,K} + \partial_p^K \circ (\partial_p^K)^*, \tag{2.5}$$

with the *up persistent p-Laplacian* given by $\Delta_{p,\text{up}}^{K,L} = \left(\partial_{p+1}^{L,K}\right)^* \partial_{p+1}^{L,K}$. The relation between each operator is shown in the diagram from Mémoli et al. (2022) below.

$$\begin{tikzcd}
C^K_{p+1} \arrow[rr, "\partial^K_{p+1}"] \arrow[dd, hook, dashed] & & C^K_p \arrow[r, shift right, "\partial^K_p"'] \arrow[dl, shift right, "\left(\partial^{L,K}_{p+1}\right)^*"'] \arrow[dd, hook, dashed] & C^K_{p-1} \arrow[l, shift right, "\left(\partial^K_p\right)^*"'] \arrow[dd, hook, dashed] \\
& C^{L,K}_{p+1} \arrow[ur, shift right, "\partial^{L,K}_p"'] \arrow[dl, hook', dashed] & & \\
C^L_{p+1} \arrow[rr, "\partial^L_{p+1}"] & & C^L_p \arrow[r, "\partial^L_p"] & C^L_{p-1}
\end{tikzcd}$$

The persistent Laplacian captures the same information as the persistent Betti numbers. In particular, the p-th persistent Betti number $\beta_p^{K,L} = \text{nullity } \Delta_p^{K,L}$ (Mémoli et al., 2022, Theorem 2.7). We investigate the usage of the persistent Laplacian for data analysis in Chapter 4.

2.4 Matrix Representations

As presented so far, each of these operators and representations are purely abstract. We detail here how to represent each of them in matrix form. With the notable exception of the persistent Laplacian, the computational packages to implement most of these representations are very well-developed. In fact, we provide the first Python implementation of the persistent Laplacian as part of Chapter 4, and the only implementation that computes the persistent Laplacian for cubical complexes.

2.4.1 Boundary operator

Let K be a simplicial complex and C_p^K, C_{p-1}^K be the chain groups generated by $[\sigma_0], \ldots, [\sigma_{n_p}]$ and $[\tau_0], \ldots, [\tau_{n_{p-1}}]$ respectively. The boundary operator $\partial_p : C_p^K \to C_{p-1}^K$ is given by

$$\partial_p([\sigma_i]) = \sum_{j=1}^{n_{p-1}} \lambda_{j,i}[\tau_j], \text{ for each } i = 1, \ldots, n_p.$$

The matrix representing the boundary operator is then given by

$$\mathbf{B}_p = \begin{array}{c} \\ \left(\begin{array}{cccc} \lambda_{0,0} & \lambda_{0,1} & \ldots & \lambda_{0,n_p} \\ \lambda_{1,0} & \lambda_{1,1} & \ldots & \lambda_{1,n_p} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{n_{p-1},0} & \lambda_{n_{p-1},1} & \ldots & \lambda_{n_{p-1},n_p} \end{array} \right) \end{array} \begin{array}{c} \\ [\tau_0] \\ [\tau_1] \\ \vdots \\ [\tau_{n_{p-1}}] \end{array}$$

with column labels $[\sigma_0] \quad [\sigma_1] \quad \ldots \quad [\sigma_{n_p}]$

for some $\lambda_{i,j}$ with $1 \leq i \leq n_p$ and $1 \leq j \leq n_{p-1}$, with the corresponding basis elements (in terms of fundamental chains) shown alongside the rows and columns of the boundary matrix, to aid interpretation. This is a matrix whose (i, j)-entry is ± 1 if the jth (oriented) $(p-1)$-simplex is a constituent of the ith (oriented) p-simplex, and 0 otherwise, with sign depending on orientations. It can be computed by Algorithm 1, which takes as input matrices $C_p \in \mathbb{R}^{(p+1)\times n_p}$ and $C_{p-1} \in \mathbb{R}^{p\times n_{p-1}}$ representing the ordered simplices $\sigma_0, \dots, \sigma_{n_p}$ and $\tau_0, \dots, \tau_{n_{p-1}}$ respectively, in terms of their constituent 0-simplices. Specifically, if C_p consists of oriented p-simplices $\sigma_0, \dots, \sigma_{n_p}$ with $\sigma_i = [\alpha_{i,0}, \dots, \alpha_{i,p}]$ for $i = 1, \dots, n_p$ and $\alpha_{i,j}$ being 0-simplices, then C_p is represented as a matrix as

$$\mathbf{C}_p = \begin{array}{c} \left(\begin{array}{cccc} \alpha_{0,0} & \alpha_{0,1} & \dots & \alpha_{0,p} \\ \alpha_{1,0} & \alpha_{1,1} & \dots & \alpha_{1,p} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{n_p,0} & \alpha_{n_p,1} & \dots & \alpha_{n_p,p} \end{array}\right) \end{array} \begin{array}{c} \sigma_0 \\ \sigma_1 \\ \vdots \\ \sigma_{n_p} \end{array} .$$

In Algorithm 1 the `ismember`(A, B) function represents the package of the same name available for R and Python which finds the positions I of elements of A in B. The boundary operator corresponds to multiplication of the matrix C_p by the *boundary matrix* B_p. Since the coboundary operator is the adjoint of the boundary operator, we can simply take the transpose of the matrix, i.e., the matrix B_p^T corresponds to the coboundary operator $\delta_p = \partial_p^*$.

Algorithm 1 Boundary matrix

```
Input Simplex matrices $C_p, C_{p-1}$
Output Boundary matrix $B_p$
$n_p \leftarrow |C_p|$
$n_{p-1} \leftarrow |C_{p-1}|$
$B_p \leftarrow$ ZeroMatrix($n_{p-1} \times n_p$)
for $i$ in 0..$p$ do
    $B_{\text{temp}} \leftarrow C_p[:, [0, \dots, i-1, i+1, \dots, p]]$
    $I \leftarrow$ ismember($B_{\text{temp}}, C_{p-1}$)
    for $j$ in 1..$n_p$ do
        $B[I_j, j] = (-1)^{i-1}$
    end for
end for
return $B_p$
```

2.4.2 Homology groups

Recall that homology groups are the quotient of the cycle group $Z_p = \ker \partial_p$ by the boundary group $B_p = \text{im } \partial_{p+1}$. A (not necessarily square) matrix is in *Smith normal form* if it is in the form

$$\begin{pmatrix} a_0 & 0 & 0 & & & \dots & 0 \\ 0 & a_1 & 0 & & & \dots & 0 \\ 0 & 0 & \ddots & & & & \\ \vdots & \vdots & & a_r & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ 0 & 0 & & & & & 0 \end{pmatrix}$$

for $0 < a_0 \leq \cdots \leq a_n$ and is such that $a_i | a_{i+1}$ for all i. Any integer-valued matrix can be reduced to Smith normal form via row and column operations (Howard, 2002, Theorem 5.8). As the rows and columns correspond to bases of C_p and C_{p-1} in our case we also keep track of the corresponding change to the bases. We reduce the boundary matrix B_p into Smith normal form to obtain the matrix $\tilde{B}_p$, updating the bases with row/column operations to get a basis $e_1, \dots e_{n_p}$ of C_p and a basis $f_1, \dots, f_{n_{p-1}}$ of C_{p-1}. In particular, we get

$$\tilde{B}_p = \begin{array}{c c} \begin{array}{cccccc} e_1 & \dots & e_r & e_{r+1} & \dots & e_{n_p} \end{array} & \\ \begin{pmatrix} a_1 & & & & & \\ & \ddots & & & & \\ & & a_{r_p} & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{pmatrix} & \begin{array}{c} f_0 \\ \vdots \\ f_r \\ f_{r+1} \\ \vdots \\ f_{n_{p-1}} \end{array} \end{array} .$$

Then $e_{r_p+1}, \dots, e_{n_p}$ is a basis for Z_p and $a_1 f_1, \dots a_{r_p} f_{r_p}$ is a basis for B_{p-1} (Zomorodian, 2005). It follows that the Betti number is given by $\beta_p = n_p - r_p - r_{p+1}$, as rank $Z_p = n_p - r_p$, rank $B_p = r_{p+1}$, and

$$\beta_p = \operatorname{rank} H_p = \operatorname{rank} Z_p / B_p = \operatorname{rank} Z_p - \operatorname{rank} B_p.$$

Persistent homology. Surprisingly, the entire persistent homology can also be computed with one matrix reduction. This algorithm is first given in Zomorodian and Carlsson (2005); we follow an algorithm for finite fields described in Edelsbrunner and Harer (2010). Given a simplicial complex K, begin by ordering all simplices $\sigma_1, \dots, \sigma_n$ so that whenever σ_i enters the filtration before σ_j we have that $i < j$, and whenever σ_i is a face of σ_j, again we have that $i < j$. The boundary matrix for this computation is an $n \times n$ matrix representing simplices of every dimension, and is given by

$$B(i,j) = \begin{cases} 1, & \sigma_i \text{ is a face of } \sigma_j \text{ with codimension } 1, \\ 0, & \text{else.} \end{cases}$$

Define $\texttt{low}(j)$ to be the row index of the lowest 1 in column j, or undefined if there is no 1 in column j. We update the boundary matrix B by iteratively adding columns from left to right. When $\texttt{low}(j_0) \neq \texttt{low}(j)$ for every non-zero column j_0, j we say that the matrix is *reduced*. This process, shown in Algorithm 2, will update the boundary matrix B until it is reduced. In fact, we can read the persistent homology off the reduced boundary matrix $\tilde{B}$. In particular, the zero columns correspond to the birth of a homology class. Suppose the zero column is column j. Then the addition of σ_j to the filtration gave birth to the topological feature, and so that is the birth time of the homology class. To find the death time, inspect row j. If there is a lowest 1 in row j, say at column k, then σ_k entering the filtration killed the homology class, and that similarly gives the death time. If there is no lowest 1 in that row, then the homology class never dies. The proof of this algorithm is in Edelsbrunner and Harer (2010, VII.1).

Algorithm 2 Persistent Homology Reduction

```
Input Boundary matrix $B$
Output Reduced matrix $\tilde{B}$
for $j$ in $1..n$ do
    while there exists $j_0 < j$ such that $\texttt{low}(j_0) = \texttt{low}(j)$ do
        Add column $j_0$ to column $j$
    end while
end for
```

2.4.3 Graph and combinatorial Laplacian

As the Laplacian is expressed directly in terms of the boundary operator it is relatively simple to express it now that we have an algorithm to compute a matrix representation of ∂_p. We have seen that the graph Laplacian is equal to

$$\Delta_{0,\mathrm{up}} = \partial_1 \partial_1^*,$$

and we can compute ∂_1 in matrix form B_1 using the algorithm above. Therefore the graph $\mathbf{\Delta}_1 = B_1 B_1^T$. It can be shown that $B_1 B_1^T = D - A$, where D is the degree matrix and A is the adjacency matrix, recovering the usual definition of the graph Laplacian matrix (Horak and Jost, 2013). Similarly, the combinatorial Laplacian for $p > 0$ is given by $\Delta_p = \partial_{p+1} \partial_{p+1}^* + \partial_p^* \partial_p$, which can be represented via the above computation of the boundary matrix as $B_{p+1} B_{p+1}^T + B_p^T B_p$.

2.4.4 Persistent Laplacian

Recall that the persistent Laplacian for a simplicial pair $K \hookrightarrow L$ is given by $\Delta_p^{K,L} = \left(\partial_{p+1}^{L,K}\right)^* \circ \partial_{p+1}^{L,K} + \partial_p^K \circ (\partial_p^K)^*$. We have matrix representations for the non-persistent boundary operator ∂_p^K, but not for the persistent part of this operator, given by $\Delta_{p,\text{up}}^{K,L} = (\partial_{p+1}^{L,K})^* \circ \partial_{p+1}^{L,K}$. Mémoli et al. (2022) show that the up persistent p-Laplacian can be computed via the Schur complement of certain matrices. In particular, given a block matrix

$$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \in \mathbb{R}^{n \times n}$$

with $D \in \mathbb{R}^{d \times d}$, the generalised Schur complement of D in M is $M/D = A - BD^\dagger C$, where † denotes the Moore-Penrose generalised inverse (Penrose, 1955; Agarwal and Flaut, 2017). Let I_K^L be the indices of rows in $\Delta_{p,\text{up}}^L$ which correspond to p-simplices not in K. Then $\Delta_{p,\text{up}}^{K,L} = \Delta_{p,\text{up}}^L / \Delta_{p,\text{up}}^L[I_K^L, I_K^L]$. We have developed an implementation of this in Python, and there is another implementation in C++ called HERMES (Wang et al., 2020), although this is restricted to a limited number of simplicial complex constructions and does not use the Schur complement to find the persistent Laplacian, instead relying on a projection of the non-persistent boundary matrix onto the basis of the group $C_p^{K,L}$.

Chapter 3

Literature Review

In this chapter we review the literature on statistics for persistence diagrams (Section 3.1) and topological machine learning (Section 3.2). We aim to provide high-level results and methodologies, contextualising how pieces of research fit into the broader Topological Data Analysis literature. We start by introducing statistics for persistence diagrams, both at the data level (answering questions like how does sampling potentially noisy data effect the computation of the persistence diagram?) and at diagram level (how do we compute standard statistical objects like means of persistence diagrams?). The former provides guarantees that persistence diagrams are a reasonable way to represent data, and the latter is useful in Chapter 5, when we use means of persistence diagrams to cluster them. We end this chapter by reviewing how topological tools are currently utilised in machine learning in Sections 3.2 and 3.3.

3.1 Statistics for Persistence Diagrams

3.1.1 Data-level statistics

Consider a set of points $X = \{x_1, \ldots, x_n\}$ sampled independently and identically distributed (i.i.d.) from a manifold $\mathbb{X} \subset \mathbb{R}^d$ which is the support of some measure μ. We know from the previous section that we can stably represent the topology of X at all scales, but two key questions remain.

1. When does $X_\epsilon := \bigcup_{x \in X} B_\epsilon(x)$ have the homotopy-type of the manifold $\mathbb{X}$? I.e., when is the topology of our approximation correct for a fixed scale?

2. When is the persistence diagram of X close to the persistence diagram of $\mathbb{X}$?

The first question is answered by Niyogi et al. (2008) in the case that μ is uniform. Given a condition number τ associated with $\mathbb{X}$, Niyogi et al. show that whenever $0 < \epsilon < \tau/2$

and

$$n > \gamma_1 \left(\log(\gamma_2) + \log\left(\frac{1}{\delta}\right)\right),$$

the homology of X_ϵ is equal to the homology of $\mathbb{X}$ with probability greater than $1 - \delta$. The values of γ_1, γ_2 are defined in Niyogi et al. (2008), but depend on ϵ and $\mathbb{X}$. This work was extended by Balakrishnan et al. (2012), who bound the minimax risk that the homology of X_ϵ is different to the homology of $\mathbb{X}$ even when some points are affected by noise that take them out of the vicinity of $\mathbb{X}$.

A notable first attempt at the second question was carried out by Bubenik and Kim (2006), where they give bounds on the distance between barcodes for points sampled on a d-hypersphere by the parametrised von Mises–Fisher distribution. They extend this work in Bubenik et al. (2010), where they consider bounds for nonparametric estimators for manifolds. It is answered more conclusively by Fasy et al. (2014a). They start by defining a distance with respect to any closed subset $A \subset \mathbb{R}^d$ as

$$d_A(x) = \inf_{a \in A} ||x - a||.$$

Then $X_\epsilon = \{y : d_X(y) < \epsilon\}$. Furthermore, this distance function allows them to use the stability theorem for persistence diagrams to show that

$$W_\infty(\mathcal{D}(X), \mathcal{D}(\mathbb{X})) \leq ||d_X - d_{\mathbb{X}}||_\infty = \mathbf{H}(X, \mathbb{X}),$$

where $\mathbf{H}$ is the Hausdorff distance, defined by

$$\begin{aligned}\mathbf{H}(A, B) &= \max\left\{\max_{x \in A}\min_{y \in B} ||x - y||, \max_{x \in B}\min_{y \in A} ||x - y||\right\}\\ &= \inf\{\epsilon : A \subset B_\epsilon \text{ and } B \subset A_\epsilon\}.\end{aligned}$$

Therefore the question of upper bounding the distance between $\mathcal{D}(X)$ and $\mathcal{D}(\mathbb{X})$ reduces to upper bounding $\mathbf{H}(X, \mathbb{X})$. Using this insight, they give explicit expressions for an ϵ (dependent on n) such that

$$\liminf_{n \to \infty} \mathbb{P}\left[W_\infty(\mathcal{D}(X), \mathcal{D}(\mathbb{X})) \leq \epsilon\right] \geq 1 - \delta.$$

Similar bounds were also given by Chazal et al. (2014b), however the presence of unknown constants means that finding explicit confidence intervals using their bounds is not tractable. Niyogi et al. (2011) show that if noisy data centred on a sum of Gaussians is sampled, you can recover the homology of the Gaussians with high confidence. Blumberg et al. (2012) consider distributions of persistence barcodes. By considering the process of computing the barcode of X sampled from $\mathbb{X}$ according to μ, you define an empirical measure that can be used to show that the barcode of X converges to the barcode of $\mathbb{X}$ as sample size increases. They also use this interpretation to give confidence intervals for persistence barcodes.

Outlier robustness. Although we've seen stability properties that guarantee small errors in the data only causes small changes in the persistence diagram, in practice outliers (a small number of data points with a large error) can cause persistence diagrams to be significantly incorrect. Instead of producing filtrations directly from sampled points, we can use sampled points to construct a density estimator in some outlier-resistant way, then compute the persistence diagram of that density estimator. To do so we must compute the persistence diagram via the sublevel sets of a function. Given a real valued function f, the sublevel sets are $f^{-1}((-\infty, c])$, with c varying over $(-\infty, \infty)$ as the scale parameter. There are several functions used in the literature to produce outlier-resistant persistence diagrams, all of which offer confidence bounds on their persistence diagrams. Chazal et al. (2011) introduce a distance-to-measure (DTM) function and Vishwanath et al. (2020) use reproducing kernels.

When these are computed in practice, the persistence diagrams of the sublevel sets of the function are not computed. Rather, the density estimator f is computed from the sampled points and evaluated at points on a grid over the domain of the function. This grid is triangulated and a filltration is induced from the values of the density estimator on the vertices. This was implemented by Fasy et al. (2014b).

Universal Probability Law. Bobrowski and Skraba (2022) study persistence diagrams generated from random point clouds and find a universal probability law. In particular, they find that the distribution of the death/birth ratio is *universal*, in the sense that the ratio is independent of the distribution that the point cloud is sampled from (as you the number of points sampled tends to infinity). This is analagous to the Central Limit Theorem for sampled random variables, and allows for hypothesis testing with persistence diagrams.

3.1.2 Diagram-level statistics

To define statistics at a diagram level we consider both statistics on the space of diagrams and on functional summaries of diagrams.

On diagrams. The first paper to consider statistics on the space of persistence diagrams was Mileyko et al. (2011). They consider the space

$$\mathcal{D}_p = \left\{ D : W_p(D, \Delta) < \infty \right\},$$

where Δ is the diagonal with infinite multiplicity. The role of the diagonal provides the main distinction between the Wasserstein distance between persistence diagrams and between probability measure. They also show that means and expectations exist on $\mathcal{D}_p$: the Fréchet mean and variance (Fréchet, 1944) extend the standard mean/variance to general metric spaces. Given a probability measure $\mathcal{P}$ on $\mathcal{D}_p$ with finite second moment, the *Fréchet variance* is given by

$$\mathrm{Var}_{\mathcal{P}} := \inf_{D \in \mathcal{D}_p} \left(F_p(D) = \int_{\mathcal{D}_p} W_p(D, \hat{D}) \mathrm{d}\mathcal{P}(\hat{D}) \right),$$

and the *Fréchet mean* is any diagram D that attains the variance:

$$\mathbb{E}_p = \{D : F_p(D) = \mathrm{Var}_{\mathcal{P}}\}.$$

Mileyko et al. (2011) show that $\mathbb{E}_{\mathcal{P}}$ exists whenever $\mathcal{P}$ has compact support. Therefore under mild assumptions the space of persistence diagrams has means and variances: concepts which underpin most of statistics.

With a proof that the Fréchet mean is well-defined, Turner et al. (2012) developed an algorithm to compute it. They show that their algorithm converges to a local minimum, and give a law of large numbers for Fréchet means of diagrams with samples drawn i.i.d according to some measure. We use an extension of this algorithm to compute the weighted Fréchet mean in Section 5.2.2. However, Turner et al.'s algorithm, as well as the contemporary algorithms to compute the Wasserstein distance, relied on the fundamentally combinatoric Hungarian algorithm (Munkres, 1957) to compute the optimal matching problem that is core to both problems. This problem was tackled by Lacombe et al. (2018), who framed the problem of computing the Fréchet mean as an optimal transport problem, and so was able to use the entropic Sinkhorn algorithm (Cuturi, 2013; Cuturi and Doucet, 2014) to compute them. As shown in Figure 3.1, this algorithm is far more scalable, and can be parallelised across GPU's for additional performance. An alternate approach to speed up the computation of the Fréchet mean, proposed by Vidal et al. (2020), is to iteratively consider only the most persistent points, only considering the least persistent (and therefore most noisy) points towards the end of the computation. Another technique to speed up the process of finding the optimal matching required by the Wasserstein distance was provided by Kerber et al. (2017), who follow the approach of Efrat et al. (2001) to improve the speed to compute the bottleneck distance by pruning the search for potentially matched points in the persistence diagram. They also improve the speed to compute the 1-Wasserstein distance by implementing a geometric version of the auction algorithm (Bertsekas, 1988), an iterative method to solve the optimal matching problem.

Turner (2020) studies the median of distributions f persistence diagrams. Munch et al. (2015) introduce a 'probablistic Fréchet mean' that, unlike the ordinary Frêchet mean, is unique and continuous. The Expected Persistence Diagram (EPD) is a statistical summary of multiple persistence diagrams. Divol and Lacombe (2021) propose a version of the empirical EPD which is quick to compute, is guaranteed to have small support, and approximates the EPD with near-optimal rates.

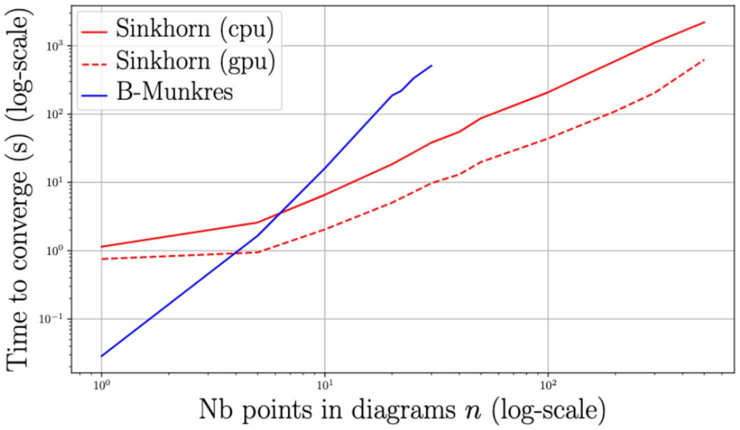


FIGURE 3.1: Turner et al.'s algorithm (B-Munkres) vs Lacombe et al.'s algorithm (Sinkhorn). Figure reproduced from Lacombe et al. (2018)

Persistence Landscapes. Another approach to define statistics on persistence diagrams is to map a diagram to a function in a Banach space called a *persistence landscape* (Bubenik, 2015). This comes with several advantages, the primary being that you can view the persistence landscape as a Banach space-valued random variable, which gives you access to a wealth of pre-existing theory (Ledoux and Talagrand, 2011). Also important is that means of landscapes are unique and easy to precisely compute, neither of which is true for Fréchet means. Formally, the persistence landscape is a function $\lambda : \mathbb{N} \times \mathbb{R} \to [-\infty, \infty]$, although it can be thought of as a sequence of functions $\lambda_k : \mathbb{R} \to [-\infty, \infty]$, where $\lambda_k(t) = \lambda(k, t)$. Define

$$\lambda_k(t) = \sup\left\{m \geq 0 : \beta^{t-m,t+m} \geq k\right\},$$

where $\beta^{i,j}$ is the pth persistent Betti number. To help visualise the persistence landscape, we extend it to a function $\bar{\lambda} : \mathbb{R}^2 \to [-\infty, \infty]$ given by

$$\bar{\lambda}(x,t) = \begin{cases} \lambda(\lceil x \rceil, t), & x > 0, \\ 0, & x \leq 0. \end{cases}$$

An example of a persistence landscape, including an image of the above extension is shown in Figure 3.2.

Consider a set of landscapes λ^i computed from data $X^i = \{x_1^i, \dots, x_m^i\}$ sampled from $\mathbb{X}$ according to some measure μ, for $i = 1, \dots, n$. Then a unique mean can be computed pointwise, i.e.,

$$\operatorname{mean}(\lambda^1, \dots, \lambda^n) = \frac{1}{n}\sum_{i=1}^{n} \lambda^i : \mathbb{N} \times \mathbb{R} \to [-\infty, \infty].$$

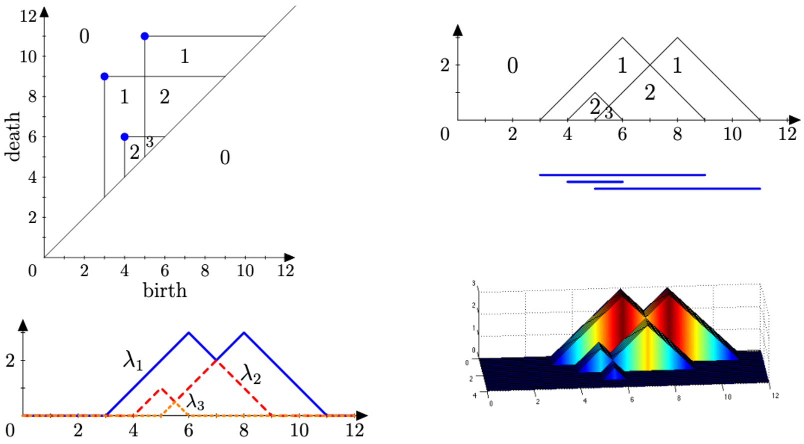


FIGURE 3.2: A persistence diagram (top left) with its rank function counting the bars in the corresponding barcode (top right), and the persistence landscape (bottom left) with its extension $\bar{\lambda}$ (bottom right). Figure reproduced from Bubenik (2015).

The process of computing the landscape of points sampled from μ gives a Banach space-valued random variable. Using known results about such random-variables (Ledoux and Talagrand, 2011) gives a Central Limit Theorem, Strong Law of Large Numbers, and confidence intervals for persistence landscapes.

The persistence landscape was further studied by Chazal et al. (2014a, 2015) who show that the mean landscape can be used an unbiased estimate for the expected value of a sampled landscape, and a biased estimate for the landscape of $\mathbb{X} = \text{supp}(\mu)$. They also propose a similar functional summary, that they call the silhouette. Robins and Turner (2015) perform PCA on rank functions, which are closely tied to persistence landscapes, to analyse colloids and sphere packing. Amongst other things, landscapes have been used to analyse protein bindings (Kovacev-Nikolic et al., 2016), financial crashes (Gidea and Katz, 2018), microbiomes (Petrov et al., 2020), and swarm behaviour Corcoran and Jones (2016). A comprehensive list of applications can be found in Bubenik (2020).

3.2 Topological Machine Learning

With the theory of persistence and relevant statistics better understood, it became natural to ask how to use these tools within machine learning.

A note on weighting functions: many techniques require weighting points in the persistence diagram. This can be done in the interest of stability, as without giving a low weight to points with low persistence a point moving off the diagonal will introduce a discontinuity (Adams et al., 2017). In addition, the Homology Inference Theorem (Cohen-Steiner et al., 2007) tells us that when we have a sufficiently dense sample of points, the points in the diagram that are most representative of the features of the distribution have low birth values and high persistence, so it makes sense to weight higher persistence points more. However, its been found that points with small (Patrangenaru et al., 2018) or medium (Bendich et al., 2016) persistence can be more discriminative in some machine learning tasks: it is likely that the value of points with different levels of persistence will in fact rely on the underlying data, and the weighting should be chosen with that in mind.

3.2.1 Mapper

The Mapper algorithm (Singh et al., 2007) was one of the first algorithms to take advantage of a topological viewpoint of data. Although it doesn't use persistence diagrams, it uses topological ideas to build an algorithm that can map complex high-dimensional datasets into a simplicial complex. This low-dimensional representation of the dataset is typically more interpretable than the data itself, and can be used to highlight areas of interest.

Suppose we have a dataset X and a continuous map $f : X \to Z$, for some parameter space Z. If $Z = \mathbb{R}$ then mapper computes a stochastic Reeb graph (Reeb, 1946); if $Z = \mathbb{R}^2$, then you get a 2-dimensional simplicial complex; and if $Z = S^1$ then you produce a map from the graph to the unit circle. Suppose now that $Z = \mathbb{R}$. Let $I = \text{range}(f|_X)$, and $\mathcal{S}$ be a partition of I. Specifically, let $\mathcal{S} = \{I_j\}$ be a collection of intervals covering I, described by the intervals length l and percentage overlap p. Then for each $I_j \in \mathcal{S}$, define $X_j = \{x \in X : f(x) \in I_j\}$, so that $\{X_j\}$ defines a cover of X. Cluster each X_j (using a clustering method of your choice) to get clusters $\{X_{jk}\}$. Finally, you obtain a simplicial complex with vertices $\{X_{jk}\}_{j,k}$ and an edge from X_{jk} to X_{lm} whenever $X_{jk} \cap X_{lm} \neq \emptyset$. In this way you map large dimensional datasets into a simplicial complex in the plane. The results of Mapper depend heavily on your choice of filter function f, as well as your clustering technique and parameters l, p, but experiments have proven Mapper capable of being very successful when complemented with sufficient human input: semi-supervised labelling of representative samples of the resulting simplicial complex is used to classify the underlying data. Lum et al. (2013) offered applications of Mapper to datasets including tumours, voting, and player performance in the NBA. Famously, Nicolau et al. (2011) used Mapper to identify a new subtype of breast cancer with high survival rates (see Figure 3.3). A spin-off company, Ayasdi, was founded with the Mapper algorithm at the core of its algorithm. Although less detail is offered on their workings, they use the Mapper algorithm for applications including protein analysis (Gilmore et al., 2016; Sardiu et al., 2017), traumatic brain injury biomarkers (Nielson et al., 2017), and infection tolerance (Torres et al., 2016; Louie et al., 2016).

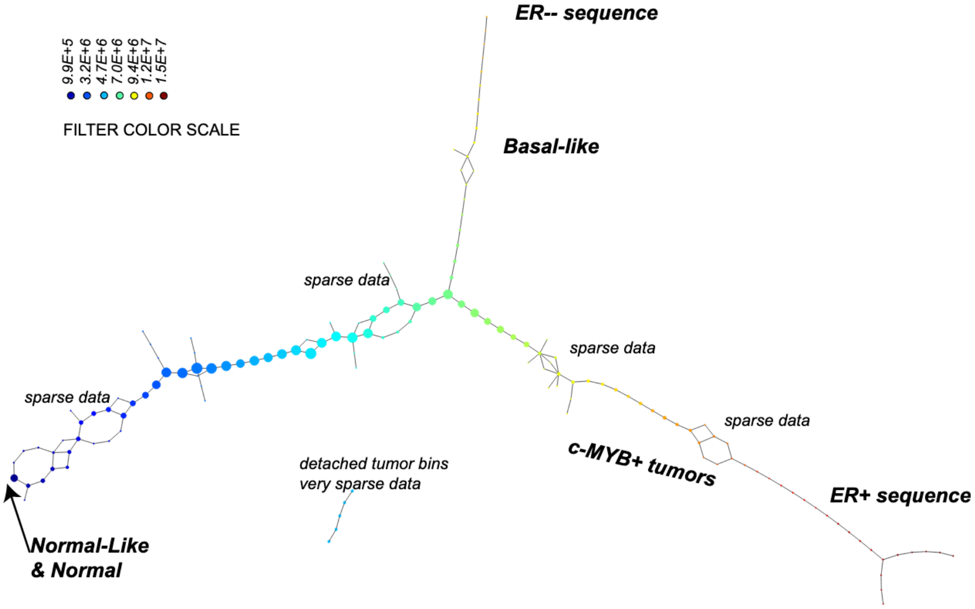


FIGURE 3.3: The results of the Mapper algorithm on breast cancer tumour genetic data. The resulting complex has structure that is deeply tied to the underlying data. The c-MYB+ tumour type was discovered using Mapper. Figure reproduced from Nicolau et al. (2011).

3.2.2 Vectorisation and kernel methods

Because the space of persistence diagrams with the Wasserstein distance is non-Hilbertian (Turner and Spreemann, 2019), it is difficult to apply existing ML methods that require more than the metric structure offered under the Wasserstein distance. Techniques to map diagrams into a more applicable form fall into two categories: finite vectorisation, or the definition of a positive-definite kernel on diagram space. Most of these techniques have strong stability properties and empirically perform well. In the words of Bubenik (2020), "perhaps since the persistence diagram is such a rich invariant, it seems that any reasonable way of encoding it in a vector works fairly well". Some featurisation methods do, however, have both stronger theoretical properties and empirical performance, and regardless we will see shortly that any mapping into a Hilbert space necessarily deforms the Wasserstein metric structure.

Feature vectors. Adcock et al. (2016) define a ring of algebraic functions on barcodes that they use as coordinates to vectorise diagrams. They use these coordinates with success as inputs to SVM's, but they are not stable with respect to the Wasserstein distance. Kališnik (2018) gives tropical coordinates for persistence barcodes that she shows are stable with respect to the Wasserstein distance. The idea behind both of these papers is to identify the space of persistence barcodes with the orbit of some action, then find a generating set of the space that can be used to produce feature vectors. Fabio and Ferri (2015) extended work originally developed for size functions (Ferri and Landi, 1999) that identifies a persistence diagram's points with the complex

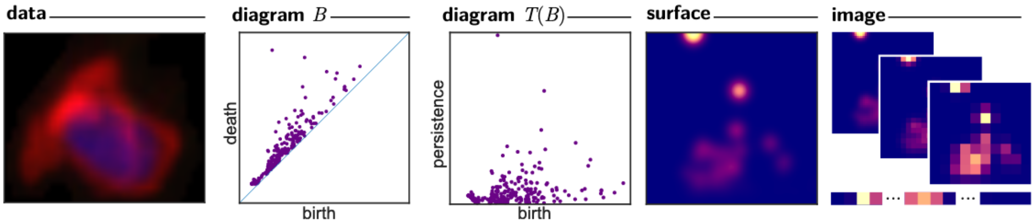


FIGURE 3.4: The pipeline from data to persistence image. Figure reproduced from Adams et al. (2017).

roots of some polynomial, then uses the coefficients of that polynomial as a vector representation. Pachauri et al. (2011) bin a kernel density estimate of a persistence diagram to obtain a finite vector, which they use to analyse cortical thickness to study neural disorders. Carrière et al. (2015) map a persistence diagram to a vector of its points pairwise distances, proving that the mapping is stable with respect to the bottleneck distance. Bendich et al. (2016) transforms a persistence diagram from birth/death axes to the equivalent birth/persistence axes, then bins the plane and counts the number of points in each bin to form a vector representation of the diagram. This approach is used by Bendich et al. (2016) to analyse brain arteries, but the vector representations are not stable. This was fixed by Adams et al. (2017) with the development of *persistence images*, a vectorisation technique that is commonly used in practice. They place a Gaussian, weighted by persistence, on each point of a diagram with birth/persistence axes. One obtains the persistence image by integrating the summated Gaussians over some binning of the space. This gives a stable representation of the persistence diagram. Specifically, given a diagram $D = \{(x, y)\}$, define a map $T((x, y)) = (x, y - x)$ and let ϕ_u be the normalised symmetric Gaussian in the plane, with mean μ and variance σ^2. Let f be a weighting function. Then the persistence surface is defined as

$$\rho_D(z) = \sum_{u \in T(D)} f(u)\phi_u(z).$$

Discretise a relevant subdomain of ρ_D into a grid $\{p_1, \dots, p_n\}$ with n 'pixels'. Then integrating over each p_i gives the persistence image:

$$I(\rho_D) = \left[\iint_{p_i} \rho_D \mathrm{d}y\mathrm{d}x \right]_{i=1}^{n}.$$

Figure 3.4 gives an example of the entire pipeline. Adams et al. (2017) use an SVM on the resulting vectors to discriminate between synthetic data and to classify discrete dynamical models. Kimura et al. (2018) use PCA on persistence images to identify trigger sites in materials science. They were also used to analyse 3d surface scans by Zeppelzauer et al. (2016). Chen et al. (2015) offer a statistical analysis of the persistence surface, which they call the persistence intensity function. Turkes et al. (2022) evaluate the efficacy of persistence images and persistence landscapes as feature vectors fed to

SVMs across a range of tasks, demonstrating that they can powerfully capture structures such as curvature and convexity.

More recent approaches include *template functions* given by Perea et al. (2019). They describe a general vectorisation technique using template functions that are only required to be continuous and compactly supported. They provide realisations using tent maps and interpolating polynomials, and match or exceed the state of the art for previous vectorisations. Zieliński et al. (2019) adapt bag-of-words vectorisation, a technique that vectorises by binning the diagram, but the bins are decided by k-means clustering. Chevyrev et al. (2020) realise the barcode as a path in some vector space, and use the path signature as a feature vector. Moon and Lazar (2020) provide a two-stage hypothesis test specifically for vectorised persistence diagrams. Hacker (2020) develop k-simplex2vec, an extension of the celebrated node2vec algorithm (Grover and Leskovec, 2016). This can learn mappings of simplicial complexes into Euclidean space in a way that respects the structure of the simplicial complex.

Kernel methods. Given the prevalence of kernel methods in machine learning (Hofmann et al., 2008), it makes sense to look for a kernel on the space of persistence diagrams and gain access to those techniques for persistence diagrams. The p-Wasserstein distance does not naturally lead to a valid kernel on the space of persistence diagrams for any $1 \leq p \leq \infty$ (Reininghaus et al., 2015), but work has been done on defining a valid positive-definite kernel (Cuturi, 2009) on persistence diagram space. The first kernel for persistence diagrams was proposed by Reininghaus et al. (2015), who propose a kernel given by the solution to a heat diffusion problem with Dirac delta representations of diagrams, along with their reflections in the diagonal, as the initial conditions. When solved, this gives the kernel between persistence diagrams D and E given by

$$k(D,E) = \frac{1}{8\pi\sigma} \sum_{\substack{p \in D \\ q \in E}} \exp\left(\frac{-||p-q||^2}{8\sigma}\right) - \exp\left(\frac{-||p-\bar{q}||^2}{8\sigma}\right),$$

where σ can be interpreted as a scale parameter and $\bar{q}$ denotes the reflection of q in the diagonal. This gives a stable positive-definite kernel on persistence diagrams. Note that without the addition of the reflection in the diagonal, this is similar to the persistence surface in persistence images (Adams et al., 2017). Reininghaus et al.'s kernel was used by Kwitt et al. (2015) as a basis for statistics on diagrams. Another function on diagrams that we've encountered is the persistence landscape (Bubenik, 2015). In fact, Reininghaus et al. (2015) show that for the 2-Wasserstein distance, the landscape admits a positive-definite kernel. Similar to the persistence landscape is the Persistence Indicator Function (Rieck et al., 2019b) (sometimes called the Betti curve), which is also shown to admit a kernel. Another stable kernel was proposed by Kusano et al. (2016) who add weighted Gaussians to define the *Persistence Weighed Gaussian Kernel*. Carriere et al. (2017) develop a stable kernel based on an approximation of the 1-Wasserstein distance called the Sliced Wasserstein distance (Rabin et al., 2011). Unlike the other kernels we've seen, they analyse how their kernel deforms the original metric structure on diagram space, and prove that their kernel is discriminative with respect to the original diagrams. They further show that their kernel empirically outperforms the previous kernels, and offer a technique to more efficiently approximate their kernel matrix. Le and Yamada (2018) use the Fisher information metric (Amari and Nagaoka, 2000), a Riemmanian distance between distributions, to define another kernel for persistence diagrams.

Metric deformation. The deformation analysis of Carriere et al. (2017) is studied more generally. We know that when defining a positive-definite kernel on diagram space, you are implicitly mapping into a Hilbert space. As the p-Wasserstein metric structure on the space of persistence diagrams does not admit an isometry into a Hilbert space for any $1 \leq p \leq \infty$ (Turner and Spreemann, 2019), we necessarily distort diagram space with this implicit mapping. An effort has been made to lower-bound the distortion caused. Carrière and Bauer (2019) study embeddings in which the metric is distorted linearly, showing that for infinite dimensional Hilbert spaces a lower bound is dependant on the cardinality of the diagram, and for finite dimensional Hilbert spaces finding such an embedding is impossible. Coarse embeddings into Hilbert spaces are also studied, which require the distortion of the metric is uniform. A coarse embedding of persistence diagram space with the p-Wasserstein distance is shown to be impossible for $p = \infty$ by Bubenik and Wagner (2019) and for $p > 2$ by Wagner (2019). The first positive result regarding embedding persistence diagrams into Hilbert space was given by Mitra and Virk (2021), and states that the space of persistence diagrams on n points with the p-Wasserstein distance does coarsely embed into Hilbert space for any p. However, they also show that the space of persistence diagrams with finitely many points does not coarsely embed into Hilbert space for $p = \infty$.

3.2.3 Deep learning

Although we've seen tailored representations of persistence diagrams, deep learning has demonstrated over the last decade that learning a task-specific representation of persistence diagrams could be more effective. The first efforts to use machine learning with persistence diagrams focused on learning parameters to better represent the diagram. Later efforts integrated various learnt and fixed representations of persistence diagrams, generally referred to as *persistence-based summaries*, into machine learning techniques in various ways. To use deep learning, its important to understand how we can differentiate through persistence diagrams.

Differentiability. Let K be a simplicial complex, f the filtration map, and ω some parameters that we wish to optimise. The differentiability of persistence diagrams was studied by Poulenard et al. (2018), who introduce a map $\pi : D \to K$, given by $(x, y) \mapsto (v_x, v_y)$, which maps points in the persistence diagram to the simplices in the underlying filtration which give rise to the topological feature that the diagram point represents. Slightly perturbing the filtration function f can always make this map bijective (Skraba et al., 2020), so it can be used to map a point in the diagram back to the simplex (and thus, the data points) that cause the birth or death of a topological feature. Using this map, they show that the derivative of the persistence map is locally well-defined, and show that for a point (x, y) in a persistence diagram we can get the derivative

$$\left(\frac{\partial x}{\partial \omega}, \frac{\partial y}{\partial \omega}\right) = \left(\frac{\partial f}{\partial \omega}\pi((x,y))|_b, \frac{\partial f}{\partial \omega}\pi((x,y))|_d\right).$$

Given a function $\mathcal{F} : D \to \mathbb{R}$ that we wish to optimise, we can use the chain rule to obtain the derivative of $\mathcal{F}$ on $D = \{(x_i, y_i)\}_{i \in \mathcal{I}}$ with respect to the parameters ω by computing

$$\begin{aligned}\frac{\partial \mathcal{F}}{\partial \omega} &= \sum_{i \in \mathcal{I}} \frac{\partial \mathcal{F}}{\partial x_i} \cdot \frac{\partial x_i}{\partial \omega} + \frac{\partial \mathcal{F}}{\partial y_i} \cdot \frac{\partial y_i}{\partial \omega} \\ &= \sum_{i \in \mathcal{I}} \frac{\partial \mathcal{F}}{\partial x_i} \cdot \frac{\partial f}{\partial \omega}\pi(x)|_b + \frac{\partial \mathcal{F}}{\partial y_i} \cdot \frac{\partial f}{\partial \omega}\pi(y)|_d.\end{aligned}$$

Poulenard et al. (2018) used their differentiability for what they call *function simplification and alignment*, forms of regularisation and loss that would be utilised by later work in a deep learning context. Leygonie et al. (2019) studied the differentiability of maps to and from persistence diagrams using the theory of diffeological spaces; their framework comprehensively answers the question of differentiability for persistence. Carrière et al. (2020) simplify the process of optimising through the persistence map, which they characterise as simply a permutation of a vector containing the filtration indices. With this characterisation they give a proof of convergence for stochastic gradient descent of a loss function $\mathcal{L}$ composed with the persistence map, as long as $\mathcal{L}$ is locally Lipschitz. Solomon et al. (2021) introduce a novel backpropogation scheme that is both faster and more stable.

Learnt vectorisation. Instead of fixed vectorisations, Hofer et al. (2017, 2019b) first proposed learning stable task-specific representations of persistence diagrams in the form of an input layer for neural networks that accepted diagrams. The representation is similar to some kernel functions for diagrams (Reininghaus et al., 2015; Kusano et al., 2016), but now the parameters of the representation are learnt rather than fixed. Using this approach achieved the state of the art in classifying social network graphs. Figure 3.5 shows the pipeline for their layer. PersLay is a general topological layer architecture proposed by Carrière et al. (2020). It is defined by

$$\text{PersLay}(D) = \Gamma(\{w(p) \cdot \phi(p)\}_{p \in D}),$$

where w is a weighting function, ϕ is a function on a point of the diagram (e.g., a Gaussian centred at the point), and Γ is a permutation invariant function (such as

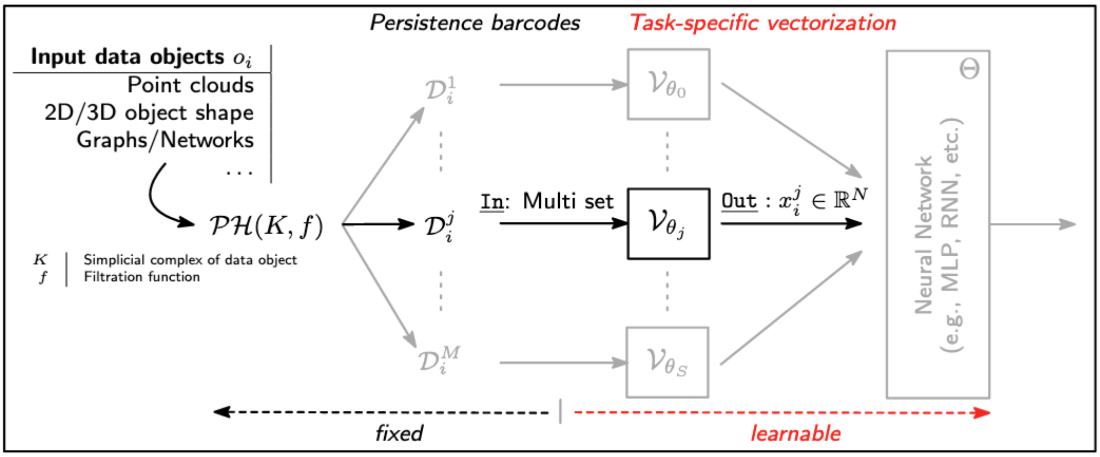


FIGURE 3.5: Instead of fixed representations, Hofer et al. (2017) propose learning task-specific vectorisations using a parameterised function. Figure reproduced from Hofer et al. (2019b).

max, min, average, etc). Note that this formulation is very general, and by selecting specific functions Γ, w and ϕ this can represent many of the diagram representations that we've already seen. As we can parameterise Γ, w and ϕ, we can learn them using PersLay. Using a variety of different functions for Γ, w and ϕ, they are able to match or surpass the previous state of the arts for dynamical systems analysis and graph classification. Note that PersLay can be used with topological representations other than persistence diagrams: Elkin and Kurlin (2020) introduce the *mergegram*, an extension of the 0-persistent homology with desirable theoretical properties that can outperform other invariants when using PersLay (Elkin and Kurlin, 2021). Kim et al. (2020) learn representations for peristence landscapes using the previously introduced DTM function (Chazal et al., 2011) for outlier robustness, and show that doing so reduces the metric deformation implicit to any mapping of a diagram into a Hilbert space. They compute a collection of weighted persistence diagrams, and process them with a parameterised differentiable map into a vector. Their layer is stable and differentiable, so can learn the parameters, and is shown to be experimentally comparable to the similar approach by Hofer et al. (2017). Zhao and Wang (2019) define a weighted kernel, where the weighting function is learnt. Using this kernel as a basis for classification, they match or outperform the state of the art on some graph classification tasks.

Topological loss and regularisation. Chen et al. (2018) first proposed a topological regularisation term. They define the *robustness* of a connected component, a concept that is closely related to persistence. Their regulariser is the sum of squares of the robustness of each of the connected components of the decision boundary. They analyse the differentiability of their regulariser and empirically demonstrate that it improves classifier performance on several synthetic and real-world datasets. Gabrielsson et al. (2020) define a topological loss function inspired by the tropical coordinates vectorisation technique developed by Adcock et al. (2016). Specifically, given a diagram $D = \{(b_i, d_i)\}_i$

with points in descending order of persistence, the loss is given by

$$\mathcal{L}(p, q, i_0, D) = \sum_{i=i_0}^{|D|} |d_i - b_i|^p \left(\frac{d_i + b_i}{2} \right)^q ,$$

where p, q define a polynomial, and i_0 represents prior topological knowledge. By ignoring the first i_0 most persistent points, you can promote having i_0 topological features. This can be used either as topological regularisation: having a low value for i_0 promotes topological simplicity; or to incorporate topological prior knowledge by choosing a value that you know is correct (e.g., an 8 in MNIST has two loops). They empirically demonstrate that topological regularisation leads to cleaner generated data and incorporating topological prior knowledge can improve the generation quality. Gallego-Posada and Forré (2021) investigate a topological regularisation term that aims to preserve the simplicial structure of the data. Interestingly, it had been hypothesised that the use of topological features could improve ML models' robustness against adversarial attacks (Chakraborty et al., 2018). Gabrielsson et al. demonstrate for the first time that this is true: adding their topological loss to an MNIST classifier took an adversarial attack (Goodfellow et al., 2015) from 100% success rate to 25.2% success rate. Clough et al. (2020) also offer a topological loss function, which uses topological prior knowledge in the form of Betti numbers.

Autoencoders and GANs. Hofer et al. (2019a) propose a topological loss for autoencoders called the *connectivity loss*. Imposing structure on the latent space of autoencoders has been shown to improve performance (Makhzani and Frey, 2014; Vincent et al., 2010; Rifai et al., 2011). Hofer et al. control the structure of the latent space using prior topological knowledge. Intuitively, the connectivity loss encourages the autoencoder to topologically organise the latent space by controlling the number of connected components. In their example, they focus on one-class learning and use their loss to encourage the latent space to have two connected components. Moor et al. (2019) also propose a form of topological autoencoders. They restrict the latent space to have the same topology as the input space by adding a topological term to the reconstruction loss. They introduce a *directional topological loss*: any diagram D can be represented by the distance matrix of the underlying points A and the pairings π that link birth/death simplices to the data points that give rise to them. Moore et al. link the birth/death simplices of the input space to the pairwise distances of the latent space , and vice versa. This means that rather than just attempting to match topology, they pass information about the topologically important points between the input and latent space. With synthetic data, they find that allows them to maintain important structural information that ordinary autoencoders lose: see Figure 3.6 for details. Dey and Das (2020) use topological regularisation to constrain the topology of the latent space in generative adversarial networks (GANs). Zhou et al. (2021) define a topological metric for disentanglement on the generative model in GANs. An interesting insight on any dimensionality reduction

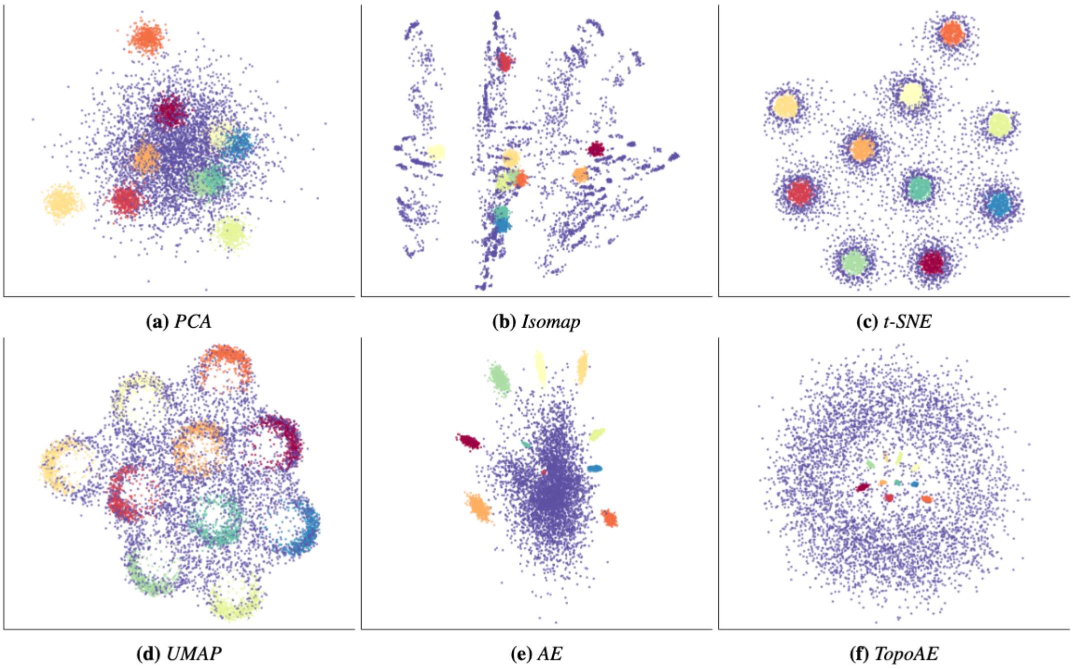


FIGURE 3.6: Given a dataset with nested spheres, only the topological autoencoder preserves the nesting relationship in the latent space. Figure reproduced from Moor et al. (2019).

technique is given by Landweber et al. (2016), who show that any function $\mathbb{R}^m \to \mathbb{R}^n$, with $m > n$, is either discontinuous (and therefore make close points distant) or collapses an unbounded domain to a single point (so is losing information), meaning that any dimensionality reduction methodology can be deceptive.

Convolutional networks. Zhao et al. (2020) use persistence images to enhance graph neural networks. Specifically, a spatial graph convolution uses reweighting vectors that contain information about local structure in the graph that ordinarily uses node degrees or learnt self-attention mechanisms. Zhao et al. learn a mapping from pixels of the persistence image to the reweighting vector in order to include information about loops in the local graph structure. They outperform existing benchmarks across a large variety of tasks. Both Hofer et al. (2019c) and Rieck et al. (2019a) also use the persistence diagram for additional topological information when learning from graph data. Gabrielsson and Carlsson (2019) topologically analyse convolutions in CNNs, demonstrating that the integration of topological information can improve network performance and proving that topological complexity correlates to generalisation ability. They also analyse topological groupings of convolutions to aid interpretability. Yan et al. (2021) compute the (extended) persistent homology of local neighbourhoods of graphs to inform link prediction with Graph Neural Networks. Rieck (2023) evaluates the context and efficacy of persistent homology within graph learning.

Image processing. Interpreting pixels as a cubicial complex gives a natural structure for filtrations on images. Using this filtration, persistence-based summaries have been used to find topologically discriminative shapes in image data. An early work by Letscher and Fritts (2007) split the image into regions using persistent homology, then merged regions in order of persistence for semi-supervised segmentation. Qaiser et al. (2016) split tumour images into patches, then use the k-nearest neighbour algorithm on the Kullback-Leibler divergence (Kullback and Leibler, 1951) distances on persistence diagrams to match similar patches together. Assaf et al. (2017) use the cubicial complex to find the most persistent connected components and treat those as segmented objects for labelling. Luo et al. (2019) also select regions with large persistence for segmentation, and offers an extensive comparison to other image segmentation techniques. Clough et al. (2020) combine their loss with CNN-based segmentation to give cleaner segmentation via the incorporation of topological priors. Hu et al. (2019) use a similar approach, giving a topological loss based on enforcing topological priors (via known Betti numbers) to improve image segmentation. They extended this work in Hu et al. (2021). Rieck et al. (2020) take a time series of fMRI images to compute a stack of persistence images: one for each time step. They show that clustering the stacks of persistent images results in meaningful groupings of data. Other topological invariants for images have been studied, for example, Kurlin (2015a) introduces an invariant called the *homologically persistent skeleton*, which they also show to be stable under perturbations and later extend to point clouds (Kurlin, 2015b; Kalisnik et al., 2019; Smith and Kurlin, 2021).

Neural network analysis. Persistent homology has been used to analyse the structure of neural networks. Given a diagram D, Rieck et al. (2019c) define *neural persistence* as

$$||D||_p = \left(\sum_{(b,d)\in D} |d-b|^p \right)^{1/p}.$$

They find that the neural persistence of perceptrons that are trained to convergence differs significantly to perceptrons that diverge or have random weights. Used for early-stopping, it can achieve comparable results to validation loss as a convergence indicator. Guss and Salakhutdinov (2018) show that the topological capacity of a network, defined in terms of the Betti number of the decision boundary, is a limiting factor in its ability to generalise. Gebhart and Schrater (2017, 2019) demonstrate that adversarial examples can be characterised and detected through the use of persistent homology. Carlsson and Gabrielsson (2020) used Mapper to analyse the topology of pretrained networks, leading to increased interpretability of each layer in a network. Liu et al. (2020) compute the persistent homology of neural networks to detect feature interactions across the network. Wu et al. (2020) use connected components in datasets to clean potentially noisy data.

3.2.4 Unsupervised learning

Initially, persistent homology was used in clustering algorithms to automatically determine the correct number of clusters (Chazal et al., 2013). Using the algorithm proposed by Turner et al. (2012), it became possible to perform clustering directly on the space of persistence diagrams. Hard k-means clustering was implemented by Maroulas et al. (2017), who also performed the convergence analysis. Lacombe et al. (2018) implemented a version of the same algorithm using their faster technique to compute Fréchet means. In the next chapter, we develop fuzzy c-means clustering for persistence diagrams by extending Turner et al.'s algorithm to the weighted case, as well as providing the same convergence guarantees as traditional fuzzy clustering. Stratification learning assumes that data is sampled from a collection of different manifolds, and attempts to infer which points belong to the same strata. Persistent homology has been used for this task (Bendich et al., 2010, 2012). We extend the fuzzy c-means clustering unsupervised learning algorithm to the space of persistence diagrams in Chapter 5.

3.3 The Laplacian in Machine Learning

The graph Laplacian has many applications in machine learning. We saw in Section 2.3.1 how it forms the basis of optimal graph embeddings, which underpins spectral clustering (Pothen et al., 1990; Demmel, 1999). The eigenbasis of the graph Laplacian provides convolutions for graph neural networks (Kipf and Welling, 2017). Deng et al. (2022) use the spectrum of the Laplacian to construct a trusted spectral representation of graphs to increase resilience against adversarial attacks on graph neural networks.

The (higher-order) combinatorial Laplacian has more limited applications in the literature, but with more interest in higher-order interactions the combinatorial Laplacian is being increasingly used. Notably, it is used by Bodnar et al. (2021) to provide updates for an extension of graph neural networks to simplicial complexes. They define an attention mechanism over their simplicial neural nets in Goh et al. (2022). Research groups led by Wei and Xia have recently been releasing a body of work looking at the combinatorial Laplacian as a feature vector. They compute the combinatorial Laplacian at fixed intervals over a filtration, looking at applications in fullerene analysis (Wang et al., 2019), protein binding (Meng and Xia, 2021; Wee and Xia, 2022), chromosome structures (Gong et al., 2022), and drug design (Jiang et al., 2021). In Qiu and Wei (2023) they investigate applications to protein engineering, linking the persistent homology and combinatorial Laplacian back to the persistent Laplacian, providing evidence for the application of persistent Laplacian to data science. In fact, in Chapter 4 we use the persistent Laplacian for data science for the first time.

Part II

Persistence-Based Summaries

Chapter 4

The Persistent Laplacian

In Section 2.3.3 we described the persistent Laplacian - an extension of the combinatorial Laplacian to the persistent case that has recently been introduced. In this first research chapter we consider the persistent Laplacian as the theoretical grounding of a new type of persistence-based summary, which we propose and evaluate. It makes sense as a persistence-based summary: we have seen that its kernel contains all the information of the homology groups, and we know that the spectrum of its non-zero image contains detailed information about its structure in the case of the 0-Laplacian - its not far fetched to consider that is also the case for the higher Laplacians. In fact, we provide some initial evidence of the efficacy of the higher-order non-zero image, by examining its utility as a feature vector.

In the first section of this chapter, however, we look at some motivation for seeking out persistence-based summaries that contain more than just the topological information in persistence diagrams. Having completed the literature review of applications of persistence-based summaries in machine learning, we were left considering an obvious initial question: is there a quantifiable advantage to using topological information in machine learning? In Section 4.1 we present our attempts to provide a statistical learning theoretic justification for the inclusion of persistence diagrams when learning. We show, however, that restricting hypothesis classes via topological prior knowledge in the form of a persistence diagram does not result in even a nonuniformly learnable class of functions. That is, the additional knowledge provided by including topological information does not reduce the complexity of the hypothesis class sufficiently to provide useful information in the eyes of statistical learning theory. With hindsight, we feel that this result follows intuition: structures that have the same homology groups can be different enough that vastly different underlying data can have the same persistence diagram. Thus there will be plenty of cases in which topological information is far from sufficient for good ML. This offers the motivation for studying the persistent Laplacian, as it gives a new persistence-based summary that can be used in machine learning which contains both topological and geometric information.

Publications and contributions. A version of the work on topological learnability in which we incorrectly believed the hypothesis class to be learnable was presented as a spotlight presentation at the NeurIPS 2020 workshop on TDA & Beyond (Bishop et al., 2020). The application of the persistent Laplacian was accepted to ICML (Davies et al., 2023b). In the statistical learning theory work, the idea was conceived jointly between myself and the co-first author of the paper, Nick Bishop. He provided the background on statistical learning theory and I provided the background on the fibre of the persistence map and the role of topological loss terms in machine learning. I proved the main result from that work, namely Theorem 4.6. In the application of the persistence Laplacian to data science, I conceived the research. Our coauthor Zhengchao Wan verified that the Schur complement formulation of the persistent Laplacian holds for cubical complexes. I developed our representation as a feature vector and ran the experiments, ablation studies, and comparisons to non-topological models.

4.1 Motivation: Topological Learnability

In Section 3.2.3 we saw many techniques that have recently been introduced to add persistence-based terms to loss functions, either seeking to integrate knowledge of topological priors or to regularise the algorithm by encouraging topological simplicity. In both cases, the implicit role of this term is to topologically restrict the class of functions from which the learnt function can be selected. There has been empirical success in specific topologically-motivated settings, but there exists no theoretical justification for the inclusion of such topological restrictions in a general setting. We investigate the use of *Statistical Learning Theory* to characterise the utility of Topological Data Analysis in machine learning. In the context of statistical learning theory, if a hypothesis class of functions is *learnable* then there are bounds on the number of sampled points required to probably approximately learn the best classifier. In this section we prove that restricting a hypothesis class to all decision boundaries that give rise to a specific persistence diagram is not sufficient to make the hypothesis class (nonuniformly) learnable. In other words, topological prior knowledge does not restrict the possible functions we can learn sufficiently for any classical results on learnability. This result, which is potentially unsurprising given the extent to which homotopic spaces can differ, justifies our investigation of the persistent Laplacian for data science: an operator which contains both topological and geometric information.

Our setup is the following. Given prior knowledge about the topology in the form of a persistence diagram D, topological loss terms implicitly restrict the decision boundaries to the class of functions

$$\mathcal{PH}^{-1}(D) = \{f : \mathcal{PH}(f) = D\}\,,$$

where $\mathcal{PH}$ is the persistence map. This class of functions has been studied by Curry (2018). Given a real-valued function $f : [0,1] \to \mathbb{R}$, we identify its graph as the decision boundary of a binary classifier $h_f : [0,1] \times \mathbb{R} \to \{0,1\}$ by

$$h_f((x,y)) = \begin{cases} 0, \ f(x) \leq y, \\ 1, \ f(x) > y. \end{cases}$$

Thus we are studying the hypothesis class

$$\mathcal{H}_D = \{h_f : f \in \mathcal{PH}^{-1}(D)\}.$$

4.1.1 The fibre of the persistence map

We use work by Curry (2018) on the fibre of the persistence map to understand the hypothesis class $\mathcal{H}_D$. We say two continuous functions $f, g : [0,1] \to \mathbb{R}$ are *graph-equivalent* if there is an orientation preserving homeomorphism $\phi : [0,1] \to [0,1]$ such that $f = g \circ \phi$. We say a continuous function $f : [0,1] \to \mathbb{R}$ is *Morse-like* if it is graph equivalent to a piecewise linear function where every critical point is isolated and has a distinct critical value. Let $\mathcal{PH}_0 : \mathcal{M} \to \mathcal{D}$, where $\mathcal{M}$ is the set of Morse-like real-valued functions on the interval with local minima at $x = 0$ and $x = 1$, and $\mathcal{D}$ is the space of persistence diagrams. Then Curry (2018, Theorem 6.12) tells us that for any persistence diagram D,

$$\mathcal{PH}_0^{-1}(D) = \bigcup_{i \in \mathcal{I}} \mathcal{H}_{f_i},$$

where $\mathcal{I}$ is a finite indexing set over some collection of functions $\{f_i\}_i$ and $\mathcal{H}_{f_i} = \{g \in \mathcal{M} : g \sim f_i\}$ with $\sim$ denoting graph equivalence. Note that Smith and Kurlin (2022) studies the families of metric spaces with identical persistence in dimension 1, although we do not use this result (in contrast to this, Widdowson and Kurlin (2023) introduce an invariant which *is* a complete and Lipschitz continuous invariant of finite unlabelled pointclouds in Euclidean space).

4.1.2 Statistical learning theory

Statistical Learning Theory aims to quantify the efficacy of machine learning algorithms. In particular, given an algorithm that can draw potential classifiers from some hypothesis class, it aims to tell you whether the algorithm can probably, approximately correctly learn to classify any labelled dataset. We use these tools to investigate the efficacy of topological loss functions.

In the standard supervised machine learning setting a learner is given a sample x from a data space $\mathcal{X}$ and would like to accurately predict a corresponding target label y from a label space $\mathcal{Y}$. The learner has access to a finite sample $S = \{(x_i, y_i)\}_{i=1}^m$ of training examples sampled from a distribution μ over $\mathcal{X} \times \mathcal{Y}$.

The goal of the learner is to select a hypothesis $h : \mathcal{X} \to \mathcal{Y}$ from a hypothesis set $\mathcal{H}$ which achieves low expected error with respect to the distribution μ. The error of a hypothesis rule h on a labelled sample (x, y) is given via a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ by evaluating $\ell(h(x), y)$. More formally, the goal of the learner is to select a hypothesis which attains low risk, where risk is formalised by

$$L_\mu(h) = \mathop{\mathbb{E}}_{(x,y)\sim\mu} \left[\ell(h(x), y)\right].$$

A learning algorithm is a mapping from samples to the hypothesis set $\mathcal{H}$. Of course, the learner would like to employ a learning algorithm which selects a hypothesis that attains low risk with high probability. These motivations are captured by the agnostic probably-approximately-correct (PAC) framework, first introduced by Valiant (1984).

Definition 4.1. A hypothesis class $\mathcal{H}$ is *agnostic PAC learnable* if there exists a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm such that for every $\epsilon, \delta \in (0,1)$ and for every distribution μ over $\mathcal{X} \times \mathcal{Y}$, when running the learning algorithm on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ independent and identically distributed (i.i.d.) examples generated by μ, the algorithm returns a hypothesis h such that, with probability of at least $1 - \delta$ (over the choice of the m training examples),

$$L_\mu(h) \leq \min_{h' \in \mathcal{H}} L_\mu(h') + \epsilon.$$

We call any such algorithm an *agnostic PAC learning algorithm* for the set $\mathcal{H}$.

In other words, a hypothesis set is agnostic PAC learnable if there exists a learning algorithm such that for any (ϵ, δ), there exists an $m \in \mathbb{N}$ such that when the algorithm receives at least m training examples, it returns a hypothesis rule that attains risk ϵ-close to the that of the best hypothesis in the set with probability $1 - \delta$. Of course, any learner would like to employ an agnostic PAC learning algorithm when one is available, as they give theoretical guarantees on the risk achieved as the sample size increases. Typically, m is referred to as the sample complexity necessary for a given algorithm to guarantee an ϵ-efficient hypothesis with probability $1 - \delta$.

In the context of binary classification, we have $\mathcal{Y} = \{0, 1\}$. In this setting, a useful property to consider when analysing the PAC learnability of a hypothesis set $\mathcal{H}$ is the Vapnik-Chervonenkis (VC) dimension of $\mathcal{H}$ (Vapnik and Chervonenkis, 1971, Section 1.1).

Definition 4.2. Let $C = \{c_1, \ldots, c_m\} \subset \mathcal{X}$ and $\mathcal{H}_C$ be the restriction of $\mathcal{H}$ to C defined by

$$\mathcal{H}_C = \{(h(c_1), \ldots, h(c_m)) \ : \ h \in \mathcal{H}\}.$$

Then we say that $\mathcal{H}$ *shatters* C if $|\mathcal{H}_C| = 2^{|C|}$.

Intuitively, this definition means that given any labelling of the points C, there exists a classifier in $\mathcal{H}$ that will give assign the points that labelling. The following definition captures the scale of this. That is, the number of points that a hypothesis class is able to label in any possible way.

Definition 4.3. The *VC dimension* of a hypothesis class $\mathcal{H}$, denoted by $\mathrm{VCdim}(\mathcal{H})$ is the maximal size of a set $C \subset \mathcal{X}$, which can be shattered by $\mathcal{H}$. If $\mathcal{H}$ shatters sets of arbitrary size, we say that $\mathcal{H}$ has infinite VC dimension.

In fact, it is well-known that VC dimension fully characterises the PAC learnability of a hypothesis class in the case of binary classification. That is, a hypothesis set $\mathcal{H}$ is PAC learnable if and only if it has finite VC dimension (Vapnik, 2000).

In many cases we desire learnability guarantees for complex hypothesis sets which do not have finite VC dimension. In such cases, we can consider a relaxation of the PAC framework called nonuniform learnability (Benedek and Itai, 1988) in which we can derive weaker, but still useful, guarantees for generalisation.

Definition 4.4. A hypothesis class $\mathcal{H}$ is nonuniformly learnable if there exists a learning algorithm, A, and a function $m_{\mathcal{H}}^{\mathrm{NUL}} : (0,1)^2 \times \mathcal{H} \to \mathbb{N}$ such that, for every $\epsilon, \delta \in (0,1)$ and for every $h \in \mathcal{H}$, if $m \geq m_{\mathcal{H}}^{\mathrm{NUL}}(\epsilon, \delta, h)$ then for every distribution μ, with probability at least $1 - \delta$ over the choice of $S \sim \mu^m$, it holds that

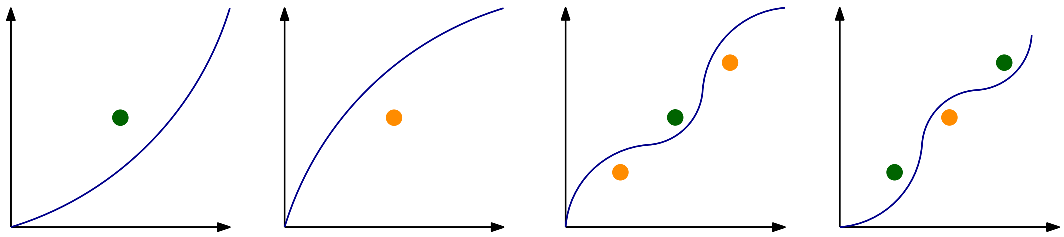
$$L_\mu(A(S)) \leq L_\mu(h) + \epsilon.$$

Similarly to the framework of PAC learning, nonuniform learnability of hypothesis classes for binary classification can be characterised through VC dimension (Benedek and Itai, 1988).

Theorem 4.5. *A hypothesis class $\mathcal{H}$ of binary classifiers is nonuniformly learnable if and only if it is a countable union of agnostic PAC learnable hypothesis classes.*

4.1.3 Topologically restricted hypothesis classes

Following the work by Curry on the fibre of the persistence map (Curry, 2018), we concern ourselves with Morse-like real-valued functions on the unit interval with local minima at $x = 0, 1$, and the 0th persistence diagram. Although this is clearly a restricted setting, it allows us to perform an initial investigation into the learnability of topologically restricted hypothesis classes. Curry (2018) tells us that $\mathcal{PH}_0^{-1}(D)$ is finite when we introduce graph equivalence on the set of functions. Moreover, Theorem 4.5 tells us that a hypothesis class of binary classifiers is nonuniformly learnable if and only if it is a



(i) Shattering one point

(ii) Shattering several points

FIGURE 4.1: We can shatter arbitrarily many points between two critical points using graph equivalent functions, proving that the VC dimension of $\mathcal{H}_f$ is unbounded.

countable union of agnostic PAC learnable hypothesis classes. Therefore, to show that restricting a hypothesis class of functions to those with a unique persistence diagram is nonuniformly learnable, we would have to show that the class of graph equivalent functions is agnostic PAC learnable, or equivalently, that it has finite VC dimension. Therefore our focus is on the collection of classifiers with decision boundaries that are graph equivalent to some function f, which we denote $\mathcal{H}_f$.

We initially believed that $\text{VCdim}(\mathcal{H}_f) = 0$. However, to prove that $\text{VCdim}(\mathcal{H}_f) > 0$ we just need to find one point that can be shattered by $\mathcal{H}_f$. Recall that two functions are graph equivalent if there is an orientation preserving homeomorphism between them, and a continuous function f is Morse-like if it is graph equivalent to a piecewise linear function where every critical point is isolated and has a distinct critical value. By applying orientation-preserving transformations, we can 'stretch and squeeze' the x axis, but we can never change the number of critical points or their values. Imagine the most simple case: a point to be classified lying between two critical points that are connected with a line. We can easily find orientation-preserving homeomorphisms that allow us to classify that point as either 0 or 1, as shown in Figure 4.1i. Therefore we have shattered that point. In fact, if there are arbitrarily many points lying on a line between two critical points, then we can shatter them, as demonstrated in Figure 4.1ii. Therefore the VC dimension of $\mathcal{H}_f = \infty$.

Now, our counterexample to the nonuniform learnability of $\mathcal{H}_f$ came about when we introduced inflection points. As these allow our classifier to change from convex to concave without introducing additional critical points, this lets graph equivalent functions be extremely expressive. We considered that perhaps restricting the number of inflection points allowed within $\mathcal{H}_f$ could lead to a finite VC dimension? Such a restriction is realistic: the number of inflection points is finite in many common learning algorithms, including neural networks. However, there remains sets of points that can be shattered, as we show in the following theorem (which also captures the more expressive case where we have unbounded inflection points).

Theorem 4.6. *Let $f : [0,1] \to \mathbb{R}$ be a Morse-like real-valued function on the unit interval with local minima at $x = 0, 1$, and $\mathcal{H}_f$ be the class of functions that are graph equivalent to f. Additionally suppose that the number of inflection points (i.e., points where the function f changes*

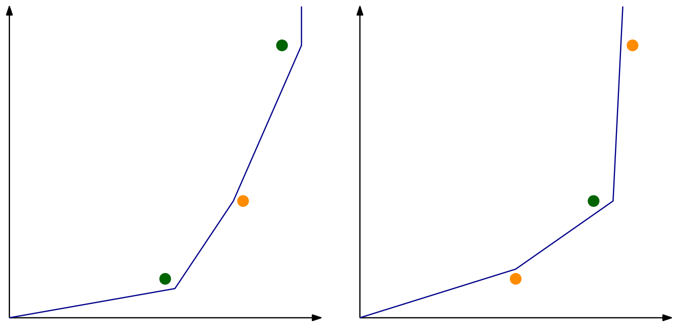


FIGURE 4.2: Bounding the number of inflection points does not prevent an arbitrary number of points from being shattered.

from convex to concave) is finite. Then $\mathrm{VCdim}(\mathcal{H}_f) = \infty$ *and* $\mathcal{H}_D$ *is not (nonuniformly) learnable.*

Proof. Without loss of generality, consider a section of f that is increasing and concave, i.e., it lies between two critical or inflection points $(x_0, y_0), (x_{n+1}, y_{n+1})$. Place n points $(x_1, y_1), \dots, (x_n, y_n)$ between them so that $2x_i = x_{i-1}$ and $y_i = 2y_{i-1}$ for all $i = 1, \dots, n+1$. Define a new list of points by

$$(\alpha_i, \beta_i) = \begin{cases} (x_i + \epsilon, y_i), & \text{if } (x_i, y_i) \text{ is labelled } 1, \\ (x_i - \epsilon, y_i), & \text{if } (x_i, y_i) \text{ is labelled } 0, \end{cases}$$

for some small $\epsilon > 0$. The decision boundary defined by connecting $\{(\alpha_i, \beta_i)\}_i$ into a piecewise linear function can attain any labelling of the n points, and so shatters them. Furthermore, it is increasing, has no critical points, and is concave (an example is shown in Figure 4.2). Therefore n points can be shattered by such a function for arbitrary n, and $\mathrm{VCdim}(\mathcal{H}_f) = \infty$. □

In fact, we cannot find any reasonable conditions on the hypothesis classes that would make them learnable. This highlights a key problem with learning with persistence diagrams: although they provide topological information which can be useful in specific cases, our analysis indicates that they cannot always provide additional useful information. This is due to the amount of information discarded when we are only interested in shape up to homotopy equivalence: when learning we need more than just information about the topological invariants. Motivated by this, we now investigate the persistent Laplacian for machine learning: an operator on simplicial complexes that contains all of the information in homology groups in its kernel, as well as additional geometric information in its non-zero image.

4.2 The Persistent Laplacian for Data Science

The persistent Laplacian is a recent theoretical extension of the combinatorial Laplacian to the persistent case. Recall from Section 2.3.3 that the persistent Laplacian is defined as follows. Let $K \subseteq L$ be a simplicial pair and consider the subspace of C_p^L given by

$$C_p^{L,K} = \left\{ c \in C_p^L \;:\; \partial_p^L(c) \in C_{p-1}^K \right\} \subseteq C_p^L, \tag{4.1}$$

so that, $C_p^{L,K}$ consists of all simplices in C_p^K that have their boundary in $C_{p-1}^K \subseteq C_{p-1}^L$. We write $\partial_p^{L,K}$ for the restriction of the boundary operator to this subspace. The *persistent p-Laplacian* is defined by Mémoli et al. (2022) as

$$\Delta_p^{K,L} = \left(\partial_{p+1}^{L,K}\right)^* \circ \partial_{p+1}^{L,K} + \partial_p^K \circ (\partial_p^K)^*, \tag{4.2}$$

with the *up persistent p-Laplacian* given by $\Delta_{p,\mathrm{up}}^{K,L} = \left(\partial_{p+1}^{L,K}\right)^* \partial_{p+1}^{L,K}$. The relation between each operator is shown in the diagram from Mémoli et al. (2022) below.

$$\begin{tikzcd}
C_{p+1}^K \arrow[rr, "\partial_{p+1}^K"] \arrow[dd, hook, dashed] & & C_p^K \arrow[r, shift right, blue, "\partial_p^K"'] \arrow[dl, shift left, blue, "\left(\partial_{p+1}^{L,K}\right)^*"'] \arrow[dd, hook, dashed] & C_{p-1}^K \arrow[l, shift right, blue, "\left(\partial_p^K\right)^*"'] \arrow[dd, hook, dashed] \\
 & C_{p+1}^{L,K} \arrow[ur, shift left, blue, "\partial_p^{L,K}"'] \arrow[dl, hook, dashed] & & \\
C_{p+1}^L \arrow[rr, "\partial_{p+1}^L"] & & C_p^L \arrow[r, "\partial_p^L"] & C_{p-1}^L
\end{tikzcd}$$

We have also seen that this operator contains both topological information, as its kernel is isomorphic to the homology groups (Theorem 2.3), and geometric information via its non-zero eigenvalues. This combination of different types of structural information makes it a natural candidate for investigation in the context of data science. In the remainder of this chapter, we consider the persistent Laplacian as a feature vector for machine learning for the first time and assess its usefulness in practical applications. Namely, we empirically investigate its efficacy in comparison to other topological and geometric techniques at MNIST classification and the prediction of molecular properties from the MoleculeNet dataset Wu et al. (2018). Additionally, we evaluate the importance of utilising the filtration in the persistent Laplacian and different magnitude eigenvalues to the success on the downstream task. We provide the first evidence that the persistent Laplacians outperforms the current mainstream topological tools, paving the way for its wider use by the ML and TDA communities.

4.2.1 The persistent Laplacian for cubical complexes

Computer vision is the area of computer science which concerns itself with processing images and videos, and makes up a significant part of machine learning research. In fact, one of the classical benchmark datasets for any new technique in machine learning is MNIST, a dataset of 28x28 images of handwritten digits (LeCun et al., 2010). In order to make the persistent Laplacian applicable to image datasets (including MNIST, which we wish to benchmark our dataset on) we need to consider a new type of complex. Up until now we have only considered simplicial complexes - collections of p-dimensional triangles that enabled us to represent the structure of our datasets. Although we could construct simplicial complexes from pixel data, it makes much more sense (and is the standard approach) to instead use *cubical complexes*. A cubical complex is built from p-cubes. Like simplices, a 0-cube is still a point $[x_0]$ and a 1-cube is still an edge $[x_0, x_1]$, but a 2-cube is a square $[x_0, x_1, x_2, x_3]$ and a 3-cube is a cube $[x_0, \ldots, x_7]$, as shown in Figure 4.3. Just as a simplex has an orientation given by an ordering of its vertices and chain groups C_p consisting of formal sums of p-simplices, we can orient p-cubes and consider the chain groups C_p generated by p-cubes.

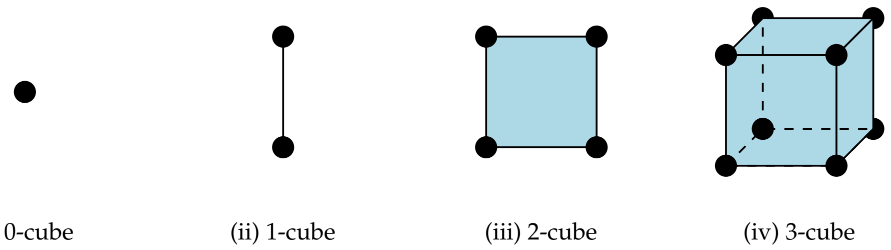

(i) 0-cube (ii) 1-cube (iii) 2-cube (iv) 3-cube

FIGURE 4.3: Examples of p-cubes.

We can also defined the boundary operator taking p-cubes to their $(p-1)$-boundaries, $\partial_p : C_p \to C_{p-1}$. Elementary p-cubes can be seen as products of p unit intervals $I_n = [m, m+1]$. For example, a unit 2-cube in $\mathbb{R}^2$ is a product of two unit intervals $[0,1] \times [0,1]$. In fact, this definition helps characterise cubical complexes in $\mathbb{R}^d$: a subset K of $\mathbb{R}^d$ is said to be a *cubical complex* if it is a finite union of p-cubes, where each p-cube is a product of p unit intervals, and the intersection of every cube in K is also in K. We define the boundary of a p-cube recursively. The boundary of a 0-cube $[m]$ is defined to be zero, so that $\partial_0[m] = 0$. The boundary of a 1-cube $[m, m+1]$ is $\partial_1[m, m+1] = [m+1] - [m]$, aligining with our expectation for the boundary of a 1-simplex. Given $p > 1$, the boundary of a p-cube $Q = I_0 \times \cdots \times I_{p-1}$ is

$$\partial Q = (\partial I_0 \times \cdots \times I_{p-1}) + (I_1 \times \partial I_2 \times \cdots \times I_{p-1}) + \cdots + (I_1 \times \cdots \times \partial I_{p-1}).$$

With C_p and ∂_p defined, we can construct our chain complex as in Section 2.1.2.

$$\cdots \xrightarrow{\partial_{p+1}} C_p \xrightarrow{\partial_p} C_{p-1} \xrightarrow{\partial_{p-1}} \cdots.$$

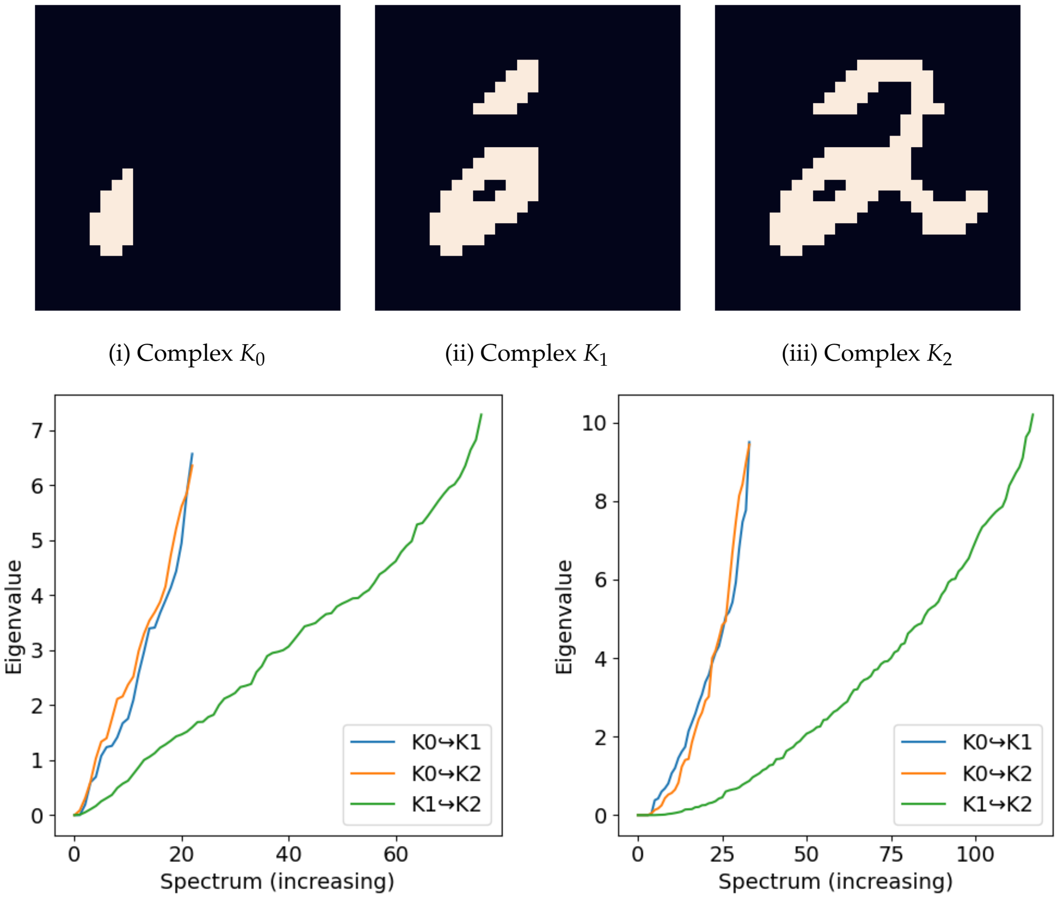
In fact, it does not matter to the algebraic structures we are now considering whether or not there are simplicial or cubical complexes beneath the chain complex, and we can define $C_p^{K,L}$ via Equation 2.4 then restrict the boundary operator and define the persistent Laplacian identically to Equation 2.5. For more details on cubical persistent homology, please see Strömbom (2007).

We also need to compute the persistent Laplacian. In Section 2.4.4 we saw that a succinct matrix representation of the persistent Laplacian exists. In particular, Mémoli et al. (2022) show that the matrix representation for up persistent Laplacians of simplicial pairs can be computed via Schur complement of certain matrices. More precisely, let $K \hookrightarrow L$ be a simplicial pair. We let $\mathbf{\Delta}_{\text{up},p}^{L}$ denote the matrix representation of the p-th up Laplacian $\Delta_{\text{up},p}^{L}$ for L. Then, if we let I_K^L denote the indices of rows (or columns) in $\mathbf{\Delta}_{\text{up},p}^{L}$ which correspond to p-simplices not belonging to K, then $\mathbf{\Delta}_{\text{up},p}^{K,L} = \mathbf{\Delta}_{\text{up},p}^{L} / \mathbf{\Delta}_{\text{up},p}^{L}(I_K^L, I_K^L)$ is, in fact, the matrix representation of $\Delta_{\text{up},p}^{K,L}$ under the canonical choice of basis of C_p^K. In our paper (currently under review), my collaborator Zhengchao Wan verified that the computation of the persistent Laplacian via the Schur complement still holds for cubical complexes. Therefore we can continue to use the persistent Laplacian for cubical complexes, simply by updating the boundary operator to compute the boundary of cubical complexes.

4.2.2 As a feature vector

One problem with using the persistent Laplacian as a feature vector remains. Typically in Topological Data Analysis we will be working with a filtration: a collection of complexes K_i such that $K_i \subseteq K_j$, with i a continuous real parameter in some range $[i_0, i_T]$. Algorithms from persistent homology theory can handle the entire filtration efficiently and generate persistence diagrams as the topological summary of the filtration, enabling us to understand the whole filtration via the births and deaths of topological features. On the other hand, the persistent Laplacian $\Delta_p^{K,L}$ is defined for just two complexes: a complex pair (either a simplicial pair or a cubical pair) $K \hookrightarrow L$. Given a filtration, which complex pairs do we select to compute the persistent Laplacian?

We compute the persistent Laplacian only for pairs in a small subset of the filtration: start by selecting a resolution R. We investigate the effect of R in Section 4.9ii. Then, we compute an increment $I = (i_T - i_0)/R$ to get a subset of the filtration $F = [i_0, i_0 + I, i_0 + 2I, \ldots, i_0 + RI = i_T]$. We then consider complex pairs $K_i \hookrightarrow K_j$ where $i, j \in F$ and $i \leq j$. An example of a sequence of complexes is given in Figure 4.4, where we show three sequential cubical complexes generated from an MNIST digit. The filtration is a simplified *height filtration*: we are adding non-zero pixels to the filtration as we move left to right across the image. Given the complex pairs, we compute their Laplacians and eigenvalues using our codebase. In Figure 4.4 we compute the persistent Laplacian for the three inclusion pairs generated from the subset of the filtration parameter $F = [0, 1, 2]$



(i) Complex K_0

(ii) Complex K_1

(iii) Complex K_2

(iv) Eigenvalues of 0-PL

(v) Eigenvalues of 1-PL

FIGURE 4.4: A filtration of cubical complexes $K_0 \hookrightarrow K_1 \hookrightarrow K_2$ of an MNIST digit, alongside the eigenvalues of the persistent Laplacian. The x axis is the eigenvalue position and the y axis is its value (i.e., the y value at $x = a$ is the value of the ath smallest eigenvalue).

and for dimension $p = 0, 1$. We can see the differences between the eigenvalues in their plots.

As feature vectors need to have a consistent length and as the number of eigenvalues depends on the size of the underlying complexes, which can change throughout the filtration, it is not sufficient to simply compute eigenvalues of the Laplacian and concatenate them. To solve this we set a length parameter that truncates or zero pads to ensure a consistent length regardless of the underlying complex. The effect of this truncation/zero padding is investigated in Section 4.3.3. Flattening the eigenvalues across complex pairs and dimensions gives the final feature vector. With resolution R, max dimension D and truncating/zero padding eigenvalues to length L, we end with a $\frac{1}{2}R(R+1)DL$ dimensional feature vector. The entire process is captured in Figure 4.5, which showcases the pipeline.

We know from Mémoli et al. (2022) that the time complexity for the computation of

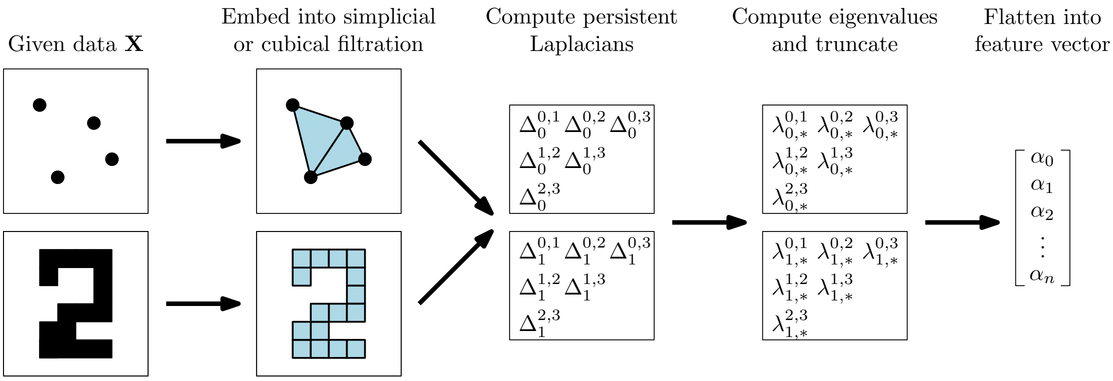

Given data $\mathbf{X}$
Embed into simplicial or cubical filtration
Compute persistent Laplacians
Compute eigenvalues and truncate
Flatten into feature vector
$\Delta_0^{0,1}$ $\Delta_0^{0,2}$ $\Delta_0^{0,3}$
$\Delta_0^{1,2}$ $\Delta_0^{1,3}$
$\Delta_0^{2,3}$
$\Delta_1^{0,1}$ $\Delta_1^{0,2}$ $\Delta_1^{0,3}$
$\Delta_1^{1,2}$ $\Delta_1^{1,3}$
$\Delta_1^{2,3}$
$\lambda_{0,*}^{0,1}$ $\lambda_{0,*}^{0,2}$ $\lambda_{0,*}^{0,3}$
$\lambda_{0,*}^{1,2}$ $\lambda_{0,*}^{1,3}$
$\lambda_{0,*}^{2,3}$
$\lambda_{1,*}^{0,1}$ $\lambda_{1,*}^{0,2}$ $\lambda_{1,*}^{0,3}$
$\lambda_{1,*}^{1,2}$ $\lambda_{1,*}^{1,3}$
$\lambda_{1,*}^{2,3}$
α_0
α_1
α_2
$\vdots$
α_n

FIGURE 4.5: The pipeline to compute feature vectors from data using the persistent Laplacian. We first map the data into a filtration of cubical or simplicial complexes, then compute the persistent Laplacian for a selection of complex pairs and dimensions, before taking the eigenvalues ($\lambda_{p,*}^{i,j}$ denotes the p-eigenvalues of $\Delta_p^{i,j}$), truncating/zero-padding (if necessary), then flattening into a feature vector.

the persistent Laplacian for one complex pair is similar to (and sometimes better) than that of persistent homology. However, when using this technique over a filtration, we are computing it $\frac{1}{2}R(R+1)$ times, so increasing the time complexity similarly. If the time taken is going to be a restraint for this technique, we recommend reducing R - we investigate the effect this will have on the performance of the feature vector in Section 4.3.3. We report the runtimes for all of the techniques considered in this Chapter in Appendix B.1.2.

4.3 Experiments

We evaluate the persistent Laplacian against other topological and geometric representations of data. The main baseline we are comparing against is persistent homology, as this is the primary technique used for topological representations of data in TDA. Our specific vectorisation of the persistence diagram once computed depends on the application: we would typically recommend using persistence images (Adams et al., 2017) but we use other vectorisations if they are better according to published baselines. We also compare against the graph Laplacian and (where there is data with more than edgewise interactions) the combinatorial Laplacian, as these represent the non-persistent versions of the k-persistent Laplacian for $k = 0$ and $k > 0$ respectively. We run experiments on the MNIST handwritten digit dataset (LeCun et al., 2010) to provide a base demonstration of value in a machine learning setting, and the MoleculeNet dataset (Wu et al., 2018), a collection of molecular data along with chemical properties to predict. The MoleculeNet data serves to evaluate the applicability of the persistent Laplacian to realistic application scenarios. In the remainder of this section we expand on our exact methodology for each dataset and baseline, and specify precisely the data and tasks we consider.

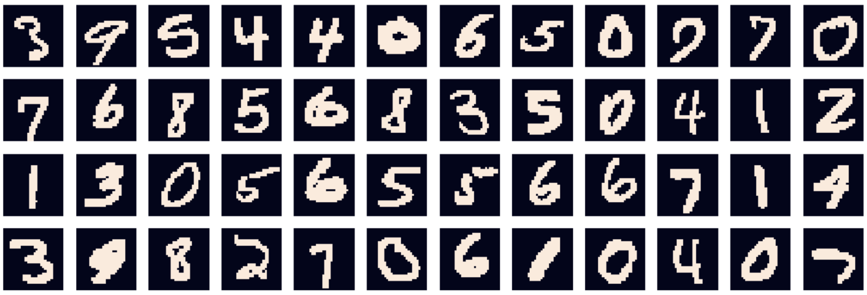


FIGURE 4.6: Examples of digits sampled from the MNIST dataset.

4.3.1 MNIST

MNIST is a standard machine learning benchmark dataset consisting of 28x28 grayscale images (LeCun et al., 2010). Each image is one handwritten digit and the task is to classify that digit - we display example digits sampled from MNIST in Figure 4.6. Each of our baselines takes a filtration as input, so we need to represent grayscale images. According to the Giotto-TDA documentation[1] the best performing filtration for persistent homology is the height filtration with direction $[1, 0]$. To implement this we first threshold the image, discarding all pixels with a grayscale value less than 0.4 (which is in fact the optimal value for our primary competitor, persistent homology). The distance from the plane defined by our direction (in this case, the leftmost edge of the image) gives the value at which each cube (a pixel or 2x2 collection of pixels, as we are using a cubical complex) enters the filtration. We then compute the persistent Laplacian feature vector as defined in Section 4.2.2, with resolution $R = 4$ and using dimensions $p = 0, 1$. We compute the persisent homology and diagrams using Giotto-TDA, then vectorise it as proposed by Garin and Tauzin (2019) and implemented in the Giotto-TDA baseline, using persistence entropy and diagram amplitudes with a collection of metrics. Note that they ensemble vectorisations of 17 different filtrations, using different directions and starting points for directional and radial filtrations. In order to control the filtration, we only use the filtration which performs best with persistent homology in their experimentation. This is the directional filtration with direction $[1, 0]$. The graph Laplacian and combinatorial Laplacian are vectorised identically to the persistent Laplacian, only as they are not persistent we simply compute them at the start values of the complex pairs, i.e., we compute the eigenvalues for the graph and combinatorial Laplacian for $R = 4$ values of the filtration. In the notation of Section 4.2.2, given discrete values of our filtration $F = [\alpha, \beta, , \dots]$, we compute $\Delta_p^{K_\alpha}, \Delta_p^{K_\beta}, \dots$ with $p = 0$ for the graph Laplacian and $p = 0, 1$ for the combinatorial Laplacian. We then represent the eigenvalues in the same way as the combinatorial Laplacian.

[1] https://giotto-ai.github.io/gtda-docs/0.3.1/notebooks/MNIST_classification.html

We vastly exceed the performance of persistent homology with the persistent Laplacian feature vector on MNIST. In particular, we attain 82% accuracy with a simple representation, which vastly outperforms the 62.5% accuracy attained by persistent homology, even with the ensembled representations utilised in the benchmark from Giotto-TDA. We also outperform the graph Laplacian and the versions of the combinatorial Laplacian that we evaluate. The full results can be seen in Figure 4.7. We consider this performance strong evidence for the suitability of the persistent Laplacian for feature representation in Topological Data Analysis.

Feature Importance. We also evaluate the feature importance that the MNIST model assigns to each eigenvalue, in terms of the Mean Decrease in Impurity metric Scornet (2023). In Figure 4.8, we display the feature importance grouped both by the positions of the complex pair in the filtration, and by the size of the eigenvalue, in order to ascertain the effect of each factor on the discriminative information the corresponding eigenvalues contain. For the complex pair feature importance plots, our x axis is the start value of the filtration, and the y axis is the shift forwards through the filtration. For example, if the start value is 1 and the shift forward is 2, then our complex pair is $K_1 \hookrightarrow K_3$. We compute the feature importance plots for the MNIST model. From the feature importance by filtration plots (Figure 4.8), it is clear that the model is using features from each of the complex pairs that we generate. This means that *persistence is providing valuable discriminative information* that the model is utilising. This is particularly true in the 0-persistent Laplacian, but also for the 1-persistent Laplacian. Next, we consider the plots that relate the size of the eigenvalue to the feature importance. Note that the smallest eigenvalue of the 0-persistent Laplacian has zero importance. This makes sense, as we know from Theorem 2.3 that this eigenvalue will always be 0, as there is always at least one connected component, and the nullity of 0-PL is the number of connected components. The spike immediately following it also makes sense, as the number of following zero eigenvalues gives topological information, namely the number of connected components, and the smallest following non-zero eigenvalues are known to contain significant information about the structure of the graph. Moving onto the 1-persistent Laplacian, we see, unsurprisingly, that the zero eigenvalues having high importance, as they quantify the number of holes present in the complex. It is interesting that there remains high feature importance for the small non-zero eigenvalues, as this implies they are also capturing discriminative information. On the whole, we can see from our feature importance plots that the combined features of the persistent Laplacian are providing value to the model: the zero eigenvalues and small non-zero eigenvalues, corresponding to topological and geometric information respectively, and complex pairs corresponding to the persistent version of the Laplacian all score highly when we examine the feature importance.

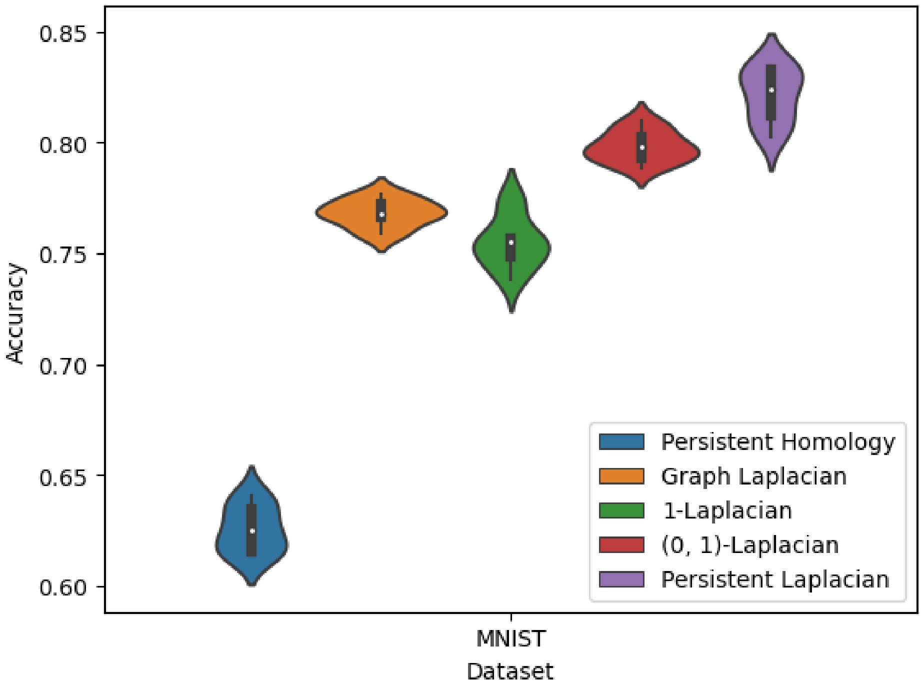
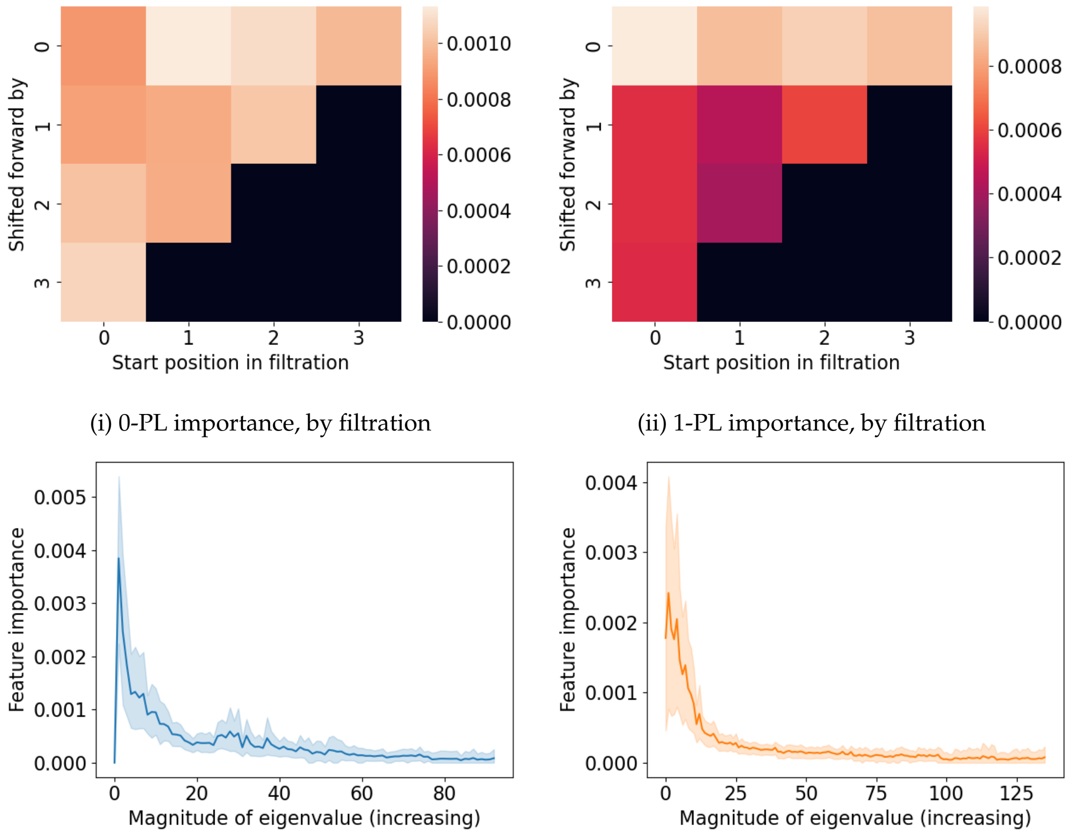


FIGURE 4.7: The classification accuracy of random forests trained on different representations of MNIST. The violin plots show the distribution over 10 repeats of each method. The persistent Laplacian performs notable better than persistent homology, and performs best across techniques.

4.3.2 MoleculeNet

MoleculeNet (Wu et al., 2018) is a collection of benchmark datasets designed to evaluate machine learning methods for prediction of molecular properties. We focus on two subsets of MoleculeNet: QM7 (Blum and Reymond, 2009; Rupp et al., 2012) and QM7b (Blum and Reymond, 2009; Montavon et al., 2013). The data comes in two forms: 3D coordinates of the atoms in each molecule, and the Coulomb matrix of each molecule, a matrix $M = [m_{i,j}]_{i \in I}$ for some list of nuclei I, where $m_{i,j}$ is the electrostatic interaction between atomic nuclei in the molecule. We map the 3D coordinates into a simplicial complex using the Vietoris-Rips construction Vietoris (1927), as described in Section 2.1.1. We represent the Coulomb matrices by letting each nucleus $i \in I$ enter the filtration as a point at time 0, then adding edges ij at their electrostatic interaction value $m_{i,j}$. In this way we use the electrostatic interactions to induce the filtration. As we are only adding vertices and edges to this filtration, we do not consider the 1-Laplacian in our benchmarks, as it would be the same as the graph Laplacian. Each of the MoleculeNet tasks we consider is a regression task for which we must predict specific electrochemical properties. For QM7 the task is to predict the atomisation energy of each molecule Blum and Reymond (2009), and we do so with both the 3D coordinates and Coloumb matrices (CM). For QM7b, there are only CM matrices available, and the specific tasks are reported in Table B.1.



(i) 0-PL importance, by filtration

(ii) 1-PL importance, by filtration

(iii) 0-PL importance, by value

(iv) 1-PL importance, by value

FIGURE 4.8: We aggregate feature importance by the complex pair and value of the eigenvalues, and across dimensions 0 and 1. We can see that the persistence is being utilised, as there are discriminative features across the eigenvalues of each complex pair.

Table 4.1 shows our results for the persistent homology-based representations (PH), Graph Laplacian (GL), combinatorial Laplacian (CL, when higher-order interactions are present in the data), and the persistent Laplacian representations (PL). We report R^2 for the tasks from MoleculeNet, as they are regression tasks. We see that we typically exceed the performance of persistent homology with the persistent Laplacian across the MoleculeNet tasks, although there are some tasks on which PH performs better. In fact, in some of the tasks (which we have left in for completeness) topological tools do not work effectively at all. On the other hand, we see an almost identical performance between the persistent Laplacian and its non-persistent versions; we suggest this is due to the level of information available in the filtration in these cases. It makes sense that the persistent version of a tool – in this case, persistent homology – is only useful if the filtration we are analysing contains discriminative information.

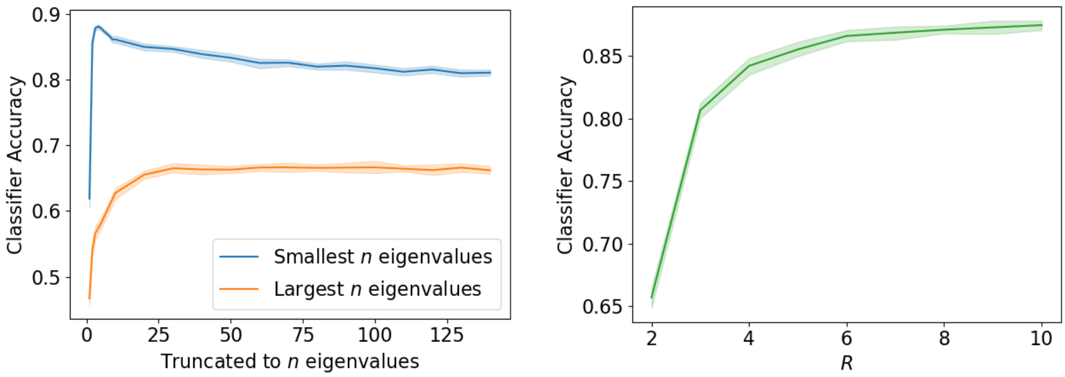
TABLE 4.1: Results across Persistent Homology (PH), Graph Laplacian (GL), Combinatorial Laplacian (CL) and Persistent Laplacian (PL) methods, on the MoleculeNet QM7 and QM7b datasets. Subtasks are from 3D coordinates (3D) and Coloumb Matrices (CM), and the numbers refer to tasks explained in Table B.1. We report the mean absolute error (MAE), as recommended by Wu et al. (2018). The MAE is the sum of absolute errors across the test set, divided by the number of samples in the test set. GL and PL use $R = 10$ and truncate to the mean number of eigenvalues across spectra. Where – is present we were using the coulomb matrices to construct graphs, so there are no higher-order interactions to study with the combinatorial Laplacian.

Dataset	PH	GL	CL	PL
QM7-3D	92.43 ± 0.629	69.23 ± 0.388	68.85 ± 0.421	68.86 ± 0.431
QM7-CM	232.0 ± 6.435	12.42 ± 0.191	–	12.38 ± 0.188
QM7b-CM-0	73.12 ± 1.893	12.39 ± 0.599	–	12.35 ± 0.597
QM7b-CM-1	1.916 ± 0.047	1.731 ± 0.038	–	1.729 ± 0.037
QM7b-CM-2	0.122 ± 0.004	0.096 ± 0.004	–	0.095 ± 0.004
QM7b-CM-3	0.484 ± 0.009	0.393 ± 0.008	–	0.393 ± 0.007
QM7b-CM-4	0.510 ± 0.009	0.362 ± 0.008	–	0.361 ± 0.007
QM7b-CM-5	0.651 ± 0.027	0.427 ± 0.010	–	0.426 ± 0.011
QM7b-CM-6	0.485 ± 0.011	0.406 ± 0.006	–	0.406 ± 0.007
QM7b-CM-7	0.548 ± 0.009	0.393 ± 0.010	–	0.392 ± 0.009
QM7b-CM-8	0.302 ± 0.005	0.272 ± 0.005	–	0.272 ± 0.005
QM7b-CM-9	0.305 ± 0.009	0.226 ± 0.006	–	0.226 ± 0.007
QM7b-CM-10	0.335 ± 0.007	0.306 ± 0.006	–	0.305 ± 0.006
QM7b-CM-11	0.236 ± 0.006	0.197 ± 0.004	–	0.197 ± 0.004
QM7b-CM-12	0.696 ± 0.015	0.352 ± 0.011	–	0.352 ± 0.012
QM7b-CM-13	0.726 ± 0.016	0.285 ± 0.009	–	0.284 ± 0.009

4.3.3 Ablation studies on eigenvalue truncation and R

Mapping the persistent Laplacian into a feature vector over a whole filtration as we describe in Section 4.2.2 requires two parameters: the truncation parameter, which determines how many of the eigenvalues in the spectra of each persistent Laplacian we consider, and the resolution R, which determines how we uniformly sample the continuous filtration to choose complex pairs to compute the persistent Laplacian of. We ran ablation studies using the MNIST dataset to consider how each of these choices effects the performance of the downstream classification task. In each case we fixed every parameter in the pipeline except for the one that we were investigating, and examined how changing that parameter effected the accuracy of the random forest classifier trained on the representations of the MNIST digits. The results of these experiments are shown in Figure 4.9.

Truncation. We investigated two different truncation strategies: truncating to the smallest n eigenvalues and to the largest n eigenvalues, the results of which are shown in Figure 4.9i. We found that truncating to the smallest eigenvalues (i.e., dropping the large eigenvalues) performed significantly better than truncating the largest eigenvalues (i.e., dropping the small eigenvalues). This aligns with (i) what we would expect, as the zero



(i) The effect of truncation.

(ii) The effect of R.

FIGURE 4.9: We investigate the effect of eigenvalue truncation and the parameter R on the efficacy of the persistent Laplacian on the MNIST classification task.

and small eigenvalues of the Laplacian contain topological and structural information about the underlying data, and (ii) what we have seen in the feature importance investigation (Figure 4.8), where the smallest eigenvalues are significantly more important to the performance of the model. In practice, then, we would recommend using only a small number of eigenvalues - the exact number will be be data-specific. Using too few eigenvalues results in a sharp drop-off in performance, and using too many leads to a gradual drop in performance, which we believe comes from significantly increasing the dimensionality of the resultant feature vector whilst adding a minimal amount of discriminative information.

Resolution. For very low values of R, the performance is not good - presumably because the model cannot exploit the additional information contained in the filtration. As we increase R, and finer details about the filtration become available via the feature vector, performance increases. We would expect an increase in R to correspond to an increase in the performance of the model, as additional details are becoming available. In particular, note that we fully capture the information contained in the persistent homology whenever R is sufficiently large that $I \leq j - i$ for every critical pair $K_i \subset K_j$ in the filtration (using the notation in Section 4.2.2). In Figure 4.9ii we see the performance flattening off for larger R, which likely happens as $I \to \hat{j} - \hat{\imath}$, where $K_{\hat{\imath}} \subset K_{\hat{j}}$ is the critical pair with the smallest filtration gap.

We also ran the MoleculeNet experiments with both $R = 5$ and $R = 10$, seeing a significant improvement in the accuracy with the increase of R. In particular, we can see in Table 4.2 that the persistent Laplacian went from only sometimes outperforming persistent homology to consistently being the best performing method. This improvement in performance demonstrates again that we are recovering more of the information within the persistent homology as we increase R.

TABLE 4.2: Results from Persistent Homology (PH) and Persistent Laplacian (PL) methods on the QM7b dataset. Note the increase in performance when increasing R from 5 to 10. Results are reported in MAE.

Dataset	PL ($R = 5$)	PL ($R = 10$)
QM7-3D	81.65 ± 0.418	68.86 ± 0.431
QM7-CM	23.21 ± 0.270	12.38 ± 0.188
QM7b-CM-0	22.58 ± 0.176	12.35 ± 0.597
QM7b-CM-1	1.989 ± 0.016	1.729 ± 0.037
QM7b-CM-2	0.112 ± 0.002	0.095 ± 0.004
QM7b-CM-3	0.518 ± 0.004	0.393 ± 0.007
QM7b-CM-4	0.481 ± 0.006	0.361 ± 0.007
QM7b-CM-5	0.613 ± 0.003	0.426 ± 0.011
QM7b-CM-6	0.533 ± 0.005	0.406 ± 0.007
QM7b-CM-7	0.532 ± 0.007	0.392 ± 0.009
QM7b-CM-8	0.346 ± 0.003	0.272 ± 0.005
QM7b-CM-9	0.263 ± 0.003	0.226 ± 0.007
QM7b-CM-10	0.394 ± 0.004	0.305 ± 0.006
QM7b-CM-11	0.225 ± 0.002	0.197 ± 0.004
QM7b-CM-12	0.458 ± 0.004	0.352 ± 0.012
QM7b-CM-13	0.378 ± 0.002	0.284 ± 0.009

4.3.4 Comparison to non-topological methods

We have demonstrated that the persistent Laplacian can outperform other topological baselines in MNIST and consistently outperform persistent homology in MoleculeNet, as well as discussed the additional information it can represent due to its theoretical properties. We consider some additional baselines here. Firstly, for MNIST, we evaluated the efficacy of using the flattened raw image as input into a random forest, as well as a convolutional neural network (LeCun and Bengio, 1998). Both significantly outperformed our topological methods, with the random forest scoring an accuracy of 0.9387 ± 0.0022 and the CNN scoring an accuracy of 0.9918 ± 0.0002. In comparison, the persistent Laplacian is the best scoring topological method, with an accuracy of 0.821 ± 0.011 (with $R = 5$ and truncating to the mean number of eigenvalues). In Table 4.3 we compare the performance of the persistent Laplacian to techniques from shallow and deep learning. Note that we do not use the QM7-3D dataset, as the coulomb matrix is typically used as the representation of the data for non-topological methods Rupp et al. (2012). In particular, we evaluated the performance of random forests (RF) and kernel ridge regression (KRR) on the flattened coulomb matrices. We also report the deep learning SOTA using deep tensor neural networks (DTNN) (Wu et al., 2018; Schütt et al., 2017) and gated graph recurrent neural networks (Shindo and Matsumoto, 2019). We find that we outperform the shallow methods, but are once again beaten by methods from deep learning.

We would argue that despite methods from deep learning beating the persistent Laplacian as a feature vector, we retain several advantages over deep methods.

(i) Firstly, the persistent Laplacian is being used as a feature vectorisation for downstream input into a simple classifier. It is a technique that relies heavily on theory, making the feature vector itself related to real-world understanding of shape and structure. In comparison, deep learning is famously a black box, and attempts to improve explainability and interpretability struggle in practice Belle and Papantonis (2021).

(ii) Secondly, the featurisation of the persistent Laplacian relies only on two understandble parameters (Section 4.3.3), and is easily reproducible. In comparison, deep learning techniques are often very sensitive to many opaque hyperparameters. Reproducing reported models is often impossible, with the 'reproducibility crisis' a known problem in the deep learning research community (Jean-Paul et al., 2019) .

Despite these points, clearly deep learning is a vastly powerful tool. In fact, topological tools are often at their most powerful when partnered with deep learning, as evidenced by the increasing popularity of Topological Machine Learning (Hensel et al., 2021). Research such as ours, which introduces a featurisation of the persistent Laplacian for representing data and evaluates the persistent Laplacian on ML baselines, provides a strong base for future work on the integration of the persistent Laplacian into Topological machine learning.

The place of topological tools within deep learning is something we will discuss more in the following discussion section.

4.4 Discussion

In this work we clearly demonstrate that we can outperform persistent homology: our experiments show that the persistent Laplacian can always outperform persistent homology across different datasets and tasks. We have explained that this is due to its appealing theoretical properties: it contains the homology in its zero eigenvalues (recovering the full persistent homology with sufficiently large R), and additional non-topological information is contained in its non-zero eigenvalues. Often, however, the performance of the graph Laplacian rivals that of the persistent 0-Laplacian. It appears that the MoleculeNet dataset does not benefit from the inclusion of persistence when computing the Laplacian. We included the results in our thesis regardless, as they demonstrate the efficacy of the persistent Laplacian in comparison to persistent homology, but further work should consider when the addition of persistence in the computation of the Laplacian provides additional discriminative data. Overall, the

TABLE 4.3: We also compare the persistent Laplacian to non-topological baselines. We trained random forests (RF) and kernel ridge regression (KRR) models on the flattened Coulomb matrices, as is standard in the literature (Rupp et al., 2012). We also report the results from the SOTA in deep learning: deep tensor neural networks (DTNN) (Schütt et al., 2017) and gated graph recurrent neural networks (GGRNN) (Shindo and Matsumoto, 2019). Our techniques outperform the shallow methods, but are beaten by deep learning. In the discussion (Section 4.4) we consider the place of topological techniques within the context of deep learning.

	Shallow Learning		TDA	Deep Learning	
Dataset	RF	KRR	PL	DTNN	GGRNN
QM7-CM	11.20 ± 0.261	26.16 ± 0.651	12.38 ± 0.188	8.75	–
QM7b-CM-0	38.17 ± 0.730	140.0 ± 3.752	12.35 ± 0.597	21.5	13.7
QM7b-CM-1	2.473 ± 0.027	2.668 ± 0.061	1.729 ± 0.037	1.26	1.02
QM7b-CM-2	0.130 ± 0.003	0.164 ± 0.004	0.095 ± 0.004	0.074	0.072
QM7b-CM-3	0.708 ± 0.011	1.192 ± 0.042	0.393 ± 0.007	0.192	0.140
QM7b-CM-4	0.768 ± 0.016	0.813 ± 0.02	0.361 ± 0.007	0.159	0.0915
QM7b-CM-5	1.057 ± 0.022	1.185 ± 0.034	0.426 ± 0.011	0.296	0.121
QM7b-CM-6	0.713 ± 0.011	1.181 ± 0.043	0.406 ± 0.007	0.214	0.176
QM7b-CM-7	0.853 ± 0.016	0.896 ± 0.022	0.392 ± 0.009	0.174	0.0940
QM7b-CM-8	0.437 ± 0.009	0.906 ± 0.027	0.272 ± 0.005	0.155	0.142
QM7b-CM-9	0.376 ± 0.012	0.426 ± 0.013	0.226 ± 0.007	0.129	0.092
QM7b-CM-10	0.495 ± 0.010	1.134 ± 0.036	0.305 ± 0.006	0.166	0.142
QM7b-CM-11	0.271 ± 0.007	0.312 ± 0.007	0.197 ± 0.004	0.139	0.118
QM7b-CM-12	0.693 ± 0.017	1.348 ± 0.026	0.352 ± 0.012	0.173	0.100
QM7b-CM-13	0.624 ± 0.015	1.362 ± 0.029	0.284 ± 0.009	0.149	0.0578

results suggest that this new theoretical development, which is able to seamlessly and efficiently combine topological and geometric information, should be more broadly researched and utilised in data analysis applications, particularly where key information for successful analysis is contained in the filtration.

In fact, the primary limitation of the persistent Laplacian in the context of other topological methods is its restriction to only computing individual complex pairs $K \subseteq L$, rather than being able to handle an entire filtration $(K_i)_i$ simultaneously, as persistent homology is. This is the reason for our resolution parameter R, which determines how regularly we uniformly sample from the filtration. Future work that extends the persistent Laplacian to the entire filtration would significantly improve its suitability to a far broader range of data analysis applications. We would suggest this as a key work for those interested in continuing this line of work.

Chapter 5

Fuzzy c-Means Clustering for Persistence Diagrams

Unsupervised learning refers to a class of machine learning algorithms that learn classes for data without being provided with a labelled dataset. Instead, they infer labels by assuming that points that are clustered together, i.e., nearby in some space and according to some metric, are in the same class. When combined with Topological Data Analysis, we can build unsupervised learning algorithms that infer labels from the topology of the underlying data. In k-means clustering, each point in a dataset is assigned to a cluster that minimises the distances from each cluster centre, which in turn are computed as the mean of the points assigned to each cluster. Alternatively computing the assignations and the cluster centres minimises the total distance from points to clusters. In fuzzy clustering, each point is assigned *k membership values* in $[0, 1]$ that represents the extent to which its assigned to each cluster. In this chapter we extend fuzzy c-means (FCM) clustering to the space of persistence diagrams, enabling fuzzy topological unsupervised learning. It is widely accepted that many real-world datasets are not clearly delineated into hard categories (Campello, 2007). Thus any algorithm that accounts for this is desirable, as evidenced by the large number of publications studying and using fuzzy clustering algorithms (Li and Lewis, 2016; Yang et al., 2019; Pantula et al., 2020). Our algorithm enables practitioners to study the fuzzy nature of data through a topological lens directly in the space of persistence diagrams.

Fréchet means of persistence diagrams, which we introduce in Section 3.1.2, are the natural concept to use for our cluster centres. They were first shown to be well-defined in persistence diagram space by (Mileyko et al., 2011, Section 4.1) and an algorithm for computation was given by (Turner et al., 2012, Algorithm 1). We extend their algorithm to compute the weighted Fréchet mean, verifying that their proof of convergence holds in the weighted case. We perform the convergence analysis for the fuzzy c-means algorithm in persistence diagram space, giving the same guarantees as Euclidean FCM clustering. Namely, we show that every convergent subsequence of iterates tends to a local minimum or saddle point. As this guarantee could lead to non-convergence in practice, we empirically evaluate convergence on randomly generated persistence diagrams with varying properties and find that it converges every time. We evaluate our algorithm using a variety of distances on persistence diagrams with the fuzzy RAND index (Campello, 2007), a standard measure of fuzzy cluster quality. We find that distances that take longer to compute result typically lead to a higher quality clustering, whereas approximations result in lower quality clusters but can be computed faster. By enabling the use of these distances, we can take advantage of the computational speed ups of vectorisations or approximate distances, whilst still having an end product in persistence diagram space.

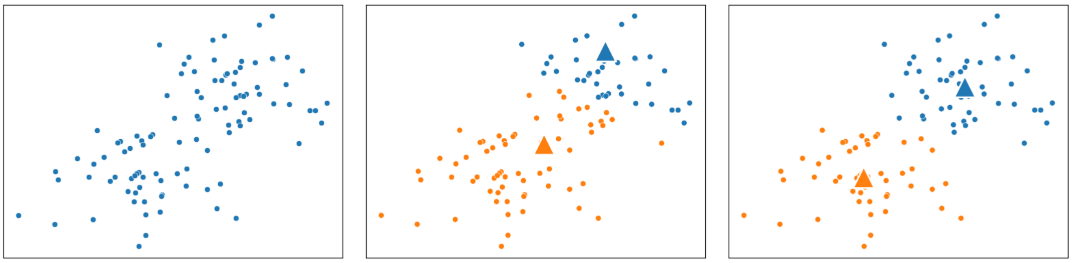
We demonstrate the practical value of fuzzy clustering persistence diagrams with an application to pre-trained model selection. This application is inspired by the somewhat surprising previous work showing that pre-trained deep learning models perform better on tasks if they have topologically similar decision boundaries (Ramamurthy et al., 2019). As one model can perform well on multiple tasks, this is a naturally fuzzy problem, and so ideally suited to our algorithm. We use our method to cluster pre-trained models and tasks (unseen datasets) using only the persistence diagrams of their decision boundaries. Not only is our algorithm able to successfully cluster models to the task it's originally trained on based just on the topology of its decision boundary, but we demonstrate that higher fuzzy cluster membership values imply better performance on tasks that the model has not been trained on. This gives a fuzzy unsupervised method for matching pre-trained models to tasks via their topologies.

Publications and contributions. The work in this chapter was published in Proceedings of Machine Learning Research via the NeurIPS Workshop on Symmetry and Geometry in Neural Representations (Davies et al., 2023a). The code and a generalisation to Riemannian manifolds were published in a white paper on topological and geometric learning (Miolane et al., 2021). I conceived of the project, performed the convergence analysis, and ran the experiments by myself. The materials science toy dataset was motivated and found via discussions with Jack Aspinall, a co-author on the paper (Davies et al., 2023a).

5.1 Clustering Algorithms

5.1.1 *k*-means clustering

Given n points in $\mathbb{R}^d$, for some $d \in \mathbb{N}^+$, we are interested in assigning each point to one of k labels, for a given $k \in \mathbb{N}^+$. The standard k-means clustering algorithm was first published by Lloyd (1982). Begin by randomly assigning each of the points x_i to a



(i) Data generated by two multivariate Gaussians in the plane.

(ii) After one iteration the clusters are not optimally labelling the data.

(iii) After 15 iterations the algorithm has learnt a better labelling.

FIGURE 5.1: An example of Euclidean k-means clustering on points in the plane. The class is indicated by the colour of the points and the triangles represent the computed cluster centres y_j.

label $1, \ldots, k$. Let $S_j, j \in [1, \ldots, j]$ be the points x_i assigned to label j (this is the 'naieve' method to find initial clusters). Compute the cluster centres via $y_j \in \mathbb{R}^d$ by taking the mean of the points with label j, computing

$$y_j = \frac{1}{|S_j|} \sum_{x_i \in S_j} x_i, \text{ for } j = 1, \ldots, k. \tag{5.1}$$

Then assign each point to its nearest cluster centre via

$$Y_j = \left\{ x_i : |x_i - y_j|^2 \leq |x_i - y_l|^2 \text{ for all } l = 1, \ldots, k \right\}. \tag{5.2}$$

By alternating these two steps, you aim to minimise the least squares distance from each point to each centroid, learning labels in an unsupervised fashion for each of data points. The algorithm converges when each Y_j is constant, i.e., when the labels no longer change. Note that convergence to a global minimum is not guaranteed (Lloyd, 1982). An example of this algorithm on simple data generated by two Gaussians in the plane is shown in Figure 5.1.

As we saw in Section 3.1.2, Mileyko et al. (2011) have shown that Fréchet means are well-defined in the space of persistence diagrams and Turner et al. (2012) has developed an algorithm to compute them. This was used by Maroulas et al. (2017) for k-means clustering in persistence diagram space, simply replacing the mean of the points in S_j in Equation 5.1 with the Fréchet mean of the diagrams in S_j, and the Euclidean distance with the Wasserstein distance when assigning diagrams to clusters.

5.1.2 Examples of k-means clustering on toy data

We now give some examples of k-means clustering, via our implementation of k-means clustering in persistence diagram space, and ran experiments on toy data to demonstrate its efficacy at inferring labels based on the topology of the underlying data.

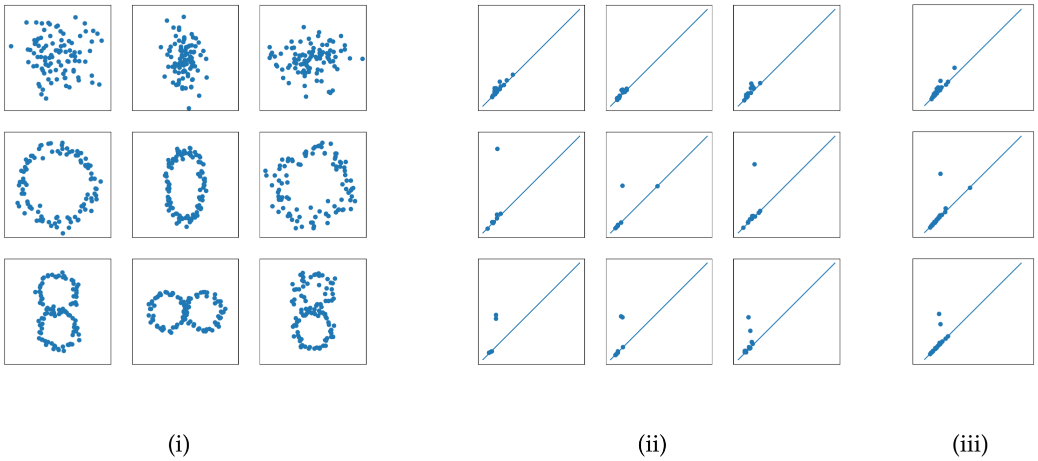


FIGURE 5.2: An example *k*-means clustering of persistence diagrams with synthetic data. In i we show nine synthetic datasets, consisting of noise, one hole, or two holes. In ii we compute the 1-persistence diagrams, which we recall counts the number of holes. We cluster these persistence diagrams, resulting in three learnt cluster centres, shown in iii. The cluster centres have zero, one, and two significantly off-diagonal points: the clustering algorithm has learnt the topological features of the datasets.

Synthetic Data Example. Figure 5.2 shows an example with synthetic data. In Figure 5.2i we generate nine datasets, three of which are just noise, three of which are shaped like rings, and three of which are figure of eights. Our goal is to use clustering on the space of persistence diagrams to infer classes for the datasets based on their topology. We compute the 1-persistence diagrams (Figure 5.2ii) and run our implementation of *k*-means clustering. In Figure 5.2iii we examine the clusters that the algorithm has converged to, and we can see that they have zero, one, and two off-diagonal points in the persistence diagram,, corresponding to zero, one, and two loops in the datasets. The *k*-means clustering has worked as expected, and grouped the nine datasets by their topology.

Materials Science Example. In materials science the topology of chemical structures is key for understanding new materials. A key property for machine learning in materials science has been identified as "invariance to the basis symmetries of physics [...] rotation, reflection, translation" (Jonathan Schmidt and Marques, 2019). Persistence diagrams, which capture affine transformation-invariant properties of datasets, are thus very well-suited for application in this domain. We demonstrate this with a toy example from materials science. The large majority of solids are comprised of regularly repeating unit cells of atoms. This *periodic structure* directly determines the properties of a material (Hull and Bacon, 2011) and it has been predicted that machine learning will reveal presently unknown links between structure and property by identifying new trends across materials (Meredig, 2019; Wei et al., 2019). The most common periodic structures, particularly amongst pure metals, are face-centred cubic (FCC) structures and body-centred cubic (BCC) structures (Putnis, 1992).

We use atomic positions for the unit-cells of iron MP-150 and iron MP-13 from the Materials Project (Jain et al., 2013), representing BCC and FCC structures respectively, for the first toy dataset (displayed in Figures 5.3i and 5.3ii). For our second toy dataset we use diamond (Figure 5.3iii) and cis-hinged polydiacetylene (Figure 5.3iv) unit-cell atomic positions from the Samara Carbon Allotrope Database (Hoffmann et al., 2016). We simulate distinct collections of periodic structures by transforming the atomic coordinates, with no information about bonds given to the algorithms. The properties of persistence diagrams mean that we can cluster the atomic coordinates derived from the same base unit-cell regardless of the transformations applied to the coordinate system. We can see this holds in Figure 5.3. We use the distance matrix between diagrams and cluster centres to represent persistence diagram space in the plane (Kruskal, 1964)) to assist with visualisation, and we can see that the persistence diagrams are clustered together (Figures 5.3ix, x), enabling unsupervised learning using the topology of the materials.

5.1.3 Euclidean fuzzy c-means clustering

It is widely accepted that many real-world datasets do not fall into clearly delineated clusters (Campello, 2007). In this case, it is desirable to compute partial membership values for a data point x_i to each cluster centre y_i. In particular, the values $r_{ij} \in [0,1]$ denote the extent to which data point x_i is assigned to cluster centre y_j. They are computed so that $\sum_{j=1}^{k} r_{ij}^2 = 1$ for each $i = 1, \ldots, n$. The fuzzy c-means clustering algorithm was first given by Dunn (1973), with the convergence studied by Bezdek (1980). An error with his convergence proof was fixed in Bezdek et al. (1987).

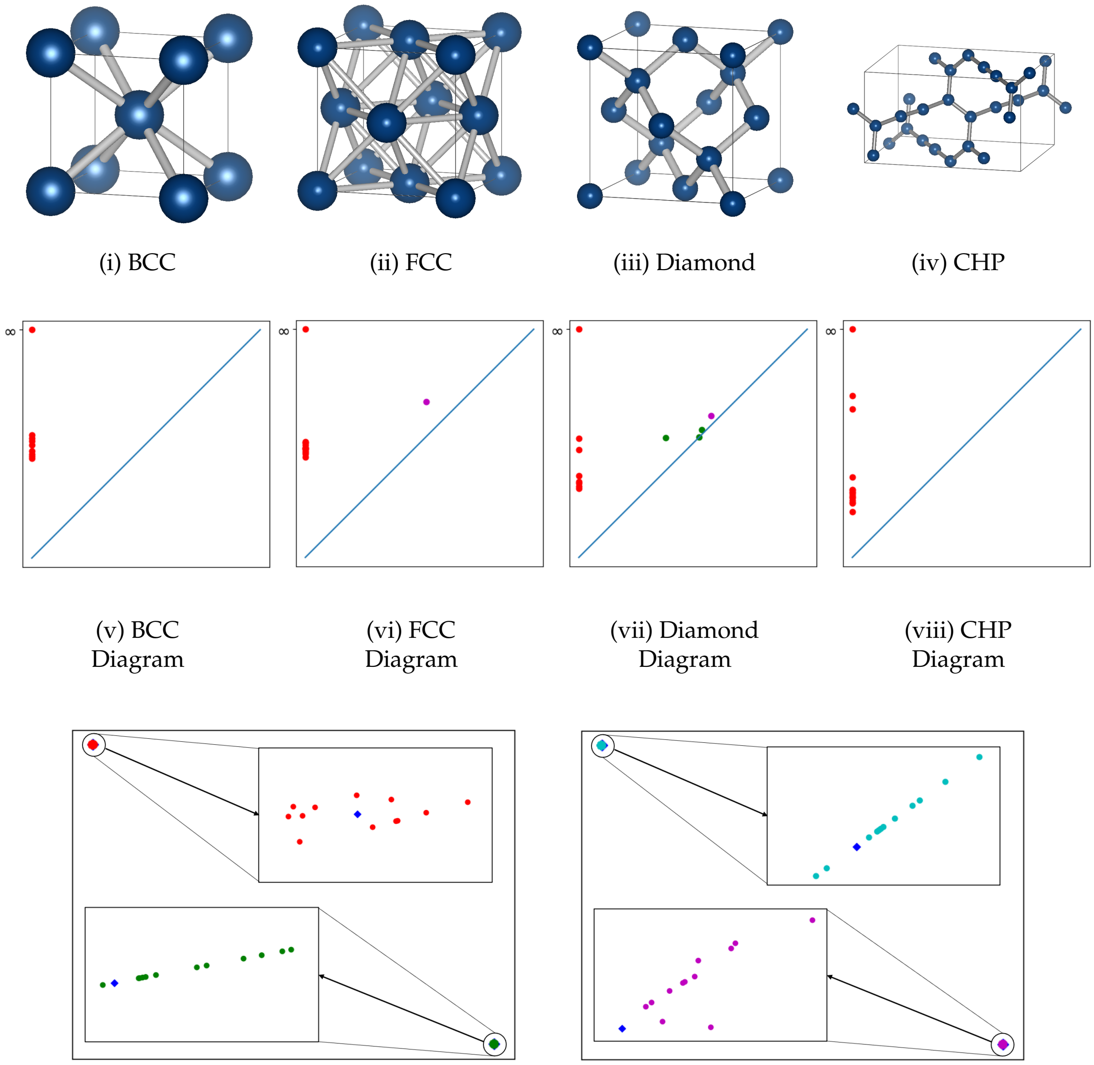
The algorithm to compute the membership values and fuzzy clustering is very similar to k-means. Instead of assigning each point x_i to a label, you compute fuzzy membership values via

$$r_{ij} = \left(\sum_{l=1}^{k} \frac{|x_i - y_j|}{|x_i - y_l|} \right)^{-1},$$

and update the cluster centres via a weighted mean

$$y_j = \frac{1}{\sum_{i=1}^{n} r_{ij}^2} \sum_{i=1}^{n} r_{ij}^2 x_i.$$

In the following section we extend this formulation to the space of persistence diagrams, verifying that the convergence properties first proven by Bezdek (1980) still hold in our setting. The example that we introduced in Figure 5.1 is re-run in Figure 5.4 with the fuzzy c-means algorithm. Note now the continuous membership, represented by a colour map, rather than just two binary classes.



(i) BCC

(ii) FCC

(iii) Diamond

(iv) CHP

(v) BCC Diagram

(vi) FCC Diagram

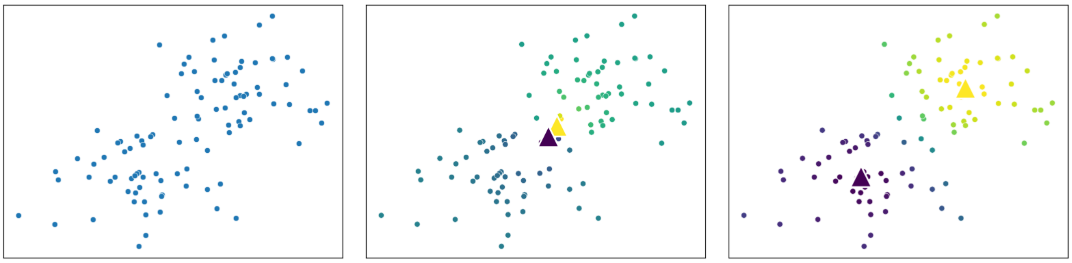
(vii) Diamond Diagram

(viii) CHP Diagram

(ix) Planar representation of BCC and FCC persistence diagram clustering.

(x) Planar representation of diamond and CHP persistence diagram clustering.

FIGURE 5.3: We cluster the persistence diagrams of periodic structures, demonstrating persistence diagrams ability to distinguish different periodic structures based on their topology. From left to right we display four examples of the periodic structures, followed by their persistence diagrams (red is 0-PH, green is 1-PH, and magenta is 2-PH). In the final panels we show persistence diagram space represented in the plane. Each circular point is a persistence diagram, and the diamond points are the learnt cluster centres.



(i) Data generated by two multivariate Gaussians in the plane.

(ii) After one iteration the cluster centres are almost identical.

(iii) After 15 iterations the clustering is far better.

FIGURE 5.4: An example of Euclidean fuzzy c-means clustering on the same data as used in Figure 5.1. The cluster centres are represented by the triangles. The membership values are represented by the continuous colour map on the points. Note that points between two cluster centres now partially belong to both.

5.2 Algorithmic Design

5.2.1 Clustering persistence diagrams

We use the 2-Wasserstein L_2 distance as a metric on the space of persistence diagrams, as defined in Equation 2.1. Recall that given diagrams D_1, D_2, the distance is defined as

$$W_2(D_1, D_2) = \left(\inf_{\gamma : D_1 \to D_2} \sum_{x \in D_1} ||x - \gamma(x)||_2^2 \right)^{1/2},$$

where the infinum is taken over all bijections $\gamma : D_1 \to D_2$. We also saw that in order for the Wasserstein distance to be well defined between persistence diagrams with different numbers of points, we add the points $\{(a, a) : a \in \mathbb{R}\}$ to each diagram with infinite multiplicity; we call this the diagonal and denote it Δ. If an off-diagonal point is matched to the diagonal the transportation cost is simply the shortest distance to the diagonal. We work in the space $\mathcal{D}_2 = \{D : W_2(D, \Delta) < \infty\}$, as that is required for the convergence. Doing so is standard practice in applications of TDA to machine learning (e.g., Chepushtanova et al. (2015)). To ensure that our persistence diagrams are all in this space, we map points at infinity to a hyperparameter T that is much larger than other death values in the diagram. As in our applications we are using the Rips complex, there will only ever be one point at infinity (which will be in 0-PH). This hyperparameter ensures that the one point at infinity will always be matched to the corresponding point at infinity when computing the Wasserstein distance between diagrams.

Our r_{jk} is the same as traditional FCM clustering, adapted with the Wasserstein distance. That is,

$$r_{jk} = \left(\sum_{l=1}^{c} \frac{W_2(\mathrm{M}_k, D_j)}{W_2(\mathrm{M}_l, D_j)} \right)^{-1}. \tag{5.3}$$

To update M_k, we compute the weighted Fréchet mean of the persistence diagrams $\{D_j\}_{j=1}^n$ with the weights $\{r_{jk}^2\}_{j=1}^n$. Specifically,

$$M_k \longleftarrow \arg\min_{\hat{D}} \sum_{j=1}^{n} r_{jk}^2 W_2(\hat{D}, D_j)^2, \text{ for } k = 1, \dots, c. \tag{5.4}$$

As the weighted Fréchet mean extends weighted centroids to $\mathcal{D}_2$, this gives our fuzzy cluster centres. The computation of the weighted Fréchet mean is covered in Section 5.2.2. By alternatively updating Equations 5.3 and 5.4 we get a sequence of iterates. We now analyse the convergence properties of these iterates.

Algorithm 3 FPDCluster

```
**Input** Diagrams $\mathrm{D} = \{\mathrm{D}_j\}_{j=1}^n$, number of clusters $c$, maximum iterations MAXITER
**Output** Cluster centres $\mathrm{M} = \{\mathrm{M}_k\}_{k=1}^c$, membership values $R = \{r_{jk}\}$
D = ADDDIAGONALS(D)
M = INITCENTRES(D)
for count in 1..MAXITER do
    for $j$ in 1..$n$ do
        for $k$ in 1..$c$ do
            $r_{jk} \leftarrow \left(\sum_{l=1}^{c} \frac{W_2(\mathrm{M}_k, \mathrm{D}_j)}{W_2(\mathrm{M}_l, \mathrm{D}_j)}\right)^{-1}$
        end for
    end for
    for $k$ in 1..$c$ do
        $\mathrm{M}_k \leftarrow$ WFRECHETMEAN$(\mathrm{D}, R_k)$
    end for
end for
return M, $R$
```

We first need to consider our separate updates as a single update procedure. Let $F : \mathrm{M} \mapsto R$ be defined by Equation 5.3 and $G : R \mapsto \mathrm{M}$ be defined by Equation 5.4, and for $R = \{r_{jk}\}$ and $\mathrm{M} = \{\mathrm{M}_k\}$ consider the sequence

$$\left\{T^{(l)}(R, \mathrm{M}) : l = 0, 1, \dots\right\}, \text{ where } T(R, M) = (F \circ G(R), G(R)).$$

We wish to show convergence of the iterates of T to a local minimum or saddle point of the cost function

$$J(R, \mathrm{M}) = \sum_{j=1}^{n} \sum_{k=1}^{c} r_{jk}^2 W_2(\mathrm{M}_k, \mathrm{D}_j)^2.$$

The two stage update process of T is too complicated to use standard fixed point theorems, so following (Bezdek, 1980, Section 3) we shall use Zangwill's Convergence Theorem (Zangwill, 1969), which we state in the general case before demonstrating how it corresponds to convergence in our setting. This theorem allows us to study the convergence of an algorithm, i.e., an iterated function. It studies whether iterates of the algorithm converge to a desired point in the so-called *solution set*, which we consider in terms of our cost function, which has to fulfill specific conditions to be a *descent function*.

Theorem 5.1 (Zangwill's Convergence Theorem)**.** *First, let $X = \mathbb{R}^d$ for some $d > 0$ have the standard topology. Let $A : X \to 2^X$ be a point-to-set map on X. Given $x_0 \in X$, generate a sequence $\{x_k\}_{k=1}^{\infty}$ such that $x_{k+1} \in A(x_k)$ for every k. Let $\Gamma \subset X$ be a subset of X called the solution set, and suppose that the following hold.*

(i) The sequence $\{x_k\} \subset S \subset X$ for a compact set S.

(ii) There exists a continuous function $Z : X \to \mathbb{R}$ such that if $x \notin \Gamma$ then $Z(y) < Z(x)$ for all $y \in A(x)$, and if $x \in \Gamma$ then $Z(y) \leq Z(x)$ for all $y \in A(x)$. The function Z is called a descent function.

(iii) The map A is closed on $X \setminus \Gamma$.

Then every convergent subsequence of $\{x_k\}$ tends to a point in the solution set Γ.

Our algorithm is the update function T. We define our solution set as

$$\Gamma = \left\{(R, \mathrm{M}) : J(R, \mathrm{M}) < J(\hat{R}, \hat{\mathrm{M}}) \; \forall \; (\hat{R}, \hat{\mathrm{M}}) \in B((R, \mathrm{M}), r)\right\}$$

for some $r > 0$, where the ball surrounding R is the Euclidean ball in $\mathbb{R}^{nc}$ and the ball surrounding M is $\cup_{k=1}^{c} B_{W_2}(\mathrm{M}_k, r)$. This set contains the local minima and saddle points of the cost function (Bezdek et al., 1987). We wish to show that our cost function $J(R, \mathrm{M})$ is the descent function Z. We proceed by verifying each of the requirements for Zangwill's Convergence Theorem.

Lemma 5.2. *Every iterate $T^{(l)}(R, \mathrm{M}) \in [0, 1]^{nc} \times \mathrm{conv}(\mathrm{D})^c$, where*

$$\mathrm{conv}(\mathrm{D}) = \bigcup_{k=1}^{c} \bigcup_{\gamma_j} \bigcup_{i=1}^{m} \mathrm{conv}\{\gamma_j(y^{(i)}) : j = 1, \ldots, n\},$$

with γ_j a bijection $\mathrm{M}_k \to \mathrm{D}_j$ and $\mathrm{conv}\{\gamma_j(y^{(i)}) : j = 1, \ldots, n\}$ the ordinary convex hull in the plane. Furthermore, $[0, 1]^{nc} \times \mathrm{conv}(\mathrm{D})^c$ is compact.

Proof. By construction, every $r_{jk} \in [0, 1]$. Since $j = 1, \ldots, n$ and $k = 1, \ldots, c$, we can view R as a point in $[0, 1]^{nc}$, and so every iterate of R is in $[0, 1]^{nc}$. We shall show that for a fixed k and a fixed bijection $\gamma_j : \mathrm{M}_k \to \mathrm{D}_j$, each updated $y^{(i)}$ is contained in a convex combination of $\{\gamma_j(y^{(i)}) : j = 1, \ldots, n\}$. Where $\gamma_j(y^{(i)}) = \Delta$, let $\gamma_j(y^{(i)}) = w_\Delta$, where w_Δ is the point closest to $y^{(i)}$ on the diagonal Δ, as this is the update point we use for the diagonal. Since there are a finite number of off-diagonal points, each updated M_k is therefore contained in the union over all bijections and all points $y^{(i)}$ of the convex combination of $\{\gamma_j(y^{(i)}) : j = 1, \ldots, n\}$. By also taking the union over each k, we show that every iterate of M must be contained in the finite triple-union of the convex combination of each possible bijection. To show that each updated $y^{(i)}$ is contained in a convex combination of $\{\gamma_j(y^{(i)}) : j = 1, \ldots, n\}$, recall that $y^{(i)} = \left(\sum_{j=1}^n r_{jk}^2\right)^{-1} \sum_{j=1}^n r_{jk}^2 \gamma_j(y^{(i)})$. Letting $t_j^{(i)} = r_{jk}^2 \left(\sum_{j=1}^n r_{jk}^2\right)^{-1}$, clearly each $t_j^{(i)} > 0$ and $\sum_{j=1}^n t_j^{(i)} = 1$. Since $y^{(i)} = \sum_{j=1}^n t_j^{(i)} \gamma_j(y^{(i)})$, each $y^{(i)}$ is contained in the convex combination. Therefore $T^{(l)}(R, \mathrm{M}) \in [0,1]^{nc} \times \mathrm{conv(D)}^c$ for each $l = 0, 1, \ldots$.

Now, $[0,1]$ is closed and bounded, so is compact. The convex hull of points in the plane is closed and bounded, so $\mathrm{conv}\{\gamma_j(y^{(i)}) : j = 1, \ldots, n\}$ is compact. Since finite unions and finite direct products of compact sets are compact, $[0,1]^{nc} \times \mathrm{conv(D)}^c$ is also compact. □

Lemma 5.3. *The cost function $J(R, \mathrm{M})$ is a descent function, as defined in Theorem 5.1(ii).*

Proof. The cost function J is continuous, as it's a sum, product and composition of continuous functions. Furthermore, we have that for any $(R, \mathrm{M}) \notin \Gamma$,

$$J(T(R, \mathrm{M})) = J(F \circ G(R), G(R)) < J(R, G(R)) < J(R, M),$$

where the first inequality is due to Bezdek (1980, Proposition 1), and the second inequality comes from the definition of the Fréchet mean. If $(R, \mathrm{M}) \in \Gamma$ then the strict inequalities include equality throughout. □

We finish by applying Zangwill's Convergence Theorem to the setting we have described.

Theorem 5.4. *For any (R, M), every convergent subsequence of $\{T^{(l)}(R, \mathrm{M}) : l = 0, 1, \ldots\}$ tends to a local minimum or saddle point of the cost function J.*

Proof. We proceed with Zangwill's Convergence Theorem (Theorem 5.1). Our algorithm is the update function T, our solution set is Γ, and our descent function is the cost function $J(R, \mathrm{M})$. By Lemma 5.2, every iterate is contained within a compact set. By Lemma 5.3, J is a descent function. Finally, since our function T only maps points in the plane to points in the plane, it is a closed map. The theorem follows by applying Theorem 5.1. □

We have shown that the FCM algorithm in persistence diagram space has the same convergence guarantee as Euclidean FCM: that every convergent subsequence will converge to a minimum or saddle point of the cost function. However, so far we have assumed that we can compute the weighted Fréchet mean; in the next section we add weights to the algorithm given by Turner et al. (2012) so we can compute it in practice.

5.2.2 Computing the weighted Fréchet mean

In Algorithm 3 we add copies of the diagonal to ensure that each diagram has the same cardinality; denote this cardinality as m. To compute the weighted Fréchet mean, we need to find $\mathrm{M}_k = \{y^{(i)}\}_{i=1}^{m}$ that minimises the Fréchet function in Equation 5.4. The Wasserstein distance requires the optimal matching given by a bijection $\gamma_j : y^{(i)} \mapsto x_j^{(i)}$ for each j. Supposing we know these bijections, we can rearrange the Fréchet function into the form $F(\mathrm{M}_k) = \sum_{j=1}^{n} r_{jk}^2 W_2(\mathrm{M}_k, D_j)^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} r_{jk}^2 ||y^{(i)} - x_j^{(i)}||^2$.

In this form, the summand is minimised for $y^{(i)}$ by the weighted Euclidean centroid of the points $\{x_j^{(i)}\}_{j=1}^{n}$. Therefore to compute the weighted Fréchet mean, we need to find the correct bijections. To this end, we adapt the algorithm for the computation of the unweighted case given by (Turner et al., 2012, Algorithm 1).

We start by using the Hungarian algorithm (Munkres, 1957) to find an optimal matching between M_k and each D_j. Given a D_j, for each point $y^{(i)} \in M_k$, the Hungarian algorithm will assign an optimally matched point $x_j^{(i)} \in D_j$. Specifically, we find matched points

$$\left[x_j^{(i)}\right]_{i=1}^{m} \longleftarrow \text{Hungarian}(\mathrm{M}_k, \mathrm{D}_j), \text{ for each } j = 1, \ldots, n. \tag{5.5}$$

Now, for each $y^{(i)} \in \mathrm{M}_k$ we need to find the weighted average of the matched points $\left[x_j^{(i)}\right]_{j=1}^{n}$. However, some of these points could be copies of the diagonal, so we need to consider three distinct cases: that each matched point is off-diagonal, that each one is a copy of the diagonal, or that the points are a mixture of both. We start by partitioning $1, \ldots, n$ into the indices of the off-diagonal points $\mathfrak{J}_{\mathrm{OD}}^{(i)} = \left\{j : x_j^{(i)} \neq \Delta\right\}$ and the indices of the diagonal points $\mathfrak{J}_{\mathrm{D}}^{(i)} = \left\{j : x_j^{(i)} = \Delta\right\}$ for each $i = 1, \ldots, m$. Now, if $\mathfrak{I}_{\mathrm{OD}} = \emptyset$ then $y^{(i)}$ is a copy of the diagonal. If not, let $w = \left(\sum_{j \in \mathfrak{J}_{\mathrm{OD}}^{(i)}} r_{jk}^2\right)^{-1} \sum_{j \in \mathfrak{J}_{\mathrm{OD}}^{(i)}} r_{jk}^2 x_j^{(i)}$ be the weighted mean of the off-diagonal points. If $\mathfrak{J}_{\mathrm{D}}^{(i)} = \emptyset$, then $y^{(i)} = w$. Otherwise, let w_Δ be the point on the diagonal closest to w. Then our update is

$$y^{(i)} \longleftarrow \frac{\sum_{j \in \mathfrak{J}_{\mathrm{OD}}^{(i)}} r_{jk}^2 x_j^{(i)} + \sum_{j \in \mathfrak{J}_{\mathrm{D}}^{(i)}} r_{jk}^2 w_\Delta}{\sum_{j=1}^{n} r_{jk}^2}, \text{ for } i = 1, \ldots, m. \tag{5.6}$$

We summarise this process in Algorithm 4.

The following theorem shows that Algorithm 4 will converge to the weighted Fréchet mean, as required.

Theorem 5.5. *Given diagrams* D_j*, membership values* r_{jk}*, and the Fréchet function* F *given by*

$$F(\hat{\mathrm{D}}) = \sum_{j=1}^{n} r_{jk}^2 F_j(\hat{\mathrm{D}}), \text{ with } F_j(\hat{\mathrm{D}}) = W_2(\hat{\mathrm{D}}, \mathrm{D}_j)^2, \tag{5.7}$$

Algorithm 4 Weighted Fréchet Mean

```
Input Diagrams $\mathrm{D} = \{\mathrm{D}_j\}_{j=1}^n$, Weights $R_k = \{r_{jk}\}_{j=1}^n$ (fixed $k$)
Output Weighted Fréchet mean $\mathrm{M}_k = \{y^{(i)}\}_{i=1}^m$
$m \leftarrow \max_{1 \leq j \leq n} |\, \mathrm{D}_j \,|$
for $j$ in $1..n$ do
    $\left[x_j^{(i)}\right]_{i=1}^m \leftarrow$ HungarianAlgorithm$(\mathrm{M}_k, \mathrm{D}_j)$
end for
while $\left\{\left[x_j^{(i)}\right]_{i=1}^m\right\}_{j=1}^n \neq \left\{\left[\hat{x}_j^{(i)}\right]_{i=1}^m\right\}_{j=1}^n$ do
    for $i$ in $1..m$ do
        $\mathfrak{J}_{\mathrm{OD}}^{(i)} = \{j : x_j^{(i)} \neq \Delta\}$
        $\mathfrak{J}_{\mathrm{D}}^{(i)} = \{j : x_j^{(i)} = \Delta\}$
        if $\mathfrak{J}_{\mathrm{OD}}^{(i)} = \emptyset$ then
            $y^{(i)} \leftarrow \Delta$
        else
            $w = \left(\sum_{j \in \mathfrak{J}_{\mathrm{OD}}^{(i)}} r_{jk}^2\right)^{-1} \sum_{j \in \mathfrak{J}_{\mathrm{OD}}^{(i)}} r_{jk}^2 x_j^{(i)}$
            if $\mathfrak{J}_{\mathrm{D}}^{(i)} = \emptyset$ then
                $y^{(i)} \leftarrow w$
            else
                $y^{(i)} \leftarrow \frac{\sum_{j \in \mathfrak{J}_{\mathrm{OD}}^{(i)}} r_{jk}^2 x_j^{(i)} + \sum_{j \in \mathfrak{J}_{\mathrm{D}}^{(i)}} r_{jk}^2 w_\Delta}{\sum_{j=1}^n r_{jk}^2}$
            end if
        end if
    end for
    $\left\{\left[\hat{x}_j^{(i)}\right]_{i=1}^m\right\}_{j=1}^n \leftarrow \left\{\left[x_j^{(i)}\right]_{i=1}^m\right\}_{j=1}^n$
    for $j$ in $1..n$ do
        $\left[x_j^{(i)}\right]_{i=1}^m \leftarrow$ Hungarian$(\mathrm{M}_k, \mathrm{D}_j)$
    end for
end while
return $\{y^{(i)}\}_{i=1}^m$
```

then $\mathrm{M}_k = \{y^{(i)}\}_{i=1}^m$ *is a local minimum of F if and only if there is a unique optimal pairing from* M_k *to each of the* D_j*, denoted* γ_j*, and each* $y^{(i)}$ *is updated via Equation 5.6.*

Since the proof of this theorem essentially amounts to adding r_{jk}^2 coefficients to the proof given by Turner et al. (2012, Theorem 3.3), we omit it from the main body of this thesis. The details are available in Appendix A.

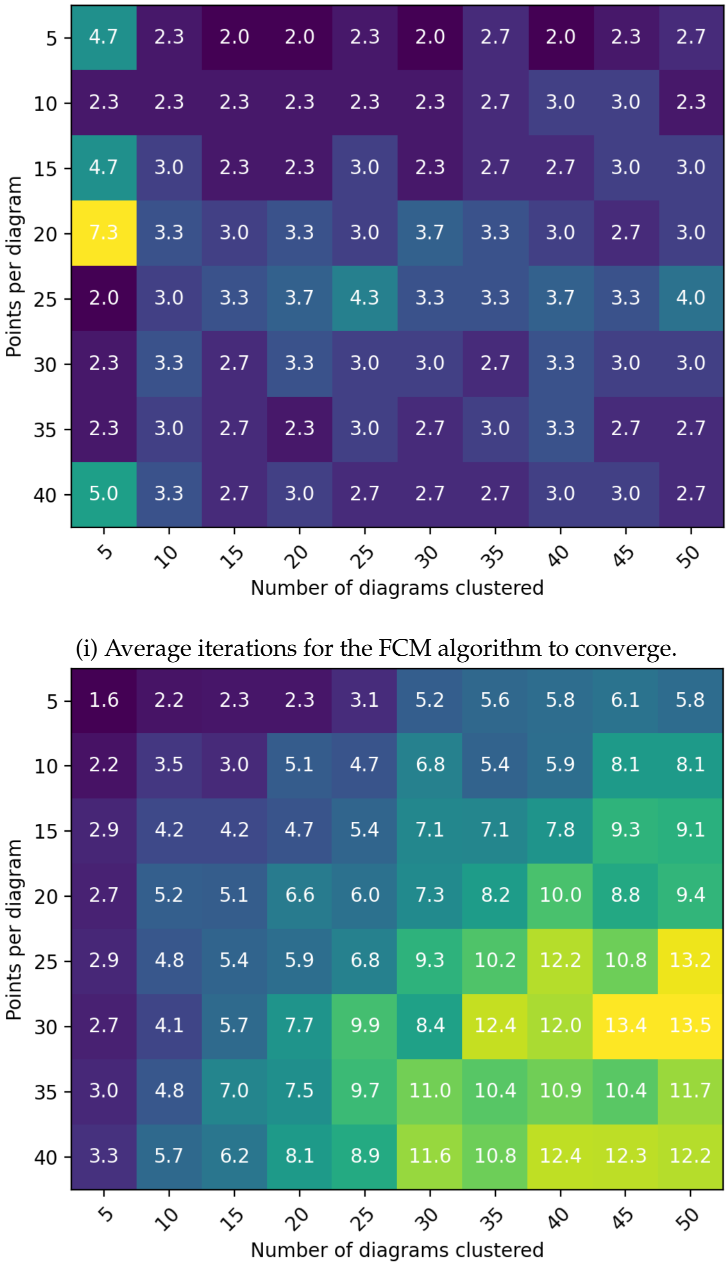
5.3 Empirical Behaviour Analysis

We have shown a theoretical guarantee that every convergent subsequence will tend to a local minimum, but in practice it remains important that our algorithm will converge within a reasonable timeframe. To empirically investigate whether or not this happens in practice we ran experiments on a total of 825 uniformly randomly generated persistence diagrams, recording the number of iterations and cost functions for both the FCM clustering and the weighted Fréchet mean (WFM) computation. We considered the FCM to have converged when the cost function remained within $\pm 0.5\%$. As explained in Section 5.2.2, the WFM converges when the matching stays the same. The results, shown in Figure 5.5 demonstrate that the FCM clustering consistently converges within 5 iterations, regardless of the number of diagrams and points per diagram (note that the time per iteration increases as the number of points/diagrams increases, even if the number of iterations remains stable). We had no experiments in which the algorithm did not converge. The WFM computation requires more iterations as both number of diagrams and number of points per diagram increases, but we once again experienced no failures to converge in each of our experiments. In general, running the algorithm offered no difficulties on a 2018 MacBook Pro (with a 1.4 GHz i5 and 8GB RAM) and we believe the algorithm is ready for use by practitioners in the TDA community.

The use of the Wasserstein distance in the clustering still means that some datasets with many points will be difficult to compute the distances for. To explore solutions to this, we investigated the substitution of different distances in Equation 5.3. Specifically, we evaluated the quality of learnt clusters using the fuzzy RAND index (Campello, 2007) - a standard measure of cluster quality - when using the 2-Wasserstein distance, bottleneck distance, sliced Wasserstein kernel (Carriere et al., 2017), heat kernel, and L2 distance between persistence images (Chepushtanova et al., 2015). More details on each of these distances are available in Section 3.2. We find that typically the more expensive optimal matching-based distances perform best, whereas approximations and embedding-based distances are faster but score lower. The distance between persistence images may provide the best compromise in practice. These results are shown in Figure 5.6.

5.4 Application to Pre-Trained Model Selection

Previous work has suggested that deep learning models will perform better on unseen datasets which have a similar topological complexity to the model's decision boundary (Ramamurthy et al., 2019). In fact, there is an increasing amount of work studying the link between topology and neural network performance (Rieck et al., 2019c; Guss and Salakhutdinov, 2018). To this end we utilise our algorithm to cluster the topology of the decision boundaries of pre-trained models and tasks (labelled datasets). Given a task we find the nearest cluster centre, then select the models nearest to that centre. Even though the only information utilised for the model selection is the topology of the decision boundaries, we find that it consistently selects the top performing model as the first choice, and additional choices perform above average, despite not being trained on the task. This further confirms that the topology of the decision boundary is indicative of



(i) Average iterations for the FCM algorithm to converge.

(ii) Average iterations for the WFM algorithm to converge.

FIGURE 5.5: Heatmaps showing average number of iterations for fuzzy clustering of persistence diagrams (5.5i) and the weighted Fréchet mean computation (5.5ii) to converge. Convergence of the FCM algorithm is determined when the cost function is stable to within ±0.5%. Convergence experiments were carried out on a total of 825 persistence diagrams, including three repeats.

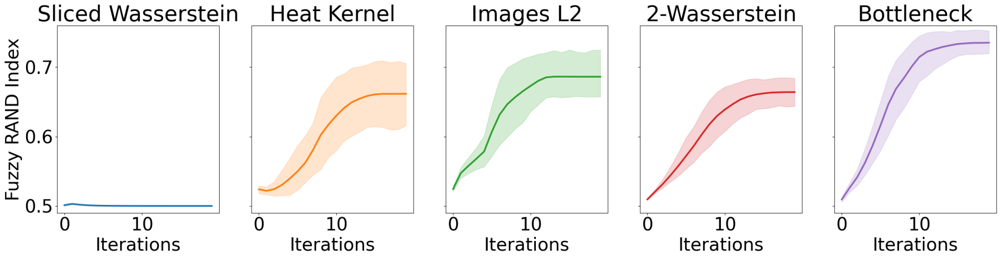


FIGURE 5.6: For computational speedups practitioners may wish to use different distances in the clustering algorithm. We use the fuzzy RAND index (Campello, 2007) to evaluate cluster quality when using some common distances. In practice, you may be best using the distances between persistence images, although the more expensive bottleneck distance will provide the best results.

generalisation ability to unseen tasks. Furthermore, our algorithm is able to exploit this information to learn cluster centres that consistently select the best performing models on tasks.

Specifically, given a dataset with n classes, we fix one class to define $n-1$ *tasks*: binary classification of the fixed class vs each of the remaining classes. On each of these tasks, we train a *model*. We compute the decision boundary of the model f, defined as $(x_1, \ldots, x_m, f(x))$ where $f(x)$ is the model's prediction for $x = (x_i)_i$, and the decision boundary of the tasks, defined via the labelled dataset as $(x_1, \ldots, x_m, y)$ where y is the true label. We compute the 1-persistence diagrams of the tasks' and models' decision boundaries and cluster them to obtain membership values and cluster centres. To view task and model proximity through our clustering, we find the cluster centre with the highest membership value for each task, and consider the models closest to that cluster centre. Note that model selection is naturally a fuzzy task: one model can (and does) perform well on multiple tasks. Therefore this is a task best suited to fuzzy clustering.

In particular, in order to assign each task to the top-ranked models, we need to have a path from a task to the nearest cluster centre, then from that cluster centre to the k-nearest models (note that when we refer to models/tasks, we're implicitly referring to the persistence diagram of their decision boundary). We can always find that route when fuzzy clustering, as the fractional membership values mean that we have information about the proximity of every model/task with every cluster centre. However, with hard clustering we cannot always find that route. Firstly, the hard labelling means that you lose a lot of information about the proximity of models/tasks to cluster centres. Therefore, in order to find a route, we need a every task to be assigned to a cluster centre that also has a model assigned to it. However, there are no guarantees that will happen. We show an example where no path exists in Figure 5.7.

To assess the ability of our model/task clustering, we performed the above experiment on three different datasets: MNIST (LeCun et al., 2010), FashionMNIST (Xiao et al., 2017), and Kuzushiji-MNIST (Clanuwat et al., 2018). MNIST (introduced in Section

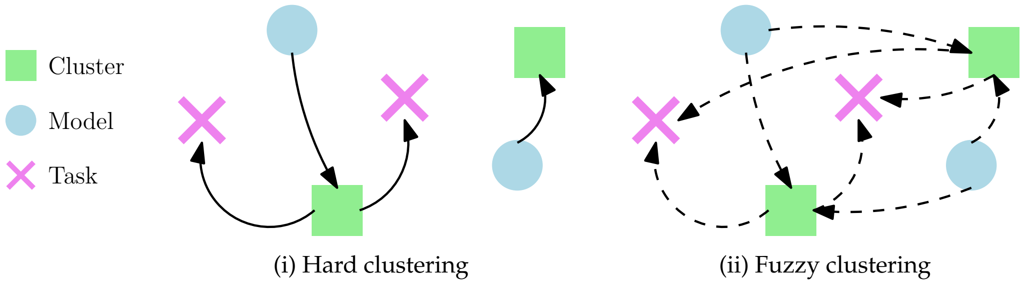
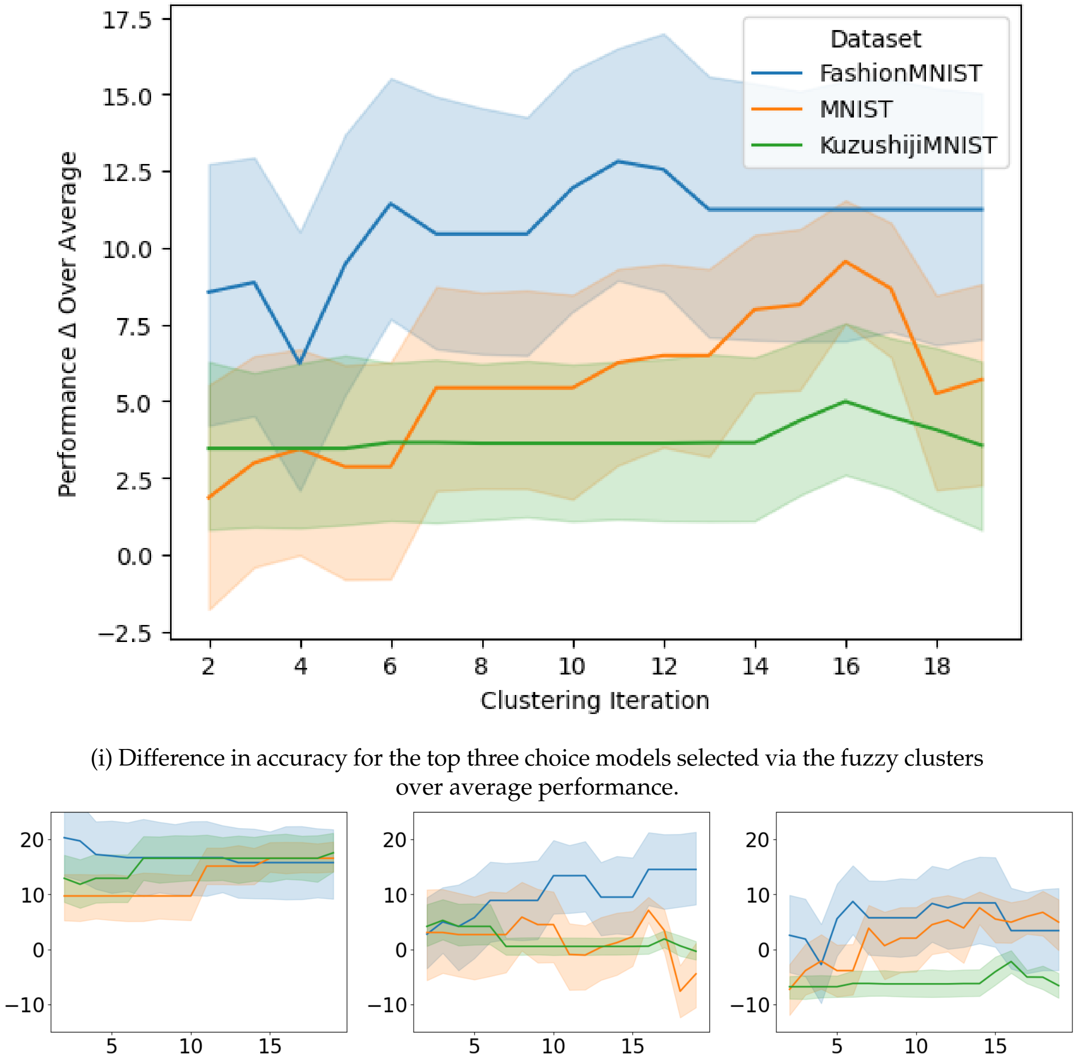


FIGURE 5.7: We show a planar visualisation of space when clustering the persistence diagrams of the decision boundaries of different pre-trained models and tasks. To match models to tasks there must be a path between every model and task. When hard clustering (i) there is no guarantee that there will be such a path; fuzzy clustering (ii) guarantees that there will be, and we can use the membership values to evaluate its quality.

4.2.1) is a dataset of images of handwritten images. FashionMNIST and Kuzushiji-MNIST are 'MNIST-like' datasets, as they are image datasets of the same size (both in terms of image size and number of samples). FashionMNIST shows images of common items of clothing, and the task is to classify the item of clothing. Kuzushiji-MNIST is a dataset of handwritten Japanese characters; the task is to classify the character. Our goal is to evaluate whether or not the clustering is capturing information about model performance on tasks, so as a baseline we use the average performance of all models on a fixed task, averaged over all tasks. Given a task, we find the its nearest cluster centre, then the three nearest models. We compute the difference in performance of the selected tasks over the average performance of the models on each task. We display the results in Figure 5.8i. We see a significant increase in performance, indicating that the topological fuzzy clustering has selected the model trained on the task, despite only having information about the topology of the decision boundary. We also show the performance of the 1st, 2nd, and 3rd choice models separately, in Figures 5.8ii, 5.8iii, and 5.8iv respectively. There is a marked decrease in performance as we descend in the order of our choices, although we generally still see a statistically significant increase in performance over average performance, indicating that the fuzzy clusters are capturing information about model generalisation to unseen tasks. The worst performance, particularly on the 3rd choice model, is from the Kuzushiji MNIST data. We hypothesise that this is because the dataset is not particularly amenable to generalisation, so a model not trained on the specific task will perform significantly worst than the highest achieving model (which we are selecting in the 1st and 2nd choices), even when selected using this technique. We repeat each experiment three times using sequential seeds, resulting in a total of 81 trained models.



(i) Difference in accuracy for the top three choice models selected via the fuzzy clusters over average performance.

(ii) 1st choice model

(iii) 2nd choice model

(iv) 3rd choice model

FIGURE 5.8: Difference in accuracy over average task performance when using the fuzzy clustered persistence diagrams of decision boundaries for model selection. Given a task, we find its nearest cluster centre, and use fuzzy membership values to select the nearest models. Figure i shows the performance of the top 3 models collectively, and ii, iii, and iv show the performance of the 1st, 2nd, and 3rd closest models separately.

5.5 FCM for Riemannian Manifolds

As part of a Geometric and Topological Machine Learning workshop at ICLR 2021 we integrated our implementation of FCM for persistence diagrams into two major code packages in the space. The first was Giotto-TDA (Tauzin et al., 2020), a well known package used for Topological Data Analysis. This made FCM available to people as part of a broader TDA ecosystem by using Giotto-TDA to compute persistence diagrams. We also adapted it for Geomstats (Miolane et al., 2020). Geomstats is a library that enables machine learning on manifolds. Since Fréchet means are defined on manifolds, we are still able to use the FCM algorithm we give in Algorithm 3. Furthermore, the convergence properties we show in Theorem 5.5 still hold, as the proof relies only on the definition of the weighted Fréchet mean; it does not matter in what setting that mean is taken. The only remaining question, therefore, is how to compute the weighted Fréchet mean on manifolds. Fortunately, Geomstats provides functions to do that, and we were able to fairly trivially implement the fuzzy c-means algorithm for manifolds. The implementation is available on their GitHub. See Figure B.1 for an example of the implementation working on the surface of a sphere.

5.6 Discussion

We have extended the fuzzy c-means clustering algorithm to persistence diagram space, proving the same convergence guarantee as the Euclidean case. Note that in reality practitioners may just embed into a feature vector and perform Euclidean clustering, bypassing the expensive process of computing Fréchet means and distances on persistence diagrams. Despite this the method remains useful, in particular if you wish to stay in persistence diagram space. This could be the case if, for example, you wish to backpropogate through the clustering, and cannot differentiate through your vectorisation strategy. The application of our algorithm matching pre-trained deep learning models to unseen datasets is particularly interesting. The link between the topology of decision boundaries and the generalisation ability of the model to unseen datasets remains relatively unexplored, and I would suggest this area as both important and interesting for future work.

Part III

Applications to Cyber Security

Chapter 6

A Review of TDA for Cyber Security

Cyber security is defined by the UK's National Cyber Security Centre as 'the way in which individuals and organisations reduce the risk of cyber attack[s]'[1], where cyber attacks are 'malicious attempts to damage, disrupt or gain unauthorised access to computer systems, networks or devices, via cyber means'[2]. The stakes are high, as the cost of cyber crime to the UK economy has been estimated as £27 billion a year (Detica, 2011), and cyber attacks are considered a Tier 1 national security risk in the UK National Security Strategy (HM Government, 2010). Cyber systems regularly comprise of intricate technical systems which record vast amounts of data, and so machine learning is increasingly being used to understand and exploit systems that would otherwise require a human operator to put in large amounts of time and have deep technical expertise. In addition, cyber security data intrinsically contains a large amount of structure, which is naturally very suited to topological tools. As such, we consider applications of persistence-based summaries to cyber security in this part (Part III) of the thesis.

We define cyber security data as referring to any dataset collected from operational systems or simulations of systems that pertains to the protection of these systems. In this thesis, this often refers to logs of actions taken on computers, but cyber data could equally relate to information about computer networks, blockchain transactions, or many other facets of computer security. In fact, cyber security data is a perfect candidate for topological tools. It often consists of higher-order interactions which we can study with persistence-based summaries and other methodologies from Topological Data Analysis. The global summaries offered by topology, which can be a weakness in some domains, is a strength in cyber security; Winding et al. (2006) say that "little information can be gained from analysing individual records, [as] often the presence of intrusion behaviour or other anomalous activity can only be detected by looking at the aggregate behaviour of related records". In Chapter 7 we are interested in representing computer

[1] https://www.ncsc.gov.uk/section/about-ncsc/what-is-cyber-security

[2] https://www.ncsc.gov.uk/information/ncsc-glossary

logs using their higher-order interactions. For now (in Chapter 6), we contextualise the applications of Topological Data Analysis to cyber security by reviewing the literature surrounding this area. We will see that TDA is applicable across many areas of cyber security; topological tools can be used on problems as widely arrayed as fingerprinting encrypted network traffic to identifying ransomware payments on blockchains.

Recall from Part I that two of the main tools in TDA are Mapper and persistent homology. Mapper is an algorithm that represents large high-dimensional datasets as a much smaller graph in a topology-preserving way, enabling its visualisation and analysis. Persistent homology produces a persistence diagram: a concise summary of the topology of a dataset which can be vectorised and used as input to other machine learning methods. Both strands of TDA have found applications in cyber security. Vast networks are common in this domain, and Mapper can reduce the size of large networks in a principled way, allowing human operators to better visualise and understand the datasets they are working with. Indeed, the primary application of Mapper that we see is to reduce large datasets of network traffic into more interpretable graphs. Clusters in the mapper graph are shown to represent one type of anomaly (Coudriau et al., 2016), and proximity to potential anomaly's within the mapper graph can alert analysts to traffic worth further investigation (Bihl et al., 2020). Persistence diagrams have also been used with success on cyber data. Persistence diagrams concisely summarise the global structure of higher-order interactions, so their application to cyber security seems very natural. This is demonstrated in the literature, with persistence diagrams being applied to successfully identify anomalies in network logs (Bruillard et al., 2016) and identify IoT devices from encrypted packet data (Collins et al., 2020).

The structure of this literature review is as follows. In Section 6.1 we summarise techniques to represent cyber data as either vectors or graphs. In Sections 6.2 and 6.3 we review the literature on Mapper and persistent homology for cyber respectively. In Appendix C we list the public cyber datasets found during this review.

Publications and contributions. This review was accepted as a paper and presented at the AAAI 2022 AI for Cyber Security workshop (Davies, 2022a). This chapter is entirely my own work.

6.1 Representing Cyber Data

Cyber data often comes in forms that is not immediately amenable to machine learning techniques. As such, choosing how to represent the data is an important step in any analysis workflow for cyber security. Much of the data naturally has a graph structure, as most things we consider live on a network. Some researchers choose to represent the data in a way that respects this graph structure, whereas others choose to represent it in a way that disregards it; both options have had success in the literature. In this section we cover cyber-specific representations of logs that disregard the graph structure, as well as general graph embedding techniques.

We often see Mapper applied to vectors that are built in a graph-agnostic way, even when the initial data has a graph structure. Mapper is capable of being applied directly to graphs - there is nothing in the algorithm that prevents us from using it on graph structured data, we just have to choose our partitioning of the data in a way that respects the graph. Hajij et al. (2018) do exactly that, but their code is not built into the main TDA libraries and is still in development, which might explain why we see a graph-agnostic representation in use with Mapper. In the case of persistent homology, again the majority of papers represent the data in a graph-agnostic way before computing the Vietoris-Rips complex. However, Collins et al. (2020) use the existing network structure and create their filtration based on inter-packet time, showing that such an approach is feasible.

6.1.1 Graph-agnostic representations

Winding et al. (2006) proposed vectorising network log data by summing numeric fields and counting enumerated fields. They suggested the following feature vectors:

- source IP, number of destination IP addresses;
- destination IP, number of failed access attempts;
- source IP, destination IP;
- destination perspective vector (consisting of destination IP, number of successful accesses, number of failed accesses, count of destination points, number of inbound bytes, number of outbound bytes).

Figure 6.1 gives an example of this vectorisation scheme for the source IP and number of destination IP addresses. This technique is regularly used to represent log data in anomaly detection research (Jakub and Branišová, 2015; Bruillard et al., 2016; Meng et al., 2019; Bihl et al., 2020). Based on its regular usage in the literature we use it as a baseline in Chapter 7, referring to it as the 'counts' baseline.

Bruillard et al. (2016) proposed mapping NetFlow data into feature vectors that summarise either (i) the cumulative number of packets sent/received by an IP of interest, (ii) the total number of packets sent/received by each IP, or (iii) the total number of packets sent/received by each IP to an IP of interest. These vectors are created over a window of a given length that advances over the whole dataset. The length of window and increment time are parameters that the authors vary and investigate the change in performance.

TABLE 6.1: An example of the feature vector creation process proposed by Winding et al. (2006) which is regularly used to represent logs in the literature. Categorical variables are counted and numerical variables are summed, so the collection of logs recording connections to each source IP address is mapped into the feature vector $(3, 1, 2, 21)$.

Log ID	Source IP	Connections
0	126.173.234.214	10
1	129.210.183.30	1
2	126.173.234.214	3
3	68.223.170.91	1
4	126.173.234.214	2
5	68.223.170.91	4

Log Reference	Feature Vector
126.173.234.214	3
129.210.183.30	1
68.223.170.91	2
Connections	21

6.1.2 Event data as graphs

Aksoy et al. (2019) consider a sliding window of 60 seconds that advances 20 seconds at a time over event logs. For the events in each window they construct a graph - the specific graphs they consider are listed below.

- *Authentication graphs* are unweighted graphs built from authentication data that has source user/destination user as edges.
- *Authentication failure graphs* are as above, but restricted to failed authentication events.
- *Process graphs* are built from start/stop records of processes. The graph consists of edges between computers and process names.
- *DNS graphs* are built from DNS lookup events with edges from source computer to resolved computer.
- *Flow graphs* are built from the records of every network flow event. The edges are between the source computer and the destination computer.

Collins et al. (2020) use encrypted packet data to build a filtration. They pick a device of interest and connect it to the last n devices that it exchanged packets with. They induce a filtration by the inter-packet arrival time, a metric that has previously shown to contain discriminative information when the traffic is encrypted. When all edges of a 2-simplex are present, they add the 2-simplex (Figure 6.8i). By doing so they avoid computing the expensive Vietoris-Rips complex, instead using the natural structure of the data and a relevant feature to induce a filtration.

6.1.3 Graph embeddings

Given the prevalence of graph-structured datasets in cyber, we offer a brief review of techniques to embed graphs into $\mathbb{R}^d$.

Early methods. Belkin and Niyogi (2001) showed that the eigenvalues of the graph Laplacian give an optimal embedding into $\mathbb{R}^D$, introducing this as a technique for finding low-dimensional embeddings over 20 years ago. Another early popular technique for semi-supervised learning on graphs was label propagation (Zhu and Ghahramani, 2002; Zhu et al., 2003). It spreads labels through a graph under the assumption that nodes that are in close proximity are likely to have the same label. Belkin et al. (2006) introduced manifold regularisation, an additional loss term that uses the graph Laplacian to regularise based on inferred structure of the data. Weston et al. (2012) extend this regularisation scheme to deep networks with their deep semi-supervised learning framework. These techniques can still provide useful baselines for graph learning problems.

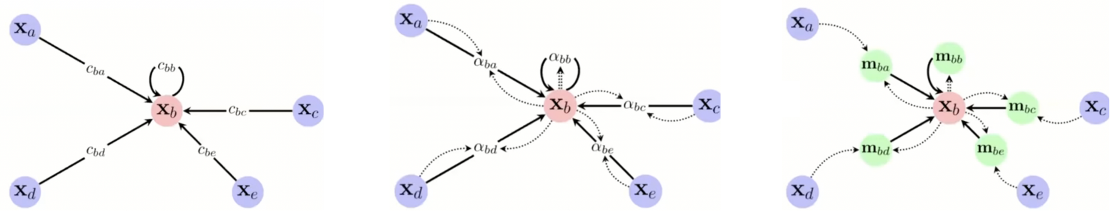
Skip-gram based models. Skip-gram was introduced by Mikolov et al. (2013a,b) to improve word embeddings for natural language processing (NLP). Given a word from a sentence, the idea of skip-gram is to maximise the average log probability that your model can predict the next word. Node2Vec (Grover and Leskovec, 2016) and LINE (Tang et al., 2015) adapted this for graph-structured data by maximising the probability that, given a node v, you can predict a node in the neighbourhood of v. Since the neighbourhoods over several hops can grow prohibitively large, they are sampled by random walks, with different adaptations using different sampling techniques.

Graph neural networks. Graph neural networks (GNNs) learn to aggregate feature vectors across nodes by combining features with those of their neighbours. Initially graph neural networks were classified as either spectral - aggregating neighbours in the spectral domain - or spatial - aggregating otherwise. However, true spectral filters require an expensive computation of the Fourier basis, so in general this was approximated, resulting in networks that were essentially spatial. As such, GNNs are now categorised as either convolutional, attentional, or message-passing (Figure 6.1). The difference lies in how information from the neighbourhood is aggregated. For convolutional networks, an updated feature vector h_i of a node x_i is decided by

$$h_i = \phi\left(x_i, \sum_{j \in N(x_i)} c_{i,j}\psi(x_j)\right),$$

where $N(x_i)$ is the neighbourhood of x_i. In this case, $c_{i,j}$ are fixed weights, normally determined by the graph's structure, ϕ is a nonlinearity and ψ is a learnt transformation. Popular examples of convolutional networks are Chebynet (Defferrard et al., 2016), GCNs (Kipf and Welling, 2017), and SGCs (Wu et al., 2019). For attentional networks the update step is

$$h_i = \phi\left(x_i, \sum_{j \in N(x_i)} a(x_i, x_j)\psi(x_j)\right),$$



(i) Convolutional (ii) Attentional (iii) Message passing

FIGURE 6.1: Graph neural networks can be sorted into the following taxonomy. Figure reproduced from Veličković (2021).

where a is a learnt function on x_i, x_j. This allows the model to determine how important an edge should be considered, rather than relying on the structure of the graph to inform that. Popular examples of attentional networks are MoNet (Monti et al., 2017), GAT (Veličković et al., 2018), and GaAN (Zhang et al., 2018). Interestingly, in the case where the graph is fully connected, this is the same as the celebrated transformer NLP model (Vaswani et al., 2017). Finally, message passing networks are the most general way to aggregate neighbours. Their update step is

$$h_i = \phi\left(x_i, \sum_{j \in N(x_i)} \psi(x_i, x_j)\right).$$

In this case, the coefficient is discarded, and ψ learns what features should be aggregated. Popular examples are interaction networks (Battaglia et al., 2016), MPNNs (Gilmer et al., 2017), and GraphNets (Battaglia et al., 2018).

Embedding large graphs. The techniques we've discussed so far cannot scale to large graphs. Computing eigenvalues for Laplacian-based models is prohibitively expensive even for smaller graphs; Skip-gram based models struggle to compute all the random walks without running out of memory for larger graphs; and many graph neural networks hold the entire adjacency matrix in memory, which is clearly unfeasible for large graphs. GraphSage (Hamilton et al., 2017) observes that to compute the gradient for a GNN with L layers at one node only requires neighbours up to L hops away. Furthermore, it samples neighbours rather than fully compute the neighbourhood, and mini-batches, i.e., only updates the gradients for a limited number of nodes at a time. By doing so it massively increases the size of graphs that can be processed with GNNs. However, there is redundant computation within this method as the same node can be sampled multiple times when neighbhourhoods overlap. ClusterGCN (Chiang et al., 2019) and GraphSAINT (Zeng et al., 2019) improve on GraphSage by proposing two techniques to remove these redundant computations.

Both Facebook and Twitter, as two organisations with very large graphs to process, have developed their own techniques to process very large graphs. Twitter proposed SImple scalable Graph Neural networks (SIGN), based on the idea that you don't necessarily require deep networks to learn from graphs; perhaps one-hop neighbourhoods are sufficient for learning. Under this assumption you can pre-compute the diffusion step, meaning you only have to multiply by weights in the training step. This gives a speed increase of up to two orders of magnitude over ClusterGCN and GraphSAINT, along with the ability for large scalability and parallelisation. However, the authors of this work suggest in a blog post[3] that SIGN is best used as as a fast benchmark for other models, so the lack of depth does ultimately seem to effect the performance of the network. Pytorch Big Graph (PBG) is Facebook's contribution (Lerer et al., 2019). It works on multi-relational graphs: graphs where you can have different types of edges. For example, in a Facebook graph an edge could represent a friendship, a like, a comment, or similar. PBG introduces several tricks to allow multi-relational graph neural networks to work on graphs with billions of nodes and trillions of edges.

6.2 Mapper for Cyber Security

In this section we review the literature on Mapper applications to cyber, splitting it by application domain. We find that it is most often used as an investigative and visualisation tool, although in some cases can provide quantitative information.

Anomaly detection. Bihl et al. (2020) apply Mapper to firewall logs from an enterprise-level organisation. The authors use a graph agnostic representation technique developed by Winding et al. (2006) that we cover in Section 6.1.1 (counting categorical fields and summing numerical fields). After representing the logs they compute the histogram matrix (HMAT) introduced by Gutierrez et al. (2018). This was intended to provide a way for analysts to visualise logs that may be anomalous over some block of time. Bihl et al. (2020) associate an HMAT matrix with each node in the mapper graph, then anomalous nodes in the Mapper graph flag to analysts that adjacent nodes may also be worth further investigation (Figure 6.2). However, even within the paper this claim goes relatively untested, and the tool is framed more as an exploratory technique than something that can reliably find anomalous events.

Network telecopes. Ranges of IPs with no active domains can be used to analyse traffic not directed towards real hosts. Since devices listening to this traffic are not real, any traffic to them is likely to be malicious (Fachkha and Debbabi, 2015). In the literature such traffic is called internet background radiation (IBR) and the listening devices are referred to as network telescopes. Coudriau et al. (2016) uses Mapper on data from a darknet telescope to find scans and DDoS attacks, outperforming traditional clustering algorithms when attacks don't cover the full range of possible ports or IPs (Figure 6.3). Mapper is also able to distinguish types of attacks from complex data. Applying

[3] https://blog.twitter.com/engineering/en_us/topics/insights/2021/simple-scalable-graph-neural-networks

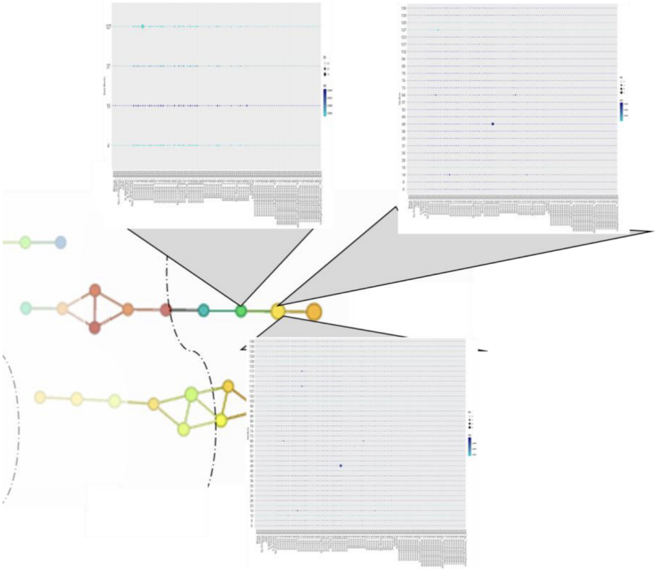
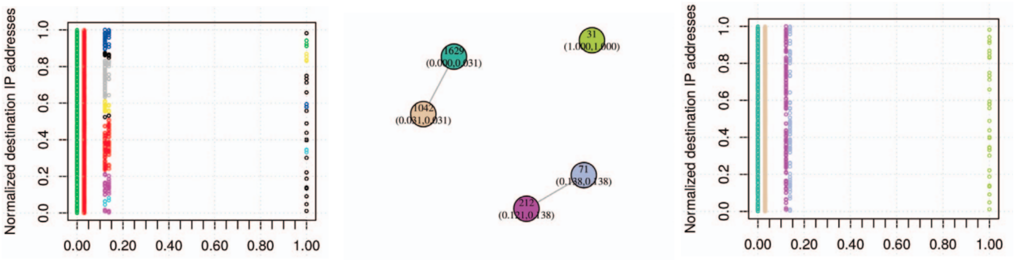


FIGURE 6.2: Mapper applied to HMAT matrices. Nodes in close proximity to those flagged as potentially anomalous could be worth further investigation. Figure reproduced from Bihl et al. (2020).



(i) The dataset clustered using DBSCAN.

(ii) The mapper graph of the dataset.

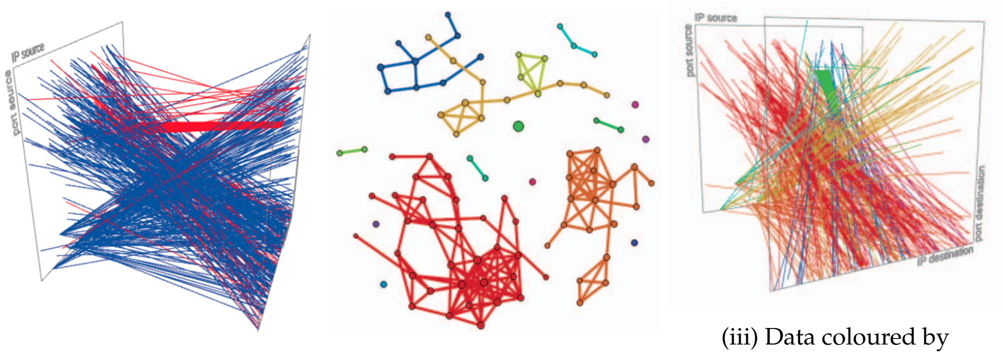
(iii) The dataset clustered with mapper.

FIGURE 6.3: Packets received by a network telescope plotted with respect to destination and source IP addresses. The vertical lines show one source IP scanning a range of destination IPs. In this example, DBSCAN cannot pick up the less dense scans, but Mapper does. Figure reproduced from Coudriau et al. (2016).

Mapper to source and destination IP/port data from a network telescope and manually examining the clusters showed that that each cluster represented a distinct attack, including the attempted exploitation of a known router vulnerability and scanning (Figure 6.4). Figure 6.5 shows visualisations of portscans and DDoS attacks that were captured by Mapper in the network telescope dataset. Of the many port scans identified by Mapper in Figure 6.5, only four were found by the ruleset of the intrusion detection system Suricata.

Narita (2021) also compute the Mapper graphs of network telescope data, but their analysis is less complete, instead focusing on describing the resultant graphs without attempting to link it to different attack types.

Tor traffic detection. If encrypted Tor traffic (which is often used for illegal activities

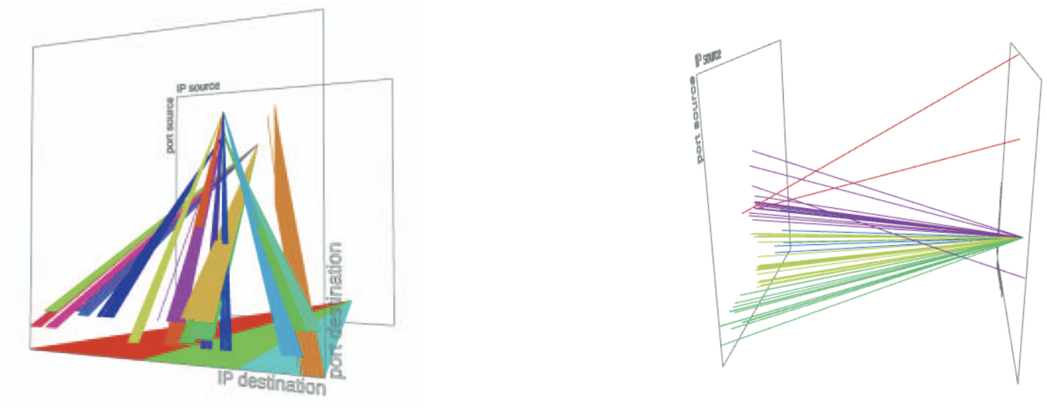


(i) Original data.

(ii) Mapper graph.

(iii) Data coloured by cluster.

FIGURE 6.4: The data collected by a network telescope is shown in (a). In (b) the data is clustered using Mapper. Manual examination of the clusters showed that the large green dot is attempting to exploit a known router vulnerability, the red component is trying to access Telnet or SSH, the orange component is a sparse port scan from a single address, and the yellow component is a randomised scan and some noise. In (c), the initial data is coloured according to the Mapper clusters. We can see that the clusters are disorderly within the data, despite Mapper pulling out distinct attack types. Figure reproduced from Coudriau et al. (2016).



(i) Portscans

(ii) DDoS

FIGURE 6.5: Portscans and DDoS attacks captured by Mapper in network telescope data. Figure reproduced from Coudriau et al. (2016).

online) is detected as it passes through a network then connections can be immediately shut down and further investigated. Veen (2018) investigates supervised and unsupervised methods for Tor traffic detection, using XGBoost with great success to detect Tor traffic. They investigate Mapper as an unsupervised technique for visualising and predicting Tor traffic within network traffic. There is some success, with qualitative observations that Tor traffic tends to be on the extremes of the graph (Figure 6.6).

Ransomware payments. Akcora et al. (2019) build a Mapper graph from Bitcoin transaction data. If a certain proportion of addresses that are within a node of the Mapper graph are known to receive ransomware payments then the remaining addresses have their risk scores increased. Repeating over the entire map and thresholding the acceptable

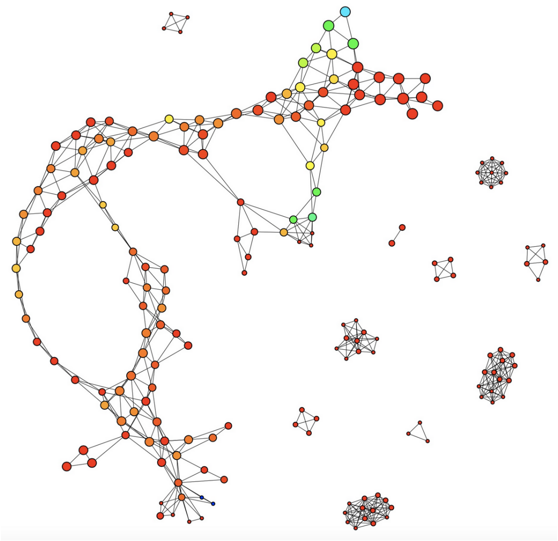


FIGURE 6.6: Mapper graph of packet data within a network. Extremities of the graph tend to be Tor traffic. Figure reproduced from Veen (2018).

risk gives a list of suspicious Bitcoin addresses. This approach outperforms DBScan and XGBoost on feature vectors built from the data.

Attack graphs. Sophisticated attacks on networks can consist of multiple steps, many of which are not problematic when viewed individually. However, as a whole they can constitute a powerful attack. This sequence of actions as part of a sophisticated attack naturally admits a graph structure, referred to as an attack graph. This graph is susceptible to analysis with topological techniques, which was the focus of work by Navarro et al. (2018). They apply mapper to event networks in the hope that the reduced graph is able to be analysed by humans who can find repeated structures across attacks.

6.3 Persistence-Based Summaries for Cyber Security

We have seen that persistence diagrams capture specific topological information about data which can be vectorised and fed into machine learning methods for further analysis and prediction. We see in this section that this can be used for detection and classification in far more quantitative ways than Mapper.

Anomaly detection. Bruillard et al. (2016) uses the distance between persistence diagrams computed from NetFlow data to detect anomalies. Vectorising the NetFlow data via a sliding window over the data counting packets (as described in Section 6.1.1), the authors compute a baseline persistence diagram from a collection of feature vectors B. For every other vector x, they compute the Wasserstein distance between the persistence diagrams of B and $B \cup \{x\}$. The larger the distance, the more topologically different

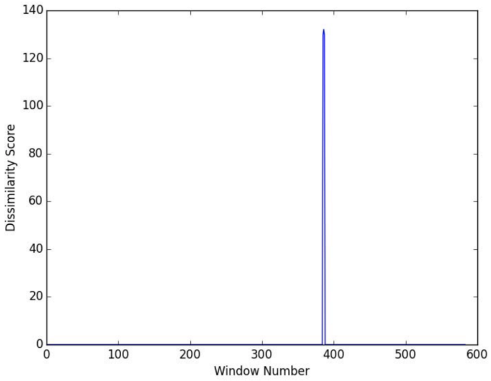


FIGURE 6.7: The spike in topological dissimilarity indicates a predicted anomaly. In fact, this was a port scan. Figure reproduced from Bruillard et al. (2016).

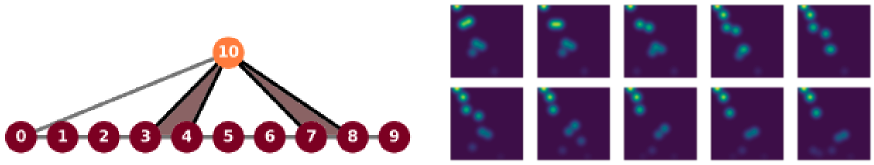
the points are, and they found that spikes in topological dissimilarity are indicative of anomalies (Figure 6.7).

Activity prediction. Gabdrakhmanova (2018) attempt to predict the activity of network switches based on previous activity. They compute persistence diagrams for each two hour period and summarise them as the approximate Euler characteristic: alternating sums of the total persistence of the diagrams over increasing dimensions. It is not specified exactly how activity predictions are made with that, but it appears they input them into a neural network to predict the next Betti number. It is unclear from the paper whether or not they had any success doing so, as no evaluation of the predictions is present.

Gabdrakhmanova (2018) summarises persistence diagrams as the Euler characteristic. Specifically, this is the alternating sum of the Betti numbers, which are approximated by the persistence of points in persistence diagrams, increasing by dimension. Given the levels of recent activity at a network switch (say, at an ISP), they wish to predict the future levels of activity. They have bits/minute data for 7 days. They split the data into two hour chunks, computing a persistence diagram for each two hour period. From that, they estimate the Euler characteristic of that period, as described above.

IoT device fingerprinting. Postol et al. (2019) show that TDA works particularly well for classifying incomplete and noisy data from internet of things (IoT) devices on networks. They compute the persistence diagrams of vectorised IoT network traffic, and use feature selection/logistic regression on functional embeddings of the diagrams to classify types of IoT devices (differentiating between cameras, sensors, and multipurpose devices like tablets). With data collected over a period of months this worked very well, but over periods of days it did not. This implies that the pattern of life for different IoT devices can be better understood with these methods over a longer period of time.

Collins et al. (2020) also study IoT data, only they use the natural network structure of the graph, introducing a filtration by the inter-packet arrival time (IAT), a feature which



(i) A simplicial complex built from packet data.

(ii) Examples of 1-persistence images of packet data using IAT.

FIGURE 6.8: The 1-persistent homology of packet data can be used to accurately fingerprint internet of things (IoT) devices on networks, even if the traffic is encrypted. Figure reproduced from Collins et al. (2020).

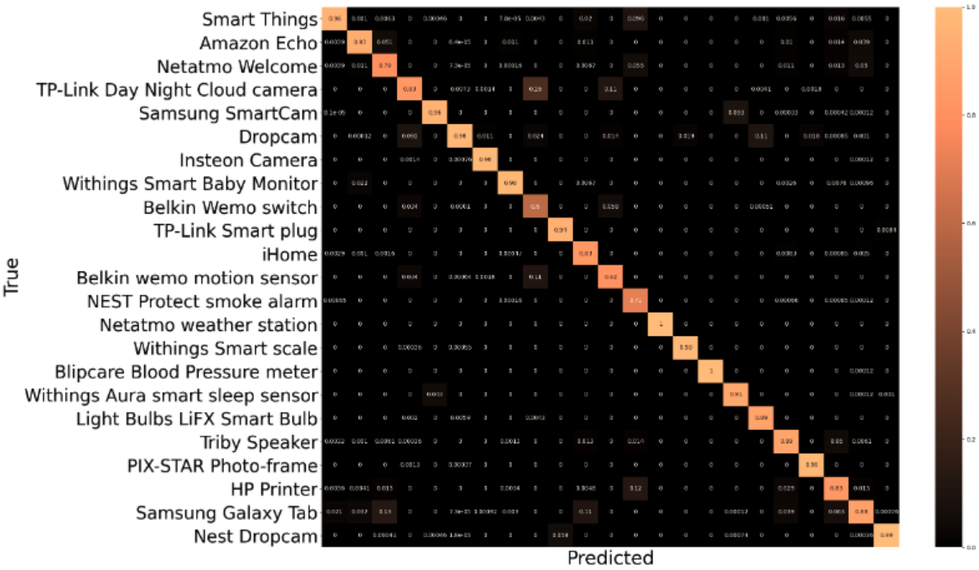


FIGURE 6.9: The normalised confusion matrix of IoT device predictions from the 1-persistent homology of packet data. Figure reproduced from (Collins et al., 2020).

has been previously shown to contain valuable information for traffic classification and anomaly detection (Uluagac et al., 2013). By using the intrinsic structure of the graph, they get rid of the need to compute the often prohibitively expensive Rips complex (Figure 6.8i). Next they compute the 1-homology (holes) of this filtration, in comparison to previous work on network data that has computed the 0-homology (connected components). Finally they vectorise the persistence diagrams by mapping them to persistence images (Figure 6.8ii), and use those to train a CNN. With these higher-dimensional topological features, they are able to identify IoT devices with high accuracy, recall, and precision (the confusion matrix is shown in Figure 6.9).

(Hyper)graph metrics. We end this review by briefly mentioning some relevant work using graph metrics for cyber security data. One way to compare graphs is the relative Hausdorff distance, which offers a fast but nuanced way to compare two graphs based on their complementary cumulative degree histograms (CCDH). The CCDH of a graph G is $(N(k))_{k=1}^{\infty}$, where $N(k)$ denotes the number of vertices in G with degree at least k. Comparing these gives us the relative Hausdorff distance, which Aksoy et al. (2019) use to detect anomalies in graph sequences built from the Los Alamos dataset (we discuss open source datasets in Section C). They build authentication graphs, authentication failure graphs, process graphs, DNS graphs, and flow graphs. Over longer time windows, they find that a spike in pairwise RH distance is indicative of a red-team event. Joslyn et al. (2019) investigate the use of hypergraphs to represent DNS data.

List of datasets. We have collated a list of relevant datasets for practitioners interested in working on cyber security data. This is included in Appendix C.

Chapter 7

Structural Representations of Host-Based Logs

Detecting anomalous behaviour in computer logs is a crucial part of any organisation's cyber security operations. Undetected incursions can result in privacy breaches for the users of the system, and both financial and reputational damage to the organisation itself. In order to understand what is happening on the computer networks they are responsible for, cyber security professionals collect logs: records of what is happening on their computers (hosts) and networks. Ideally, these logs will be analysed in close to real-time by an Intrusion Detection System (IDS), which in turn is monitored by human operators in a Security Operations Centre (SOC). The reality is, however, that most organisations will not have the budget or motivation to properly set up a SOC. Even in a properly resourced SOC, it can remain difficult to successfully detect security incidents.

Given the high stakes and vast amounts of data, this is a natural domain in which to apply machine learning. As such, an increasing amount of research effort is being put into *data-driven cyber security*, which aims to exploit the large amount of data typically collected in the cyber security domain. Using these logs in machine learning requires representing them as a format which is amenable to machine learning methods; typically this means mapping them into real-valued vectors. Although these logs are intrinsically extremely structured, consisting entirely of pairwise and higher-order interactions, standard methods for representing logs entirely disregard this structure. Recall from Section 6.1.1 the counts vector, which vectorises logs by counting categorical variables and summing numerical variables. This is often used to represent logs in the literature, including as input to complicated downstream models, but entirely disregards their underlying structure. Methods that do account for the structure typically build graphs from logs. This disregards (i) higher-order interactions in the data, and (ii) the time of the logs. The time of the logs in particular is considered crucial in determining whether behaviour is malicious: there is a big difference between an email attachment launching unknown software (which is a typical attack vector), rather than the other way around.

In this chapter we propose a methodology to directly represent computer logs into a filtration of simplicial complexes. This allows us to (i) account for higher-order interactions, whilst also (ii) using the time of the logs as a filtration parameter. We evaluate the peristent homology of these filtrations as a feature vector for classifying malicious behaviour in computer logs. We also evaluate the graph and hypergraph Laplacian as topological baselines, and the counts vector as a non-topological baseline.

Publications and contributions. The work in this chapter was accepted as a paper and presented at the SIAM Data Mining workshop on Applications of Topological Data Analysis (Davies, 2022b; Darling et al., 2022). The application of TDA to cyber security was proposed by collaborators from the Alan Turing Institute, who also suggested the baseline and provided the data. I developed the representation technique, implemented it, ran the experiments, and performed the analysis.

7.1 Background

7.1.1 Host-based logging

Security logging is the practice of recording activity on a computer network. Logs are typically processed by automatic Intrusion Detection Systems (IDS), which are in turn monitored within Security Operations Centres (SOCs), providing an invaluable early warning to cyber security professionals that threat actors are operating within their computer networks. These logs are typically either network logs or host-based logs (Khraisat et al., 2019). Network logs are extracted from a network by packet capture or NetFlow (Hofstede et al., 2014), typically by a network tap or compatible router situated at network choke points (Hay et al., 2008). Although these taps should be set up so that any remote infraction on the network necessarily must be logged, the large amount of data passing through a typical corporate network means that a network-based intrusion detection system (NIDS) will struggle to evaluate all of the data it is exposed to (Bhuyan et al., 2014).

In comparison, host-based logs are collected by a service running locally on each computer (i.e., each host). They are typically collected from hosts performing a variety of tasks; to give just a few examples, host-based logs may be collected from end-user machines, servers, and databases (Khraisat et al., 2019). They log detailed information on the processes running on the host, so they are able to detect attacks via vectors that avoid network activity, one example of which is an insider threat (Creech and Hu, 2014). Our experiments focus on host-based logs captured by Windows System Monitor[1] (Sysmon),

[1] https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon

TABLE 7.1: An example of a Sysmon log.

UtcTime	`4/9/2015 07:03:12.343 PM`
ProcessGuid	`{7acfffcf-fbf0-5533-0000-00104820887f}`
ProcessId	`18704`
Image	`C:\Windows\System32\SearchFilterHost.exe`
CommandLine	`⋆C:\Windows\system32\SearchFilterHost.exe⋆ 069...`
CurrentDirectory	`C:\Windows\system32`
User	`NT AUTHORITY\SYSTEM`
OriginalFileName	`Cmd.exe`
CurrentDirectory	`C:\Windows\system32`
LogonGuid	`{7acfffcf-3b9b-5524-0000-0020e7030000}`
LogonId	`0x3E7`
TerminalSessionId	`0`
IntegrityLevel	`Medium`
Hashes	`SHA1=[hash],MD5=[hash],IMPHASH=[hash]`
ParentProcessGuid	`{7acfffcf-4ed3-5527-0000-0010e196db1c}`
ParentProcessId	`5756`
ParentImage	`C:\Windows\System32\SearchIndexer.exe`
ParentCommandLine	`C:\Windows\System32\SearchIndexer.exe /Embedding`

Windows Sigcheck[2], and Windows Event Forwarding[3] (WEF), set-up and recorded using the open-source Logging Made Easy project (LME). Sysmon is a persistent service that allows you to record:

- process creation events (including the full command line for current and parent processes);
- the hash of process image files;
- logon sessions;
- loading of drivers and DLLs;
- access to files (including tracking modified file creation times); and
- network connections (including source and destination IP addresses and ports).

Sigcheck shows file versioning, timestamps, and digital signature details. WEF collects these logs and forwards them on to downstream collectors, where they can be stored and processed by other tooling which is part of the LME setup. An example host-based log collected by Sysmon is shown in Table 7.1.

[2] https://learn.microsoft.com/en-us/sysinternals/downloads/sigcheck

[3] https://learn.microsoft.com/en-us/windows/security/threat-protection/use-windows-event-forwarding-to-assist-in-intrusion-detection

7.1.2 Hypergraphs

For the fist time in this thesis we consider the hypergraph as a structure to represent higher-order interaction data. A hypergraph is a generalisation of a graph $G = (V, E)$ in which each *hyperedge* $e \in E$ is a subset of V; that is, each edge can now contain arbitrarily many nodes, rather than just two. They also generalise simplicial complexes, removing the constraint that for each simplex σ in a simplicial complex K, each face $\sigma' \subseteq \sigma$ of σ must also be in K. This generalisation is why we are interested in considering hypergraphs: just because more than two objects interact, does not mean that each subset of these objects interact. Therefore hypergraphs can offer a more faithful representation of the underlying data. We consider them for this application to investigate whether this increases the performance of the representations. We exemplify the distinction between graphs, simplicial complexes, and hypergraphs in Figure 7.1.

We represent hypergraphs with the hypergraph Laplacian, which is typically defined in two ways within the literature. The first definition considers subsets of nodes of the same cardinality: this definition is akin to the combinatorial Laplacian. The other definition considers all hyperedges as one set, corresponding more closely to an extension of the graph Laplacian.

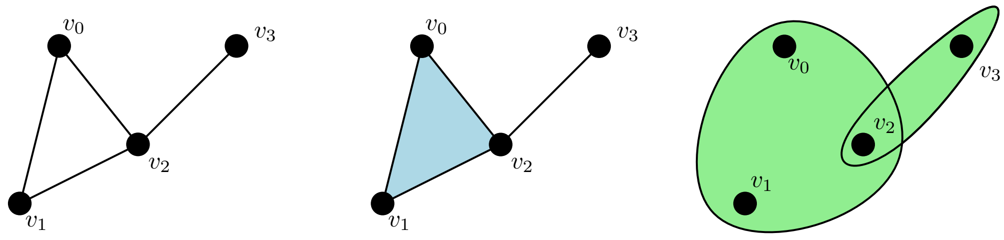
Chung (1992b) first studied the hypergraph Laplacian, and falls into the first camp: she considered the subsets of hyperedges containing k vertices, which we denote E_k. This restriction is akin to considering k-simplices, although note the off-by-one error: a k-simplex consists of $k + 1$ nodes, whereas in the notation of Chung (1992b), a k-edge consists of k nodes. She defines two $(k - 1)$-edges as adjacent if they are both part of the same k-edge, and using this defines analogues of the degree and adjacency matrix. In particular, she defines the degree of a $(k - 1)$-edge $e \in E_{k-1}$ as the number of k-edges it is contained within, i.e.,

$$d(e) = |\{f \in E_k : e \subset f\}|\,.$$

The diagonal matrix D is an $|E_{k-1}| \times |E_{k-1}|$ matrix with diagonal entries $D(e, e) = d(e)$ for $e \in E_{k-1}$ and $D(e, f) = 0$ for $e, f \in E_{k-1}, e \neq f$. The adjacency matrix A is a binary matrix of size $|E_{k-1}| \times |E_{k-1}|$ defined as $A(e, f) = 1$ when $e, f \in E_{n-1}$ and $e \cup f \in E_k$ (i.e., when the two $k - 1$ edges are both in a k edge), and 0 otherwise. She also defines the complete graph matrix K as an $|E_{k-1}| \times |E_{k-1}|$ binary matrix, which is given by $K(e, f) = 1$ if $e, f \in E_{n-1}$ and $e \cap f = v$ for a single vertex $v \in V$, i.e., if two edges differ only by one node, and 0 otherwise. The k-Laplacian for $k \geq 3$ is then given by

$$\mathcal{L}(H) = D - A + \rho(K + (k - 1)I,$$

where I is the $(k - 1) \times (k - 1)$ identity matrix, and ρ is the density of G: the average node degree d over the number of nodes $|V|$. She develops a homology theory for hypergaphs and investigates the spectrum of this Laplacian.



(i) The graph representation.

(ii) The simplicial complex representation.

(iii) The hypergraph representation.

FIGURE 7.1: Consider the higher-order interactions (v_0, v_1, v_2) and (v_2, v_3). Since a simplicial complex requires closure under taking subsets of interactions, the only totally accurate representation of those two interactions is with a hypergraph.

The hypergraph Laplacian was also studied by Jost and Mulas (2019), who generalise the Laplace operator for graphs by defining a boundary operator on functions defined on the vertices of the hypergraph. By considering the Laplacian as a composition of this boundary operator with its adjoint (cf. Equation 2.4.3) they consider hyperedges collectively, rather than separating them into k-edges.

However, we focus on the the definition given by Hayashi et al. (2020), mainly because they provide an implementation of their Laplacian in the package HyperNetX[4]. They define a random walk on a hypergraph $H = (V, E)$ as follow. Let $\omega : E \to \mathbb{R}^+$ be any function which assigns positive weights to the edges of a hypergraph. Let the current node of our random walk be $X_t = v_t \in V$. To move to the next vertex we (i) select a hyperedge $e \ni v_t$ with probability proportional to $\omega(e)$, then (ii) select a vertex $v \in e$ uniformly at random and set $X_{t+1} = v$. The function ω gives rise to a transition probability matrix of the random walk $P \in [0,1]^{|V| \times |V|}$, defined so that $P(v, v')$ is the probability of transitioning to node v' if you are currently at node v. The hypergraph random walk Laplacian is then given by $I - P$. More generally, the eigenvalues of the graph Laplacian are related to those of the probability transition matrix (Hayashi et al., 2020), and random walks on graphs are deeply tied to the study of their Laplacians (Chung, 1992a). Hayashi et al. (2020) demonstrate that the previous critique that some hypergraph Laplacians can equally be represented as graph Laplacians (Agarwal et al., 2006) does not hold in their case.

7.1.3 Previous work

In Chapter 6 we reviewed applications of TDA to Cyber Security, covering a broad array of topics which come under the umbrella of cyber security. We briefly recap the work which is most relevant to this section, i.e., that which is about mapping log data into feature vectors. We also introduce some additional non-topological techniques for representing logs.

[4] https://pnnl.github.io/HyperNetX/build/index.html

The counts vector. In Section 6.1.1 we introduced the counts vector (Winding et al., 2006). As shown in Table 6.1, for categorical variables it counts the number of times each category occurs, and sums numerical variables. Despite its simplicity, it has been used to great effect in a number of applications (Jakub and Branišová, 2015; Bruillard et al., 2016; Meng et al., 2019; Bihl et al., 2020). It will be a baseline for this application.

Representing logs as graphs. Log data is highly structured, and as such it makes sense to represent it in graphs. Aksoy et al. (2019) built a variety of graphs from logs, using attributes like users, processes, or IP addresses as nodes, then adding edges based on interactions between the identifiers in the logs. We gave more details in Section 6.1.2. Han et al. (2020) build provenance graphs from log data that enables them to attribute malicious processes to the root cause within the machine in real-time, in particular targeting the activities of Advanced Persistent Threats (APTs). Hassan et al. (2018) also built provenance graphs, summarising them using finite automata to reduce network and storage overheads when processing large amounts of data. Bowman et al. (2020) built authentication graphs from log data, which they processed with graph neural networks to detect lateral movement by APTs.

Representing logs as simplicial complexes. There is a limited amount of work in which simplicial complexes are considered for representing logs. Bruillard et al. (2016) represent logs using the counts vectorisation, then compute the Vietoris-Rips complexes of those representations to apply persistent homology. Collins et al. (2020) represent intercepted packet data from IoT devices into simplicial complexes, making each node an intercepted packet, connecting them to a central node representing the device of interest, and inducing a filtration by the inter-packet arrival time (IAT): the time between each packet is sent. This is shown in Figure 6.8.

Deep learning. Despite the increasing complexity of deep models used for behaviour classification, a lot of models still use simple representation strategies. According to a recent evaluation of models by Le and Zhang (2022) DeepLog (Du et al., 2017) is the state-of-the-art for anomaly detection in logs, but it uses a very simple representation which assigns each log to an integer based on its event type. Le and Zhang (2022) also evaluate LogAnomaly (Meng et al., 2019), which uses the counts vector to represent the log data. Another approach is to semantically represent logs using language models like FastText or Word2Vec. Although this has been done recently, for example in Lu et al. (2018) and Zhang et al. (2019), these models were evaluated as worst than those using simple counts representations by Le and Zhang (2022). It is clear that even the best performing deep models still rely on simple log representations. Our method focuses on improving the quality of those representations.

TABLE 7.2: A simplified example of host-based logs, which we use to demonstrate our methodology. Each timestep represents one log entry.

Timestep	0	1	2
Process ID	18704	19043	18704
Event Type	Process Creation	Network Event	File Access
New Process ID	19043	–	–
Source IP	–	192.168.1.12	–
Destination IP	–	192.168.1.16	–
Path	–	–	`C:\passwords.txt`

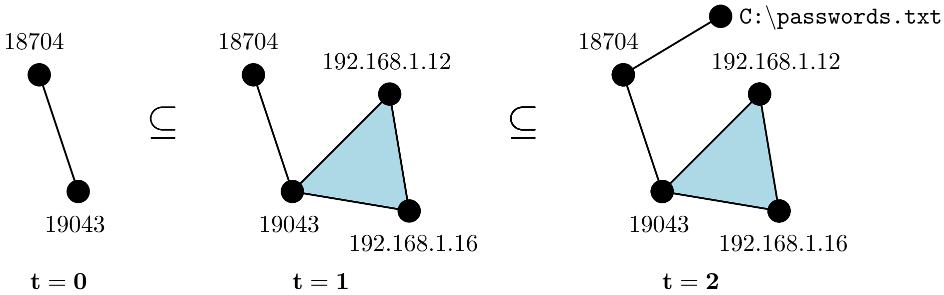


FIGURE 7.2: An example of how we map logs into a filtration of simplicial complexes, using the example logs in Table 7.2. Each unique identifier in the logs becomes a 0-simplex (node), and higher order simplices correspond to log entries containing multiple identifiers. They are added to the filtration at their timestamp.

7.2 Methodology

7.2.1 Higher-order temporal representations

Previous work on representing the structure of logs has focused either on graphs or simplicial complexes. Using graphs discards higher-order interactions and the timestamp of logs, both of which could provide crucial discriminative information. Similarly, the previous representation into simplicial complexes by Bruillard et al. (2016) computed the Vietoris-Rips complex of the counts vectors, which does not utilise the natural structure and temporal aspect of logs data. Collins et al. (2020) constructs a simplicial complex by using packet data between IoT devices as nodes, adding 1-simplices to the filtration based on the inter-packet arrival time, and adding higher-order simplices when each of the requisite 1-simplices is present. In comparison, we investigate the application of Topological Data Analysis to a tailored filtration which utilises both the higher-order interactions and the temporal nature of a filtration to naturally represent the structure inherent to logs data.

In particular, we construct a filtration of simplicial complexes as follows. The 0-simplices are the unique identifiers within the host-based logs. Table 7.2 displays some simple example logs that we will use to help illustrate our method: for example, the unique identifiers in Table 7.2 are the process IDs 18704 and 19043, the IP addresses 192.168.1.12

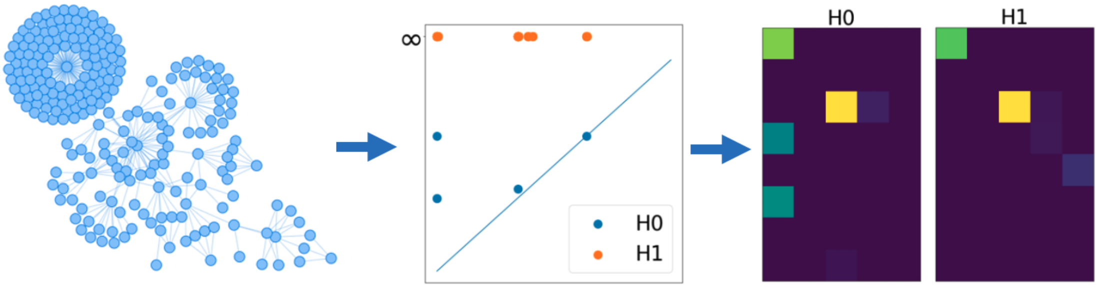


FIGURE 7.3: Having mapped a collection of logs into a filtration, we compute the persistence images. We show the end of the filtration constructed from the logs of trojan horse malware, represented as a graph. We compute the persistence diagram from the full filtration of simplicial complexes (not shown), then the persistence images. Note the big cluster of nodes: this is the trojan horse creating files as it downloads additional malware, a structural characteristic than can be captured by our representations.

and 192.168.1.16, and the file path `C:\passwords.txt`. They are our 0-simplices. *Each higher-order simplex is an individual log entry which contains multiple unique identifiers*. For example, the log entry at $t = 0$ in Table 7.1 contains two process IDs, so corresponds to a 1-simplex $[18704, 19043]$, and the log entry at $t = 1$ contains three unique identifers, so corresponds to a 2-simplex generated by the process ID 19043 and the two IP addresses 192.168.1.12 and 192.168.1.16. This is demonstrated in Figure 7.2. The time of the log entry is the time the corresponding simplex enters the filtration - any faces of the simplex that are not already in the complex are also added at this time. In this way each simplex represents a higher-order interaction between the unique identifiers within the host-based logs, and the filtration parameter represents the time at which this event happens. This construction faithfully represents the structure of the logs.

7.2.2 Representations

Given our higher-order temporal representation that naturally represents the structure inherent to host-based logs, we give an overview of the techniques we are interested in evaluating on this representation. The primary technique we were interested in investigating using this representation was persistent homology.

Persistence diagrams. Given that we have a filtration of simplicial complexes, we are able to follow a fairly standard topological machine learning pipeline. In particular, we computed the persistence diagrams of the filtrations, then vectorised the persistence diagrams as persistence images[5], techniques which we describe in Sections 2.2.4 and 3.2.2 respectively. This process is shown in Figure 7.3. We investigated the effect of induced 2-simplices and pixel size of the persistence images on the performance of our downstream classification task; theses details are available in Appendix B.3.

[5] Computed using Dionysus and Persim.

Hypergraph Laplacian. The construction of a hypergraph $H = (V, E)$ from logs that we consider is extremely similar to the filtration of simplicial complexes: each vertex $v \in V$ is a unique identifier within the logs, and each edge $e \in E$ contains the nodes corresponding to the identifiers in one log entry. The differences are twofold: firstly, we just construct one hypergraph, rather than a filtration, as there is no persistent hypergraph Laplacian. Secondly, hypergraphs do not require that we add each subset of a higher-order interaction: we can simply add one hyperedge for each log entry. We compute the hypergraph Laplacian using the HyperNetX package discussed in Section 7.1.2, then compute its spectrum. As the size of the spectrum depends on the size of the hypergraph, which will vary over different collections of logs, we featurise the spectrum of the hypergraph Laplacian by sorting the spectrum, then truncating or zero padding the resultant vector so that each feature vector is the same length.

Graph Laplacian. We also investigate the graph Laplacian as an representation. We were not yet familiar with the persistent Laplacian (Mémoli et al., 2022), so we also construct a graph using all logs for the graph Laplacian, rather than consider a filtration of graphs. We construct our graph $G = (V, E)$ by adding every unique identifier within the logs as a vertex $v \in V$, and adding every pairwise interaction within an individual logs as an edge E. This is equivalent to taking the last complex from our filtration of simplicial complexes and restricting to the 1-skeleton. Our process from now is identical to the hypergraph Laplacian: we compute the graph Laplacian of each constructed graph, sort the spectrum, then truncate or zero pad in order to create a feature vector.

Counts vector. The final baseline that we evaluate is the counts vector, first introduced in Section 6.1.1 and revisited in Section 7.1.3. This vector does not utilise a mapping into a simplicial complex or similar, instead recall that, given a collection of logs, it counts the unique entries in categorical variables, and sums numerical variables. Despite its simplicity, it has been regularly utilised to great effect in the time since it was introduced.

7.3 Experiments

7.3.1 Dataset

We evaluate our representation technique on two synthetic host-based datasets. All logs are generated by Sysmon, set up and recorded using the open-source Logging Made Easy (LME) project. One data point is a collection of logs from one *run*, in which a computer running LME is turned on, malicious or benign activity takes place on the host, then the machine is switched off. The collection of logs recorded during this run gets labelled as benign or malicious depending on the activity. The first dataset is the 'malware' dataset; in this dataset the malicious activity is malware being executed on the host, and the benign activity is cleanware (benign software) being executed on the host.

TABLE 7.3: The number of examples of benign and malign runs in each dataset.

	Malware Dataset	Adversary Dataset
Benign	768	800
Malign	841	800
Total	1609	1600

TABLE 7.4: The event types we include in the computation of feature vectors at low and high acuity.

Low Acuity	High Acuity
Process creation	Process creation
Network event	Network event
	Process termination
	File creation

The task is to classify each run (i.e., each collection of logs) based on whether malware or cleanware was executed on the machine. The second dataset is the 'adversary' dataset. The malign activity is simulated with Mitre's CALDERA adversary emulation platform[6]. This is an open-source project that runs an agent on the host, emulating the activities of an adversary active within the host. This can be more sophisticated than simply running malware already present on the machine (as in the malware dataset), including the use of fileless attacks: malicious activity using only software already present on the host. The benign dataset is the emulated activity of a benign end-user. The number of benign and malign runs in each dataset is shown in Table 7.3. Note that each run contains many thousands of individual log entries, taking place over an average time of approximately three minutes.

7.3.2 Details

We represent the logs using the methodology described in Section 7.2. In particular, we compute the counts vector, the persistent homology and the spectrum of the graph Laplacian and hypergraph Laplacian for each run, at two different levels of acuity. The data acuity refers to how many types of logs we consider: in a low acuity setting we restrict to fewer event types, to investigate the effect on the performance of the classifier. We specify the event type of the logs considered at each level of acuity in Table 7.4. We use a 10-fold cross validated random forest[7] as our classifier. The simple nature of the classifier allows us to focus on the efficacy of the feature representations. Further details of the classifier are available in Appendix B.3.

[6] `https://caldera.mitre.org/`

[7] Using sci-kit learn (Pedregosa et al., 2011).

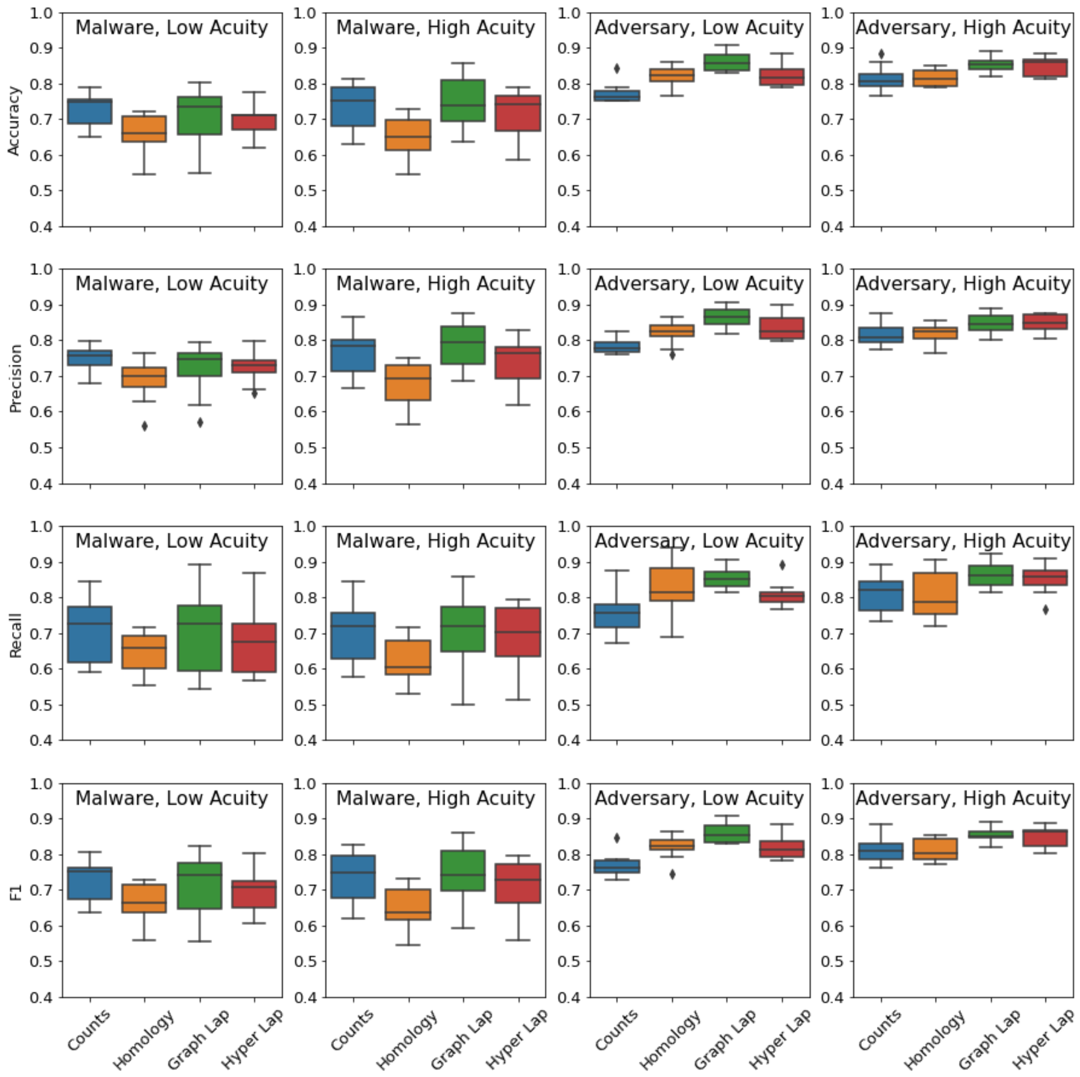


FIGURE 7.4: The results for each representation across the malware and adversary datasets, low and high data acuities, and with accuracy, precision, recall, and F1 metrics. Although the performance is often close, on balance we find that across metrics and settings we can see that structural representations generally rival or outperform the counts baseline, with the graph Laplacian our best method overall.

7.3.3 Results

Figure 7.4 shows our results in detail. We evaluate the accuracy, precision, recall, and F1 for each experiment across 10 folds. The variations in method, dataset scenario and data acuity give us several angles for analysis.

First we consider the effect data acuity has on the results. Somewhat predictably, higher data acuity generally comes with an increase in performance across metrics and scenarios. In the adversary dataset the counts baseline is particularly increased relative to the other representation strategies. On the other hand, the relatively stable performance of the structural representations suggests that they are resilient to lower data acuity scenarios.

Since increasing the data acuity increases the computational complexity, being able to have less degradation in performance relative to the counts vector when decreasing data acuity is valuable.

In the malware dataset, we find that the counts methodology outperforms persistent homology significantly, and is marginally better than the graph and hypergraph Laplacians. Although this initial result is dissapointing, as the primary technique we were aiming to evaluate was persistent homology, the performance of the graph Laplacian is interesting. In the adversary setting, the counts vector is generally worst than persistent homology, but is significantly outperformed by the graph and hypergraph Laplacians. Again, this this indicates that although the primary method we were interested in was persistent homology, the graph Laplacian is performing significantly better.

Explainability. We ran feature importance analysis on our models using the Mean Decrease in Impurity (MDI) feature importance technique for random forests (Scornet, 2023). For the persistent homology features we get a ranking over the persistence images, shown on the left of Figure 7.5, which tell us that topological features that are born at the start of the filtration and persist throughout it are most important. The execution of the cleanware or malware is immediately after start-up, suggesting that our methods are generally picking up on actions that correspond to the software. For the graph Laplacian we investigated the feature importance, with the eigenvalues on the x-axis sorted by magnitude (largest first). We see on the right of Figure 7.5 that generally the largest eigenvalues are most discriminative, although there is a spike towards the smaller eigenvalues that we hypothesise is due to the importance of small eigenvalues in characterising the structure of graphs Nica (2016). As each element in an eigenvector corresponds to a node in the graph we can plot those values on the nodes, leading to a visualisation that could be used to flag nodes (and therefore logs) of interest.

7.4 Discussion

Although the structural feature vectors do not outperform the counts baseline across the board, we believe this remains a result of interest: the fact that the topological and spectral techniques are able to perform at a similar or better level than a proven baseline using just the structural information contained in the logs is a positive result. The graph Laplacian in particular can rival or outperform the baseline across all scenarios, demonstrating that the structure of host-based logs contains discriminative information on their content. As the topological and spectral feature vectors only use the global structure that is intrinsic to the log, the fact that our techniques are able to rival the counts baseline means that *just the structural information is somehow fingerprinting the anomalous behaviour.* There was no guarantee that this would be the case.

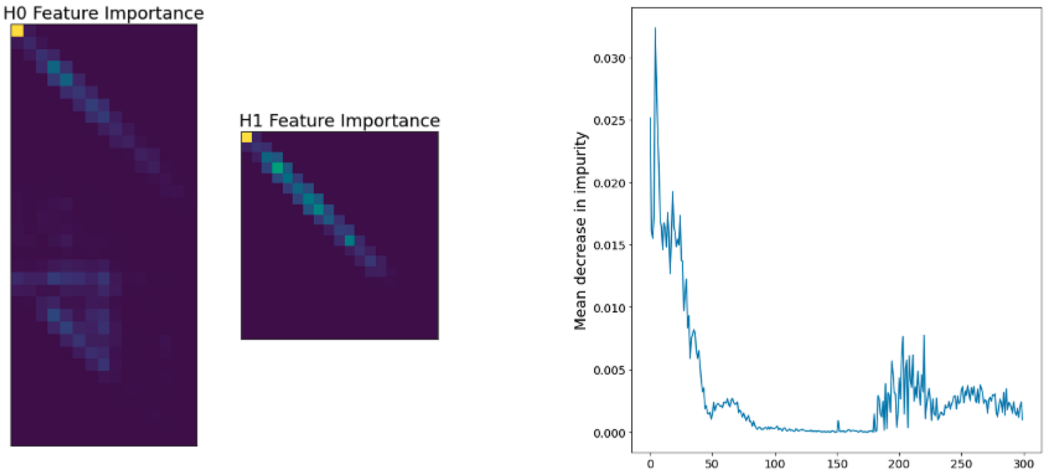


FIGURE 7.5: Feature importance of the 0 and 1-persistence images and spectrum of the Laplacian (ordered by size of the eigenvalue) for the malware dataset. The topological features that are born early and persist the longest are most valuable, as are the large eigenvalues.

Furthermore, it is unlikely that any one technique is going to be the silver bullet to find malware and malicious activity within computer logs. We have demonstrated that the structural representations can be used to detect both malware execution and adversary activity within host-based logs, and although the individual performance is far from what could be used as a sole indicator of anomalous activity, the reality is that any tool built is going to be part of a suite of indicators that can flag potentially malicious activity for further investigation.

Future work. As our datasets have balanced classes, which wouldn't be the case in a realistic anomaly detection scenario (where benign behaviour would far outweigh malicious behaviour), future work could be to adapt these structural vectors to a setting where they are part of a wider anomaly detection pipeline, feeding into a more sophisticated model built for anomaly detection. Further future work should also further investigate the explainability aspect of these structural feature vectors. Although the initial explainability results provide some initial understanding of the important structural features within the underlying logs, further investigation could lead to a better understanding of the correspondence between malicious activity on the host and the structural features this leads to. There is also a line of work that we do not compare to, namely using transformers to learn representations (see, for example, Nedelkoski et al. (2020); Guo et al. (2021); Le and Zhang (2021)). Further work should compare and contrast with this line of research.

Within the broader context of my thesis, this research was done before our work on applications of the persistent Laplacian to data analysis. Given (i) that the graph Laplacian was our best performing structural representation, and (ii) that the temporal aspect of the logs is generally considered key information when classifying logs, the persistent

Laplacian is a natural candidate for further work in representations of computer logs that account for their intrinsic structure.

Chapter 8

Conclusions

In this thesis we have been interested in understanding and extending applications of persistence-based summaries to data analysis. We understood the current context of their application via two literature reviews: in Chapter 3 we reviewed the literature on methodologies for topological machine learning and data analysis, and in Chapter 6 we reviewed the literature pertaining to applications of persistence-based summaries to cyber security. Our research contributions were also aligned by this motivation: in Part II we investigated the efficacy of topological loss terms in deep learning using statistical learning theory, applied the recently developed persistent Laplacian to data analysis for the first time, extended the fuzzy c-means clustering algorithm to persistence diagram space, and looked at applications of that algorithm to deep learning model selection. In Part III we considered cyber security as a specific application domain, proposing and evaluating a new methodology to use persistence-based summaries with host-based logs. We finish by discussing the specific contributions, limitations, and future work of each chapter.

We began by investigating the use of topological loss terms in deep learning via statistical learning theory in Section 4.1. Although this initial investigation into topological loss terms provided a new framework for thinking about the utility of topological machine learning, we ultimately showed that we cannot prove classical learnability bounds when restricting a hypothesis class to the preimage of a persistence diagram. However, just because the loss term will not always be useful does not mean there is no utility in using topological information. In some ways, it is not surprising that persistence diagrams cannot be guaranteed to always provide useful information: they capture invariants up to isometry, which, although demonstrated in the literature to be of great use in some domains, can be less useful in other applications. Interesting future work would be to consider persistent Laplacian-based loss terms; my intuition is that the additional geometric information provided by the non-zero eigenvalues is likely to enable some sort of learnability bound to be proven.

The second part of Chapter 4 proposed a feature vectorisation scheme for the recently developed persistent Laplacian. Evaluating the persistent Laplacian against topological and non-topological baselines empirically demonstrated that it can consistently outperform persistent homology. The theory also supports this: as the resolution parameter we define gets large we recover the entire persistent homology, and the non-zero eigenvalues are known to provide additional information that is deeply tied to the structure of the underlying data. We do not outperform the deep learning baselines that we report, which is indicative of the wider tension between topological representations of data and deep learning: deep learning models with rapidly increasing numbers of parameters are likely to outperform topological methods. As we discussed when reporting the non-topological baselines (Section 4.3.4), topological techniques do still retain advantages over deep learning methods: they offer a concise representation of structure with strong underlying theory, and can be easier to both interpret and train than deep learning methods. The potential for integrating topological techniques into deep learning also promises a cooperative approach. An interesting piece of future work would be a comprehensive investigation into the scenarios where topological tools retain advantages over deep learning; Turkes et al. (2022) have some interesting work in this direction. I believe that the application of the persistent Laplacian is the most interesting contribution of this thesis, offering a novel data analysis technique based on recently developed theory. As such, further development and usage of the persistent Laplacian in data science is the future work that I would be most excited to see carried out. Integrating the persistent Laplacian into deep learning via a topological loss term would also be valuable future work. Perhaps the biggest weakness of the persistent Laplacian currently is that it can only be computed for complex pairs $K \subseteq L$. We would recommend a theoretical effort to enable the persistent Laplacian to be computed over an entire filtration simultaneously. This is likely to unlock its full potential for data science.

In Chapter 5 we extended the fuzzy c-means clustering algorithm to persistence diagram space, proving the same convergence guarantee as the Euclidean case. The reality is that in most cases the better option is to map the persistence diagram into a feature vector and perform Euclidean clustering, bypassing the expensive process of computing Fréchet means and distances on persistence diagrams. Despite this we have received interest in our released code from TDA researchers, so we hope it can provide value. It is of particular use if you wish to stay in persistence diagram space, which is sometimes the case, for example if you wish to backpropogate through the clustering and cannot differentiate through your vectorisation strategy. In my opinion the most interesting result in this chapter is the application to pre-trained deep learning model selection. The link between the topology of decision boundaries and the generalisation ability of the model to unseen datasets remains relatively unexplored, and I would suggest this area as promising future work.

In Chapter 7 we investigated the application of persistence-based summaries to host-based logs, proposing representing logs as a filtration of simplicial complexes that directly captures both their intrinsic structure and temporal nature. Although we evaluated persistent homology, we found that the graph Laplacian, which disregards both the higher-order interactions and temporal aspect of the logs, was our best performing representation. However, the fact that we are able to rival a non-topological baseline which is regularly used in the literature using just the structural information is a significant result. A natural extension of this work is to use the persistent Laplacian on our mapping of host-based logs into a filtration of simplicial complexes. This would be able to exploit the higher-order structure and temporal features that we anticipated providing much more information in these experiments.

References

Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18(8):1–35, 2017. URL `http://jmlr.org/papers/v18/16-337.html`.

Aaron Adcock, Erik Carlsson, and Gunnar Carlsson. The ring of algebraic functions on persistence bar codes. *Homology, Homotopy and Applications*, 18(1):381–402, 2016. . URL `https://doi.org/10.4310/hha.2016.v18.n1.a21`.

Ravi P Agarwal and Elena Cristina Flaut. *An introduction to linear algebra*. CRC Press, 2017.

Sameer Agarwal, Kristin Branson, and Serge Belongie. Higher order learning with graphs. In *Proceedings of the 23rd international conference on Machine learning*, pages 17–24, 2006.

Cuneyt Gurcan Akcora, Yitao Li, Yulia R Gel, and Murat Kantarcioglu. Bitcoinheist: Topological data analysis for ransomware detection on the bitcoin blockchain. *arXiv preprint arXiv:1906.07852*, 2019.

Sinan G Aksoy, Kathleen E Nowak, Emilie Purvine, and Stephen J Young. Relative hausdorff distance for network analysis. *Applied Network Science*, 4(1):1–25, 2019.

J. W. Alexander. A proof of the invariance of certain constants of analysis situs. *Transactions of the American Mathematical Society*, 16:148–154, 1915.

Shun-ichi Amari and Hiroshi Nagaoka. *Methods of Information Geometry*, volume 191. American Mathematical Society, 01 2000.

Rabih Assaf, Alban Goupil, Mohammad Kacim, and Valeriu Vrabie. Topological persistence based on pixels for object segmentation in biomedical images. In *2017 Fourth International Conference on Advances in Biomedical Engineering (ICABME)*. IEEE, October 2017. . URL `https://doi.org/10.1109/icabme.2017.8167531`.

Sivaraman Balakrishnan, Alessandro Rinaldo, Don Sheehy, Aarti Singh, and Larry Wasserman. Minimax rates for homology inference. In Neil D. Lawrence and Mark Girolami, editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pages 64–72, La Palma, Canary Islands, 21–23 Apr 2012. PMLR. URL https://proceedings.mlr.press/v22/balakrishnan12a.html.

Peter W Battaglia, Razvan Pascanu, Matthew Lai, Danilo Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. *NeurIPS*, 2016.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Ulrich Bauer. Ripser: efficient computation of vietoris-rips persistence barcodes, August 2019. Preprint.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Nips*, volume 14, pages 585–591, 2001.

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(11), 2006.

Vaishak Belle and Ioannis Papantonis. Principles and practice of explainable machine learning. *Frontiers in Big Data*, 4, July 2021. . URL https://doi.org/10.3389/fdata.2021.688969.

P. Bendich, S. P. Chin, J. Clark, J. Desena, J. Harer, E. Munch, A. Newman, D. Porter, D. Rouse, N. Strawn, and A. Watkins. Topological and statistical behavior classifiers for tracking applications. *IEEE Transactions on Aerospace and Electronic Systems*, 52(6): 2644–2661, 2016.

Paul Bendich, Sayan Mukherjee, and B. Wang. Towards stratification learning through homology inference. *arXiv: Geometric Topology*, 2010.

Paul Bendich, B. Wang, and S. Mukherjee. Local homology transfer and stratification learning. In *SODA*, 2012.

Paul Bendich, J. S. Marron, Ezra Miller, Alex Pieloch, and Sean Skwerer. Persistent homology analysis of brain artery trees. *The Annals of Applied Statistics*, 10(1):198–218, March 2016. . URL https://doi.org/10.1214/15-aoas886.

Gyora M. Benedek and Alon Itai. Nonuniform learnability. In Timo Lepistö and Arto Salomaa, editors, *Automata, Languages and Programming*, pages 82–92, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg. ISBN 978-3-540-39291-0.

Dimitri Bertsekas. The auction algorithm: A distributed relaxation method for the assignment problem. *Annals of Operations Research*, 14:105–123, 12 1988. .

E Betti. Sopra gli spazi di un numero qualunque di dimensioni. *Annali di Matematica pura ed applicata*, 4:140–158., 1871.

J. C. Bezdek. A convergence theorem for the fuzzy isodata clustering algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2(1):1–8, Jan 1980. ISSN 1939-3539. .

J. C. Bezdek, R. J. Hathaway, M. J. Sabin, and W. T. Tucker. Convergence theory for fuzzy c-means: Counterexamples and repairs. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(5):873–877, 1987.

Monowar H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. Network anomaly detection: Methods, systems and tools. *IEEE Communications Surveys & Tutorials*, 16(1):303–336, 2014. .

Norman Biggs. *Algebraic Graph Theory*. Cambridge University Press, Cambridge, second edition, 1993. ISBN 0521458978.

Trevor Bihl, Robert Gutierrez, Kenneth Bauer, Brad Boehmke, and Cade Saie. Topological data analysis for enhancing embedded analytics for enterprise cyber log analysis and forensics. In *Proceedings of the Annual Hawaii International Conference on System Sciences*. Hawaii International Conference on System Sciences, 2020. . URL `https://doi.org/10.24251/hicss.2020.238`.

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Nicholas Bishop, Thomas Davies, and Long Tran-Thanh. Hypothesis classes with a unique persistence diagram are not nonuniformly learnable. In *NeurIPS 2020 Workshop on Topological Data Analysis and Beyond*, 2020. URL `https://openreview.net/forum?id=Ay-RgChnje`.

L. C. Blum and J.-L. Reymond. 970 million druglike small molecules for virtual screening in the chemical universe database GDB-13. *J. Am. Chem. Soc.*, 131:8732, 2009.

Andrew Blumberg, Itamar Gal, Michael Mandell, and Matthew Pancia. Robust statistics, hypothesis testing, and confidence intervals for persistent homology on metric measure spaces. *Foundations of Computational Mathematics*, 14, 06 2012. .

Omer Bobrowski and Primoz Skraba. On the universality of random persistence diagrams. *arXiv preprint arXiv:2207.03926*, 2022.

Cristian Bodnar, Fabrizio Frasca, Yu Guang Wang, Nina Otter, Guido Montúfar, Pietro Lio, and Michael Bronstein. Weisfeiler and lehman go topological: Message passing simplicial networks. *arXiv preprint arXiv:2103.03212*, 2021.

Benjamin Bowman, Craig Laprade, Yuede Ji, and H. Howie Huang. Detecting lateral movement in enterprise computer networks with unsupervised graph AI. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, pages 257–268, San Sebastian, October 2020. USENIX Association. ISBN 978-1-939133-18-2. URL https://www.usenix.org/conference/raid2020/presentation/bowman.

Leo Breiman. *Classification and regression trees*. Routledge, 2017.

L. E. J Brouwer. Beweis der invarianz der dimensionzahl. *Mathematische Annalen*, 70: 161–165, 1911.

Paul Bruillard, Kathleen Nowak, and Emilie Purvine. Anomaly detection using persistent homology. In *2016 Cybersecurity Symposium (CYBERSEC)*, pages 7–12, 2016. .

P. Bubenik. Statistical topological data analysis using persistence landscapes. *Journal of Machine Learning Research*, 16:77–102, 01 2015.

Peter Bubenik. The persistence landscape and some of its properties. *Abel Symposia*, page 97–117, 2020. ISSN 2197-8549. . URL http://dx.doi.org/10.1007/978-3-030-43408-3_4.

Peter Bubenik and Peter Kim. A statistical approach to persistent homology. *Homology, Homotopy and Applications*, 9, 08 2006. .

Peter Bubenik and Alexander Wagner. Embeddings of persistence diagrams into hilbert spaces. *CoRR*, abs/1905.05604, 2019. URL http://arxiv.org/abs/1905.05604.

Peter Bubenik, Gunnar Carlsson, Peter T. Kim, and Zhi-Ming Luo. Statistical topology via morse theory persistence and nonparametric estimation. pages 75–92, 2010. . URL https://doi.org/10.1090/conm/516/10167.

Ricardo JGB Campello. A fuzzy extension of the rand index and other related indexes for clustering and classification assessment. *Pattern Recognition Letters*, 28(7):833–841, 2007.

Gunnar Carlsson. Topology and data. *Bulletin of The American Mathematical Society*, 46: 255–308, 04 2009. .

Gunnar Carlsson and Rickard Brüel Gabrielsson. Topological approaches to deep learning. In Nils A. Baas, Gunnar E. Carlsson, Gereon Quick, Markus Szymik, and Marius Thaule, editors, *Topological Data Analysis*, pages 119–146, Cham, 2020. Springer International Publishing. ISBN 978-3-030-43408-3.

Gunnar Carlsson and Mikael Vejdemo-Johansson. *Topological Data Analysis with Applications*. Cambridge University Press, 2021. .

Mathieu Carrière and Ulrich Bauer. On the metric distortion of embedding persistence diagrams into separable hilbert spaces. In *Symposium on Computational Geometry*, 2019.

Mathieu Carriere, Marco Cuturi, and Steve Oudot. Sliced wasserstein kernel for persistence diagrams. In *International conference on machine learning*, pages 664–673. PMLR, 2017.

Mathieu Carrière, Frédéric Chazal, Marc Glisse, Yuichi Ike, and Hariprasad Kannan. Optimizing persistent homology based functions. *CoRR*, abs/2010.08356, 2020. URL `https://arxiv.org/abs/2010.08356`.

Mathieu Carrière, Frédéric Chazal, Yuichi Ike, T. Lacombe, Martin Royer, and Y. Umeda. Perslay: A neural network layer for persistence diagrams and new graph topological signatures. In *AISTATS*, 2020.

Mathieu Carrière, Steve Oudot, and Maks Ovsjanikov. Stable topological signatures for points on 3d shapes. *Computer Graphics Forum*, 34, 07 2015. .

Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *CoRR*, abs/1810.00069, 2018. URL `http://arxiv.org/abs/1810.00069`.

Frédéric Chazal, David Cohen-Steiner, and Quentin Mérigot. Geometric Inference for Measures based on Distance Functions. *Foundations of Computational Mathematics*, 11 (6):733–751, 2011. . URL `https://hal.inria.fr/inria-00383685`.

Frédéric Chazal, Brittany Terese Fasy, Fabrizio Lecci, Alessandro Rinaldo, and Larry Wasserman. Stochastic convergence of persistence landscapes and silhouettes. In *Proceedings of the thirtieth annual symposium on Computational geometry*. ACM, June 2014a. . URL `https://doi.org/10.1145/2582112.2582128`.

Frédéric Chazal, Marc Glisse, Catherine Labruère, and Bertrand Michel. Convergence rates for persistence diagram estimation in topological data analysis. In *International Conference on Machine Learning*, pages 163–171. PMLR, 2014b.

Frédéric Chazal, Brittany Fasy, Fabrizio Lecci, Bertrand Michel, Alessandro Rinaldo, and Larry Wasserman. Subsampling methods for persistent homology. In *International Conference on Machine Learning*, pages 2143–2151. PMLR, 2015.

Frédéric Chazal, Vin De Silva, Marc Glisse, and Steve Oudot. *The structure and stability of persistence modules*, volume 10. Springer, 2016.

Frédéric Chazal, Leonidas J. Guibas, Steve Y. Oudot, and Primoz Skraba. Persistence-based clustering in riemannian manifolds. *J. ACM*, 60(6), November 2013. ISSN 0004-5411. . URL `https://doi.org/10.1145/2535927`.

Chao Chen, Xiuyan Ni, Qinxun Bai, and Yusu Wang. Toporeg: A topological regularizer for classifiers. *CoRR*, abs/1806.10714, 2018. URL http://arxiv.org/abs/1806.10714.

Yen-Chi Chen, Daren Wang, Alessandro Rinaldo, and Larry Wasserman. Statistical analysis of persistence intensity functions, 2015.

Sofya Chepushtanova, Tegan Emerson, Eric Hanson, Michael Kirby, Francis Motta, Rachel Neville, Chris Peterson, Patrick Shipman, and Lori Ziegelmeier. Persistence images: An alternative persistent homology representation. 07 2015.

I. Chevyrev, Vidit Nanda, and H. Oberhauser. Persistence paths and signature features in topological data analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42:192–202, 2020.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 257–266, 2019.

Fan Chung. *Spectral Graph Theory*. American Mathematical Society, 1992a.

Fan RK Chung. The laplacian of a hypergraph. In *Expanding graphs*, pages 21–36, 1992b.

Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature. 2018.

J. Clough, N. Byrne, I. Oksuz, V. A. Zimmer, J. A. Schnabel, and A. King. A topological loss function for deep-learning based image segmentation using persistent homology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020.

David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, Jan 2007. ISSN 1432-0444. . URL https://doi.org/10.1007/s00454-006-1276-5.

Joseph R Collins, Michaela Iorga, Dmitry Cousin, and David Chapman. Passive encrypted iot device fingerprinting with persistent homology. *UMBC Faculty Collection*, 2020.

Padraig Corcoran and Christopher B. Jones. Spatio-temporal modeling of the topology of swarm behavior with persistence landscapes. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPACIAL '16, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450345897. . URL https://doi.org/10.1145/2996913.2996949.

Marc Coudriau, Abdelkader Lahmadi, and Jérôme François. Topological analysis and visualisation of network monitoring data: Darknet case study. In *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, 2016. .

Gideon Creech and Jiankun Hu. A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns. *IEEE Transactions on Computers*, 63(4):807–819, 2014. .

Justin Curry. The fiber of the persistence map for functions on the interval. *Journal of Applied and Computational Topology*, 2:301–321, 2018.

Marco Cuturi. Positive definite kernels in machine learning. *arXiv preprint arXiv:0911.5367*, 2009.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2292–2300. Curran Associates, Inc., 2013. URL `http://papers.nips.cc/paper/4927-sinkhorn-distances-lightspeed-computation-of-optimal-transport.pdf`.

Marco Cuturi and Arnaud Doucet. Fast computation of wasserstein barycenters. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 685–693, Bejing, China, 22–24 Jun 2014. PMLR. URL `https://proceedings.mlr.press/v32/cuturi14.html`.

R. W. R. Darling, John A. Emanuello, Emilie Purvine, and Ahmad Ridley. Proceedings of TDA: Applications of Topological Data Analysis to Data Science, Artificial Intelligence, and Machine Learning Workshop at SDM 2022. 2022.

Thomas Davies. A Review of Topological Data Analysis for Cybersecurity. In *Workshop on Artificial Intelligence for Cyber Security*, 2022a. URL `https://arxiv.org/abs/2202.08037`.

Thomas Davies. Topological data analysis for anomaly detection in host-based logs. In *SDM Workshop On Applications of Topological Data Analysis to Data Science, Artificial Intelligence, and Machine Learning*, 2022b.

Thomas Davies, Jack Aspinall, Bryan Wilder, and Tran-Thanh Long. Fuzzy c-means clustering in persistence diagram space for deep learning model selection. In Sophia Sanborn, Christian Shewmake, Simone Azeglio, Arianna Di Bernardo, and Nina Miolane, editors, *Proceedings of the 1st NeurIPS Workshop on Symmetry and Geometry in Neural Representations*, volume 197 of *Proceedings of Machine Learning Research*, pages 137–157. PMLR, 03 Dec 2023a. URL `https://proceedings.mlr.press/v197/davies23a.html`.

Thomas Davies, Zhengchao Wan, and Ruben J Sanchez-Garcia. The persistent Laplacian for data science: Evaluating higher-order persistent spectral representations of data. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan

Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 7249–7263. PMLR, 23–29 Jul 2023b. URL https://proceedings.mlr.press/v202/davies23c.html.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852, 2016.

James Demmel. Cs267: Notes for lecture 23, April 1999. URL https://people.eecs.berkeley.edu/~demmel/cs267/lecture20/lecture20.html.

Chenhui Deng, Xiuyu Li, Zhuo Feng, and Zhiru Zhang. Garnet: Reduced-rank topology learning for robust and scalable graph neural networks. In Bastian Rieck and Razvan Pascanu, editors, *Proceedings of the First Learning on Graphs Conference*, volume 198 of *Proceedings of Machine Learning Research*, pages 3:1–3:23. PMLR, 09–12 Dec 2022. URL https://proceedings.mlr.press/v198/deng22a.html.

Detica. The cost of cyber crime, 2011. URL https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/60943/the-cost-of-cyber-crime-full-report.pdf. Accessed on 01/03/2023.

Adrish Dey and Sayantan Das. Topo sampler: A topology constrained noise sampling for {gan}s. In *NeurIPS 2020 Workshop on Topological Data Analysis and Beyond*, 2020. URL https://openreview.net/forum?id=OTxZfmVFlTO.

Tamal Krishna Dey and Yusu Wang. *Computational Topology for Data Analysis*. Cambridge University Press, 2022. ISBN 9781009098168.

Vincent Divol and Theo Lacombe. Estimation and quantization of expected persistence diagrams. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2760–2770. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/divol21a.html.

Jozef Dodziuk. Difference equations, isoperimetric inequality and transience of certain random walks. *Transactions of the American Mathematical Society*, 284(2):787–787, February 1984. . URL https://doi.org/10.1090/s0002-9947-1984-0743744-x.

Gérard Dray, C Raissi, J Brissaud, Pascal Poncelet, Mathieu Roche, and Maguelonne Teisseire. Web analysis traffic challenge: Description and results. In *Discovery Challenge ECML/PKDD*, 2007.

Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017*

ACM SIGSAC Conference on Computer and Communications Security, CCS '17, page 1285–1298, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349468. . URL https://doi.org/10.1145/3133956.3134015.

J. C. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, January 1973. . URL https://doi.org/10.1080/01969727308546046.

Beno Eckmann. Harmonische funktionen und randwertaufgaben in einem komplex. *Commentarii mathematici Helvetici*, 17:240–255, 1944/45. URL http://eudml.org/doc/138857.

Herbert Edelsbrunner and John Harer. *Computational Topology - an Introduction.* American Mathematical Society, 2010. ISBN 978-0-8218-4925-5.

Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, Nov 2002. ISSN 1432-0444. . URL https://doi.org/10.1007/s00454-002-2885-2.

Alon Efrat, Alon Itai, and Matthew Katz. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 31:1–28, 09 2001. .

Y. Elkin and V. Kurlin. The mergegram of a dendrogram and its stability. In *Proceedings of MFCS (Mathematical Foundations of Computer Science)*, pages 32:1–32:13, 2020. .

Y. Elkin and V. Kurlin. Isometry invariant shape recognition of projectively perturbed point clouds by the mergegram extending 0d persistence. *Mathematics*, 9(17), 2021.

Barbara Di Fabio and Massimo Ferri. Comparing persistence diagrams through complex vectors. In *Image Analysis and Processing — ICIAP 2015*, pages 294–305. Springer International Publishing, 2015. . URL https://doi.org/10.1007/978-3-319-23231-7_27.

Claude Fachkha and Mourad Debbabi. Darknet as a source of cyber intelligence: Survey, taxonomy and characterization. *IEEE Communications Surveys & Tutorials*, 18:1–1, 01 2015. .

Brittany Fasy, Fabrizio Lecci, Alessandro Rinaldo, Larry Wasserman, Sivaraman Balakrishnan, and Aarti Singh. Confidence sets for persistence diagrams. *The Annals of Statistics*, 42:2301–2339, 03 2014a.

Brittany Terese Fasy, Jisu Kim, Fabrizio Lecci, and Clément Maria. Introduction to the r package tda. *arXiv preprint arXiv:1411.1830*, 2014b.

Massimo Ferri and Claudia Landi. Representing size functions by complex polynomials. *Proc. Math. Met. in Pattern Recognition*, pages 16–19, 1999.

Anatolij Timofeevič Fomenko and Dmitry Fuchs. *Homotopical topology*, volume 273. Springer, 2016.

Patrizio Frosini. Measuring shapes by size functions. In David P. Casasent, editor, *Intelligent Robots and Computer Vision X: Algorithms and Techniques*, volume 1607, pages 122 – 133. International Society for Optics and Photonics, SPIE, 1992. . URL https://doi.org/10.1117/12.57059.

M. Fréchet. L'intégrale abstraite d'une fonction abstraite d'une variable abstraite et son application á la moyenne d'un élément aléatoire de nature quelconquee. *Revue Scientifique*, 82:483–512, 1944.

N. Gabdrakhmanova. Constructing a neural-net model of network traffic using the topologic analysis of its time series complexity. In Boris Kryzhanovsky, Witali Dunin-Barkowski, and Vladimir Redko, editors, *Advances in Neural Computation, Machine Learning, and Cognitive Research*, pages 91–97, Cham, 2018. Springer International Publishing.

Rickard Brüel Gabrielsson and G. Carlsson. Exposition and interpretation of the topology of neural networks. *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1069–1076, 2019.

Rickard Brüel Gabrielsson, Bradley J. Nelson, Anjan Dwaraknath, and Primoz Skraba. A topology layer for machine learning. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1553–1563. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/gabrielsson20a.html.

Jose Gallego-Posada and Patrick Forré. Simplicial regularization. In *ICLR 2021 Workshop on Geometrical and Topological Representation Learning*, 2021. URL https://openreview.net/forum?id=x9xn6HKgefz.

Sebastian Garcia, Martin Grill, Jan Stiborek, and Alejandro Zunino. An empirical comparison of botnet detection methods. *computers & security*, 45:100–123, 2014.

Adélie Garin and Guillaume Tauzin. A topological "reading" lesson: Classification of MNIST using TDA. *CoRR*, abs/1910.08345, 2019. URL http://arxiv.org/abs/1910.08345.

Thomas Gebhart and Paul Schrater. Adversary detection in neural networks via persistent homology. *ArXiv*, abs/1711.10056, 2017.

Thomas Gebhart and Paul Schrater. Adversarial examples target topological holes in deep networks. *ArXiv*, abs/1901.09496, 2019.

Marian Gidea and Yuri Katz. Topological data analysis of financial time series: Landscapes of crashes. *Physica A: Statistical Mechanics and its Applications*, 491:820–834, February 2018. . URL https://doi.org/10.1016/j.physa.2017.09.028.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

Joshua Gilmore, Mihaela Sardiu, Brad Groppe, Janet Thornton, Xingyu Liu, Dayebgadoh Gerald, Charles Banks, Brian Slaughter, Jay Unruh, Jerry Workman, Laurence Florens, and Michael Washburn. Wdr76 co-localizes with heterochromatin related proteins and rapidly responds to dna damage. *PLOS ONE*, 11:e0155492, 06 2016. .

Christopher Wei Jin Goh, Cristian Bodnar, and Pietro Lio. Simplicial attention networks. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022. URL https://openreview.net/forum?id=ScfRNWkpec.

Weikang Gong, JunJie Wee, Min-Chun Wu, Xiaohan Sun, Chunhua Li, and Kelin Xia. Persistent spectral simplicial complex-based machine learning for chromosomal structural analysis in cellular differentiation. *Briefings in Bioinformatics*, 23(4), May 2022. . URL https://doi.org/10.1093/bib/bbac168.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2015.

Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 855–864, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. . URL https://doi.org/10.1145/2939672.2939754.

Haixuan Guo, Shuhan Yuan, and Xintao Wu. Logbert: Log anomaly detection via bert. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

William H. Guss and R. Salakhutdinov. On characterizing the capacity of neural networks using algebraic topology. *ArXiv*, abs/1802.04443, 2018.

Robert J Gutierrez, Kenneth W Bauer, Bradley C Boehmke, Cade M Saie, and Trevor J Bihl. Cyber anomaly detection: Using tabulated vectors and embedded analytics for efficient data mining. *Journal of Algorithms & Computational Technology*, 12(4):293–310, 2018. . URL https://doi.org/10.1177/1748301818791503.

Celia Hacker. k-simplex2vec: a simplicial extension of node2vec. *arXiv preprint arXiv:2010.05636*, 2020.

Mustafa Hajij, Paul Rosen, and Bei Wang. Mapper on graphs for network visualization. *arXiv preprint arXiv:1804.11242*, 2018.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

Xueyuan Han, Thomas Pasquier, Adam Bates, James Mickens, and Margo Seltzer. Unicorn: Runtime provenance-based detector for advanced persistent threats. In *Proceedings 2020 Network and Distributed System Security Symposium*. Internet Society, 2020. . URL https://doi.org/10.14722/ndss.2020.24046.

Wajih Ul Hassan, Lemay Aguse, Nuraini Aguse, Adam Bates, and Thomas Moyer. Towards scalable cluster auditing through grammatical inference over provenance graphs. In *Network and Distributed Systems Security Symposium*, 2018.

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

Allen Hatcher. *Algebraic topology*. Cambridge Univ. Press, Cambridge, 2000. URL https://cds.cern.ch/record/478079.

Andrew Hay, Daniel Cid, Rory Bary, and Stephen Northcutt. *Getting Started with OSSEC*, pages 1–27. 12 2008. ISBN 9781597492409. .

Koby Hayashi, Sinan G. Aksoy, Cheong Hee Park, and Haesun Park. Hypergraph random walks, laplacians, and clustering. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, page 495–504, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368599. . URL https://doi.org/10.1145/3340531.3412034.

Felix Hensel, Michael Moor, and Bastian Rieck. A survey of topological machine learning methods. *Frontiers in Artificial Intelligence*, 4, May 2021. . URL https://doi.org/10.3389/frai.2021.681108.

Kate Highnam, Kai Arulkumaran, Zachary Hanif, and Nicholas R Jennings. Beth dataset: Real cybersecurity data for anomaly detection research. *ICML Workshop on Uncertainty and Robustness in Deep Learning*, 2021.

Peter Hilton. A brief, subjective history of homology and homotopy theory in this century. *Mathematics Magazine*, 61(5):282–291, December 1988. . URL https://doi.org/10.1080/0025570x.1988.11977391.

HM Government. A strong britain in an age of uncertainty: The national security strategy, 2010. URL https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/61936/national-security-strategy.pdf. Accessed on 01/03/2023.

Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

Christoph Hofer, Roland Kwitt, Marc Niethammer, and Andreas Uhl. Deep learning with topological signatures. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1634–1644. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/6761-deep-learning-with-topological-signatures.pdf.

Christoph Hofer, Roland Kwitt, Marc Niethammer, and Mandar Dixit. Connectivity-optimized representation learning via persistent homology. volume 97 of *Proceedings of Machine Learning Research*, pages 2751–2760, Long Beach, California, USA, 09–15 Jun 2019a. PMLR. URL http://proceedings.mlr.press/v97/hofer19a.html.

Christoph D. Hofer, Roland Kwitt, and Marc Niethammer. Learning representations of persistence barcodes. *Journal of Machine Learning Research*, 20(126):1–45, 2019b. URL http://jmlr.org/papers/v20/18-358.html.

Christoph D. Hofer, Roland Kwitt, and Marc Niethammer. Graph filtration learning. *CoRR*, abs/1905.10996, 2019c. URL http://arxiv.org/abs/1905.10996.

Roald Hoffmann, Artem Kabanov, Andrey Golov, and Davide Proserpio. Homo citans and carbon allotropes: For an ethics of citation. *Angewandte Chemie International Edition*, 55, 07 2016. .

Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36(3):1171–1220, June 2008. . URL https://doi.org/10.1214/009053607000000677.

Rick Hofstede, Pavel Celeda, Brian Trammell, Idilio Drago, Ramin Sadre, Anna Sperotto, and Aiko Pras. Flow monitoring explained: From packet capture to data analysis with NetFlow and IPFIX. *IEEE Communications Surveys &: Tutorials*, 16(4):2037–2064, 2014. . URL https://doi.org/10.1109/comst.2014.2321898.

Danijela Horak and Jürgen Jost. Spectra of combinatorial laplace operators on simplicial complexes. *Advances in Mathematics*, 244:303–336, 2013.

Ralph Howard. Rings, determinants, the smith normal form, and canonical forms for similarity of matrices. class notes. https://people.math.sc.edu/howard/Classes/700c/notes2.pdf, 2002. Accessed: 2023-02-13.

Xiaoling Hu, Fuxin Li, Dimitris Samaras, and Chao Chen. Topology-preserving deep image segmentation. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5657–5668. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/8803-topology-preserving-deep-image-segmentation.pdf.

Xiaoling Hu, Yusu Wang, Li Fuxin, Dimitris Samaras, and Chao Chen. Topology-aware segmentation using discrete morse theory. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=LGgdb4TS4Z.

D. Hull and D.J. Bacon. *Introduction to Dislocations (Fifth Edition)*. Butterworth-Heinemann, Oxford, fifth edition edition, 2011. ISBN 978-0-08-096672-4. . URL http://www.sciencedirect.com/science/article/pii/B9780080966724000190.

Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand Ceder, and Kristin a. Persson. The Materials Project: A materials genome approach to accelerating materials innovation. *APL Materials*, 1(1):011002, 2013. ISSN 2166532X. . URL http://link.aip.org/link/AMPADS/v1/i1/p011002/s1&Agg=doi.

Breier Jakub and Jana Branišová. Anomaly detection from log files using data mining techniques. *Lecture Notes in Electrical Engineering*, 339:449–457, 01 2015. .

S. Jean-Paul, T. Elseify, I. Obeid, and J. Picone. Issues in the reproducibility of deep learning results. In *2019 IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*, pages 1–4, 2019. .

Peiran Jiang, Ying Chi, Xiao-Shuang Li, Zhenyu Meng, Xiang Liu, Xian-Sheng Hua, and Kelin Xia. Molecular persistent spectral image (Mol-PSI) representation for machine learning models in drug design. *Briefings in Bioinformatics*, 23(1), 12 2021. ISSN 1477-4054. . URL https://doi.org/10.1093/bib/bbab527. bbab527.

Silvana Botti Jonathan Schmidt, Mário R. G. Marques and Miguel A. L. Marques. Recent advances and applications of machine learning in solid-state materials science. *npj Computational Materials*, 5(1):83, 2019. ISSN 2057-3960. . URL https://doi.org/10.1038/s41524-019-0221-0.

Cliff Joslyn, Sinan Aksoy, Dustin Arendt, Louis Jenkins, Brenda Praggastis, Emilie Purvine, and Marcin Zalewski. High performance hypergraph analytics of domain name system relationships. In *HICSS 2019 symposium on cybersecurity big data analytics*, 2019.

Jürgen Jost and Raffaella Mulas. Hypergraph laplace operators for chemical reaction networks. *Advances in mathematics*, 351:870–896, 2019.

S. Kalisnik, V. Kurlin, and D. Lesnik. A higher-dimensional homologically persistent skeleton. *Advances in Applied Mathematics*, 102:113–142, 2019.

Sara Kališnik. Tropical coordinates on the space of persistence barcodes. *Foundations of Computational Mathematics*, 19(1):101–129, January 2018. . URL https://doi.org/10.1007/s10208-018-9379-y.

Michael Kerber, Dmitriy Morozov, and Arnur Nigmetov. Geometry helps to compare persistence diagrams. *Journal of Experimental Algorithmics (JEA)*, 22:1–20, 2017.

Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1), July 2019. . URL `https://doi.org/10.1186/s42400-019-0038-7`.

Kwangho Kim, Jisu Kim, Manzil Zaheer, Joon Kim, Frederic Chazal, and Larry Wasserman. Pllay: Efficient topological layer based on persistent landscapes. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15965–15977. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/b803a9254688e259cde2ec0361c8abe4-Paper.pdf`.

Masao Kimura, Ippei Obayashi, Yasuo Takeichi, Reiko Murao, and Yasuaki Hiraoka. Non-empirical identification of trigger sites in heterogeneous processes using persistent homology. *Scientific Reports*, 8(1), February 2018. . URL `https://doi.org/10.1038/s41598-018-21867-z`.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

G. Kirchhoff. Ueber die auflösung der gleichungen, auf welche man bei der untersuchung der linearen vertheilung galvanischer ströme geführt wird. *Annalen der Physik*, 148:497–508, 1847.

G. Kirchhoff. On the solution of the equations obtained from the investigation of the linear distribution of galvanic currents. *IRE Transactions on Circuit Theory*, 5(1):4–7, 1958. .

Violeta Kovacev-Nikolic, Peter Bubenik, Dragan Nikolić, and Giseon Heo. Using persistent homology and dynamical distances to analyze protein binding. *Statistical Applications in Genetics and Molecular Biology*, 15(1), January 2016. . URL `https://doi.org/10.1515/sagmb-2015-0057`.

J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, March 1964. . URL `https://doi.org/10.1007/bf02289565`.

S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, March 1951. . URL `https://doi.org/10.1214/aoms/1177729694`.

V. Kurlin. A homologically persistent skeleton is a fast and robust descriptor of interest points in 2d images. In *Lecture Notes in Computer Science (Proceedings of CAIP 2015)*, volume 9256, pages 606–617, 2015a.

V. Kurlin. A one-dimensional homologically persistent skeleton of an unstructured point cloud in any metric space. *Computer Graphics Forum*, 34(5):253–262, 2015b.

Genki Kusano, Y. Hiraoka, and K. Fukumizu. Persistence weighted gaussian kernel for topological data analysis. In *ICML*, 2016.

Roland Kwitt, Stefan Huber, Marc Niethammer, Weili Lin, and Ulrich Bauer. Statistical topological data analysis - a kernel perspective. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3070–3078. Curran Associates, Inc., 2015. URL http://papers.nips.cc/paper/5887-statistical-topological-data-analysis-a-kernel-perspective.pdf.

Theo Lacombe, Marco Cuturi, and Steve Oudot. Large scale computation of means and clusters for persistence diagrams using optimal transport. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9770–9780. Curran Associates, Inc., 2018.

Peter S Landweber, Emanuel A Lazar, and Neel Patel. On fiber diameters of continuous maps. *The American Mathematical Monthly*, 123(4):392–397, 2016.

Tam Le and Makoto Yamada. Persistence fisher kernel: A riemannian manifold kernel for persistence diagrams. In *NeurIPS*, 2018.

Van-Hoang Le and Hongyu Zhang. Log-based anomaly detection without log parsing. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 492–504. IEEE, 2021.

Van-Hoang Le and Hongyu Zhang. Log-based anomaly detection with deep learning: How far are we? In *2022 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 2022.

Yann LeCun and Yoshua Bengio. *Convolutional Networks for Images, Speech, and Time Series*, page 255–258. MIT Press, Cambridge, MA, USA, 1998. ISBN 0262511029.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553): 436–444, 2015.

Michel Ledoux and Michel Talagrand. *Probability in Banach Spaces*. Classics in Mathematics. Springer-Verlag, Berlin, 2011. ISBN 978-3-642-20211-7.

Gottfried Wilhelm Leibniz and Karl Gerhardt. *Leibnizens Mathematische Schriften*. A. Asher, Berlin, 1863.

Adam Lerer, Ledell Wu, Jiajun Shen, Timothee Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich. Pytorch-biggraph: A large-scale graph embedding system. *arXiv preprint arXiv:1903.12287*, 2019.

David Letscher and Jason Fritts. Image segmentation using topological persistence. In *Computer Analysis of Images and Patterns: 12th International Conference, CAIP 2007, Vienna, Austria, August 27-29, 2007. Proceedings 12*, pages 587–595. Springer, 2007.

Jacob Leygonie, Steve Oudot, and Ulrike Tillmann. A framework for differential calculus on persistence barcodes. *ArXiv*, abs/1910.00960, 2019.

J. Li and H. W. Lewis. Fuzzy clustering algorithms — review of the applications. In *2016 IEEE International Conference on Smart Cloud (SmartCloud)*, pages 282–288, 2016. .

André Lieutier. Talk: Persistent harmonic forms, 2014. URL https://project.inria.fr/gudhi/files/2014/10/Persistent-Harmonic-Forms.pdf.

Richard P Lippmann, David J Fried, Isaac Graf, Joshua W Haines, Kristopher R Kendall, David McClung, Dan Weber, Seth E Webster, Dan Wyschogrod, Robert K Cunningham, et al. Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. In *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, volume 2, pages 12–26. IEEE, 2000.

Zirui Liu, Qingquan Song, Kaixiong Zhou, Ting-Hsiang Wang, Ying Shan, and Xia Hu. Detecting interactions from neural networks via topological analysis. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6390–6401. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/473803f0f2ebd77d83ee60daaa61f381-Paper.pdf.

S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, March 1982. . URL https://doi.org/10.1109/tit.1982.1056489.

Alexander Louie, Kyung Han Song, Alejandra Hotson, Ann Thomas Tate, and David S. Schneider. How many parameters does it take to describe disease tolerance? *PLOS Biology*, 14(4):1–21, 04 2016. . URL https://doi.org/10.1371/journal.pbio.1002435.

Siyang Lu, Xiang Wei, Yandong Li, and Liqiang Wang. Detecting anomaly in big data system logs using convolutional neural network. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pages 151–158. IEEE, 2018.

P. Lum, G. Singh, A. Lehman, T. Ishkanov, M. Vejdemo-Johansson, M. Alagappan, J. Carlsson, and G. Carlsson. Extracting insights from the shape of complex data using topology. *Scientific Reports*, 3, 2013.

Hengrui Luo, Justin Strait, and A. Saha. Combining geometric and topological information in image segmentation. *ArXiv*, abs/1910.04778, 2019.

Alireza Makhzani and B. Frey. k-sparse autoencoders. *CoRR*, abs/1312.5663, 2014.

Vasileios Maroulas, Joshua Mike, and Andrew Marchese. K-means clustering on the space of persistence diagrams. In *SPIE*, page 29, 08 2017. .

Facundo Mémoli, Zhengchao Wan, and Yusu Wang. Persistent laplacians: Properties, algorithms and implications. *SIAM Journal on Mathematics of Data Science*, 4(2):858–884, 2022.

Weibin Meng, Ying Liu, Yichen Zhu, Shenglin Zhang, Dan Pei, Yuqing Liu, Yihao Chen, Ruizhi Zhang, Shimin Tao, Pei Sun, et al. Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In *IJCAI*, volume 19, pages 4739–4745, 2019.

Zhenyu Meng and Kelin Xia. Persistent spectral-based machine learning (perspect ml) for protein-ligand binding affinity prediction. *Science Advances*, 7(19):eabc5329, 2021. . URL https://www.science.org/doi/abs/10.1126/sciadv.abc5329.

Bryce Meredig. Five high-impact research areas in machine learning for materials science. *Chemistry of Materials*, 31(23):9579–9581, 2019. . URL https://doi.org/10.1021/acs.chemmater.9b04078.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *ICLR workshop*, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013b.

Yuriy Mileyko, Sayan Mukherjee, and John Harer. Probability measures on the space of persistence diagrams. *Inverse Problems - INVERSE PROBL*, 27, 12 2011. .

Nina Miolane, Alice Le Brigant, Johan Mathe, Benjamin Hou, Nicolas Guigui, Yann Thanwerdas, Stefan Heyder, Olivier Peltre, Niklas Koep, Hadi Zaatiti, Hatem Hajri, Yann Cabanes, Thomas Gerald, Paul Chauchat, Christian Shewmake, Bernhard Kainz, Claire Donnat, Susan P. Holmes, and Xavier Pennec. Geomstats: A python package for riemannian geometry in machine learning. *CoRR*, abs/2004.04667, 2020. URL https://arxiv.org/abs/2004.04667.

Nina Miolane, Matteo Caorsi, Umberto Lupo, Marius Guerard, Nicolas Guigui, Johan Mathe, Yann Cabanes, Wojciech Reise, Thomas Davies, António Leitão, Somesh Mohapatra, Saiteja Utpala, Shailja Shailja, Gabriele Corso, Guoxi Liu, Federico Iuricich, Andrei Manolache, Mihaela Nistor, Matei Bejan, Armand Mihai Nicolicioiu, Bogdan-Alexandru Luchian, Mihai-Sorin Stupariu, Florent Michel, Khanh Dao Duc, Bilal Abdulrahman, Maxim Beketov, Elodie Maignant, Zhiyuan Liu, Marek Cerný, Martin Bauw, Santiago Velasco-Forero, Jesús Angulo, and Yanan Long. ICLR 2021 challenge for computational geometry & topology: Design and results. *CoRR*, abs/2108.09810, 2021. URL https://arxiv.org/abs/2108.09810.

Atish Mitra and Žiga Virk. The space of persistence diagrams on n points coarsely embeds into hilbert space. *Proceedings of the American Mathematical Society*, 149(6): 2693–2703, March 2021. . URL https://doi.org/10.1090/proc/15363.

Grégoire Montavon, Matthias Rupp, Vivekanand Gobre, Alvaro Vazquez-Mayagoitia, Katja Hansen, Alexandre Tkatchenko, Klaus-Robert Müller, and O Anatole von Lilienfeld. Machine learning of molecular electronic properties in chemical compound space. *New Journal of Physics*, 15(9):095003, 2013. URL http://stacks.iop.org/1367-2630/15/i=9/a=095003.

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124, 2017.

Chul Moon and Nicole A Lazar. Hypothesis testing for shapes using vectorized persistence diagrams. *arXiv preprint arXiv:2006.05466*, 2020.

M. Moor, Max Horn, Bastian Alexander Rieck, and K. Borgwardt. Topological autoencoders. *ArXiv*, abs/1906.00722, 2019.

Nour Moustafa, Benjamin Turnbull, and Kim-Kwang Raymond Choo. An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. *IEEE Internet of Things Journal*, 6(3):4815–4830, 2019. .

Elizabeth Munch, Katharine Turner, Paul Bendich, Sayan Mukherjee, Jonathan Mattingly, and John Harer. Probabilistic fréchet means for time varying persistence diagrams. *Electronic Journal of Statistics*, 9(1):1173–1204, 2015. ISSN 1935-7524. . URL http://dx.doi.org/10.1214/15-EJS1030.

J. R. Munkres. Algorithms for the assignment and transportation problems. *Journal of The Society for Industrial and Applied Mathematics*, 10:196–210, 1957.

James R. Munkres. *Topology*. Prentice Hall, New Delhi, 2nd. edition, 1974.

James R. Munkres. *Elements of Algebraic Topology*. Addison-Wesley, London, 1984.

Vidit Nanda. Computational algebraic topology lecture notes. `http://people.maths.ox.ac.uk/nanda/cat/TDANotes.pdf`, 2021. Accessed: 2021-09-17.

Masaki Narita. An empirical study on darknet visualization based on topological data analysis. *International Journal of Networked and Distributed Computing*, 9:52–58, 2021. ISSN 2211-7946. .

Julio Navarro, Véronique Legrand, Sofiane Lagraa, Jérôme François, Abdelkader Lahmadi, Giulia De Santis, Olivier Festor, Nadira Lammari, Fayçal Hamdi, Aline Deruyver, et al. Huma: A multi-layer framework for threat analysis in a heterogeneous log environment. In *Foundations and Practice of Security: 10th International Symposium, FPS 2017, Nancy, France, October 23-25, 2017, Revised Selected Papers 10*, pages 144–159. Springer, 2018.

Sasho Nedelkoski, Jasmin Bogatinovski, Alexander Acker, Jorge Cardoso, and Odej Kao. Self-attentive classification-based anomaly detection in unstructured logs. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 1196–1201. IEEE, 2020.

Bogdan Nica. A brief introduction to spectral graph theory. *arXiv preprint arXiv:1609.08072*, 2016.

Monica Nicolau, Arnold J. Levine, and Gunnar Carlsson. Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proceedings of the National Academy of Sciences*, 108(17):7265–7270, 2011. ISSN 0027-8424. . URL `https://www.pnas.org/content/108/17/7265`.

Jessica Nielson, Shelly Cooper, John Yue, Marco Sorani, Tomoo Inoue, Esther Yuh, Pratik Mukherjee, Tanya Petrossian, Jesse Paquette, Pek Lum, Gunnar Carlsson, Mary Vassar, Hester Lingsma, Wayne Gordon, Alex Valadka, David Okonkwo, Geoffrey Manley, and Adam Ferguson. Uncovering precision phenotype-biomarker associations in traumatic brain injury using topological data analysis. *PLOS ONE*, 12:e0169490, 03 2017. .

Partha Niyogi, Stephen Smale, and Shmuel Weinberger. Finding the homology of submanifolds with high confidence from random samples. *Discrete & Computational Geometry*, 39:419–441, 03 2008. .

Partha Niyogi, Stephen Smale, and Shmuel Weinberger. A topological view of unsupervised learning from noisy data. *SIAM J. Comput.*, 40:646–663, 01 2011. .

Emmy Noether. Ableitung der elementarteilertheorie aus der gruppentheorie. *Jahresbericht der Deutschen Mathematiker-Vereinigun*, 36:104, 1926.

Deepti Pachauri, Chris Hinrichs, Moo K. Chung, Sterling C. Johnson, and Vikas Singh. Topology-based kernels with application to inference problems in alzheimer's disease.

IEEE Transactions on Medical Imaging, 30(10):1760–1770, October 2011. . URL https://doi.org/10.1109/tmi.2011.2147327.

P. D. Pantula, S. S. Miriyala, L. Giri, and K. Mitra. Synchronicity identification in hippocampal neurons using artificial neural network assisted fuzzy c-means clustering. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1594–1600, 2020. .

Vic Patrangenaru, Peter Bubenik, Robert L. Paige, and Daniel Osborne. Challenges in topological object data analysis. *Sankhy*, 81(1):244–271, September 2018. . URL https://doi.org/10.1007/s13171-018-0137-7.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

R. Penrose. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(3):406–413, 1955. .

Jose A. Perea. A brief history of persistence. *ArXiv*, abs/1809.03624, 2018.

Jose A. Perea, Elizabeth Munch, and Firas A. Khasawneh. Approximating continuous functions on persistence diagrams using template functions. *CoRR*, abs/1902.07190, 2019. URL http://arxiv.org/abs/1902.07190.

Pavel Petrov, Stephen T Rush, Shaun Pinder, Christine H Lee, Peter T Kim, and Giseon Heo. Topological data analysis of clostridioides difficile infection and fecal microbiota transplantation. *Computational and Methodological Statistics and Biostatistics: Contemporary Essays in Advancement*, pages 427–446, 2020.

Henri Poincaré and John Stillwell. *Papers on Topology: Analysis Situs and its Five Supplements*. American Mathematical Society/London Mathematical Society, 2010.

Henri Poincaré. Analysis situs. *Journal de l'École Polytechnique.*, 2:1–123, 1895.

Michael Postol, Candace Diaz, Robert Simon, and Drew Wicke. Time-series data analysis for classification of noisy and incomplete internet-of-things datasets. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1543–1550. IEEE, 2019.

Alex Pothen, Horst D. Simon, and Kang-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430–452, 1990. . URL https://doi.org/10.1137/0611030.

Adrien Poulenard, Primoz Skraba, and Maks Ovsjanikov. Topological function optimization for continuous shape matching. *Computer Graphics Forum*, 37:13–25, 08 2018. .

Andrew Putnis. *An Introduction to Mineral Sciences*. Cambridge University Press, 1992. .

Talha Qaiser, K. Sirinukunwattana, Kazuaki Nakane, Yee-Wah Tsang, D. A. Epstein, and N. Rajpoot. Persistent homology for fast tumor segmentation in whole slide histology images. In *MIUA*, 2016.

Yuchi Qiu and Guo-Wei Wei. Persistent spectral theory-guided protein engineering. *Nature Computational Science*, 3:1–15, 02 2023. .

Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. Wasserstein barycenter and its application to texture mixing. volume 8, pages 435–446, 05 2011. .

Karthikeyan Natesan Ramamurthy, Kush Varshney, and Krishnan Mody. Topological data analysis of decision boundaries with application to model selection. In *International Conference on Machine Learning*, pages 5351–5360. PMLR, 2019.

Stefan Rass, Sandra König, Shahzad Ahmad, and Maksim Goman. Metricizing the euclidean space towards desired distance relations in point clouds. *arXiv preprint arXiv:2211.03674*, 2022.

G Reeb. Sur les points singuliers d'une formede pfaff complètement intégrable ou d'une fonction numérique. *C. R. Acad. Sci. Paris*, 222:847–849, 1946.

J. Reininghaus, S. Huber, U. Bauer, and R. Kwitt. A stable multi-scale kernel for topological machine learning. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4741–4748, 2015.

Bastian Rieck. On the expressivity of persistent homology in graph learning, 2023. URL https://arxiv.org/abs/2302.09826.

Bastian Rieck, Christian Bock, and Karsten Borgwardt. A persistent weisfeiler-lehman procedure for graph classification. In *International Conference on Machine Learning*, pages 5448–5458. PMLR, 2019a.

Bastian Rieck, Tristan Yates, Christian Bock, Karsten Borgwardt, Guy Wolf, Nicholas Turk-Browne, and Smita Krishnaswamy. Uncovering the topology of time-varying fmri data using cubical persistence. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6900–6912. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/4d771504ddcd28037b4199740df767e6-Paper.pdf.

Bastian Alexander Rieck, F. Sadlo, and H. Leitte. Topological machine learning with persistence indicator functions. *ArXiv*, abs/1907.13496, 2019b.

Bastian Alexander Rieck, Matteo Togninalli, C. Bock, M. Moor, Max Horn, Thomas Gumbsch, and K. Borgwardt. Neural persistence: A complexity measure for deep neural networks using algebraic topology. *ArXiv*, abs/1812.09764, 2019c.

Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 833–840, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195.

Vanessa Robins. Towards computing homology from approximations. *Topology Proceedings*, 24:503–532, 01 1999.

Vanessa Robins and Katharine Turner. Principal component analysis of persistent homology rank functions with case studies of spatial point patterns, sphere packing and colloids. *Physica D: Nonlinear Phenomena*, 07 2015. .

Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O. Anatole von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Phys. Rev. Lett.*, 108:058301, Jan 2012. . URL `https://link.aps.org/doi/10.1103/PhysRevLett.108.058301`.

Mihaela Sardiu, Joshua Gilmore, Brad Groppe, Laurence Florens, and Michael Washburn. Identification of topological network modules in perturbed protein interaction networks. *Scientific Reports*, 7:43845, 03 2017. .

Kristof T. Schütt, Farhad Arbabzadah, Stefan Chmiela, Klaus R. Müller, and Alexandre Tkatchenko. Quantum-chemical insights from deep tensor neural networks. *Nature Communications*, 8(1), January 2017. . URL `https://doi.org/10.1038/ncomms13890`.

Erwan Scornet. Trees, forests, and impurity-based variable importance in regression. In *Annales de l'Institut Henri Poincare (B) Probabilites et statistiques*, volume 59, pages 21–52. Institut Henri Poincaré, 2023.

Hiroyuki Shindo and Yuji Matsumoto. Gated graph recursive neural networks for molecular property prediction, 2019. URL `https://arxiv.org/abs/1909.00259`.

Gurjeet Singh, F. Mémoli, and G. Carlsson. Topological methods for the analysis of high dimensional data sets and 3d object recognition. In *PBG@Eurographics*, 2007.

P. Skraba, Gugan Thoppe, and D. Yogeshwaran. Randomly weighted d-complexes: Minimal spanning acycles and persistence diagrams. *Electron. J. Comb.*, 27:P2.11, 2020.

Philip Smith and Vitaliy Kurlin. Skeletonisation algorithms with theoretical guarantees for unorganised point clouds with high levels of noise. *Pattern Recognition*, 115:107902, 2021.

Philip Smith and Vitaliy Kurlin. Families of point sets with identical 1d persistence. *arXiv preprint arXiv:2202.00577*, 2022.

Yitzchak Solomon, Alexander Wagner, and Paul Bendich. A fast and robust method for global topological functional optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 109–117. PMLR, 2021.

Daniel Strömbom. Persistent homology in the cubical setting: theory, implementations and applications, 2007.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077, 2015.

Guillaume Tauzin, Umberto Lupo, Lewis Tunstall, Julian Burella Pérez, Matteo Caorsi, Anibal Medina-Mardones, Alberto Dassatti, and Kathryn Hess. giotto-tda: A topological data analysis toolkit for machine learning and data exploration, 2020.

Brenda Y. Torres, Jose Henrique M. Oliveira, Ann Thomas Tate, Poonam Rath, Katherine Cumnock, and David S. Schneider. Tracking resilience to infections by mapping disease space. *PLOS Biology*, 14(4):1–19, 04 2016. . URL `https://doi.org/10.1371/journal.pbio.1002436`.

Renata Turkes, Guido Montufar, and Nina Otter. On the effectiveness of persistent homology. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=DRjUkfExCix`.

Katharine Turner. Medians of populations of persistence diagrams. *Homology, Homotopy and Applications*, 22:255–282, 2020.

Katharine Turner and Gard Spreemann. Same but different: Distance correlations between topological summaries. *arXiv: Algebraic Topology*, 2019.

Katharine Turner, Yuriy Mileyko, Sayan Mukherjee, and John Harer. Frechet means for distributions of persistence diagrams. *Discrete & Computational Geometry*, 52, 06 2012. .

A Selcuk Uluagac, Sakthi V Radhakrishnan, Cherita Corbett, Antony Baca, and Raheem Beyah. A passive technique for fingerprinting wireless devices with wired-side observations. In *2013 IEEE conference on communications and network security (CNS)*, pages 305–313. IEEE, 2013.

L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, November 1984. ISSN 0001-0782. . URL `https://doi.org/10.1145/1968.1972`.

V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971. .

Vladimir N. Vapnik. *Bounds on the Rate of Convergence of Learning Processes*, pages 69–91. Springer New York, New York, NY, 2000. ISBN 978-1-4757-3264-1. . URL https://doi.org/10.1007/978-1-4757-3264-1_4.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

HJ Van Veen. Detecting encrypted tor traffic with boosting and topological data analysis. https://kepler-mapper.scikit-tda.org/en/latest/notebooks/TOR-XGB-TDA.html, 2018. Accessed: 2021-09-15.

Petar Veličković. Theoretical foundations of graph neural networks. https://www.youtube.com/watch?v=uF53xsT7mjc, 2021. Accessed: 2021-09-27.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

J. Vidal, J. Budin, and J. Tierny. Progressive wasserstein barycenters of persistence diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):151–161, 2020.

L. Vietoris. Über den höheren zusammenhang kompakter röume und eine klasse von zusammenhangstreuen abbildungen. *Mathematische Annalen*, 97(1):454–472, December 1927. . URL https://doi.org/10.1007/bf01447877.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, December 2010. ISSN 1532-4435.

Siddharth Vishwanath, Kenji Fukumizu, Satoshi Kuriki, and Bharath K. Sriperumbudur. Robust persistence diagrams using reproducing kernels. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21900–21911. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/f99499791ad90c9c0ba9852622d0d15f-Paper.pdf.

Alexander Wagner. Nonembeddability of persistence diagrams with p¿2 wasserstein metric. *ArXiv*, abs/1910.13935, 2019.

Rui Wang, Duc Duy Nguyen, and Guo-Wei Wei. Persistent spectral graph, 2019.

Rui Wang, Rundong Zhao, Emily Ribando-Gros, Jiahui Chen, Yiying Tong, and Guo-Wei Wei. Hermes: Persistent spectral graph software. *arXiv preprint arXiv:2012.11065*, 2020.

JunJie Wee and Kelin Xia. Persistent spectral based ensemble learning (PerSpect-EL) for protein–protein binding affinity prediction. *Briefings in Bioinformatics*, 23(2), 02 2022. ISSN 1477-4054. . URL https://doi.org/10.1093/bib/bbac024. bbac024.

Jing Wei, Xuan Chu, Xiang-Yu Sun, Kun Xu, Hui-Xiong Deng, Jigen Chen, Zhongming Wei, and Ming Lei. Machine learning in materials science. *InfoMat*, 1:338–358, 09 2019. .

Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural networks: Tricks of the trade*, pages 639–655. Springer, 2012.

Daniel E Widdowson and Vitaliy A Kurlin. Recognizing rigid patterns of unlabeled point clouds by complete and continuous isometry invariants with no false negatives and no false positives. In *Computer Vision and Pattern Recognition*, 2023.

Robert M. Winding, Timothy E. Wright, and M. Chapple. System anomaly detection: Mining firewall logs. *2006 Securecomm and Workshops*, pages 1–5, 2006.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

Pengxiang Wu, Songzhu Zheng, Mayank Goswami, Dimitris Metaxas, and Chao Chen. A topological filter for learning with label noise. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21382–21393. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/f4e3ce3e7b581ff32e40968298ba013d-Paper.pdf.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. 2017.

Zuoyu Yan, Tengfei Ma, Liangcai Gao, Zhi Tang, and Chao Chen. Link prediction with persistent homology: An interactive view. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11659–11669. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/yan21b.html.

H. Yang, Q. Yao, A. Yu, Y. Lee, and J. Zhang. Resource assignment based on dynamic fuzzy clustering in elastic optical networks with multi-core fibers. *IEEE Transactions on Communications*, 67(5):3457–3469, 2019. .

Ozlem Yavanoglu and Murat Aydos. A review on cyber security datasets for machine learning algorithms. 12 2017. .

W.I. Zangwill. *Nonlinear programming: a unified approach*. Prentice-Hall international series in management. Prentice-Hall, 1969. URL https://books.google.co.uk/books?id=TWhxLcApH9sC.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.

Matthias Zeppelzauer, Bartosz Zieliński, Mateusz Juda, and Markus Seidl. Topological descriptors for 3d surface analysis. In Alexandra Bac and Jean-Luc Mari, editors, *Computational Topology in Image Context*, pages 77–87, Cham, 2016. Springer International Publishing. ISBN 978-3-319-39441-1.

Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *arXiv preprint arXiv:1803.07294*, 2018.

Xu Zhang, Yong Xu, Qingwei Lin, Bo Qiao, Hongyu Zhang, Yingnong Dang, Chunyu Xie, Xinsheng Yang, Qian Cheng, Ze Li, et al. Robust log-based anomaly detection on unstable log data. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 807–817, 2019.

Qi Zhao and Yusu Wang. Learning metrics for persistence-based summaries and applications for graph classification. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 9859–9870. Curran Associates, Inc., 2019.

Qi Zhao, Ze Ye, Chao Chen, and Yusu Wang. Persistence enhanced graph neural network. In *International Conference on Artificial Intelligence and Statistics*, pages 2896–2906. PMLR, 2020.

Sharon Zhou, Eric Zelikman, Fred Lu, Andrew Y. Ng, Gunnar E. Carlsson, and Stefano Ermon. Evaluating the disentanglement of deep generative models through manifold topology. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=djwS0m4Ft_A.

Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. 2002.

Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.

Bartosz Zieliński, Michał Lipiński, Mateusz Juda, Matthias Zeppelzauer, and Paweł Dłotko. Persistence bag-of-words for topological data analysis. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4489–4495. International Joint Conferences on Artificial Intelligence Organization, 7 2019. . URL https://doi.org/10.24963/ijcai.2019/624.

Afra Zomorodian and Gunnar Carlsson. Computing Persistent Homology. *Discrete & Computational Geometry*, 33(2):249–274, 2005. ISSN 1432-0444. . URL https://doi.org/10.1007/s00454-004-1146-y.

Afra J. Zomorodian. *Topology for Computing*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2005. .

Eduard Čech, Zdeněk Frolík, and Miroslav Katětov. *Topological spaces*. Academia, Publishing House of the Czechoslovak Academy of Sciences, 1966. URL http://eudml.org/doc/277000.

Appendix A

Convergence of the Weighted Fréchet Mean

To show convergence, recall that the Fréchet mean is computed by finding the arg min of

$$F(\hat{\mathrm{D}}) = \sum_{j=1}^{n} r_{jk}^2 F_j(\hat{\mathrm{D}}), \text{ with } F_j(\hat{\mathrm{D}}) = W_2(\hat{\mathrm{D}}, \mathrm{D}_j)^2, \tag{A.1}$$

for fixed k. We start by recounting work by Turner et al. (2012), which we shall adapt for the weighted Fréchet mean. The proofs we are adapting use a gradient descent technique to prove local convergence. In order to use their techniques, we need to define a differential structure on the space of persistence diagrams. Before we begin, note that in Turner et al. (2012), the Fréchet mean is defined as the arg min of the Fréchet function $F(\hat{\mathrm{D}}) = \int W_2(\hat{\mathrm{D}}, \mathrm{D}_j)^2 d\rho(\hat{\mathrm{D}})$ with the empirical measure $\rho = n^{-1}\sum_{j=1}^{n} \delta_{\mathrm{D}_j}$. We are using the empirical measure $\rho = \left(\sum_{j=1}^{n} r_{jk}^2\right)^{-1} \sum_{j=1}^{n} r_{jk}^2 \delta_{\mathrm{D}_j}$, but for ease we drop the scalar $\left(\sum_{j=1}^{n} r_{jk}^2\right)^{-1}$ as it is positive, so doesn't affect the minimum of the function.

Turner et al. (2012, Theorem 2.5) show that in the space of persistence diagrams $\mathcal{D}_{L^2} = \{\mathrm{D} : W_2(\mathrm{D}, \Delta) < \infty\}$ an optimal bijection $\gamma : \mathrm{D}_1 \to \mathrm{D}_2$ induces a unit-speed geodesic $\phi(t) = \{(1-t)x + t\gamma(x) : x \in \mathrm{D}_1, 0 \leq t \leq 1\}$. For a point $\mathrm{D} \in \mathcal{D}_2$ we define the tangent cone T_{D}. Define $\hat{\Sigma}_{\mathrm{D}}$ as the set of all non-trivial unit-speed geodesics emanating from D. Let $\phi, \eta \in \hat{\Sigma}_{\mathrm{D}}$ and define the angle between them as

$$\angle_{\mathrm{D}}(\phi, \eta) = \arccos\left(\lim_{s,t\downarrow 0} \frac{s^2 + t^2 - W_2(\phi(s), \eta(t))^2}{2st}\right) \in [0, \pi]$$

when the limit exists. Then the space of directions $(\Sigma_{\mathrm{D}}, \angle_{\mathrm{D}})$ is the completion of $\hat{\Sigma}_{\mathrm{D}}/\sim$ with respect to $\angle_{\mathrm{D}}$, with $\phi \sim \eta \iff \angle_{\mathrm{D}}(\phi, \eta) = 0$. We now define the tangent cone as

$$T_{\mathrm{D}} = (\Sigma_{\mathrm{D}} \times [0, \infty))/(\Sigma_{\mathrm{D}} \times \{0\}).$$

Given $u = (\phi, s), v = (\eta, t)$, we define an inner product on the tangent cone by

$$\langle u, v \rangle = st \cos \angle_{\mathrm{D}}(\phi, \eta).$$

Now, for $\alpha > 0$ denote the space $(\mathcal{D}_2, \alpha W_2)$ as $\alpha\mathcal{D}_2$ and define the map $i_\alpha : \alpha\mathcal{D}_2 \to \mathcal{D}_2$. For an open set $\Omega \subset \mathcal{D}_2$ and a function $f : \Omega \to \mathbb{R}$, the differential of f at $\mathrm{D} \in \Omega$ is defined by $d_{\mathrm{D}} f = \lim_{\alpha \to \infty} \alpha(f \circ i_{\mathrm{D}} - f(\mathrm{D}))$. Finally, we say that $s \in T_{\mathrm{D}}$ is a supporting vector of f at D if $d_{\mathrm{D}} f(x) \leq -\langle s, x \rangle$ for all $x \in T_{\mathrm{D}}$.

Lemma A.1. *The following two results are proven by Turner et al. (2012).*

(i) Let $\mathrm{D} \in \mathcal{D}_2$. Let $F_j(\hat{\mathrm{D}}) = W_2(\hat{\mathrm{D}}, \mathrm{D}_j)^2$. Then if ϕ is a distance-achieving geodesic from D to $\hat{\mathrm{D}}$, then the tangent vector to ϕ at D of length $2W_2(\hat{\mathrm{D}}, \mathrm{D})$ is a supporting vector at D of f.

(ii) If D is a local minimum of f and s is a supporting vector of f at D, then $s = 0$.

If there is a unique optimal matching $\gamma_{\mathrm{D}_1}^{\mathrm{D}_3} : \mathrm{D}_1 \to \mathrm{D}_3$, we say that it is induced by an optimal matching $\gamma_{\mathrm{D}_1}^{\mathrm{D}_2} : \mathrm{D}_1 \to \mathrm{D}_2$ if there exists a unique optimal matching $\gamma_{\mathrm{D}_2}^{\mathrm{D}_3} : \mathrm{D}_2 \to \mathrm{D}_3$ such that $\gamma_{\mathrm{D}_1}^{\mathrm{D}_3} = \gamma_{\mathrm{D}_2}^{\mathrm{D}_3} \circ \gamma_{\mathrm{D}_1}^{\mathrm{D}_2}$. (Turner et al., 2012, Proposition 3.2) prove that an optimal matching at a point is also locally optimal. In particular, they show the following.

Lemma A.2. *Let $\mathrm{D}_1, \mathrm{D}_2 \in \mathcal{D}_2$ such that there is a unique optimal matching from D_1 to D_2. Then there exists an $r > 0$ such that for every $\mathrm{D}_3 \in B_{W_2}(\mathrm{D}_2, r)$, there is a unique optimal pairing from D_2 to D_3 that is induced by the matching from D_1 to D_2.*

We are now in a position to prove Theorem 5.5, which we restate below. It proves that our algorithm converges to a local minimum of the Fréchet function.

Theorem A.3. *Given diagrams D_j, membership values r_{jk}, and the Fréchet function F defined in Equation A.1, then $\mathrm{M}_k = \{y^{(i)}\}_{i=1}^m$ is a local minimum of F if and only if there is a unique optimal pairing from M_k to each of the D_j, denoted γ_j, and each $y^{(i)}$ is updated via Equation 5.6.*

Proof. First assume that γ_j are optimal pairings from M_k to each D_j, and let s_j be the vectors in T_{M_k} that are tangent to the geodesics induced by γ_j and are distance-achieving. Then by Lemma A.1(i), each $2s_j$ is a supporting vector for the function F_j. Furthermore, $2\sum_{j=1}^n r_{jk}^2 s_j$ is a supporting vector for F, as for any $\hat{\mathrm{D}}$,

$$\begin{aligned} d_{\mathrm{M}_k} F(\hat{\mathrm{D}}) = d_{\mathrm{M}_k} \left(\sum_{j=1}^n r_{jk}^2 F_j(\hat{\mathrm{D}}) \right) &= \sum_{j=1}^n r_{jk}^2 d_{\mathrm{M}_k} F_j(\hat{\mathrm{D}}) \\ &\leq \sum_{j=1}^n -r_{jk}^2 \langle 2s_j, \hat{\mathrm{D}} \rangle = - \left\langle 2 \sum_{j=1}^n r_{jk}^2 s_j, \hat{\mathrm{D}} \right\rangle. \end{aligned}$$

By Lemma A.1(ii), $2\sum_{j=1}^{n} r_{jk}^2 s_j = 0$. Putting $s_j = \gamma_j(y^{(i)}) - y^{(i)}$ and rearranging gives that $y^{(i)}$ updates via Equation 5.6, as required. Note that when $\gamma_j(y^{(i)}) = \Delta$, we let $\gamma_j(y^{(i)}) = w_\Delta$ as defined in Equation 5.6, because this minimises the transportation cost to the diagonal. Now suppose that γ_j and $\tilde{\gamma}_j$ are both optimal pairings. Then by the above argument $\sum_{j=1}^{n} r_{jk}^2 s_j = \sum_{j=1}^{n} r_{jk}^2 \tilde{s}_j = 0$, implying that $s_j = \tilde{s}_j$ and so $\gamma_j = \tilde{\gamma}_j$. Therefore the optimal pairing is unique.

To prove the opposite direction, assume that $\mathrm{M}_k = \{y^{(i)}\}$ locally minimises the Fréchet function F. Observe that for a fixed bijection γ_j, we have that

$$\begin{aligned} F(\mathrm{M}_k) &= \sum_{j=1}^{n} r_{jk}^2 W_2(\mathrm{M}_k, D_j)^2 \\ &= \sum_{j=1}^{n} r_{jk}^2 \left(\inf_{\gamma_j:\hat{\mathrm{M}}\to \mathrm{D}_j} \sum_{y\in \mathrm{M}_k} ||y-\gamma_j(y)||^2 \right) \\ &= \sum_{j=1}^{n} r_{jk}^2 \sum_{i=1}^{m} ||y^{(i)} - x_j^{(i)}||^2 \\ &= \sum_{i=1}^{m} \left(\sum_{j=1}^{n} r_{jk}^2 ||y^{(i)} - x_j^{(i)}||^2 \right). \end{aligned}$$

The final term in brackets is non-negative, and minimised exactly when $y^{(i)}$ is updated via Equation 5.6. Furthermore, the unique optimal pairing from M_k to each of the D_j's is the same for every $\hat{\mathrm{M}}$ within the ball $B_{W_2}(\mathrm{M}_k, r)$ for some $r > 0$, by Lemma A.2. Therefore, if M_k is a local minimum of F, then the $y^{(i)}$'s are equal to the values found by taking the optimal pairings γ_j and calculating the weighted means of $\gamma_j(y^{(i)})$ with the weights r_{jk}^2, as required. It will remain a minimum as long as the matching stays the same, which happens in the ball $B_{W_2}(\mathrm{M}_k, r)$, so we are done. □

Appendix B

Additional Experimental Details

B.1 The Persistent Laplacian

B.1.1 Task details

The specifics of the MoleculeNet tasks that we evaluate in Section 4.3.2 are in Table B.1.

B.1.2 Runtimes

We report the runtimes for pre-processing (embedding data into filtrations of simplicial/cubical complexes), embeddings (computing the topological summaries), and

TABLE B.1: The QM7b tasks are as follows. Each of these is a property of the molecule being described, evaluated with different methods. For full details of the tasks please see Montavon et al. (2013).

Task ID	Evaluation Method	Description
0	PBE0	Activation energy
1	ZINDO	Excitation energy with the most absorption
2	ZINDO	Highest absorption
3	ZINDO	HOMO
4	ZINDO	LUMO
5	ZINDO	1st excitation energy
6	ZINDO	Ionization potential
7	ZINDO	Electron affinity
8	KS	HOMO
9	KS	LUMO
10	GW	HOMO
11	GW	LUMO
12	PBE	Polarisability
13	SCS	Polarisability

TABLE B.2: Time taken to pre-process the raw data, in seconds.

Pre-Processing	Time (seconds)
Height Filtration[1]	0.000376
Induced from coulomb matrix (QM7/7b)	0.000376

TABLE B.3: Time taken to embed the pre-processed data, in seconds.

Embedding	Time (seconds)
Persistent Homology ($p = 0, 1$)[2]	0.000231
Cubical Homology ($p = 0, 1$)[1]	0.000231
Graph Laplacian	0.0000292
Combinatorial Laplacian	0.000382
Persistent Laplacian ($p = 0, 1$, simplicial complex pair)	0.000380
Persistent Laplacian ($p = 0, 1$, cubical complex pair)	1.855

TABLE B.4: Time taken to post-process the embedded data, in seconds.

Post-Processing	Time (seconds)
MNIST Baseline Embedding[1] (Garin and Tauzin, 2019)	0.000376
Persistence Images[3]	0.00188
Persistent Laplacian Vectorisation (Section 4.2.2)	0.00330

post-processing (mapping the topological summaries into a persistence diagram). Note that each of these timing experiments are run on one core of a 2021 MacBook Pro with an M1 Pro chip. We report the time in seconds. Note that in these tables we reference others' implementation; if there is no reference then it is our implementation. We repeat each timing experiment 100 times. The error bounds are typically in the order of 10^{-5}, so for readability we do not report them. Note that the time taken for the computation of the persistent Laplacian for one simplicial complex pair is of the same order of magnitude as the persistent Laplacian. As you increase R, and thus the number of persistent Laplacians computed increases, the time taken will similarly increase. The time taken for the cubical complex persistent Laplacian is several orders of magnitude larger - this is due to our implementation of the cubical boundary.

As we used the Vietoris-Rips persistence function in Giotto-TDA (Tauzin et al., 2020) for the QM7 3d coordinate experiments we cannot disentangle the time taken for preprocessing and computing the topological representation. We report the total time to compute the Vietoris-Rips persistent homology of one set of 3d coordinates from QM7.

[1] Using Giotto-TDA (Tauzin et al., 2020).
[2] Using Dionysus 2.
[3] Using Persim.

TABLE B.5: Time taken to compute Vietoris-Rips persistence, in seconds.

Post-Processing	Time (seconds)
Vietoris-Rips Persistence ($p = 0, 1$) [1]	0.000578
Vietoris-Rips Persistence ($p = 0, 1, 2$) [1]	0.000912

Note that this is an extremely well-optimised package that has been under development for several years - the persistent Laplacian and our implementation are recent methods that haven't been subjected to the same levels of research and optimisation.

B.2 Fuzzy c-Means Clustering for Persistence Diagrams

For models, we trained the standard Pytorch CNN. We trained them on MNIST, FashionMNIST, and KMNIST, each obtained using `torchvision.datasets`. We split the data into 9 binary datasets for classification, class 0 vs each of the remaining classes. We trained three of each model, seeded with 0, 1, and 2 respectively. MNIST and KMNIST were each trained for five epochs, FashionMNIST was trained for 14 epochs. Our train:test split was 6:1, as is standard for MNIST structured datasets. We used Ripser (Bauer, 2019) to compute the 1-persistence diagrams using default settings. We limited the number of points in the diagram to the 25 most persistent when clustering. Our percentage improvement values use the membership values after 16 iterations. We compute the standard error bounds when calculating the percentage improvement.

Figure B.1 displays an example of our fuzzy clustering algorithm working on the surface of a sphere.

B.3 Structural Representations of Host-Based Logs

Table B.6 displays the effect of adding induced 2-simplices to the simplicial complexes; we concluded that the effect was negligible. This, coupled with the fact that adding induced 2-simplices is very computationally intensive, meant that we generally did not add induced 2-simplices when running our experiments. Table B.7 investigates the effect of persistence image pixel size on the the performance of the classifier; once again the effect was negligible. Following standard procedure, we map points at infinity in the diagrams to a hyperparameter that is larger than any death value across all persistence diagrams in our dataset when mapping persistence diagrams into persistence images.

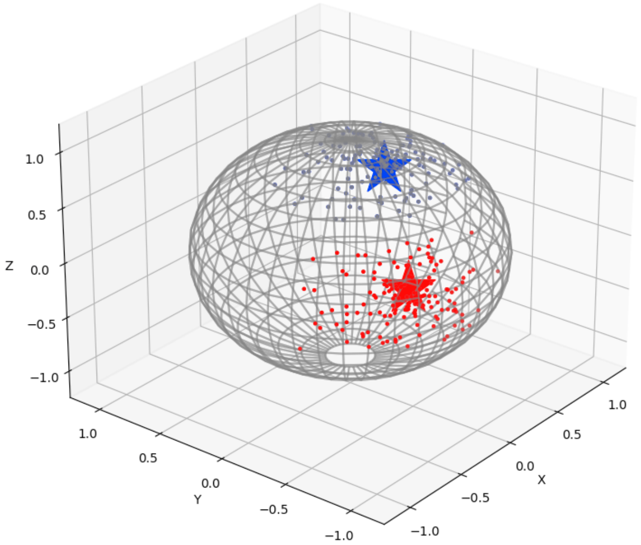


FIGURE B.1: We implemented our algorithm for manifolds using the Geomstats package (Miolane et al., 2020). In this figure we fuzzy cluster points on the surface of a sphere.

TABLE B.6: Investigation of adding induced 2-simplices, using the malware dataset with low acuity and 0/1-persistence diagrams. We found that there was no real change when adding induced 2-simplices. We hypothesise this is because most holes were never destroyed, even when adding the induced 2-simplices. As this was computationally costly we did not add induced 2-simplices in the rest of the experiments.

	No 2-simplices	Induced 2-simplices
Accuracy	66.20 ± 1.65	66.14 ± 1.49
Precision	68.85 ± 1.77	68.56 ± 1.54
Recall	64.58 ± 1.72	64.34 ± 1.98
F1	66.56 ± 1.59	66.34 ± 1.61

We use scikit-learn's RandomForestClassifier (Pedregosa et al., 2011) with default settings - the specific parameters are shown in Table B.8. We train and score using scikit-learn's `cross_validate` over 10 random folds.

TABLE B.7: Investigation of persistence image pixel size effect on performance of classifier using the malware dataset with low acuity and 0/1-persistence diagrams. We found that there was no real change when using different sizes of pixels. Based on this we selected a pixel size of 20 as it offered a compromise between resolution and the dimensionality of the persistence image vector.

Pixel size	Accuracy	Precision	Recall	F1
5	63.95 ± 2.06	66.15 ± 2.05	63.37 ± 2.54	64.59 ± 2.11
10	65.07 ± 2.19	67.10 ± 2.19	64.81 ± 2.59	65.82 ± 2.24
15	65.58 ± 1.89	67.53 ± 1.98	65.77 ± 2.10	66.54 ± 1.85
20	64.26 ± 2.22	66.31 ± 2.12	63.37 ± 3.04	64.68 ± 2.48
50	64.51 ± 1.87	66.54 ± 1.78	64.10 ± 2.23	65.23 ± 1.94
100	61.96 ± 1.77	63.83 ± 1.62	61.93 ± 2.75	62.71 ± 2.05

TABLE B.8: Parameters for the scikit-learn random forest classifier used for all experiments.

Parameter	Value
bootstrap	True
ccp_alph	0.0
class_weight	None
criterion	'gini'
max_depth	None
max_features	'auto'
max_leaf_nodes	None
max_samples	None
min_impurity_decrease	0.0
min_samples_leaf	1
min_samples_split	2
min_weight_fraction_leaf	0.0
n_estimators	100
n_jobs	None
oob_score	False
random_state	0
verbose	0
warm_start	False

Appendix C

Cyber Security Datasets

We provide a list of available datasets that may be of interest to machine learning practioners looking to work with cyber security data.

- **KDD Cup 1999** provides TCP connections, content features suggested by domain knowledge, and traffic features, along with labels for anomaly detection (Lippmann et al., 2000). It is a very popular dataset for anomaly detection (although fairly outdated these days) and is available online on their website. It is based on the data captured for the DARPA'98 IDS evaluation program.
- **ECML-PKDD 2007** provides requests to web servers and was designed for identifying web attacks with machine learning (Dray et al., 2007). It is real-world data that has been manually processed and is available on their GitHub.
- **CSIC 2010** is another dataset for web attacks and is based off real data from an e-commerce application. It is also available on their website.
- **ISOT Botnet and Ransomware** provides DNS queries for 9 botnets plus 19 malign applications, and 420GB of ransomware and benign program execution traces. It is available on their website.
- **CTU-13** consists of network logs for botnets, verified normal hosts, and unverified background noise (Garcia et al., 2014). It is available on their website.
- **ADFA** was compiled in 2013 to update intrusion detection datasets like KDD Cup that were over a decade old. The data consists of labelled system calls and is split into Linux, Windows, and Windows stealth attacks. It is available on their website. Although I haven't verified this, a labelled version of the Linux dataset appears to be available on GitHub.
- **UNSW-NB15** is a comprehensive dataset that covers a variety of different attacks and is aimed at training and evaluating intrusion detection systems (Moustafa et al., 2019). It is available on their website.

- **DARPA Operationally Transparent Cyber** (OpTC) is a dataset of event logs from 1000 hosts that have a period of baseline activity, followed by having malware injected by a red team. It is available on GitHub.
- **LANL ARCS**, the Los Alamos National Lab Advanced Research in Cyber Systems dataset, consists of three separate datasets and is available on their website. The datasets are
  - (i) Comprehensive, Multi-Source Cyber-Security Events, consisting of DNS and event data from across the LANL internal network, including labelled red team anomalies;
  - (ii) Unified Host and Network Data Set, consisting of network data from the LANL internal network;
  - (iii) User-Computer Authentication Associations in Time, which consists of 700 million successful authentication events from the LANL enterprise network.
- **BETH** is a recent anomaly detection dataset created by collecting over eight million DNS logs and kernel level events from 23 honeypots (Highnam et al., 2021). The data is real-life, contemporary and provided in a consistent format, and is available on Kaggle.

Some of these datasets are described in a review paper by Yavanoglu and Aydos (2017).