2022

Topological Hierarchies and Decomposition: From Clustering to Persistence

Kyle A. Brown
Wright State University

Topological Hierarchies and Decompositions: from Clustering to Persistence

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

by

KYLE A. BROWN
B.S. in Computer Science, Wright State University, 2016
B.S. in Mathematics, Wright State University, 2016
M.S. in Computer Science, Wright State University, 2019

2022
Wright State University

WRIGHT STATE UNIVERSITY
GRADUATE SCHOOL

April 22, 2022

I HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER MY SUPERVISION BY Kyle A. Brown ENTITLED Topological Hierarchies and Decompositions: from Clustering to Persistence BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Doctor of Philosophy.

Derek Doran, Ph.D.
Dissertation Director

Yong Pei, Ph.D.
Director, Ph.D. in
Computer Science and Engineering Program

Barry Milligan, Ph.D.
Vice Provost for Academic Affairs
Dean of the Graduate School

Committee on
Final Examination

Derek Doran, Ph.D.

Michael Raymer, Ph.D.

Thomas Wischgoll, Ph.D.

Nikolaos Bourbakis, Ph.D.

Vincent Schmidt, Ph.D.

ABSTRACT

Brown, Kyle A. Ph.D., Department of Computer Science and Engineering, Wright State University, 2022. *Topological Hierarchies and Decompositions: from Clustering to Persistence.*

Hierarchical clustering is a class of algorithms commonly used in exploratory data analysis (EDA) and supervised learning. However, they suffer from some drawbacks, including the difficulty of interpreting the resulting dendrogram, arbitrariness in the choice of cut to obtain a flat clustering, and the lack of an obvious way of comparing individual clusters. In this disseration, we develop the notion of a topological hierarchy on recursively-defined subsets of a metric space. We look to the field of topological data analysis (TDA) for the mathematical background to associate topological structures such as simplicial complexes and maps of covers to clusters in a hierarchy.

Our main results include the definition of a novel hierarchical algorithm for constructing a topological hierarchy, and an implementation of the MAPPER algorithm and our topological hierarchies in pure Python code as well as a web app dashboard for exploratory data analysis. We show that the algorithm scales well to high-dimensional data due to the use of dimensionality reduction in most TDA methods, and analyze the worst-case time complexity of MAPPER and our hierarchical decomposition algorithm. Finally, we give a use case for exploratory data analysis with our techniques.

Contents

List of Figures

Acknowledgment

I would like to take this opportunity to extend my thanks to my advisor, Doctor Doran, for the years of mentoring and being the one who first gave me the opportunity and funding to attend graduate school. I'd also like to thank the committee members for their time spent in attending the defense and reviewing my dissertation. Further, I'm thankful to Doctor Kramer for funding most of my graduate years and introducing me to TDA, as well as opportunities for internships involving TDA. Finally, I'd like to thank Matthew Piekenbrock for multiple illuminating discussions on TDA, hierarchical clustering, statistics, and mathematics in general.

Further thanks go out to my mother, for supporting me over the years. I love you mom. Last but not least, I'd like to thank my cats for being a source of unconditional love.

To Science.

Introduction

Clustering is a common task in data analysis where one seeks to partition a collection of points into a set of disjoint groups [62]. The grouping is done based on some notion of similarity or distance between points. The resulting clustering *partitions* the points by placing each one into exactly one cluster. There are many existing approaches to clustering: k-means, single-linkage, complete-linkage, average-linkage, spectral clustering, and density-based approaches such as DBSCAN to name a few. Hierarchical clustering is a particular class of clustering techniques, which produces a hierarchy of clusterings ranging from one where all points are in one group to another where each point is its own group [52]. This hierarchy can be represented as a dendrogram, which is a tree where the nodes at each level represent a clustering and the branches describe the merging or splitting of clusters. Traditional hierarchical clusterings produce a binary tree as they either merge or split clusters.

Despite its applications to data analysis, clustering has some limitations. The output of a non-hierarchical clustering is a collection of groups of points with no further information. In particular, the shapes of the clusters and their relationships are left opaque. These algorithms may also depend on threshold parameters which are chosen arbitrarily or based on an informal heuristic. Clustering can also lack robustness with respect to noise or changes in the data. With hierarchical clustering, one has a notion of relationships between clusters at different levels of the dendrogram as they merge or split. However, what is still lacking is a way to look into the features within the clusters and relate them to each other. Therefore, I will look to topological data analysis (TDA) as a way to (i) identify the shape of the data and its clusters, and (ii) provide more robustness with respect to noise and the choice of parameters.

Topological data analysis is a growing field which applies techniques from topology, a branch of mathematics, to the analysis of point-cloud data [11]. There are many techniques in TDA, but they tend to share some features; they provide qualitative information about the data, they are less dependent on a metric or the coordinate system of the data, and

they give summaries of different aspects of the data. Much of TDA focuses on *persistent homology*, which is an algebraic technique for identifying topological features of shapes and functions [26]. It works for incomplete or noisy point-cloud data as well, as it picks out features which persist across several summaries of the data as a parameter is varied. This lends it a robustness absent from other approaches, as its emphasis on qualitative features allows it to ignore small continuous variations introduced by noise. Traditionally, persistent homology is done on a filtration of simplicial complexes constructed via either the Vietoris-Rips or Čech complexes. The MAPPER algorithm produces a simplicial complex summary of point-cloud data viewed through the lens of a continuous function [84]. For some recent overviews of TDA, see [16] and [97].

In this dissertation, I introduce formally the concept of a *topological hierarchy* and describe basic notions of persistence on a topological hierarchy. A topological hierarchy can be considered a hierarchical structure, or tree, where each node represents a subset of some topological space, being a subspace of its parent. This tree is a kind of dendrogram, with branches representing further partitioning of a topological space into subspaces. I include further information about each space at the node, namely a cover of the space and an associated simplicial complex. For each node in the topological hierarchy, there is a relationship from the associated cover and simplicial complex to those of its parent. Furthermore, I show how one can define persistent homology on topological hierarchies, which allows one to relate topological features in the subspace to features in the parent spaces. To showcase the power of topological hierarchies for exploratory data analysis (EDA), I apply them and the topological hierarchical decomposition (TDA) algorithm to the FICO dataset of home equity line of credit (HELOC) loan applications [33] to provide insights onto the financial reasons an applicant was not able to pay back their loan. This shows how THDs can be considered an explainable AI approach, by providing reasoning for why a black box algorithm predicted that applicants would not be able to pay back their loans.

1.1 Outline of the Dissertation

The rest of the dissertation is structured as follows. Chapter 2 reviews existing literature on TDA, hierarchical clustering, and applications of TDA. Section 2.1 reviews the papers and articles that define the mathematical background and algorithms of TDA, namely those that cover MAPPER (Subsection 2.1.1), persistent homology (Subsection 2.1.2), barcodes and persistence diagrams (Subsection 2.1.2), zigzag persistence (Subsection 2.1.3), and multiscale MAPPER (Subsection 2.1.4). Section 2.2 reviews a few papers describing hierarchical and density-based clustering, as well as the dendrogram formalism. Section 2.3 describes various applications of TDA within multiple fields.

Chapter 3 provides the necessary mathematical background, focusing on point-set topology and homology. Section 3.1 covers point-set topology: topological spaces, closed sets, neighborhoods, and separability. Subsection 3.1.1 describes continuous maps and homeomorphisms, and Subsection 3.1.2 open covers and compactness. Subsection 3.1.3 covers the definition of a topological basis, and Subsection 3.1.4 connectedness and paths. Finally, Subsection 3.1.5 introduces metric spaces. Section 3.2 introduces a mathematical description of clustering. Section 3.3 covers some basic topological structures used in TDA: pullback covers, simplicial complexes, nerves, and MAPPER. Section 3.4 describes homology and persistence; Subsection 3.4.1 reviews definitions from algebra used in defining homology. Subsection 3.4.2 defines simplicial homology on abstract simplicial complexes. Finally, Subsection 3.4.4 describes zigzag persistence, a generalization of persistent homology.

The core of the dissertation is in Chapter 4, which introduces Topological Hierarchies and examines their mathematical and algorithmic properties. Section 4.1 discusses preliminary material used in the definition of hierarchies, namely subspace topologies, maps of covers, and nerve-like maps. The definition of a topological hierarchy is given in Section 4.2, along with examples. Subsection 4.2.1 describes how to compute a topological hierarchy on an existing indexed hierarchy with a tower of covers. Subsection 4.2.2 in-

troduces the topological hierarchical decomposition algorithm. Three methods of defining persistence on a topological hierarchy are given in Section 4.3. Section 4.4 looks at evaluating topological hierarchies as a clustering, and compares THD to hierarchical clustering on the HELOC dataset. Section 4.5 looks at the algorithmic aspects of topological hierarchies, including potential optimizations and runtime performance. Finally, Subsection 4.5.1 examines the computational complexity of MAPPER, clustering, and THDs. It also includes remarks on the space complexity of those algorithms.

Chapter 5 describes a Python library implementing MAPPER and the THD algorithm, and a dashboard web application that uses this library. Section 5.1 exhibits the results of experiments to evaluate the empirical runtime performance of the Python library. Section 5.2 describes the interface of the dashboard., and Section 5.3 compares the results of the dashboard on the FICO dataset with a previous study on the same dataset using the Ayasdi Platform. Chapter 6 describes an application of topological hierarchies and THD to a real world dataset, namely the HELOC FICO loan applicant dataset. Subsection 6.1.1 gives needed financial glossary to understand the dataset. Subsection 6.1.2 describes the FICO dataset and Subsection 6.1.3 covers the results of THD applied to the FICO dataset. Finally, Subsection 6.1.3 compares the methodology to supervised learning models.

Literature Review

This section surveys the academic literature related to the topic of the dissertation by focusing on three themes: topological structures, hierarchies, and applications of topological data analysis. The topological structures of concern are constructions of simplicial complexes from a topological space, simplicial homology, and persistent homology. Hierarchical structures are related to clusterings, especially hierarchical clustering, which produces a binary tree structure known as a dendrogram. The topological constructions covered are either directly related to the topic of the dissertation, or provide useful background for it. Hierarchical structures that arise from clustering lead to evaluation metrics that can be applied to topological hierarchies to measure their usefulness in partitioning a space. Finally, applications of topological data analysis show how the abstract definitions lead to useful approaches in making sense of data. Each paper discussed is given a recap, followed by a discussion of how it relates to the topic of the dissertation.

2.1 Topological Structures

This section covers literature on TDA and related topics. Many papers build on the original article by Edelsbrunner *et al.* that introduces persistent homology [28], and the article by Zomorodian and Carlsson that extends this definition [101]. This is the setting in which the well-known barcode and persistence diagram representations are defined. Zigzag persistence can be seen as a further generalization of persistent homology to sequences with inclusions going in either direction. Multiscale MAPPER is a synthesis of the original MAPPER construction and persistent homology, providing an environment in which the persistence of MAPPER and related structures such as Čech filtrations can be studied.

For more information on TDA, there are several surveys on the subject in the literature. Carlsson's 2009 paper [11] on the subject can be seen as an introduction to the subject, although it predates much of the important developments in the subject throughout the decade following it. Zomorodian's survey from 2012 covers simplicial complexes,

combinatorial constructions of complexes such as Čech complexes, topological invariants, simplicial homology, filtrations, persistent homology, zigzag persistence, reductions, and simplicial sets among other topics [100]. Munch in 2017 gives an overview of persistent homology and MAPPER [66]. Wasserman in 2018 covers density-based clustering, mode clustering, Morse theory, manifold learning, persistent homology, and applications [97].

2.1.1 MAPPER

The MAPPER algorithm is introduced by Singh *et al.* in [84]. The approach is motivated by the nerve of an open cover, which constructs a simplicial complex from the cover. In particular, the MAPPER construct is defined as the nerve of a pullback cover under some continuous mapping. There is a short discussion of multiresolution structure, and how a sequence of maps of covers leading to a sequence of simplicial maps is preserved under the pullback operation. The authors give a few specific examples of the MAPPER construction for simple spaces. The core content of the paper concerns a statistical implementation of MAPPER on point cloud data via the usage of single-linkage clustering. The authors define a heuristic to obtain a flat clustering, choosing a number of clusters corresponding to the first empty bin of the histogram of edge lengths, where the edge length is computed from the dendrogram. Applying their MAPPER algorithm to a point cloud on a unit sphere, they obtain the correct Betti numbers for the unit sphere using the homology detection software PLEX [82].

Next, Singh *et al.* discuss some interesting filter functions: kernel density estimates, eccentricity, and graph Laplacians, and give some applications of MAPPER. The first application is to the Miller-Reaven diabetes study, where they obtain a simplicial complex with flares corresponding to those in the 3-dimensional projection computed by the original authors. The next one shows that it is possible to reconstruct some aspects of the topology of the torus with MAPPER. The third application is to models from a 3D shape database, using MAPPER to simplify the shapes and then comparing the simplified shapes

to the original shapes and to each other.

The MAPPER algorithm is of central importance to this dissertation. Although topological hierarchies are general enough to cover other constructions such as the Vietoris-Rips complex, the dissertation focuses on the use of MAPPER for building hierarchies. The complexity analysis of THD is given in terms of MAPPER in Section 4.5.1. Section 5 concerns a Python library that implements MAPPER and topological hierarchical decomposition using this implementation of MAPPER. Therefore, an understanding of the MAPPER algorithm is crucial in the sequel.

2.1.2 Persistent Homology

This subsection covers the papers that introduce persistent homology and related notions such as barcodes. For more information on persistent homology, there have been several surveys done on the subject. The first by Edelsbrunner and Harer in 2008 covers Morse functions, tame functions, Smith normal form, bottleneck distance, and homlogical constructions that are related to persistent homology [26]. Another survey by Edelsbrunner and Morozov describes persistence diagrams, bottleneck distance, algorithms, and applications of persistent homology [29]. A more recent survey (2016) by Kerber covers quivers, barcodes, zigzag persistence, multidimensional persistence, and statistical topological data analysis [53]. Ferri gives a quick introduction to persistence followed by a discussion of applications [32]. Further applications of persistence and TDA are discussed in Section 2.3.

Persistent homology first arose in the problem of topological simplification in computer graphics and geometric modeling. Edelsbrunner *et al.* introduce persistence via filtrations in [28]. They construct simplicial complexes via the alpha complex structure, based on Voronoi regions of the union of a finite number of open balls. A filtration is a growing sequence of complexes, each which is a subcomplex of the next. The authors construct these by growing the spheres and studying the dual complex (Delaunay triangulation) of the Voronoi diagrams. They describe an algorithm for incrementally computing the Betti

numbers of such a filtration, which proceeds by identifying when a k-cycle is created or destroyed and incrementing or decrementing β_k correspondingly.

Edelsbrunner *et al.* define persistence in terms of the cycle and boundary groups and define the p^{th} persistent Betti numbers. They give an abstract algorithm for computing persistence by keeping track of the birth and death of k-cycles along a consistent basis across homology groups. They introduce a visualization involving intervals and triangles which is related to the later barcode and persistence diagram descriptions of persistence. Finally, they apply persistence to the problem of topological simplification and conclude with experimental results.

Zomorodian and Carlsson extend the definition of persistence to arbitrary d-dimensional filtered complexes [101]. They introduce the notion of a persistence module, which arises from the homology of a sequence of filtered complexes. Unlike Edelsbrunner *et al.*, they investigate persistence modules over an arbitrary ring R in the abstract. They show that for F a field, a graded $F[t]$-module can be associated to the direct sum of a set of $\mathcal{P}$-intervals (i.e. the birth and death times) via a bijection. They describe an algorithm for computing persistent homology over a field. This algorithm constructs the decomposition into sums of intervals directly without the need to compute the persistence module first. They also introduce algorithms for persistent homology when the coefficients are in a ring. They show that when the ring is a principal ideal domain, the persistent homology groups are computable in the same time as homology groups. Zomorodian and Carlsson look at persistent homology on the Klein bottle, and show the issues encountered when using $\mathbb{Z}_2$ as coefficients. Finally, they examine the application of their methods to higher-dimensional datasets, such as electric field values in 4-dimensional spacetime and the flow of air currents around a jet for varying velocities, using 1-cycles to identify vortices.

Persistent homology is used in the discussion of persistence on topological hierarchies in Section 4.3. Two of the notions of persistence described there can be described with persistent homology: the "group-to-root" persistence of a path in a hierarchy and the global

persistence of the hierarchy. Outside of this, persistent homology is not discussed, but it is a very important technique in TDA and could be an interesting direction for future investigation of topological hierarchies. In particular, two possible directions are other definitions of persistence on a hierarchy, and using persistent homology to compare and/or evaluate hierarchies.

Barcodes and Persistence Diagrams

Barcodes are introduced by Carlsson *et al.* in [14]. Much of the paper concerns constructions on differential manifolds which are out of the scope of this dissertation. Barcodes are defined as a finite multiset of intervals that are bounded below. These intervals describe the birth and death of topological features in a filtration. They define a pseudo-metric on the set of barcodes as the minimum distance between two barcodes over all possible matchings between them, and discuss algorithms for computing the optimal matching for this pseudo-metric. They discuss how to compare topological objects using barcodes, which can provide a measure of topological features that is robust to noise.

The diagrams of [28] are studied further by Cohen-Steiner *et al.* in [18]. They consider real-valued functions on a topological space which have a finite number of homological critical values, which are values at which the homology of inverse images of the half-open interval $(-\infty, a]$ changes. A persistence diagram is defined as ordered pairs of these homological critical values counted with multiplicity. This is related to Morse theory: if a function is a Morse function on a smooth manifold, then its homological critical values are its classical critical values.

Cohen-Steiner *et al.* investigate the stability of persistence diagrams under the Hausdorff distance and the bottleneck distance. They state and prove their main theorem: that the bottleneck distance of persistence diagrams is bounded by the Hausdorff distance of the functions they represent. They cover these applications: estimating the homology groups of a metric space from an incomplete sample and comparing shapes via the persistence

diagram as a stable signature. They show how their stability theorem leads to stability results specific to these cases, guaranteeing e.g. for a given sampling of a metric space with conditions on the distances the correctness of the reconstructed homology groups.

While barcodes and persistence diagrams are not used directly in the dissertation, they are important tools for the visualization and comparison of persistence modules. They relate intuitively to the formation and dissolution of topological features throughout a filtration, and also appear notably in the stability theorems for persistent homology, which are stated in terms of the bottleneck distance. In any future investigations of topological hierarchies, it would be worthwhile to investigate whether such stability results are enjoyed by any kind of topological hierarchy.

2.1.3 Zigzag Persistence

Zigzag persistence is first introduced by Carlsson and de Silva in [12]. They give needed background and persistence and consider situations where zigzag diagrams may arise in practice. The first situation is the relationships among the densest $p\%$ of a point-cloud measured according to different parameter values. The second example is topological bootstrapping, taking samples of a space and combining them to reconstruct topological features of the whole space. The third example is witness complexes for different combinations of landmark subsets, which yields a witness bicomplex zigzag structure. The properties of zigzag diagrams will entail properties that are comparable to those of persistence modules and diagrams.

Then they introduce zigzag modules proper, which are defined as sequences of vector spaces with linear maps between them going in either direction, unlike a persistence module where the maps all go one direction. To study decompositions of zigzag diagrams, Carlsson and de Silva define submodules and direct sums of zigzag diagrams, and state the result that every zigzag module has a Remak decomposition in terms of indecomposable zigzag modules, although this decomposition may not be unique, but it is unique up to a reordering

of the summands. They then introduce interval τ-modules, as zigzag modules with specific birth and death times whose vector spaces are all 0 or the field the vector spaces are over. They show that these intervals are indecomposable, and that any zigzag module can be written as a direct sum of intervals. The zigzag persistence of a zigzag module is defined to be the multiset of intervals containing the birth and death times from the decomposition of the module into interval τ-modules.

Next, Carlsson and de Silva study zigzag modules as filtrations. They define the right-filtration on a zigzag module, and show that the right filtration of an interval τ-module has a nice form in terms of the zero vector space and field K. They study the decomposition of right filtrations of vector spaces independently of zigzag persistence, and obtain an equivalent decomposition theorem into a direct sum of intervals. They describe algorithms for determining the indecomposable factors of a τ-module. They give an abstract algorithm for arbitrary vector spaces, and a concrete one where bases have been chosen and matrices are used to describe the transformations. They go on to describe further algebraic techniques in the context of zigzag persistence and modules.

The most useful application of zigzag persistence in this dissertation is in the definition of ancestor modules in Section 4.3. Unlike the other definitions of persistence on a hierarchy which are over a single branch or the entire hierarchy, these involve two branches of the hierarchy. This could provide useful topological information about features that are shared between the two branches, as well as features that are unique to each branch. Furthermore, it may be worthwhile to try and extend the definition of zigzag persistence to be able to cover three or more branches of a hierarchy at once, or even the whole hierarchy. This could provide a "local view" of the entire hierarchy at once, as opposed to the global view obtained by taking the disjoint sum of each level of the hierarchy.

2.1.4 Multiscale MAPPER

Dey, Memoli, and Wang introduce multi-scale MAPPER, which extends the MAPPER construction of [84] to a tower of covers and allows the recovery of persistent homology from it [24]. They begin by reviewing facts about topological spaces and simplicial complexes, including the important notion of a simplicial map between simplicial complexes. They relate the MAPPER construction to merge trees and Reeb graphs, which are viewed as special cases of the construction. Next, they cover maps of covers, and relate them to simplicial maps through the nerve construction, as well as showing that the induced simplicial maps by the nerve are contiguous.

Next, they introduce the core concept behind multiscale MAPPER: towers, which are sequences of objects indexed by a resolution with maps that allow passing from one resolution to another. Dey *et al.* describe a tower of covers as a special case of a tower where the objects are finite open covers and the maps are maps of covers. Similarly, they define a tower of simplicial complexes with simplicial maps between them. They then define the multiscale MAPPER as the tower of simplicial complexes obtained by applying the nerve of the pullback operation to a tower of covers. Passing to homology, they show that the resulting tower of vector spaces is a persistent module and that the techniques of persistent homology can be applied to it.

Dey *et al.* investigate the stability of multiscale MAPPER. They introduce notions of *interleaving* for towers of covers and simplicial complexes which mirror that of [15] and use these to prove the stability of multiscale MAPPER in the bottleneck distance of [18]. They show that if two towers of covers are η-interleaved, then the corresponding multiscale MAPPERs are η-interleaved and the bottleneck distance of their persistence diagrams is at most η. They then prove a stability theorem for multiscale MAPPER under perturbation of (filter) functions, assuming the towers of covers are of a special class satisfying nice properties and the filter space is a compact connected metric space. Combining these two results, they prove a stability result for multiscale MAPPER in the general case, a simultaneous

change in the tower of covers and filter function.

They turn to the computation of multiscale MAPPER from a piecewise-linear function defined on the vertices of a simplicial complex (and interpolated from these values for higher-order simplices). Dey *et al.* show that if the function satisfies a minimum diameter assumption, it is possible to compute MAPPER and multiscale MAPPER from just the 1-skeleton of the simplicial complex. They give a definition of isomorphism for multiscale MAPPERS and show that under the minimum diameter assumption on the filter function, the multiscale MAPPER resulting from the filter function on the whole complex is isomorphic to the one resulting from the function restricted to the vertices of the complex. They conclude the section by extending this result to real-valued functions on triangulable topological spaces which are approximated by piecewise-linear functions on a simplicial complex.

Dey *et al.* extend the discussion of the previous paragraph to functions mapping to an arbitrary compact metric space. They define a combinatorial version of mapper where the connected components in the pullback only consist of vertices in the underlying simplicial complex. They show that the multiscale MAPPER obtained combinatorially interleaves in a weak sense with the full multiscale MAPPER, and therefore the bottleneck distance between their persistence diagrams is bounded. They describe how the pullback can be used to induce a pseudo-metric, and how it relates to Čech filtrations in particular, in that there is an interleaving of multiscale MAPPER with the Čech filtration.

Multiscale MAPPER is considered here as a structure which can be compared to topological hierarchies. Namely, a multiscale MAPPER can be considered as a kind of hierarchy where every group is the same, and there is therefore no splitting. This leads to nice properties that the authors exploit to define persistence and prove a stability result for multiscale MAPPER. Allowing splitting and for some groups to be subsets of others leads to hierarchies, and the same nice properties no longer hold. Finally, some of the definitions used in the paper on multiscale MAPPER, namely towers of covers, are useful in the dis-

cussion of topological hierarchies. Many of the ideas behind topological hierarchies are motivated by the constructions used to define multiscale MAPPER.

2.2 Hierarchical Structures

The result of hierarchical clustering algorithms is a dendrogram in which branches represent splits of the data into smaller subsets until each cluster has only one point [80]. Much of the work in this area has been towards finding the optimal point to cut the dendrogram, to obtain the best possible flat clustering. However, the dendrogram can be a useful object on its own, which served to motivate the definition of a topological hierarchy and the THD algorithm. A topological hierarchy is not necessarily produced by a clustering algorithm; rather, I define it by extra structure on top of an existing hierarchy. My approach also embeds extra topological information at each node, relating the children to their parents in a way that enables the computation of topological features via persistence and qualitative comparisons between groups in the hierarchy.

Hartigan investigates the consistency of single-linkage clustering with respect to high-density clusters [46]. Given a probability density p on a metric space with points x and some threshold $c \geq 0$, the high-density clusters are maximal connected sets of the form $\{x|p(x) \geq c\}$. These are also called population clusters or *level sets*. Hartigan points out that the collection of level sets over all levels c has a tree structure – this is the *cluster tree*. He goes on to prove that single-linkage clustering is fully consistent in one-dimension, but in two or more dimensions is only fractionally consistent. A weaker but more intuitive conclusion is that if two level sets are separated by a valley of low enough probability, then they will be asymptotically separated into distinct clusters by single-linkage. Note that the cluster tree is not easily computable in practice, as it requires information about a probability distribution that one only has a finite sample of. In comparison, a topological hierarchy is computable for any finite metric space, but doesn't represent the same kind

of structure a cluster tree does, i.e. there is no assumption that an underlying probability density even exists, let alone what form it may take.

Stuetzle describes an algorithm for estimating the cluster tree of a density from a sample [87]. To do so, he first considers nearest neighbor distance estimation, for which the set of level sets has a nice form as the union of open balls centered at the observations. He cites the result that by breaking the longest edge of the minimum spanning tree (MST) recursively, one obtains the dendrogram of single-linkage clustering [39], and concludes that the nearest neighbor density estimate and single-linkage clustering are isomorphic. He describes his approach in terms of *runt pruning* as described by Hartigan and Mohanty [47], based on the runt size of a dendrogram node, which is the lesser of the number of leaves of the two subtrees under the node. Runt pruning only considers splitting a high-density cluster into two connected components if the runt size of the corresponding dendrogram node is larger than some threshold. Stuetzle describes an algorithm for runt pruning and then goes on to give results comparing runt pruning-based clustering with other approaches on generated and real-world data. While his approach improves on single-linkage clustering, runt pruning is still in the realm of hierachical and density-based clustering. Therefore, it has the benefits of being able to analyze its properties, while lacking the extra structure topological hierarchies add that contain information about relationships within a cluster.

Topological hierarchies have many features that are not shared with cluster trees and dendrograms. Their method of construction does not rely entirely on the metric structure of the points, but rather the topological structure as viewed through the simplicial complexes constructed on each group. In the case of a MAPPER-based topological hierarchy, the filter function can significantly impact how the data is formed into a simplicial complex, producing structures much different than the density-based cluster tree. Cluster trees and dendrograms are binary trees, while topological hierarchies are not required to be binary.

2.3 Applications of TDA

This section describes studies which apply TDA to real-world data or a standard machine learning or data analysis dataset, rather than developing new topological methods. The emphasis is on applications of MAPPER and similar simplicial complex constructions such as Vietoris-Rips complexes, as well as persistent homology. Each application described in detail is of a distinct field, such as bioinformatics, aviation, or finance. This is not a comprehensive review; there are several articles applying TDA which are not described in detail here. These include applications to biology [89, 63], chemical engineering [86], computer vision [9], cosmology [99, 49], epidemiology [88, 60, 17], finance [43, 38], manufacturing [44], neuroscience [85, 79, 96], ransomware detection [2], quantum chemistry [71], and social networks [4], among others.

Nielson *et al.* use TDA to identify interactions between preclinical spinal cord injury (SCI) and traumatic brain injury (TBI) in datasets from the Visualized Syndromic Information and Outcomes for Neurotrauma-SCI (VISION-SCI) repository [69]. They use MAPPER to visualize the data, with the Pearson correlation as the metric and three principal components of the singular value decomposition as the filter. They map behavioral outcomes onto the topological networks, showing that forelimb outcomes were most sensitive to cervical SCI. TDA reveals an interaction between SCI and concurrent TBI that depends on the anatomical location of brain lesions. They identify "potential detrimental consequences of [methylprednisolone] treatment on tissue pathology in cervical SCI, and to a lesser extent in thoracic SCI." This study shows the power of MAPPER in reducing a very high dimensional dataset into a lower-dimensional representation (the "syndromic space" that Nielson *et al.* describe), and how the resulting network structure encodes complex interrelations between the features of the original dataset.

Cole and Shiu propose the use of persistent homology for studying the landscape of string vacua in the various superstring theories of theoretical physics [19]. By using persistence to characterize the distribution of vacua, they hope to understand how the vacua

relate to low-energy physics. They choose persistent homology over black-box approaches like neural networks due to its interpretibility. They consider type IIB string theory on Calabi-Yau orientifolds in the presence of background fluxes, and study the distribution of stabilized axiodilaton and complex structure moduli vacuum expectation values using persistent homology. Viewing the distribution of string vacua as a kind of point-cloud dataset, they apply persistent homology to it to find its topological features. They identify many 1-cycles in the persistent homology of vacua, corresponding to long-lived voids in the vacua. Using persistence pairing, they were able to reconstruct the presence of isolated vacua inside some of these voids in the rigid Calabi-Yau construction, something not possible with traditional persistence where these isolated vacua would be treated as noise and not distinguished from the hole. This shows that TDA can have applications to even the most theoretical fields of physics, where there are no examples of "real-world" data.

Li *et al.* consider the application of TDA to aviation datasets [59]. They introduce persistent homology using examples of a simplicial complex built from aviation data. Next, they review existing applications of TDA to aviation in the literature. Their own case study is on the nerve topology of airport configurations. They view the surface of an airport as a network structure where ramps and runways serve as sources and sinks of aircraft, which travel along taxiways and inactive runways. A failing of this graph-based representation is that it only captures pairwise relationships. They look at the five busiest airports in the USA, obtaining the runway configurations from the Aviation System Performance Metrics (ASPM) database maintained by the US Federal Aviation Administration. They infer the active taxiway configurations from air traffic control audio, since these aren't included in the ASPM database. They construct a nerve complex by viewing the taxiways as "covers" that contain a runway or terminal when the taxiway intersects or meets them. Thus, the vertices of the complex are the runways and terminals of the airport. They note that maximal simplexes are lost when going from a high-capacity scenario to a low-capacity scenario, indicating a significant change in the airport's topology. They suggest that the

maximal simplex could be used as a measure of the airport's connectivity, to get a feel for how reachable the various assets of the airport are.

Gidea and Katz explore the daily returns of four major US stock indices during the technology crash of 2000 and the financial crisis of 2007 to 2009 [37]. They look at the stock data as a time series under a sliding window, treating the data for each window as a point-cloud to which TDA can be applied. For each point-cloud, they compute the Vietoris-Rips filtration, and from that the persistence landscape and the L^p norms of the landscape for loops. They identify that the variance and spectral density of the L^p norms of the combined time series are substantially increasing before the crashes. They conclude that this represents an increased persistence of loops as the market transitions from the ordinary state to a "heated" state. They suggest that this measure could be used as an early warning signal preceding a future market crash.

Asaad and Jassim apply persistent homology to the forensic detection of image tampering [6]. They use the local binary pattern (LBP) of Ojala *et al.* [70] to represent the local texture of a grayscale image. Uniform LBP pixels at different thresholds are used to build Vietoris-Rips complexes. They investigate the sensitivity of the VR complexes to morphing in passport photos, under the assumption that morphing will produce topologically inequivalent complexes which therefore have different topological invariants. For the splicing morphing scheme, their method correctly classified 98% of 100 images randomly selectec from the Utrecht face database. The combined morphing scheme was correctly classified 99% of the time, while the complete morphing scheme is only classified correctly 60% of the time.

Topological hierarchies provide new opportunities for applying TDA to real-world problems. For bioinformatics data, a THD can be viewed as an extension of MAPPER that provides views of the syndromic space at various resolutions and on meaningful subsets of the data decomposed by MAPPER. It could potentially identify smaller subgroups of meaningful outcomes by looking at the groups near the leaf nodes. For the application to

aviation considered by Li *et al.*, THD could potentially identify sub-topologies of an airport by decomposing a point-cloud representation of the airport into a hierarchical structure. This could be useful for studying the ability of the airport to continue operation when some parts of it are non-functional due to weather, construction, or accident. For analysis of stock returns as a time series, THD could be used to identify periods of time preceding a crash which exhibit unusual structure and are thus decomposed into subgroups in the hierarchy whose features are statistically significantly different than the rest of the data. For detecting image tampering, THD has the potential to not only identify tampered images through the same techniques of persistence, it could also find which regions of the image are the most distorted from the original.

Background

This chapter will give an introduction to the needed topological and mathematical background for the rest of the dissertation, as well as related concepts that are useful in applications. For more information on introductory topology, consult a standard textbook such as [67] for general topology, and [48] for algebraic topology. There are also a number of surveys of topological data analysis and monographs on applied or computational topology; see any of [11, 16, 66, 73, 97, 100, 27].

If A is a set, we use $2^A = \{B | B \subseteq A\}$ to denote the power set rather than $\mathcal{P}(A)$ which is often used in mathematical writings.

3.1 Basic Topology

The starting point in the study of topology is the definition of a topological space. This generalizes notion of nearness, openness, and closedness encountered in real analysis, complex analysis, and the study of Euclidean spaces $\mathbb{R}^n$. A topology can be thought of as providing a definition of qualitative nearness through its open sets. A metric structure, which provides a quantitative distance, is a stronger structure than a topology, as will be seen later.

Definition 1. Given a set X, a *topology* on X is a collection $\mathcal{T}$ of subsets of X satisfying the following axioms:

1. The set X itself is in $\mathcal{T}$; Symbolically, $X \in \mathcal{T}$.

2. The empty set $\emptyset$ is in $\mathcal{T}$; That is, $\emptyset \in \mathcal{T}$.

3. For any indexing set A such that for all $\alpha \in A$, there is an element $O_\alpha \in \mathcal{T}$, the union $\bigcup_{\alpha \in A} O_\alpha \in \mathcal{T}$. That is, $\mathcal{T}$ is closed under arbitrary union.

4. For any two $O_1, O_2 \in \mathcal{T}$, the intersection $O_1 \cap O_2$ is in $\mathcal{T}$. That is, $\mathcal{T}$ is closed under finite intersection.

The sets O in the topology $\mathcal{T}$ are called *open sets*, and this name was chosen because they generalize the properties of open sets in $\mathbb{R}^n$. Recall that a subset U of $\mathbb{R}^n$ is open if for every point x in U there is an open ball around x which is contained in U. Intuitively, U is open if there is enough wiggle room around each of its points so that small enough translations can not leave U. Finally, given a set X and a topology $\mathcal{T}$ on X, the pair $(X, \mathcal{T})$ is called a *topological space*.

Example 1. Here are some of the standard examples of topological spaces:

1. An arbitrary set X with $\mathcal{T} = \{X, \emptyset\}$ as a topology. This is known as the *trivial topology* since it doesn't contain much, if any, information about the structure of the set.

2. An arbitrary set X with the topology $\mathcal{T} = 2^X$ of all subsets of X. This is called the *discrete topology* on X. Like the trivial topology, it is not too useful in practice since any subset of X will be open.

3. The set $X = \{a, b, c\}$ with the collection of subsets $\mathcal{T} = \{X, \emptyset, \{a\}, \{b, c\}\}$ as the topology. There are many other ways to define a topology on a set of three elements, 29 in total [54]. While topologies on a finite set can be useful to understand definitions and find counterexamples, through the rest of the dissertation it is assumed that the underlying topologies are on infinite sets, usually subsets of $\mathbb{R}^n$. This leads to notions of convergence that are closer to the intuitive ones from calculus.

4. The set of real numbers $\mathbb{R}$ is a topological space with the following standard topology: a set $O \subseteq \mathbb{R}$ is considered open if for any point $p \in O$ there exists an $\epsilon > 0$ such that the interval $B_\epsilon(p) = (p - \epsilon, p + \epsilon)$ is contained in O: $B_\epsilon(p) \subseteq O$. This is

equivalent to saying that an open set is made up of a (possible infinite) union of open intervals, since we can just choose one such interval for each point in the set.

5. The previous discussion extends to $\mathbb{R}^n$ as follows: a set $O \subseteq \mathbb{R}^n$ is considered open if for any point $p \in O$ there exists n numbers $\epsilon_i > 0$ such that the Cartesian product $(p - \epsilon_1, p + \epsilon_1) \times (p - \epsilon_n, p + \epsilon_n)$ is contained in O. Equivalently, an open set in $\mathbb{R}^n$ can be made up of unions of Cartesian products of n open intervals.

6. The complex numbers $\mathbb{C}$ also form a topological space. Instead of open intervals, we work with open balls of the form $B_r(z_0) = \{z \in \mathbb{C} || z - z_0| < r\}$. A set $O \subseteq \mathbb{C}$ is open if for any point $p \in O$ there exists a number $\epsilon > 0$ such that $B_\epsilon(p) \subseteq O$. This definition can be extended to give the standard topology on $\mathbb{C}^n$.

A topological space gives us the open sets, from which we can construct various other sets. The first will be the closed sets, which are simply the complements of open sets.

Definition 2. Given a topological space $(X, \mathcal{T})$, a subset $C \subseteq X$ is said to be *closed* if $X \backslash C$ is open, that is, if $X \backslash C \in \mathcal{T}$.

For example, in $\mathbb{R}$, the complement of an open interval I is the union of two closed intervals, and a general closed set can be written as the (possibly infinite) intersection of a collection of closed intervals, just how an open set can be written as the union of open intervals. The intuition one should have for a closed set is that it contains its own boundary, i.e. points for which any neighborhood (see the next paragraph) will contain points both inside and outside of the set. However, this shouldn't be taken too far – the interval $[0, \infty)$ is closed in $\mathbb{R}$ (its complement is $(-\infty, 0)$) but is unbounded on one side. So only the boundary points that are inside the set $\mathbb{R}$ are included in a closed set.

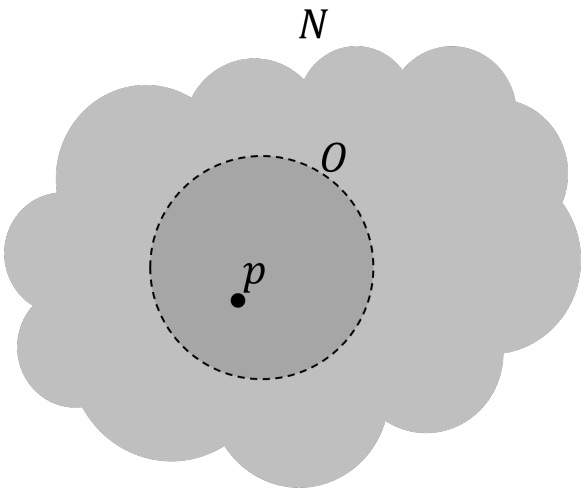


Figure 3.1: A neighborhood N of a point p which contains an open set O containing p.

Often times we want to consider a set that "surrounds" a point but may not necessarily be open. This leads to a simple generalization of an open set.

Definition 3. A *neighborhood* of a point $x \in X$ is a set $N \subseteq X$ such that $x \in U \subseteq N$ for some open set U. That is, a neighborhood of a point is a set that contains an open set containing the point.

Neighborhoods are used instead of open sets because in practice what matters isn't that the point is in some specific open set, but merely that there is "wiggle room" around the point, which doesn't require that the set we're talking about is open, only that it has some open subset containing the point. Equivalently, one can consider a point in its neighborhood to be an "interior point", i.e. one that is not on the boundary of the set.

A very important property that a topological space can have is the ability to "separate" points by open sets or neighborhoods. All but the most pathological spaces will have this property. Two distinct points $x, y \in X$ in a topological space can be separated by neighborhoods if there exists a neighborhood U of x and a neighborhood V of y such that $U \cap V = \emptyset$. This leads to the following definition.

Definition 4. A topological space is a *Hausdorff space* if every pair of distinct points in it

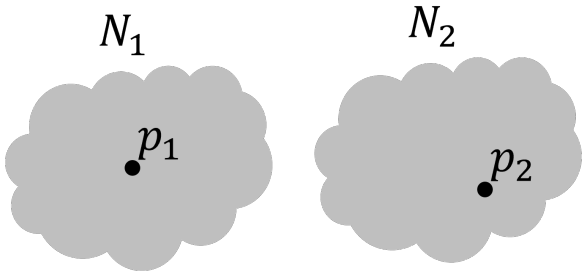


Figure 3.2: Two points p_1 and p_2 separated by neighborhoods N_1 and N_2 such that $N_1 \cap N_2 = \emptyset$.

can be separated by neighborhoods.

For example, $\mathbb{R}$ with the usual topology is Hausdorff since we can choose disjoint open intervals around a pair of arbitrary real numbers. Every topological space discussed in this dissertation is assumed to be Hausdorff to avoid pathological cases. In particular, every metric space is a Hausdorff space, and the subspace (see below) of a Hausdorff space is also Hausdorff. In a Hausdorff space, a sequence of points has a unique limit if it converges to any limit at all.

We now consider how to construct new topological spaces from existing ones. Since topological spaces are sets, we can take unions, intersections, Cartesian products, and look at subsets. All that is left is to define a topology on these constructs from the existing topologies on the sets they are built off of. Let us look at cartesian products first. This leads to the notion of a product topology:

Definition 5. Let X and Y be topological spaces. The *product topology* on the Cartesian product $X \times Y$ has as open sets $A \times B$ where A is open in X and B is open in Y.

This extends to the product topology of a finite number of sets in a natural way. There are more subtleties when an infinite Cartesian product is taken, but this will not need to be considered in this dissertation.

Example 2. We can form the product topology of any number of the spaces from Example 1. In particular, we have the product topologies on $\mathbb{R}^n$ and $\mathbb{C}^n$ given by the product of n

copies of $\mathbb{R}$ or $\mathbb{C}$ respectively. While not immediately obvious, these are equivalent to the topologies on them described in terms of open balls.

Next, let us consider an arbitrary subset of a topological space. In order to construct a topology on it, we can consider intersections of the subset with the open sets of the full space. This leads to the definition of a subspace topology, an extremely important construction in this dissertation.

Definition 6. Let $(X, \mathcal{T})$ be a topological space, and $S \subset X$. Then the *subspace topology* of S is defined as

$$\mathcal{U} = \{T \cap S | T \in \mathcal{T}\}.$$

The open sets of $\mathcal{U}$ are just the intersection of open sets in $\mathcal{T}$ with the subset S. This defines a new topological space $(S, \mathcal{U})$.

The subspace topology is the core construction used in defining topological hierarchies. A hierarchy defines a tree structure of subsets, and as long as we have a topology on the largest set, we can endow each of the subsets with the subspace topology. This is related to the fact that if $A \subset B \subset T$, then the subspace topology on B generates the subspace topology on A, and this is the same as the subspace topology on A generated by T. This applies to subsets at any (finite) level, which defines a kind of consistency of subspace topologies on a hierarchical structure.

Now, let us consider definitions that cover the notions of limits and convergence (of a sequence) in a topological space. A limit point is a generalization of the area around the limit of a sequence in $\mathbb{R}^n$, where you will always have points that are not in the sequence. An isolated point is a single point in a set that isn't "connected" to any other points of the set. And the limit of a sequence is a straightforward generalization of the limit of a sequence in $\mathbb{R}^n$, using the language of neighborhoods to avoid talking about distances.

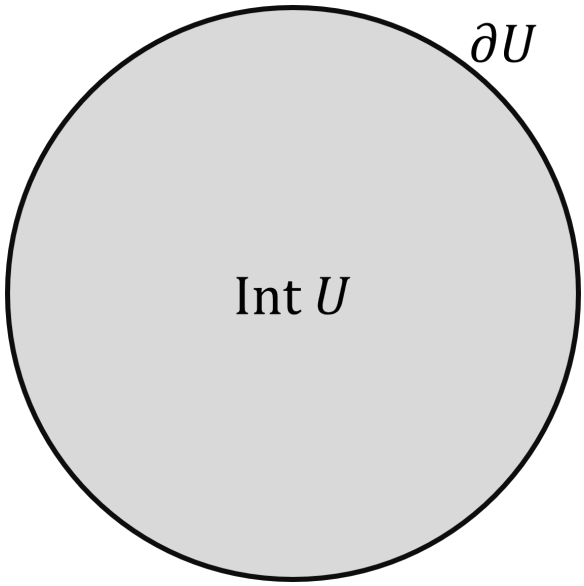


Figure 3.3: The boundary ∂U and interior Int U of a set U. The closure $\bar{U}$ is the union of the boundary and interior. The original set U would include the interior and possibly parts of or the entire boundary.

Definition 7. Let S be a subset of a topological space X. A point $x \in X$ is a *limit point* of S if every neighborhood of x contains at least one point of S distinct from x. A point $y \in S$ is an *isolated point* of S if there exists a neighborhood of y that does not contain any other points of S. A point $x \in X$ is a *limit* of a sequence $(x_n)_{n\in\mathbb{N}}$ if for every neighborhood V of x, there exists a $N \in \mathbb{N}$ such that for all $n \geq N$, the point $x_n \in V$.

Finally, there are sets that can be defined from subsets of a topological space based on their limit points. The closure of a set can be viewed as adding the boundary of the set to it, to form a closed set. In particular, the boundary of a closed set is the same set. The interior of a set removes the boundary, and the interior of an open set is the same open set. The boundary is the closure of a set minus its interior, i.e. only the "closed" part of the closure. This leads to the following definitions.

Definition 8. Let $S \subseteq X$ be some subset of a topological space. The *closure* of S, denoted $\bar{S}$, is the set S together with all of its limit points. The *interior* of S is the union of all

open sets contained in S, denoted $\mathrm{Int}S$. The *boundary* of S is the closure with the interior removed, that is, $\partial S = \bar{S} - \mathrm{Int}S$. Points in the boundary have the property that any neighborhood of them will contain points outside of S.

3.1.1 Continuous Maps

Given two topological spaces $(X, \mathcal{T})$ and $(Y, \mathcal{U})$, a function $f : X \to Y$ is often called a *mapping* of topological spaces, or simply a *map*. We are most interested in continuous maps, which are distinguished by the fact that they preserve certain properties of topological spaces. To define this, we first need to review the definitions of continuity from calculus and real analysis.

Definition 9. Let $f : A \to \mathbb{R}$ be a real-valued function, where $A \subseteq \mathbb{R}$. The function f is said to be continuous at a point $c \in A$ if

$$\lim_{x \to c} f(x) = f(c)$$

More formally, f is continuous at c if for all $\epsilon > 0$ there exists a $\delta > 0$ such that for all $x \in A$ with $|x - c| < \delta$, we have that $|f(x) - f(c)| < \epsilon$.

This more rigorous definition will become important for topology because it involves open sets; the set of values

$$\{x \in A \big| |x - c| < \delta\}$$

is exactly the open interval $(c - \delta, c + \delta)$ restricted to A, and similarly we have an open interval $(f(c) - \epsilon, f(c) + \epsilon)$. Then the $\epsilon - \delta$ definition of continuity is essentially saying that for any open interval around $f(c)$, there is an open interval around c which is mapped to the first open interval. This leads to the general definition of continuity for topological

spaces.

Definition 10. Given two topological spaces $(X, \mathcal{T})$ and $(Y, \mathcal{U})$, a function $f : X \to Y$ is said to be *continuous* if for a subset $O \subseteq Y$ which is open ($O \in \mathcal{U}$), the inverse image $f^{-1}(O)$ is open ($f^{-1}(O) \in \mathcal{T}$). If f is a bijection, it has an inverse $g : Y \to X$ such that $f \circ g = \mathrm{id}_Y$ and $g \circ f = \mathrm{id}_X$. If g also happens to be continuous, then f is said to be a *homeomorphism* and the spaces X and Y are said to be *homeomorphic*.

In topology, homeomorphic spaces are considered to be identical. For example, all circles and ellipses are homeomorphic to the unit circle, so by studying the unit circle we can determine the topological properties of every circle and ellipse. Note that the *geometric* properties of an ellipse are very different than those of a circle. We call something a *topological property* if it is preserved by homeomorphisms. For example, compactness, and being a Hausdorff space are topological properties.

3.1.2 Open Covers and Compactness

Often when looking at the global structure of a topological space, we wish to be able to find an open set for each point in the space. This leads to a family of open sets which "cover" the space in the sense that each point in the space falls in one or more of a sets.

Definition 11. Let $(X, \mathcal{T})$ be a topological space. Given a subset $A \subseteq X$, an *open cover* of A is a family of sets $\mathcal{U} = \{U_\alpha\}_{\alpha \in J}$ such that each U_α is open (in $\mathcal{T}$), and

$$A \subseteq \bigcup_{\alpha \in J} U_\alpha$$

That is, each point x of A is contained in at least one U_α. Given an open cover $\mathcal{U} =$

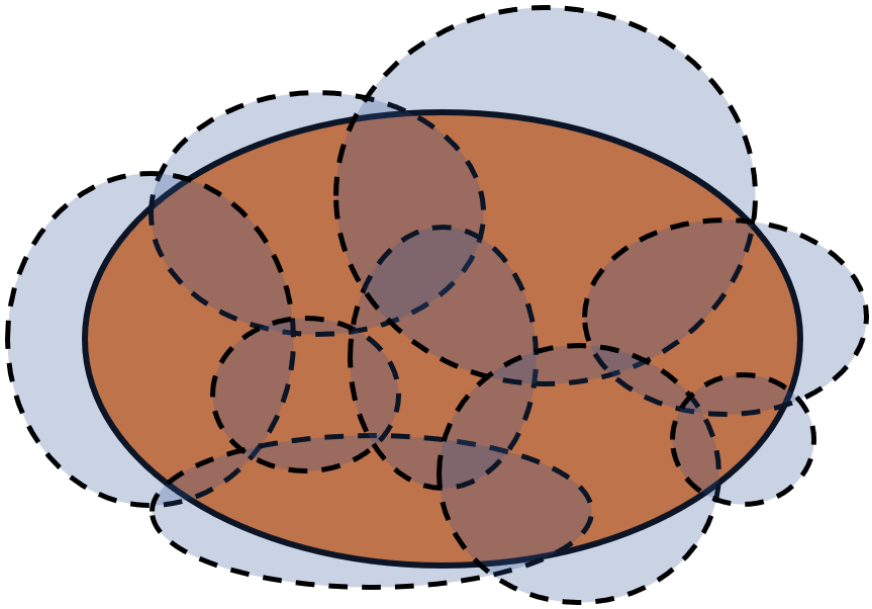


Figure 3.4: Illustration of an open cover on a subspace (the orange ellipse). The open sets of the cover are the intersections of the dashed ellipses with the orange-filled subspace.

$\{U_\alpha\}_{\alpha \in J}$, if there exists a finite set $\mathcal{V} = \{U_1, \ldots, U_n\}$ where each U_i is in $\mathcal{U}$, then $\mathcal{V}$ is said to be a *finite subcover* of $\mathcal{U}$.

An illustration of an open cover is given in Figure 3.4. This shows how an open cover of a parent space can be restricted to an open cover on the subspace topology. So far, there are no restrictions on the number of open sets in the cover. In particular, there could be an infinite number of open sets in the cover, even an uncountable infinite number. A nice property to avoid this would be the ability to replace any open cover with a finite one. This leads to the notion of compactness.

Definition 12. A topological space X is said to be *compact* if every open cover of X has a finite sub-cover.

Compactness is a notoriously opaque notion; see [77] for more motivation and a discussion of a related property, sequential compactness. In $\mathbb{R}^n$, a set is compact if and only if it is closed and bounded, a fact known as the Heine-Borel Theorem. This does not hold for general topological spaces where one does not have a definition of boundedness, but

compactness can be viewed as a generalization of these properties.

3.1.3 Topological Bases

Similar to the motivation for an open cover, we may want to associate a set with each point in a topological space. Furthermore, in the intersection between two such sets, we wish to always be able to find another set containing a point in the intersection that "fits" in the intersection. These sets will form a basis for the topology.

Definition 13. Given a set X, a *basis* of a topology on X is a collection $\mathcal{B}$ of subsets of X such that

1. For all $x \in X$, There exists a $B \in \mathcal{B}$ such that $x \in B$. That is, $\mathcal{B}$ covers X.

2. For any $B_1, B_2 \in \mathcal{B}$, if there is an $x \in X$ such that $x \in B_1 \cap B_2$, then there exists a $B_3 \in \mathcal{B}$ such that $x \in B_3 \subseteq B_1 \cap B_2$.

Note that a basis in topology is different from a basis in linear algebra, but the idea behind the definition is similar. However, a topological basis isn't immediately related to a notion of "dimension" like a basis for a vector space is. A topological basis can be viewed as the smallest number of sets needed to generate a topology in the following sense.

Given a set X and a basis $\mathcal{B}$, the *topology generated by* $\mathcal{B}$ consists of all possible unions and finite intersections of elements in $\mathcal{B}$. For example, the set of open intervals is a basis on $\mathbb{R}$ and the topology generated by this basis is just the standard topology on $\mathbb{R}$.

3.1.4 Connectedness and Paths

Recall that an isolated point in a subset of a topological space is one that is not "connected" to the rest of the subset in the sense that there is a neighborhood of the isolated point that doesn't contain any other points in the subset. Is it possible for entire *subsets* of a

topological space to be isolated instead of a single point? The answer is yes, and the following definition describes this situation.

Definition 14. A topological space X is said to be *connected* if it can not be written as the union of two disjoint non-empty open sets. This can be extended to subsets of X, where a subset $S \subseteq X$ is connected if it is a connected space under the subspace topology. A *connected component* of a topological space is an open subset which is connected. The set of connected components of X form a partition of the space.

In applications, it is desirable to have a stronger notion of connectedness. It may be difficult to find the exact forms the disjoin sets of a non-connected space takes. Instead, we consider "travelling" through the space on a curve. If we can continuously go from one point to another in a space, we can be sure that the points are connected. This leads to the definition of path-connectedness.

Definition 15. Given a topological space X, a *path* in X from a point $x \in X$ to $y \in X$ is a continuous function $\gamma : [0,1] \to X$ where $[0,1]$ is the unit closed interval in $\mathbb{R}$ and $\gamma(0) = x$ and $\gamma(1) = y$.

The topology we use on $[0,1]$ is the subspace topology induced by the standard topology on $\mathbb{R}$. If γ is injective, i.e. for all $x, y \in [0,1]$ $\gamma(x) = \gamma(y)$ implies $x = y$, we say the path γ is *simple*. Intuitively, this is a path which does not cross itself. Now we can define what it means for two points to be path-connected.

Definition 16. Two points in X are *path-connected* if there exists a path between them. The entire space is path-connected if every pair of points is path-connected. Similarly to connectedness, a subset of a topological space is path-connected if it is under the subspace topology.

Given a point x, the path-component of the space containing x consists of all the points that are path-connected with x. Every path-connected set is connected, but the converse does not hold. Hereafter, it is assumed that when "connected" is used, "path-connected" is meant.

3.1.5 Metric Spaces

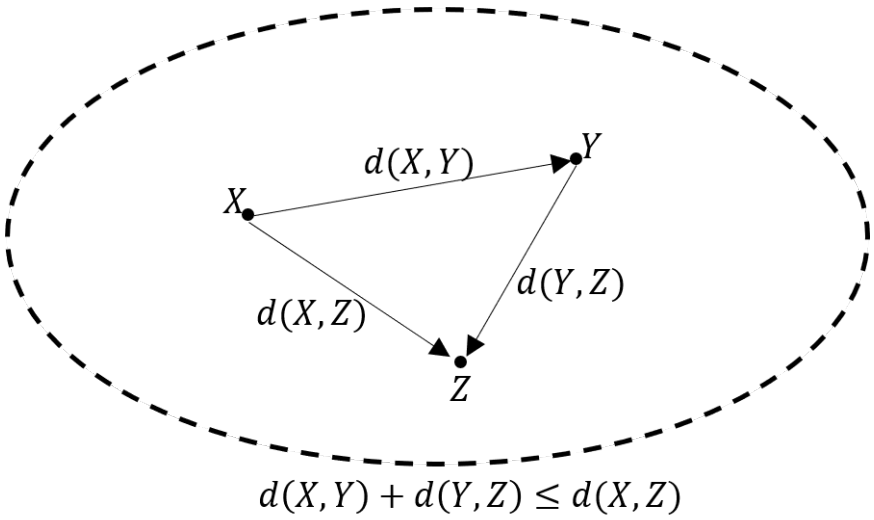


Figure 3.5: An illustration of the triangle inequality in a metric space.

An example of a large family of topological spaces is that of a metric space. Informally, a metric space is a set of points and a way of computing the *distance* between two points. This distance should satisfy the expected properties of distance, based on our experience with real numbers and Euclidean spaces. In particular, it should be non-negative, and only equal to zero when computing the distance between a point and itself.

Definition 17. A *metric space* (X, d) is a set X along with a function $d : X \to \mathbb{R}$ such that

1. $d(x, y) = 0$ if and only if $x = y$ all $x, y \in X$

2. $d(x, y) = d(y, x)$ for all $x, y \in X$

3. $d(x,z) \leq d(x,y) + d(y,z)$ for all $x, y, z \in X$

The last axiom is often called the *triangle inequality* because of its geometric interpretation in $\mathbb{R}^2$, as illustrated by Figure 3.5. From these three axioms, it can be shown that $d(x,y) \geq 0$ for all $x, y \in X$, so the metric has the desired non-negativity as well.

Example 3. Here are the standard examples of metric spaces.

1. The real numbers $\mathbb{R}$ with distance given by $d(x,y) = |x-y|$. This is the prototypical example where the triangle inequality is first encountered.

2. Generalizing this, $\mathbb{R}^n$ with Euclidean distance given by $d(x,y) = \left(\sum_i (x_i - y_i)^2\right)^{1/2}$.

3. For any $p > 0$ the space $\mathbb{R}^n$ is a metric space with distance given by $d(x,y) = \left(\sum_i |x_i - y_i|^p\right)^{1/p}$. The previous example is the case $p = 2$.

4. Complex numbers $\mathbb{C}$ with distance given by $d(z,w) = |z-w|$ where $|z| = \sqrt{z\bar{z}}$, where $\bar{z}$ is the complex conjugate.

5. The surface of a sphere, with the distance between two points being given by the arc length of a great circle connecting them.

Given that the most well-known metric spaces $\mathbb{R}^n$ are also topological spaces, is it possible to make any metric space into a topological space? The answer is yes. There is a standard way of constructing a topology on a metric space (X,d). For any $\epsilon > 0$ and $x_0 \in X$, define the *open ball* of radius ϵ centered at x_0 to be

$$B_\epsilon(x_0) = \{x \in X | d(x, x_0) < \epsilon\}$$

Then the set

$$\mathcal{B} = \{B_\epsilon(x) | \forall x \in X \text{ and } \epsilon > 0\}$$

forms a basis for a topology on X. The resulting topology is called the standard or induced topology on X.

3.2 Clustering

In order to describe the computational aspects of MAPPER more formally, it is useful to have a definition of a clustering. I follow [8], which views clustering as a family of partitions that are equivalent up to labeling. Assume a set of s labels, usually the numbers $1, \ldots, s$, and an assignment of a label to each point in a metric space defines a clustering. However, since the labels are arbitrary, we allow permutations of them to define the same clustering.

Definition 18 (Clustering)**.** Let (X, d) be a metric space, s a positive integer, and $F(X, s)$ the set of all functions $f : X \to \{1, \ldots, s\}$. For any $f, g \in F(X, s)$, we say that f and g are equivalent and write $f \sim g$ if there is a permutation π of $\{1, \ldots, s\}$ such that $f = \pi \circ g$. The equivalence class of functions $f \in F(X, s)$ is denoted $[f]$, i.e.

$$[f] = \{g \in F(X, s) | f \sim g\}$$

The set of all *clusterings* on X, denoted $\mathcal{F}$, is the quotient:

$$\mathcal{F} = F(X, s)/\sim = \{[f] | f \in F(X, s)\}$$

This definition means possible clusterings are to be considered equivalent if they produce the same clusters but label them differently, so that the clusterings are related by a permutation of the labels. Given a clustering function $f : X \to \{1, \ldots, s\}$, we denote the

ith cluster $C_i(f)$, and it is given by

$$C_i(f) = f^{-1}(i) = \{x \in X | f(x) = i\}$$

for $1 \leq i \leq s$.

We are particularly concerned with hierarchical clustering algorithms, which are typically done in an agglomerative approach, where one starts with each point as a separate cluster, and iteratively merges clusters to form a hierarchy of clusters with the entire point set as the root [52, 68]. This gives a tree structure known as a *dendrogram*, which can be viewed as a family of functions f_* such that f_d is the clustering function obtained by cutting the tree at some distance d, producing what is known as a *flat clustering* where the leaf nodes of the cut dendrogram correspond to the clusters produced by f_d.

Given the current set of clusters, hierarchical clustering schemes differ in how they determine which two clusters to merge at a given step. In general, some distance is computed for each pair of clusters based on the metric, and the two clusters with the lowest distance between them are merged. This distance between clusters is known as the *linkage criterion*. Let A and B be two clusters, then the following linkage criteria are the most commonly used to assign a distance between them:

- (*single linkage*) $d(A, B) = \min \{d(a, b) | a \in A, b \in B\}$
- (*complete linkage*) $d(A, B) = \max \{d(a, b) | a \in A, b \in B\}$
- (*unweighted average linkage*) $d(A, B) = \frac{1}{|A| \cdot |B|} \sum_{a \in A} \sum_{b \in B} d(a, b)$

For a given pair of clusters A and B, the single linkage distance will be the smallest, the complete linkage distance the largest, and the unweighted average linkage distance will fall in between them. There are other linkage criteria, but many of them rely on having a Euclidean metric, or are variations of average linkage. For a discussion of evaluating clusterings, see Section 4.4.

3.3 Topological Structures

The MAPPER construction of [84] is motivated by the notion of a pullback cover, which allows one to "pull back" an open cover in a space mapped to from another space to obtain an open cover of path-connected sets in the parent space. For some pathological functions, it is possible that the inverse image of a set in a cover is split into infinitely many connected components. We restrict ourselves to well-behaved functions that do not do this.

Definition 19 (Well-behaved function)**.** Let X be a metric space, Z a topological space, and $f : X \to Z$ a continuous map. We say that f is *well-behaved* if for any path-connected open set U in Z, the pre-image $f^{-1}(U)$ has finitely many path-connected components in X.

Using well-behaved functions guarantees that the complexes produced by MAPPER are always finite as long as we start with a finite cover, since MAPPER will at most split a finite number of open sets a finite number of times.

If we have an open cover $\mathcal{W} = \{W_i\}_{i=1}^t$ on Z and a continuous, well-behaved function $f : X \to Z$, can we obtain a cover on X? The answer is yes, since each of the pre-images $f^{-1}(W_i)$ will be open in X as f is continuous, and each $x \in X$ is mapped to some $z \in Z$ so taken as a whole $\{f^{-1}(W_i)\}_{i=1}^t$ will cover X. However, the pre-images may not all be path-connected, as a connected set in Z may be "split" under the inverse image f^{-1} into many components in X. The pullback cover is a construction which recovers an open cover of path-connected components from these inverse images.

Definition 20 (Pullback Cover)**.** Let X be a metric space, Z a topological space, $f : X \to Z$ a well-behaved continuous map, and $\mathcal{W} = \{W_i\}_{i=1}^t$ a finite open cover on Z where each W_i is path-connected. Use $U_i = f^{-1}(W_i)$ to represent the inverse images of the W_i for

$i = 1, \ldots, t$. Now, split each of the U_i into its path-connected components, so we have

$$U_i = V_1^i \cup \cdots \cup V_{s_i}^i$$

for $i = 1, \ldots, t$ where s_i is the number of path-connected components of U_i.

The *pullback cover* of $\mathcal{W}$ under f is the collection of these V_j^i:

$$f^*(\mathcal{W}) = \{V_j^i | i = 1, \ldots, t \text{ and } j = 1, \ldots, s_i\}$$

To build up to the idea of a simplicial complex, I start by defining the standard simplices in $\mathbb{R}^{n+1}$. A simplex can be thought of as a generalization of a triangle or tetrahedron to arbitrary dimensions. Since I am only concerned with topological properties of complexes, I can choose a particularly simple coordinization to describe simplices.

Definition 21. For a non-negative integer n the *standard n-simplex* is a subset Δ^n of $\mathbb{R}^{n+1}$ given by

$$\Delta^n = \left\{ (x_0, x_1, \ldots, x_n) \in \mathbb{R}^{n+1} \;\middle|\; \sum_{i=0}^{n} x_i = 1 \text{ and } x_i \geq 0 \text{ for all } i = 0, \ldots, n \right\}$$

So the standard 0-simplex is a point, the standard 1-simplex a line, the standard 2-simplex a triangle, and so on. An arbitrary simplex is generated by a continuous deformation of a standard simplex. Note that any n-simplex will contain $n+1$ $(n-1)$-simplices as subsets (when $n > 0$); these are the *faces* of the simplex. A geometric *simplicial complexes* is made up of the union of simplices of any dimension. An abstract simplicial complex is obtained by "forgetting" the geometric structure of a simplicial complex, focusing entirely on the combinatorial aspects of simplices as sets of points.

Definition 22 (Abstract simplicial complex)**.** Given a finite set V, an *abstract simplicial complex* Σ is a collection of subsets of V with the following property: whenever $\sigma \in \Sigma$ and $\tau \subset \sigma$, then $\tau \in \Sigma$. The elements of Σ are called its *faces* or *simplices* and V is called the *vertex set* and its elements *vertices*. If a simplex $\sigma \in \Sigma$ has $k+1$ vertices (i.e. $|\sigma| = k+1$), we say that σ is a *k-simplex*.

An abstract simplicial complex can be viewed as an undirected graph with higher-order "edges" in the following sense: the nodes of the graph are the elements of V, edges are pairs $\{v, v'\}$, and so on. The fact that any edge in a graph must have its endpoints also in the graph is generalized by the property of an abstract simplicial complex. From here on out I will only be using abstract simplicial complexes, so I will call them simplicial complexes for brevity as there is no possibility of confusion.

The main examples of simplicial complexes I will discuss are the nerve and Vietoris-Rips constructions. The nerve construction builds an (abstract) simplicial complex from an open cover. It can be considered to give a representation of the cover, showing how the elements of the cover overlap. Given a subset of a topological space (possibly the entire space) and an open cover of that subset, the nerve can describe certain topological properties of the space.

Definition 23 (Nerve)**.** Let X be a topological space and $\mathcal{U} = \{U_i\}_{i=1}^t$ a finite cover of X. The *nerve* of $\mathcal{U}$ is a simplicial complex, denoted $\mathcal{N}(\mathcal{U})$, constructed as follows: the vertex set is the collection of indices $\{1, \ldots, t\}$ and a set of indices $\{v_0, v_1, \ldots, v_k\}$ is a k-simplex if

$$U_{v_0} \cap U_{v_1} \cap \cdots \cap U_{v_k} \neq \emptyset$$

That is, the simplices are the indices of collections of open sets which all overlap. It is easy to see that this is a simplicial complex, as if some collection of open sets all have a non-

empty intersection, then any subset of that collection will also have non-empty intersection. The 0-simplices are identified with the open sets in the cover, with higher-order simplices being generated by the overlaps of open sets of that order.

Before I define MAPPER, I give an example of a nerve that is commonly used in TDA, and an example of a construction which doesn't use the nerve known as the Vietoris-Rips complex, or VR complex for short. The properties of the VR complex shared with the nerve complex will motivate a generalization of the nerve operation used for defining topological hierarchies.

Example 4. Given a finite metric space $X = (x_1, \ldots, x_n)$ and some $\epsilon > 0$, the *Čech complex* $\check{\mathrm{C}}_\epsilon(X)$ is a simplicial complex defined as follows: we define an open cover $\mathcal{U}_\epsilon = \{U_i\}_{i=1}^n$ on X by taking the open ball of radius ϵ around each $x_i \in X$, so that we have $U_i = B_\epsilon(x_i)$ for $i = 1, \ldots, n$. The Čech complex is simply the nerve of this cover: $\check{\mathrm{C}}_\epsilon(X) = \mathcal{N}(\mathcal{U}_\epsilon)$.

Note that the Čech complex will have as many vertices as there are points in the metric space. This means that it can become complicated quickly if points are added to the space, as the number of possible intersections between the open balls grows as $\binom{n}{k+1}$ for checking the existence of k-simplices. This motivates the Vietoris-Rips complex, which greatly reduces the number of intersections which need to be checked.

Example 5. Let (X, d) be a finite metric space and $\epsilon > 0$. The *Vietoris-Rips complex* (VR complex) of X with diameter ϵ is the simplicial complex $\mathrm{VR}_\epsilon(X)$ whose vertex set is X, and where a subset $\{x_0, \ldots, x_p\}$ of X spans a p-simplex if and only if $d(x_i, x_j) < \epsilon$ for all $0 \leq i, j \leq p$. We can describe this using the open cover:

$$\mathcal{U}_\epsilon = \{U_\epsilon(x) = B_{\epsilon/2}(x) | x \in X\}$$

Then a subset $\{x_0, \ldots, x_p\}$ of X spans a p-simplex if and only if $U_\epsilon(x_i) \cap U_\epsilon(x_j) \neq \emptyset$ for all $0 \leq i, j \leq p$. This cover is the same as the one used in the construction of the Čech complex, except with radius $\epsilon/2$. Furthermore, we have the inclusion relations:

$$\check{\mathrm{C}}_{\epsilon/2}(X) \subseteq \mathrm{VR}_\epsilon(X) \subseteq \check{\mathrm{C}}_{2\epsilon}(X)$$

I am now ready to introduce the MAPPER construction, first defined in [84], which has motivated much of my work in TDA and lead to the concept of topological hierarchies. MAPPER approaches the combinatorial problem raised by the Čech complex construction by making use of a continuous function to map an original topological space to a new one, usually of lower dimension.

Definition 24 (MAPPER)**.** Let X be a metric space, Z a topological space, $f : X \to Z$ a continuous well-behaved map, and $\mathcal{U} = \{U_i\}_{i=1}^t$ a finite open cover of Z. The *MAPPER* of f is defined as the nerve of the pullback cover of $\mathcal{U}$ under f:

$$\mathcal{M}(\mathcal{U}, f) = \mathcal{N}(f^*(\mathcal{U})) \tag{3.1}$$

f is called the *filter* or *lens* function. Elements of $\mathcal{U}$ are known as *bins*. When computing MAPPER, Z is chosen to be of lower-dimension than X.

One can view MAPPER as an extension of clustering where instead of clustering on the entire metric space, one clusters separately on each set of an open cover of the space. This leads to a definition which extends Definition 18.

Definition 25 (MAPPER Function)**.** Let X be a metric space with an open cover $\mathcal{U} = \{U_i\}_{i=1}^t$. Given clusterings $f_i \in \mathcal{F}_i = \mathcal{F}(U_i, s)$, the *MAPPER function* f assigns to each

$x \in X$ the set of cluster labels assigned to it by each of the f_i, so that we have

$$f(x) = \{f_i(x) | i = 1, \ldots, t \text{ and } x \in U_i\} \tag{3.2}$$

The MAPPER function can be used to build the simplicial complex of MAPPER by looking at overlap between the sets it gives for a fixed cluster. For example, if U_i has cluster labels $c_1^i, c_2^i, \ldots, c_{s_i}^i$, then for $i \neq j$ there will be an edge between c_a^i and c_b^j if and only if there exists $x \in X$ such that $\{c_a^i, c_b^j\} \subseteq f(x)$, i.e. both clusters are in the MAPPER function for some point x, which indicates that the clusters overlap.

3.4 Homology and Persistence

I give here a quick overview of simplicial homology, then move on to persistence. For more details on simplicial homology and persistence, see [26, 29, 101]. Simplicial homology constructs modules on a simplicial complex which capture certain features of the complex. The dimensions of these modules describe the number of these features. I will always use scalars from a field F, so the modules will always be vector spaces.

3.4.1 Algebra

Here I recall the algebraic background needed to describe homology groups. These concepts are covered in much more detail in any book on abstract algebra, such as [75, 34, 40].

Definition 26 (Group)**.** A *group* $(G, \cdot)$ is a set G together with a binary operation $\cdot : G \times G \to G$ such that

- The binary operation is associative: $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ for all $a, b, c \in G$,
- There is a unit element $e \in G$, such that $e \cdot a = a \cdot e = a$ for all $a \in G$,

- Every element $a \in G$ has an inverse $a^{-1} \in G$ such that $a \cdot a^{-1} = a^{-1} \cdot a = e$.

If the following property also holds, the group is called *Abelian*:

- The binary operation is commutative: $a \cdot b = b \cdot a$ for all $a, b \in G$.

The operation in an Abelian group is often called *addition* and denoted $+$, e.g. $a+b = b+a$, even when the underlying set is not a set of numbers. In a general group (Abelian or not), this operation is often called *multiplication* and written without the dot: $a \cdot b = ab$.

I am mainly interested in the definition of a group so that I can define rings and fields. Members of a ring or field will be the coefficients of chain complexes, which are then used to build up the homology groups from which persistent homology is constructed.

Definition 27 (Ring and Field)**.** A *ring* $(R, +, \cdot)$ is a set R together with two binary operations $+ : R \times R \to R$ and $\cdot : R \to R$, such that

- $(R, +)$ is an Abelian group,
- $\cdot$ is commutative: $ab = ba$ for all $a, b \in R$,
- $\cdot$ is associative: $a(bc) = (ab)c$ for all $a, b, c \in R$,
- $\cdot$ distributes over $+$: $a(b + c) = ab + ac$ and $(a + b)c = ac + bc$ for all $a, b, c \in R$,
- and there is a unit element $1 \in R$ for $\cdot$ such that $1a = a1 = a$ for all $a \in R$.

In a ring we denote by 0 the unit for the addition operation $+$ and $-a$ for the inverse of $a \in R$ under $+$. If, in addition we have the existence of inverses for multiplication, i.e. for all $a \in R$ there is an $a^{-1} \in R$ such that $aa^{-1} = a^{-1}a = 1$, we call the ring a *field* instead.

Note that in the literature rings are not always required to have a unit or even be commutative. I will only consider commutative rings with unit so that they are included in the properties here. The last definition needed is a vector space; while I could work with modules, for the purposes of this dissertation I only need coefficients in a field, so vector spaces will suffice.

Definition 28 (Vector Space)**.** Let $\mathbb{F}$ be a field. A vector space V over $\mathbb{F}$ is a set V of elements called *vectors* which satisfy:

- V is an Abelian group with operation $+$,
- there is an operation $\cdot : \mathbb{F} \times V \to V$ such that $\lambda \cdot v \in V$ for all $\lambda \in \mathbb{F}$ and $v \in V$,
- this operation distributes with vector addition: $\lambda_1(v_1 + v_2) = \lambda_1 v_1 + \lambda_2 v_2$ and $(\lambda_1 + \lambda_2)v_1 = \lambda_1 v_1 + \lambda_2 v_1$ for all $\lambda_1, \lambda_2 \in \mathbb{F}$ and $v_1, v_2 \in \mathbb{F}$.

The elements of V are called *vectors* and the elements of $\mathbb{F}$ *scalars*.

Let us recall the notions of linear independence, dimension, and bases in a vector space as well.

Definition 29. Let V be a vector space over a field $\mathbb{F}$. A subset $\{v_1, \ldots, v_n\} \subseteq V$ of vectors is said to be *linearly independent* if the equation

$$a_1 v_1 + \ldots a_n v_n = 0,$$

where $a_1, \ldots, a_n \in \mathbb{F}$ are scalars, is only satisfied when $a_1 = a_2 = \cdots = a_n = 0$. The subset is *linearly dependent* if there exist scalars $a_1, \ldots, a_n$, not all zero, such that

$a_1v_1 + \cdots + a_nv_n = 0$. Equivalently, they are linearly dependent if one of the vectors can be written as a linear combination of the others.

A set B of linearly independent vectors in V is said to be a *basis* of V if every vector in V can be written as a linear combination of vectors in B. The *dimension* of V is the cardinality of B, denoted $\dim V = |B|$. It can be proven that every basis of a given vector space V has the same cardinality, so the dimension of a vector space is independent of a chosen basis. Furthermore, a basis is the largest set of linearly independent vectors that can be constructed. Adding any non-zero vector to a basis will lead to a set of linearly dependent vectors.

Next, I need a way to construct a vector space from an arbitrary set and a given field. This can be done by declaring the elements of the set to be vectors and writing an arbitrary vector as a linear combination of the elements from the set, each element multiplied by a scalar from the field. This leads to the idea of a formal sum.

Definition 30 (Formal sum)**.** Let S be a non-empty set of finite cardinality and $\mathbb{F}$ a *field*. By a *formal sum*, I mean an expression of the form

$$a_1s_1 + a_2s_2 + \cdots + a_ns_n,$$

where $a_1, \ldots, a_n \in \mathbb{F}$ and $s_1, \ldots, s_n \in S$. This is to be treated as just a sequence of symbols, along with a few natural rules:

$$as_1 + as_2 = a(s_1 + s_2)$$

and

$$a_1 s + a_2 s = (a_1 + a_2)s.$$

This will give us a vector space if we add a zero vector, known as the vector space freely generated by S.

3.4.2 Simplicial Homology

With the notion of a formal sum, I can construct a vector space over a simplicial complex. This sounds like an odd thing to do, but it will allow for the study the topological properties of a simplicial complex using the powerful methods of algebra. Starting with formal sums of p-complexes, I will build up to a vector space whose dimensionality provides information about the connected components, holes, and higher dimensional analogues of a complex.

Definition 31 (Chains)**.** Let K be a simplicial complex and F a field. A p-chain c_p of simplices in K over F is a formal sum:

$$c_p = \sum_{i=0}^{n} \alpha_i \sigma_i$$

where $\alpha_i \in F$ and σ_i is a p-simplex in K for all $i = 0, \ldots, n$. The set of p-chains forms a vector space $C_p(K)$. The dimension of this vector space is exactly the number of p-simplices in the complex.

The important thing about a simplicial complex is that each p-simplex (for $p > 0$) will also contain its faces. These faces are $p-1$-simplices. In the algebraic construction of p-chains, this means there should be a way to go from a p-chain to a $p-1$-chain. This is

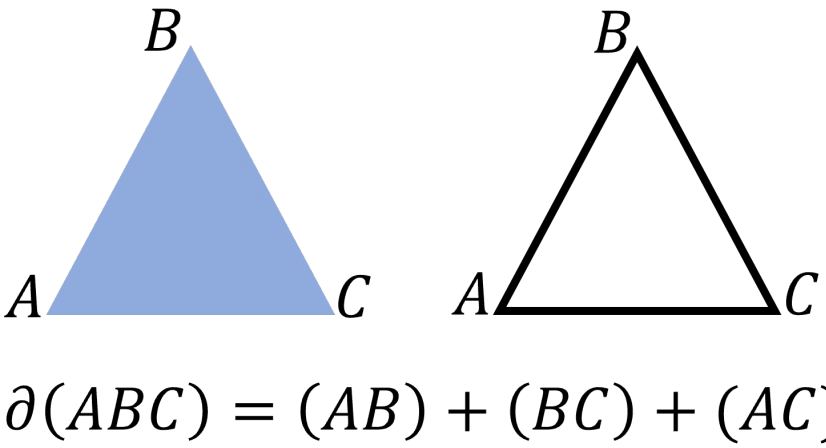


Figure 3.6: An example of the boundary operator. The simplex (ABC) on the left has the boundary $AB + BC + CA$ shown on the right.

the idea of the boundary of a chain.

Definition 32. Given a simplicial complex K and the vector space $C_p(K)$, the *boundary* $\partial_p(\sigma)$ of a p-simplex σ to be the alternating sum of the $(p-1)$-simplices comprising its faces. Then the boundary of the p-chain c_p is given by:

$$\partial_p(c_p) = \sum_{i=0}^{n} \alpha_i \partial(\sigma_i).$$

The boundary operator in action is illustrated on a 2-simplex in Figure 3.6. This shows how the intuitive idea of a boundary is captured by the boundary operator; viewed as topological spaces, the triangle ABC is bounded by the three edges AB, BC, and CA.

The boundary operator is a linear transformation $\partial_p : C_p(K) \to C_{p-1}(K)$ and satisfies the property $\partial_{p-1} \circ \partial_p = 0$, as can be proved by looking at its action on a single p-simplex. For example, given a triangle, taking its boundary twice will give a sum of points, which all cancel out due to the negative signs introduced by the alternating sums. The images of ∂_{p+1} then have the important property that they vanish under the boundary operation. However, it may be possible to have p-chains which vanish under the boundary which aren't the

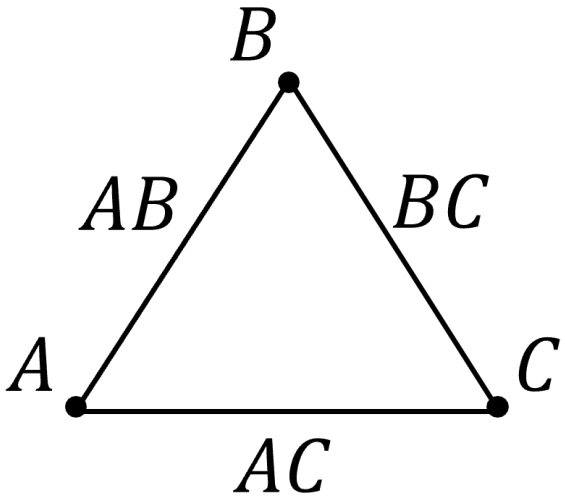


Figure 3.7: An example of a cycle. The chain $AB+BC+CA$ vanishes under the boundary operator, since the boundary ends up being $A - B + B - C + C - A = 0$.

image of some $(p + 1)$-chain. This motivates the following definition.

Definition 33 (Cycle, Boundary, Homology group)**.** A p-chain c is called a *cycle* if $\partial_p(c) = 0$. The kernel of the boundary operator,

$$Z_p(K) \coloneqq \{c \in C_p(K) | \partial_p(c) = 0\},$$

is the subspace of $C_p(K)$ consisting of the cycles.

A p-chain c is called a *boundary* if there exists a $(p+1)$-chain $c' \in C_{p+1}(K)$ such that $c = \partial_{p+1}c'$. The image of the boundary operator,

$$B_p(K) \coloneqq \{\partial_{p+1}(c) | c \in C_{p+1}(K)\},$$

is the subspace of $C_p(K)$ consisting of the boundaries.

The image of ∂_{p+1}, $B_p(K) = \{\partial_{p+1}(c_{p+1}) | c_{p+1} \in C_{p+1}(K)\}$ is a subspace of $C_p(K)$

whose elements are called boundaries. Furthermore, every boundary is a cycle, as $\partial_{p-1} \circ \partial_p = 0$, so that $B_p(K)$ is a subspace of $Z_p(K)$.

A cycle is shown in Figure 3.7. Note that this cycle is not a boundary, since there is no 2-simplex (ABC) that the cycle could be the boundary of.

A boundary isn't particularly interesting in a simplicial complex, as one can expect to find one for any p-simplex when $p > 0$. However, a cycle that *isn't* a boundary is interesting. For example, consider three edges that each share a point, such as in Figure 3.7. This makes up an un-filled triangle, and can be considered as a 2-dimensional "hole" in the space represented by the complex. Therefore, I want a construction that takes this into account. This leads to the definition of a homology group.

Definition 34. The p-th homology group $H_p(K)$ for non-negative integers p is the quotient vector space:

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

It consists of equivalence classes of cycles, where the equivalence relation is defined as follows: two cycles $c, c' \in Z_p(K)$ are considered equivalent if there exists a boundary $b \in B_p(K)$ such that $c = c' + b$. Then these two are the same equivalence class: $[c] = [c']$. Note that the equivalence class of the zero vector

The dimension of $H_p(K)$ (viewed as a vector space) is called the p-th Betti number $\beta_p = \dim H_p(K)$.

The first few Betti numbers have simple interpretations; β_0 is the number of connected components in the simplicial complex K. β_1 is the number of "loops," or unfilled closed paths up to the equivalence defined by $B_1(K)$. Furthermore, elements in a basis for $H_p(K)$ correspond directly to the features counted by the Betti numbers.

3.4.3 Persistent Homology

Persistent homology tracks the features in the homology groups $H_p(K)$ across linear transformations. The basic structure is that of a persistence module. This is a more general algebraic structure which arises when studying the homology of spaces connected by maps.

Definition 35 (Persistence Module)**.** Given a field F, a *persistence module* $\mathbb{V}$ is a sequence of vector spaces V_i over F connected by linear maps f_i:

$$V_1 \xrightarrow{f_1} V_2 \xrightarrow{f_2} \ldots \xrightarrow{f_{i-1}} V_i \xrightarrow{f_i} \ldots$$

In practice, I use finite persistence modules which are sequences $V_1 \to \cdots \to V_n$. Given such a persistence module, for $1 \leq i < j$ the (i, j)*-th persistent homology*, denoted $V^{i \to j}$, is the image of the compound transformation $f^{i \to j} : V_i \to V_j$ given by $f^{i \to j} = f_j \circ f_{j-1} \circ \cdots \circ f_{i+1} \circ f_i$. For our purposes, all vector spaces in a persistence module are assumed to be finite-dimensional.

A persistence module arises naturally when one has a sequence of simplicial complexes K_i related by inclusion, known as a *filtration*:

$$K_0 \subseteq K_1 \subseteq \cdots \subseteq K_i \subseteq \ldots$$

Applying the homology functor H_* to the complexes for a given p gives a persistence module:

$$H_p(K_0) \to H_p(K_1) \to \cdots \to H_p(K_i) \to \ldots$$

with the linear transformations being induced on the chain complexes by the inclusion relations between the simplicial complexes. More generally, one can define a persistence

module on a sequence of simplicial complexes related by simplicial maps.

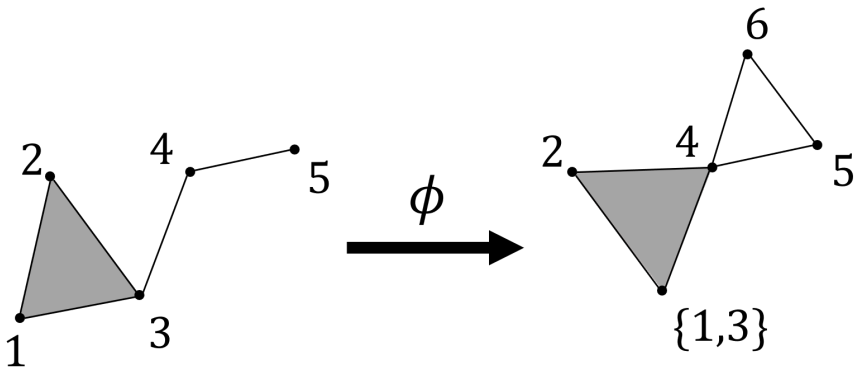


Figure 3.8: An example of a simplicial map with a vertex collapse and embeddings.

Definition 36 (Simplicial map)**.** Let K and L be simplicial complexes with respective vertex sets V_K and V_L. A *simplicial map* ϕ from K to L is a mapping of the vertex sets $\phi : V_K \to V_L$ such that for all simplices $\sigma \in K$, we have that the image $\phi(\sigma) \in L$.

An example of a simplicial map is given in Figure 3.8. Vertices with the same label are mapped to each other, except for 1 and 3 which are merged to $\{1, 3\}$ in a vertex collapse. This collapses the edge $\{1, 3\}$ to a single point and turns the 2-simplex $\{1, 2, 3\}$ into the 1-simplex $\{2, \{1, 3\}\}$. There are also simplices added in the codomain, which illustrate the embedding aspect of a simplicial map.

If one starts with a sequence $K_1 \xrightarrow{\phi_1} K_2 \xrightarrow{\phi_2} \ldots \xrightarrow{\phi_{n-1}} K_n$ of simplicial complexes K_i with simplicial maps $\phi_i : K_i \to K_{i+1}$, when homology is computed each simplicial map becomes a linear transformation $H_p(\phi_i) : H_p(K) \to H_p(L)$, yielding a persistence module.

Proposition 1. *Any finite persistence module $\mathbb{V}$ with coefficients in a field F can be decomposed uniquely up to ordering as a direct sum:*

$$\mathbb{V} = \bigoplus_{s=1}^{m} \mathbb{I}(b_s, d_s)$$

where the interval persistence modules $\mathbb{I}(b_s, d_s)$ *have as spaces:*

$$I_i = \begin{cases} F, & \text{if } b_s \leq i \leq d_s \\ 0, & \text{otherwise} \end{cases}$$

and homomorphisms are the identity between copies of F *and 0 otherwise [14]. The* b_s *are called the* birth *times and the* d_s *the* death *times. Direct sums of persistence modules are defined in the obvious way - we take the direct sums of corresponding pairs of vector spaces in the modules, and the sums of the corresponding linear transformations.*

Proof. Given in [20]. □

Using this decomposition, one can produce a representation of the persistence module as a *barcode* [36]. This will be a set of line segments in the plane, where the horizontal axis is the index of the persistence module. The vertical axis is an arbitrary ordering of the interval modules in the decomposition. If one has an interval $\mathbb{I}(b_s, d_s)$, then there is a horizontal line from b_s to d_s on the plot. An equivalent representation is that of a *persistence diagram*. This is a plot in the (x, y)-plane where one plots a point at each birth-death pair (b_s, d_s). All points will be at or above the line $y = x$ as $d_s \geq b_s$. One can define a distance metric on persistence diagrams and use this to prove a stability result for persistent homology; see [18].

Now consider a sequence of simplicial complexes $K_1 \to K_2 \to \cdots \to K_n$ related by simplicial maps. Applying homology yields persistence modules $H_p(K_1) \to \cdots \to H_p(K_n)$ for all non-negative integers p. If we decompose these persistence modules, the intervals $\mathbb{I}(b, d)$ have the following interpretation: each interval represents a generator of the homology group (equivalence class of cycles modulo boundaries) which is born at time b in the group $H_p(K_b)$ and is mapped to 0 ("dies") by the homomorphism going from $H_p(K_d)$.

For $p = 0$, the intervals represent connected components of the simplicial complexes, which die as they are merged by the addition of 1-simplices connecting them. For $p = 1$, the intervals are unfilled loops, which are born as new cycles form from the addition of edges and die as these cycles are filled by 2-simplices.

3.4.4 Zigzag Persistence

Zigzag modules generalize the persistence module of Definition 35 to allow the maps to go in either direction. They are defined in [12] and an algorithm for computing zigzag persistence for real-valued functions is given in [13].

Definition 37 (Zigzag module)**.** A *zigzag module* is a sequence $\mathbb{V}$ of vector spaces $(V_1, \ldots, V_n)$ which form a diagram:

$$V_1 \overset{p_1}{\longleftrightarrow} V_2 \overset{p_2}{\longleftrightarrow} \ldots \overset{p_{n-1}}{\longleftrightarrow} V_n$$

where each $\overset{p_i}{\longleftrightarrow}$ represents either a linear map $V_i \overset{f_i}{\longrightarrow} V_{i+1}$ going forward or a map $V_i \overset{g_i}{\longleftarrow} V_{i+1}$ going backward. The sequence of symbols f or g is called the *type* of $\mathbb{V}$, and the length of a type τ is the length of the zigzag diagram, i.e. the number of vector spaces. A persistence module is a zigzag module of type $ff \ldots f$, i.e. with only forward maps. A zigzag module $\mathbb{V}$ of type τ will also be referred to as a *τ-module*.

Two zigzag modules $\mathbb{V}$ and $\mathbb{W}$ of the same type τ can be composed into the direct sum $\mathbb{V} \oplus \mathbb{W}$, which is a zigzag module with vector spaces $V_i \oplus W_i$ and forward and backward maps $f_i \oplus h_i$ and $g_i \oplus k_i$, where forward maps in $\mathbb{W}$ are written h_i and backward maps k_i. A submodule $\mathbb{W}$ of a τ-module $\mathbb{V}$ is a τ-module such that each space W_i is a subspace of V_i and $f_i(W_i)$ is a subspace of W_{i+1} or $g_i(W_{i+1})$ a subspace of W_i for all $1 \leq i \leq n$. A zigzag

module is *decomposable* if it can be written as the direct sum of nonzero submodules, and *indecomposable* otherwise.

The features in a zigzag module are represented by indecomposable interval submodules, whose respective endpoints are the birth and death times of the feature. By decomposing a zigzag module into interval submodules, one can produce a summary of the module as a persistence diagram or barcode. This generalizes the results from the original theory of persistent homology, although the construction is more complex.

Definition 38 (Interval τ-module)**.** Let τ be a type of length n, F a field, and b, d integers such that $1 \leq b \leq d \leq n$. The *interval* τ-module with birth time b and death time d, written $\mathbb{I}_\tau(b, d)$, has as spaces:

$$I_i = \begin{cases} F, & \text{if } b \leq i \leq d, \\ 0, & \text{otherwise} \end{cases}$$

Its maps are identity maps between adjacent copies of F, and the zero map otherwise. The type is usually implicit, so the interval module will be written $\mathbb{I}(b, d)$.

Proposition 2. *Every τ-module can be written as a direct sum of interval submodules. Equivalently, the indecomposable τ-modules are exactly the interval modules $\mathbb{I}_\tau(b, d)$, where $1 \leq b \leq d \leq n$.*

Proof. See [12]. □

This means that any τ-module can be described uniquely up to isomorphism by the unordered list of the birth and death times of the interval submodules in its decomposition.

Topological Hierarchies

This chapter introduces the principal contributions of the dissertation. I introduce generalizations of maps of covers to allow the maps to be between a subspace and its superspace. Nerve-like maps allow one to describe constructs such as the nerve and Vietoris-Rips complex using the same language. Finally, these are used to introduce the main definition, that of an (abstract) topological hierarchy. I give examples of topological hierarchies, and introduce the related notion of an indexed hierarchy, where each group is associated with a real resolution or scale parameter.

Next, I describe how to build a topological hierarchy algorithmically. Based on the notion of a tower from the Multiscale MAPPER construction, I describe how to build a topological hierarchy based on decomposing a space into connected components through simplicial complexes from MAPPER, leading to the topological hierarchical decomposition algorithm. The mathematical study of topological hierarchies concludes with a short discussion on how to apply persistent homology to them.

Finally, I compare topological hierarchies and THD in particular to hierarchical clustering. I describe how to cut a topological hierarchy based on a distance measure to obtain a flat clustering. Using adjust Rand scores and adjust mutual information metrics, I compare the results of THD with hierarchical clustering on the HELOC dataset, showing that they perform comparably over a range of distances.

I conclude the chapter with a theoretical time complexity analysis of MAPPER and THD. I find a worst case $O(n^2)$ complexity for MAPPER and $O(n^3)$ for THD, where n is the number of points in the data to decompose. There is also a short discussion on potential optimizations when computing THDs, and a discussion on space complexity when computing THDs.

4.1 Preliminaries

The notion of a topological hierarchy is motivated by the construction known as the subspace topology. This is used when one has a subset of a topological space and wish to induce a topology on it. With the subspace topology, one can relate structure on the subspace back to the parent.

Definition 39 (Subspace topology)**.** Let X be a topological space and $A \subseteq X$ a subset of it. A can be made into a topological space by defining a topology on it as follows: a subset O of A is said to be open if $O = U \cap A$ for some open set U in X. That is, the topology T is precisely the open sets of X restricted to A:

$$T = \{U \cap A | U \subset X \text{ and } U \text{ is open in } X\}$$

The topology T is said to be the *subspace topology* of A and A as *subspace* of X.

It is important to note that if one has the relations $B \subseteq A \subseteq X$, one can continue to define the subspace topology on B. This will be the same topology whether one uses the subspace topology defined on A or the original topology on X when constructing the subspace topology on B. Therefore, one can start with an original topological space X and iteratively decompose it into subsets, each time using the subspace topology.

I require further structure on X and its subspace A in order to be able to define persistence. The first step is to describe how one can induce a cover on A given one on X. The cover on A will have nice properties relating it to the one on X.

Definition 40 (Induced open covers)**.** Let X be a topological space, A a subset of X endowed with the subspace topology, and $\mathcal{U} = \{U_i\}_{i=1}^{t}$ an open cover of X. The *open cover*

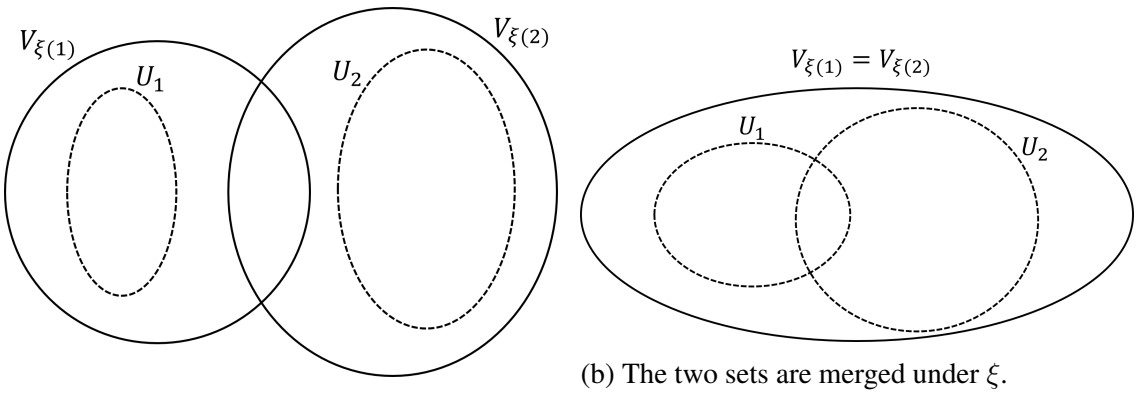
of A induced by $\mathcal{U}$, denoted $\mathcal{U}_{|A}$, has as its members:

$$V_i = U_i \cap A$$

for $i = 1, \ldots, t$. It is clear that this is indeed an open cover of A as each element of $\mathcal{U}_{|A}$ is open by the definition of subspace topology, and the union of all the elements is exactly A since $\mathcal{U}$ covers X. Furthermore, we have $V_i \subseteq U_i$ for all $i = 1, \ldots, t$.

For example, the induced open cover on Figure 3.4 can be obtained by taking the intersection of the open sets with the orange-filled subspace.

The notion of a map of covers is used to define a multiresolution structure for MAPPER (see [84, 24]) and motivates one of our constructions used to describe persistence from a subspace to a parent. The relationship described by this map of covers passes over to one when the nerve is applied, giving what is known as a simplicial map. Once one has simplicial maps, there is a known way to compute persistence by the methods of [23]. However, I must modify the definition slightly.



(a) The two sets are not merged under ξ. $\xi(1) \neq \xi(2)$.

(b) The two sets are merged under ξ. $\xi(1) = \xi(2)$.

Figure 4.1: Examples of simple maps of covers $\xi : \mathcal{U} \to \mathcal{V}$. Sets in $\mathcal{U}$ are outlined with dashed lines, and sets in $\mathcal{V}$ with solid lines.

If one has a topological space X with a subspace A and an open cover $\mathcal{U}$ on X, the

relationship between $\mathcal{U}_{|A}$ and $\mathcal{U}$ is almost that of a map of covers, where the map ξ between the index sets is the identity, as $U_i \cap A \subseteq U_i$ for all i. However, the covers are not strictly on the same space, since A may be a strict subset of X. Therefore, I extend the definition of map of covers so that the space $\mathcal{U}$ covers in the definition is allowed to be a subspace of X.

Definition 41 (Map of covers)**.** Let X be a topological space, Y a subspace of X (possibly X itself), $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ an open cover of Y, and $\mathcal{V} = \{V_\beta\}_{\beta \in B}$ an open cover of X. A *map of covers* is a function $\xi : A \to B$ such that for all $\alpha \in A$ we have $U_\alpha \subseteq V_{\xi(\alpha)}$. Sometimes I will stretch the use of notation and write $\xi(U_\alpha)$ for $V_{\xi(\alpha)}$, i.e. I will also let ξ represent the mapping $\mathcal{U} \to \mathcal{V}$. Note that this reduces to the normal definition of map of covers when $Y = X$.

Example maps of covers are shown in Figure 4.1. It is possible to have two distinct sets in $\mathcal{U}$ mapped to the same set in $\mathcal{V}$ and this is shown. If the nerve operator is applied, this leads to a vertex collapse. A situation where distinct sets in $\mathcal{U}$ which do not intersect are mapped to distinct sets in $\mathcal{V}$ which do intersect is also shown. Under the nerve, this leads to two vertices with no connection using $\mathcal{U}$ and an edge between the vertices with $\mathcal{V}$.

Finally, I want a mapping between simplicial complexes with properties that will allow us to obtain a persistence module when passing to homology. Traditional persistence uses a *filtration*, which is a growing sequence of simplicial complexes each a subset of the next one. A more general notion is that of a simplicial map, described in Definition 36, includes filtrations as a special case but also allows vertex collapses where a pair $\{u, v\}$ of distinct vertices are mapped to the same vertex $\phi(u) = \phi(v)$ under the map ϕ.

As I will show, the nerve serves to induce a simplicial map from maps of covers. However, there are constructions such as the Vietoris-Rips complex that are not nerves which also can yield simplicial maps. This motivates the next few definitions, which capture the properties that allow these operators to induce simplicial maps. The cover map concept

generalizes the associating of open sets with 0-simplices in the nerve, to allow for multiple open sets being mapped to the same 0-simplex.

Definition 42 (Cover map and embedding)**.** Let X be a topological space, $\mathcal{U}$ a finite open cover of X, and K a finite simplicial complex. A *cover map* is a surjective mapping $\Phi : \mathcal{U} \to V(K)$ such that any pair $U, U' \in \mathcal{U}$ satisfies $U \cap U' \neq \emptyset$ if and only if $\{\Phi(U), \Phi(U')\}$ is a simplex in K. A cover map is a *cover embedding* if it is a bijection.

For any cover map $\Phi : \mathcal{U} \to V(K)$, I define a *pseudoinverse* $\Phi^* : V(K) \to \mathcal{U}$ as a selection for each $v \in V(K)$ some $U \in \mathcal{U}$ so that $\Phi(U) = v$, yielding $\Phi^*(v) = U$. This is possible since Φ is surjective, so there will always be at least one U to choose from for any given v; The composition $\Phi \circ \Phi^*$ is the identity function on $V(K)$. The pseudoinverse is an actual inverse for cover embeddings since there is only one choice of U for each v.

A cover map describes a function that take sets in open covers to vertices in a simplicial complex. Two open sets that have a common intersection are either mapped to the same vertex or have vertices sharing an edge in the complex. Both the nerve and Vietoris-Rips constructions define cover embeddings, as open sets correspond directly to vertices for both constructions. Furthermore, any two open sets are joined by an edge if and only if they have a common intersection in both constructions.

Note that the psuedoinverse is not necessarily unique - if one has $\Phi(U_1) = v = \Phi(U_2)$, then the choice of $\Phi^*(v)$ as either U_1 or U_2 is arbitrary. In order to ensure there's a unique inverse, I need to redefine the open cover so that the cover map on it becomes injective. This motivates the following definition.

Definition 43 (Φ-reduced cover)**.** Let X be a topological space, $\mathcal{U}$ an open cover on X, K a finite simplicial complex, and Φ a cover map from $\mathcal{U}$ to K. For each $v \in V(K)$, define

the open set

$$U_v = \bigcup_{U \in \Phi^{-1}(v)} U$$

Then the Φ-*reduced open cover* $\mathcal{U}_\Phi$ is given by $\mathcal{U}_\Phi = \{U_v\}_{v \in V(K)}$. Furthermore, I define $\bar{\Phi} : \mathcal{U}_\Phi \to V(K)$ by $\bar{\Phi}(U_v) = v$.

It is clear that $\bar{\Phi}$ is injective. It is also surjective, a property inherited from Φ. Therefore it is bijective, making it a cover embedding from $\mathcal{U}_\Phi$ onto $V(K)$. If one defines $i : \mathcal{U} \to \mathcal{U}_\Phi$ by

$$i(U) = \bigcup_{V \in \Phi^{-1}(\Phi(U))} V,$$

then one can write $\Phi = \bar{\Phi} \circ i$. A choice of pseudoinverse Φ^* amounts to choosing a right inverse for i, i.e. choosing an open set U' in $\mathcal{U}$ for each open set U_v in $\mathcal{U}_\Phi$, $\Phi(U') = v$. Note that i is a map of covers as well.

Definition 44 (Nerve-like map)**.** Given a topological space X, a *nerve-like map* is an operation F that (i) Assigns to an open cover $\mathcal{U}$ of X or any subspace of X a simplicial complex $F(\mathcal{U})$ with a corresponding cover map $\Phi_\mathcal{U}$ and a pseudoinverse $\Phi^*_\mathcal{U}$ chosen for each U. (ii) Satisfies the following property: given a map of covers $\xi : \mathcal{U} \to \mathcal{V}$ between covers on X and its subspaces, the map $\Phi_\mathcal{V} \circ \xi \circ \Phi^*_\mathcal{U}$ from $V(F(\mathcal{U}))$ to $V(F(\mathcal{V}))$ is a simplicial map.

Remark 1. I note that a nerve-like map is a functor when its cover maps are injective, i.e. cover embeddings. Any cover map takes the identity map of covers to the identity simplicial map, since $\Phi_\mathcal{U} \circ \text{ id } \circ \Phi^*_\mathcal{U} = \Phi_\mathcal{U} \circ \Phi^*_\mathcal{U} = \text{id}$. However, the condition $F(\xi' \circ \xi) = F(\xi') \circ F(\xi)$ only holds when the cover maps are injective as it requires $\Phi^*_\mathcal{U} \circ \Phi_\mathcal{U} = \text{id}$, which is equivalent to the statement that $\Phi_\mathcal{U}$ is injective.

When a nerve-like map is a functor, I will call it a *simplicial functor*. The domain

category has as objects pairs $(X, \mathcal{U})$ where X is a topological space and $\mathcal{U}$ a cover of X. Morphisms are maps of covers $\mathcal{U} \to \mathcal{V}$ from an open cover $\mathcal{U}$ on a subspace of a space that is covered by $\mathcal{V}$. The codomain category $\mathbf{Csim}$ has as objects simplicial complexes and morphisms simplicial maps [31].

Hereafter, I assume that either any cover map I discuss is a cover embedding, or that the open cover is replaced with the Φ-reduced open cover, so that a cover embedding can be obtained from any nerve-like map up to redefinition of the open cover under a map of covers.

Proposition 3. *The nerve and Vietoris-Rips constructions are nerve-like maps with cover embeddings.*

Proof. This proof is for the nerve; the Vietoris-Rips construction is nearly the same. Let X be a topological space with cover $\mathcal{V}$, Y a subspace of X with cover $\mathcal{U}$, and $\xi : \mathcal{U} \to \mathcal{V}$ a map of covers. Let $K = \mathcal{N}(\mathcal{U})$ and $L = \mathcal{N}(\mathcal{V})$ be the nerves. The nerve defines a cover embedding - the 0-simplices correspond exactly to the open sets, and by definition of the nerve $\{i, j\}$ is a 1-simplex iff $U_i \cap U_j \neq \emptyset$. This implies that $\Phi^*_{\mathcal{U}} = \Phi^{-1}_{\mathcal{U}}$ for the nerve as well.

Now suppose that $\{v_0, v_1, \ldots, v_p\}$ is a p-simplex in $\mathcal{N}(\mathcal{U})$ and let $U_i = \Phi^{-1}_{\mathcal{U}}(v_i)$ for $0 \leq i \leq p$. Then $\bigcap_{0 \leq i \leq p} U_i \neq \emptyset$ by definition of the nerve. Since $U_i \subseteq \xi(U_i)$ for all i, one has $\bigcap_{0 \leq i \leq p} V_i \neq \emptyset$ where $V_i = (\xi \circ \Phi^{-1}_{\mathcal{U}})(v_i)$. This implies that $\Phi_{\mathcal{V}}(\{V_0, \ldots, V_p\})$ is a k-simplex in $\mathcal{N}(\mathcal{V})$, for some $0 \leq k \leq p$. Therefore, the composition $\Phi_{\mathcal{V}} \circ \xi \circ \Phi^{-1}_{\mathcal{U}}$ is simplicial, showing that the nerve is a nerve-like map. □

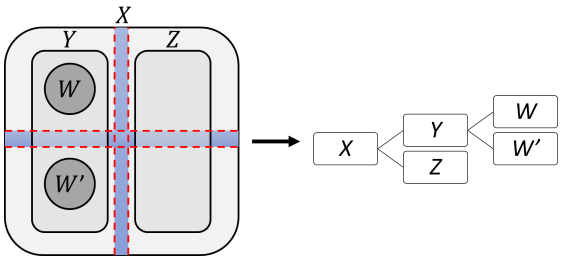
4.2 Definition

I now have the preliminaries to define a topological hierarchy. This will be a recursive partitioning of a topological space, where for each partitioning one has maps of covers going from the partitions to the space that was partitioned. Intuitively, one can think of it as a hierarchical clustering with topological information associated to each cluster. However, the splits need not always be binary for a topological hierarchy.

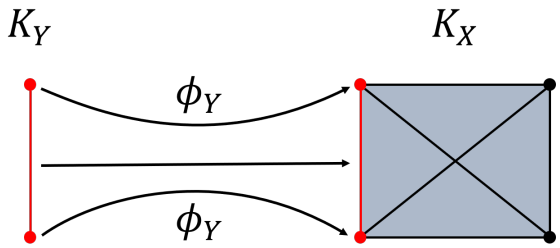
Definition 45 (Topological hierarchy)**.** Given a topological space X, a *topological hierarchy* is a tuple $(\mathcal{X}, p, c, \mathcal{U}_*, \xi_*, F)$ defining a tree structure on $\mathcal{X}$, where:

- $\mathcal{X}$ is a collection of subsets of X, each with the subspace topology.
- $p : \mathcal{X} \to \mathcal{X}$ is a function giving for each $Y \in \mathcal{X}$ its *parent* $p(Y)$. I define $p(X) = X$, so that we have $Y \subseteq p(Y)$ for any $Y \in \mathcal{X}$.
- $c : \mathcal{X} \to 2^{\mathcal{X}}$ is a function giving for each $Y \in \mathcal{X}$ its *children* $c(Y)$ so that for all $Y \in \mathcal{X}$ either $c(Y) = \emptyset$ or $\bigcup_{Z \in c(Y)} Z = Y$. Also, for any $Z, W \in c(Y)$ then $Z = W$ or $Z \cap W = \emptyset$. That is, either Y is partitioned by its children or it has no children.
- $\mathcal{U}_*$ assigns to each $Y \in \mathcal{X}$ an open cover $\mathcal{U}_Y$ of Y.
- ξ_* assigns to all $Y \in \mathcal{X}$ a map of covers $\xi_Y : \mathcal{U}_Y \to \mathcal{U}_{p(Y)}$. ξ_X is the identity map.
- A nerve-like map F which yields simplicial maps $\phi_Y = \Phi_{\mathcal{U}_{p(Y)}} \circ \xi_Y \circ \Phi^*_{\mathcal{U}_Y}$ from the maps of covers ξ_Y for all $Y \in \mathcal{X}$.

There is some further terminology I will use throughout the rest of this chapter. The members of $\mathcal{X}$ are called the *nodes* or *groups* of the topological hierarchy. For any $Y \in \mathcal{X}$,



(a) A topological space X and subspaces represented as a hierarchy. There is an open cover consisting of four overlapping sets on the spaces.



(b) The simplicial complexes $K_Y = \mathcal{N}(\mathcal{U}_Y)$ and $K_X = \mathcal{N}(\mathcal{U}_X)$ and the induced simplicial map $K_Y \to K_X$.

Figure 4.2: An illustration of Example 6. On the left, the spaces and a hierarchical representation are shown. On the right, the simplicial complexes for Y and X are shown with arrows indicating the induced simplicial map.

I say that Y is a *leaf node* if it has no children ($c(Y) = \emptyset$). Otherwise, I say there is a *split* at the node Y. Sometimes I will only want to talk about the hierarchy $(\mathcal{X}, p, c)$ as well.

Example 6. For an example of how to make an existing hierarchical structure into a topological hierarchy, suppose we have a metric space X with an open cover $\mathcal{U}$. If one runs a hierarchical clustering algorithm on X, the result is a set of clusters $\mathcal{X}$ with parent/children functions p and c. For the open covers $\mathcal{U}_*$, choose the induced covers (Definition 40) $\mathcal{U}_{|Y}$ for each $Y \in \mathcal{X}$. The maps of covers ϕ_Y are defined by the inclusion of the sets of $\mathcal{U}_{|Y}$ in the sets of $\mathcal{U}_{|p(Y)}$. Take the nerve $\mathcal{N}$ to be the nerve-like map, which induces the simplicial maps.

This example is illustrated in Figure 4.2. In Figure 4.2a, the parent space X and subspaces Y, Z, W, and W' are shown on the left with dashed lines representing the boundaries of open sets in the open cover $\mathcal{U}$ of X. In order to get interesting simplicial complexes, I assume the open sets overlap some along these boundaries. On the right of the subfigure is the hierarchical representation of this structure as a tree. The right subfigure, Figure 4.2b, shows the simplicial complexes $K_X = \mathcal{N}(\mathcal{U})$ and $K_Y = \mathcal{N}(\mathcal{U}_Y)$ and the induced simplicial map $\phi_Y : K_Y \to K_X$. The subspace Y is split into two overlapping

components by the open cover, so K_Y consists of two nodes with an edge between them. The parent space X is split into four overlapping components (I assume there's a common intersection between all four), and so K_X has four nodes with a 3-simplex (tetrahedron) and all of its faces. The simplicial map $\phi_Y : K_Y \to K_X$ embeds the nodes and edge of K_Y into one of the edges of K_X.

In order to discuss topological hierarchies with Čech and Vietoris-Rips complexes, I define a hierarchy with a positive real number associated to each node that either only increases or only decreases as one goes up the hierarchy. These parameters will be the radii of the open balls used in the construction of the simplicial complexes.

Definition 46 (Indexed hierarchy)**.** Given a set X, an *indexed hierarchy* is a hierarchy $(\mathcal{X}, p, c)$ with some real number $\epsilon_Y > 0$ for each $Y \in \mathcal{X}$ satisfying exactly one of the following:

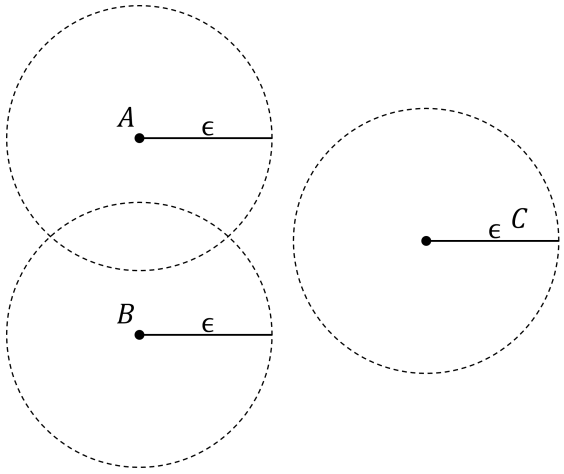
- For all $Y \in \mathcal{Y}$, $\epsilon_Y \leq \epsilon_{p(Y)}$, or
- For all $Y \in \mathcal{Y}$, $\epsilon_Y \geq \epsilon_{p(Y)}$ for all $Y \in \mathcal{Y}$.

The first case will be called an *increasing* indexed hierarchy (viewed as going from child to parent) and the second case a *decreasing* indexed hierarchy.

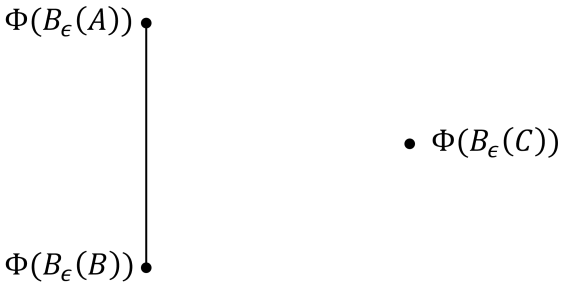
Example 7. Suppose one starts with an increasing indexed hierarchy $(\mathcal{X}, p, c, \epsilon_*)$. For each group $Y \in \mathcal{X}$, assign an open cover $\mathcal{U}_Y$ whose open sets are of the form:

$$U_y = B_{\epsilon_Y}(y)$$

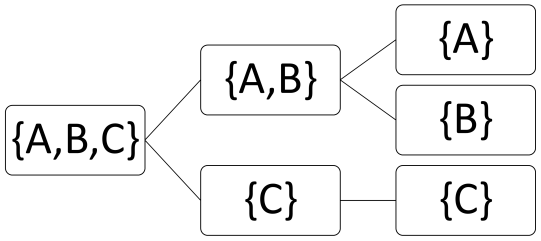
for each $y \in Y$. We have a map of covers, as $Y \subseteq p(Y)$ and $\epsilon_Y \leq \epsilon_{p(Y)}$ means that $B_{\epsilon_Y}(y) \subseteq B_{\epsilon_{p(Y)}}(y)$ for each $y \in Y$. For the nerve-like map, one can either use the nerve



(a) The point-cloud data with open balls indicated for a specific value of ϵ.



(b) The Vietoris-Rips complex corresponding to the situation in Figure 4.3a. This produces two child groups, corresponding to the two connected components of the complex.



(c) The hierarchy resulting from the process. The complex shown in Figure 4.3a corresponds to the group A, B, C in the tree.

Figure 4.3: An example of building an indexed topological hierarchy with Vietoris-Rips complexes, following Example 7.

itself, in which case we get Čech complexes, or the operation used to construct Vietoris-Rips complexes. Unlike Example 6, we did not start with a cover on X and instead used the indexing to define one.

A specific case of this construction is shown in Figure 4.3. We start with three points $A, B, C \in \mathbb{R}^2$, and a value of ϵ that produces open balls as shown in Figure 4.3a. Using Vietoris-Rips as the nerve-like map produces the hierarchy of Figure 4.3c.

4.2.1 Computing a Topological Hierarchy

The overall goal of this section is to enable the computation of topological hierarchies from a point-cloud dataset using MAPPER. This leads to an algorithm I call topological hierarchical decomposition (THD). To do this we need a way to relate the pullback covers for each MAPPER by maps of covers, which will yield simplicial maps when the nerve is applied. One such construction is a tower, which I define specifically for open covers. This parametrizes the covers of Z by some scale parameter ϵ such that there are maps of covers from the covers with smaller ϵ to covers with larger ϵ. For example, if one uses overlapping intervals of the same size, the length of the intervals could be used as ϵ so long as they remain centered on the same points for each ϵ. Another example could start with several intervals and gradually merge them, and have ϵ be related to the reciprocal of the number of intervals.

Definition 47 (Tower of covers)**.** A *tower of covers* of a topological space Z with minimum resolution $r \in \mathbb{R}$ is a collection $\mathfrak{U} = \{\mathcal{U}_\epsilon\}_{\epsilon \geq r}$ of open covers $\mathcal{U}_\epsilon$ of Z with maps of covers $u_{\epsilon,\epsilon'} : \mathcal{U}_\epsilon \to \mathcal{U}_{\epsilon'}$ such that $u_{\epsilon,\epsilon}$ is the identity function and $u_{\epsilon,\epsilon''} = u_{\epsilon',\epsilon''} \circ u_{\epsilon,\epsilon'}$ for all $r \leq \epsilon \leq \epsilon' \leq \epsilon''$.

If we have a tower of covers $\mathfrak{U}$ on Z and a continuous well-behaved map $f : X \to Z$, then when we take the pullback covers $f^*(\mathcal{U}_\epsilon)$ we obtain maps of covers $f^*(u_{\epsilon,\epsilon'}) : f^*(\mathcal{U}_\epsilon) \to f^*(\mathcal{U}_{\epsilon'})$ for all $r \leq \epsilon \leq \epsilon'$. It can be shown that the resulting collection $f^*(\mathfrak{U}) = \{f^*(\mathcal{U}_\epsilon) | \epsilon \geq r\}$ is a tower of covers; see [24] for the necessary properties of pullbacks. Let us verify that the properties of a tower hold for our extended definition of maps of covers for the subspace-induced covers.

Proposition 4. *If we have a tower of covers $\mathfrak{U}$ of a topological space X with minimum resolution $r \in \mathbb{R}$ and subspaces $Z \subseteq Y \subseteq X$, then for all $r \leq \epsilon \leq \epsilon' \leq \epsilon''$ we have that*

1. *$u_{\epsilon,\epsilon'}$ can be made into a map of covers of the subspace-induced covers $\mathcal{U}_{\epsilon|Z} \to \mathcal{U}_{\epsilon'|Y}$.*

2. *If we have the restricted maps of covers* $u_{\epsilon,\epsilon'} : \mathcal{U}_{\epsilon|Z} \to \mathcal{U}_{\epsilon'|Y}$, $u_{\epsilon',\epsilon''} : \mathcal{U}_{\epsilon'|Y} \to \mathcal{U}_{\epsilon''}$, *and* $u_{\epsilon,\epsilon''} : \mathcal{U}_{\epsilon|Z} \to \mathcal{U}_{\epsilon''}$, *then the property of a tower still holds, so that* $u_{\epsilon,\epsilon''} = u_{\epsilon',\epsilon''} \circ u_{\epsilon,\epsilon'}$.

Proof. Suppose that $\mathcal{U}_\epsilon$ is indexed by some set I, so that $\mathcal{U}_\epsilon = \{U_{\epsilon,i}\}_{i\in I}$. For the first statement, I define u to be the function $\mathcal{U}_{\epsilon|Z} \to \mathcal{U}_{\epsilon|Y}$ given by

$$u(U_{\epsilon,i} \cap Z) = U_{\epsilon',u_{\epsilon,\epsilon'}(i)} \cap Y$$

for all $i \in I$, where $U_{\epsilon',u_{\epsilon,\epsilon'}(i)}$ is the open set that $u_{\epsilon,\epsilon'}$ maps $U_{\epsilon,i}$ to. In order to reduce the number of subscripts, I will also refer to $u_{\epsilon,\epsilon'}$ as u from now on. We have that:

$$U_{\epsilon,i} \cap Z \subseteq U_{\epsilon',u(i)} \cap Z \subseteq U_{\epsilon',u(i)} \cap Y = u(U_{\epsilon,i} \cap Z)$$

with the first inclusion holding because the unrestricted u is a map of covers, and the second because Z is a subspace of Y. Therefore, the restricted u is a map of covers.

Now let $u = u_{\epsilon,\epsilon'}$, $u' = u_{\epsilon',\epsilon''}$, and $u'' = u_{\epsilon,\epsilon''}$. For their restrictions, we have:

$$\begin{aligned}
[u' \circ u](U_{\epsilon,i} \cap Z) &= u'\left[U_{\epsilon',u(i)} \cap Y\right] \\
&= U_{\epsilon'',(u'\circ u)(i)} \cap X \\
&= U_{\epsilon'',u''(i)} \cap X \\
&= u''(U_{\epsilon,i} \cap Z)
\end{aligned}$$

for all $i \in I$; the third equality holds because $u'' = u' \circ u$ for the unrestricted maps of covers. Thus, the second property is satisfied. □

This shows we can use a tower of covers with subspaces and still retain their proper-

ties. Note that we can restrict the same map of covers $u_{\epsilon,\epsilon'}$ in different ways, as long as the domain cover is restricted to a subspace of the range. I will use this to define a topological hierarchy from an indexed hierarchy using MAPPER-like constructions on towers of covers.

Example 8. Let X be a topological space, $\mathfrak{U}$ a tower of covers on X with minimum resolution r, and an increasing indexed hierarchy $(\mathcal{X}, p, c, \epsilon_*)$ on X, I construct a topological hierarchy as follows:

- For each group $Y \in \mathcal{X}$, I assign the open cover $\mathcal{U}_Y = \mathcal{U}_{\epsilon_Y|Y}$. The tower of covers gives us a map of covers $u_Y = u_{\epsilon_Y,\epsilon_{p(Y)}} : \mathcal{U}_Y \to \mathcal{U}_{p(Y)}$ restricted to the subspace-induced covers.

- For the nerve-like map, we use the nerve itself. Thus, for each group $Y \in \mathcal{X}$, we have a simplicial complex $\mathcal{N}(\mathcal{U}_Y)$, with a simplicial map induced by the map of covers $u_{\epsilon_Y,\epsilon_{p(Y)}}$ from the tower of covers, restricted to the subspace-induced covers.

For the specific case of MAPPER, the tower of covers $\mathcal{U}$ can come from the pullback of a tower of covers in the filter space Z under the continuous map $f : X \to Z$.

4.2.2 Topological Hierarchical Decomposition

I now present an algorithm which produces a topological hierarchy by decomposing a space using the 1-skeletons of simplicial complexes, known as topological hierarchical decomposition (THD). The essential feature of THD is that for each group Y in the resulting topological hierarchy, the children of Y are in a one-to-one correspondence with the connected components of the one-skeleton of the group's simplicial complex $F(\mathcal{U}_Y)$. First, I need a way to associate vertices in a simplicial complex with points in a topological space.

In order to state its properties, I make use of definitions from graph theory to talk of the connected components of a simplicial complex.

Definition 48 (Path in a simplicial complex)**.** Let K be a simplicial complex with vertex set $V(K)$. A *path* in K is a sequence of vertices $(v_1, \ldots, v_m)$ such that $\{v_i, v_{i+1}\}$ is a 1-simplex of K for all $1 \leq i < m$. Two vertices $v, v' \in V(K)$ are *connected* if there is a path $(v_1, \ldots, v_m)$ in K such that $v_1 = v$ and $v_m = v'$. If v and v' are connected, I will write $v \leftrightarrow v'$. It is clear that $\leftrightarrow$ is an equivalence relation on $V(K)$. I define the *connected components* of K to be the equivalence classes

$$[v] = \{v' \in V(K) | v \leftrightarrow v'\}$$

of connected vertices in K and the set of components to be the quotient:

$$C(K) = K/\leftrightarrow = \{[v] | v \in V(K)\}$$

Furthermore, we have for each connected component $[v]$ a subcomplex of K that consists of the vertices in $[v]$ and all their cofaces:

$$K[v] = \{\sigma \in K | \sigma \subseteq [v]\}$$

Note that in simplicial homology, elements of a basis of $H_0(K)$ correspond to its connected components, so that we have $\beta_0 = \dim H_0(K) = |C(K)|$. In order to more easily refer to the connected components themselves, I choose to use notation and concepts from graph theory over homology for defining THDs.

Definition 49 (Vertex membership function)**.** Let X be a topological space, Y a subset of X (possibly X itself), and K a simplicial complex. A *vertex membership function* for Y is a mapping $\mu : V(K) \to 2^Y$ from the vertex set of the complex K to subsets of Y such that

- For any pair of vertices $u, v \in V(K)$, if $\mu(u) \cap \mu(v) \neq \emptyset$, then $u \leftrightarrow v$.
- For any point $y \in Y$, there exists a vertex $u \in V(K)$ such that $y \in \mu(u)$.

The first property says that if u and v are not connected, then their membership sets $\mu(u)$ and $\mu(v)$ share no points. The second property guarantees that each point in y is represented by at least one vertex in K, and is equivalent to requiring that $\bigcup_{u \in V(K)} \mu(u) = Y$.

The points $y \in \mu(u)$ are said to be the *members* of the vertex u.

Example 9. Let X be a topological space, $\mathcal{U}$ a finite open cover of X, F a nerve-like map, and $K = F(\mathcal{U})$. If Y is a subset of X, define $\mu : V(K) \to 2^Y$ by

$$\mu(v) = Y \cap \bigcup_{U \in \Phi^{-1}(v)} U$$

for all $v \in V(K)$, where $\Phi : \mathcal{U} \to V(K)$ is the associated cover map. Suppose that $u, v \in K$ and $\mu(u) \cap \mu(v) \neq \emptyset$. Then:

$$\left(\bigcup_{U \in \Phi^{-1}(u)} U \right) \cap \left(\bigcup_{V \in \Phi^{-1}(v)} V \right) \neq \emptyset$$

so for some $U' \in \Phi^{-1}(u)$ and $V' \in \Phi^{-1}(v)$ we have $U' \cap V' \neq \emptyset$. Therefore, $\{u, v\}$ spans a 1-simplex, so u and v are connected. This makes μ a vertex membership function. I will call μ the *canonical (vertex) membership function* of the nerve-like map F for the open cover $\mathcal{U}$, and will indicate it by $\mu_{\mathcal{U}}$ for the open cover $\mathcal{U}$.

Proposition 5. *Let X be a topological space, Y a subset of X, K a finite simplicial complex, and $\mu : V(K) \to 2^Y$ a vertex membership function. Then μ can be used to form a partition of Y with sets $Y_u = \bigcup_{u' \in [u]} \mu(u')$.*

Proof. Let $u, v \in V(K)$ be any pair of vertices. If they are connected, $u \leftrightarrow v$ and $[u] = [v]$, so $Y_u = Y_v$. Now, assume u and v are not connected.

Since $C(K)$ is a partition of $V(K)$, $[u] \cap [v] = \emptyset$ so we will have that u' and v' are not connected for all $u' \in [u]$ and $v' \in [v]$. Furthermore, $[u]$ and $[v]$ will both be finite sets since I assumed that K was finite. We have that:

$$\begin{aligned} Y_u \cap Y_v &= \left(\bigcup_{u' \in [u]} \mu(u') \right) \cap \left(\bigcup_{v' \in [v]} \mu(v') \right) \\ &= \bigcup_{v' \in [v]} \left(\bigcup_{u' \in [u]} \mu(u') \right) \cap \mu(v') \\ &= \bigcup_{v' \in [v]} \bigcup_{u' \in [u]} (\mu(u') \cap \mu(v')) \\ &= \bigcup_{v' \in [v]} \bigcup_{u' \in [u]} \emptyset = \emptyset \end{aligned} \tag{4.1}$$

where I used the distributive property of union and intersection for the second and third equalities, and the first property of Definition 49 for non-connected vertices for the final equalities. Therefore, either $Y_u = Y_v$ or $Y_u \cap Y_v = \emptyset$. Finally, by the second property of Definition 49, the union of all the Y_u is exactly Y. This shows that $\{Y_u | u \in V(K)\}$ is a partition of Y. □

I now have the material to define a topological hierarchical decomposition, which will have membership functions on the associated simplicial complexes for each group, which determine the children of that group based on the connected components of the complexes.

Algorithm 1 Algorithm to decompose a topological space with a tower of covers, producing an increasing indexed topological hierarchy

Input
- X A topological space
- F A nerve-like map
- r The minimum resolution $r \in \mathbb{R}$
- $\mathfrak{U}$ A tower of covers on X
- ϵ_0 The largest (initial) resolution to start with. $\epsilon_0 \geq r$
- δ Amount to decrease resolution by if a split is not observed. $\delta > 0$

function SUBSPACETHD($X, F, \mathfrak{U}, r, \epsilon_0, \delta$) ▷ Build a hierarchy recursively
  $\mathcal{U}_X \leftarrow \mathcal{U}_{\epsilon_0|X}$ ▷ Cover induced by subspace at current resolution
  $K \leftarrow F(\mathcal{U}_X)$ ▷ Simplicial complex for this group
  node $\leftarrow$ new_node((X, K, ϵ_0))
  $C \leftarrow C(K)$ ▷ connected components
  if $|C| = 1$ **then** ▷ No split observed - change scale
    if $\epsilon_0 - \delta \geq r$ **then**
      child $\leftarrow$ SUBSPACETHD($X, F, \mathfrak{U}, r, \epsilon_0 - \delta, \delta$)
      return add_child(node, child)
    else
      return node ▷ Stop decomposing if resolution would be below r
    end if
  else ▷ Split observed - run SUBSPACETHD on children
    for L in C **do**
      $Y \leftarrow \bigcup_{u \in L} \mu_{\mathcal{U}_X}(u)$ ▷ Points in component
      child $\leftarrow$ SUBSPACETHD($Y, F, \mathfrak{U}, r, \epsilon_0, \delta$)
      node $\leftarrow$ add_child(node, child)
    end for
    return node
  end if
end function

Definition 50 (Topological hierarchical decomposition)**.** A *topological hierarchical decomposition* (THD) is a topological hierarchy $(\mathcal{X}, p, c, \mathcal{U}_*, \xi_*, F)$ such that for each group $Y \in \mathcal{X}$, there is a vertex membership function $\mu_Y : F(\mathcal{U}_Y) \to 2^Y$. Moreover, for all $Y \in \mathcal{X}$, I require that either Y has no children ($c(Y) = \emptyset$), or that the children of Y are exactly the partitions Y_u produced by μ_Y according to Proposition 5.

This definition leads to an algorithm, listed in Algorithm 1, for constructing a THD

given a topological space X, a tower of covers $\mathfrak{U}$ on X, and a nerve-like map F. There are a few additional parameters that are needed. Since I start from the parent group X and wish to decompose downwards, we need a starting resolution ϵ_0 for the tower of covers. This will be the largest resolution used, as it will only ever decrease or be unchanged for children groups. I only change the resolution if no split is observed, i.e. the simplicial complex for the group has one connected component. In that case, I decrease the resolution by δ. If this would result in a resolution less than r, I stop decomposition on that group.

Proposition 6. *Algorithm 1 produces a THD on an increasing indexed hierarchy.*

Proof. To show we have a topological hierarchy, we only need to show we have maps of covers and an increasing index as the other properties are clear. Let Z be a group in the hierarchy produced by the algorithm, and $Y = p(Z)$ its parent. If Z results from a split in Y, then $\epsilon_Y = \epsilon_Z$ (so $\epsilon_Z \leq \epsilon_Y$) and the open cover $\mathcal{U}_Z = \mathcal{U}_{Y|Z}$ where $\mathcal{U}_Y = \mathcal{U}_{\epsilon_Y|Y}$, with the map of covers being given by simple inclusion. Otherwise, there was no split, and $\epsilon_Z = \epsilon_Y - \delta$ and $\epsilon_Z \geq r$. Since $\delta > 0$, we have $\epsilon_Z < \epsilon_Y$. This shows we have an indexed hierarchy. For the covers, we have $\mathcal{U}_Y = \mathcal{U}_{\epsilon_Y|Y}$ and $\mathcal{U}_Z = \mathcal{U}_{\epsilon_Z|Z}$. By Proposition 4, there is a map of covers from $\mathcal{U}_{\epsilon_Z|Z}$ to $\mathcal{U}_{\epsilon_Y|Y}$ since $Z \subseteq Y$ and $\epsilon_Y > \epsilon_Z$. Therefore, we always have a map of covers from a child group's cover to its parent's cover, and we have an indexed topological hierarchy.

The property of a THD remains to be shown. Let Y be any group. If there is no split then $|c(Y)| = 1$ and the only child of Y is Y itself, which is the set Y_u in the partition where u is any vertex in $K = F(\mathcal{U}_{\epsilon_Y|Y})$ since there is only one connected component. If there is a split, line 22 of the algorithm implies that the children form a partition of Y by Proposition 5, as they are exactly the sets Y_u in that proposition. □

4.3 Persistence

I now proceed to look at a few different ways in which persistent homology can be defined for topological hierarchies. Since the focus of the rest of this chapter is on relating topological hierarchies to clustering, this section will be brief and only describe the kinds of persistence that could be extracted from a hierarchy. The first approach recovers a persistence module by starting from any group in a topological hierarchy and tracing its parents back to the root node. The next one defines a zigzag module for any two groups in a topological hierarchy by going up the hierarchy to their shared ancestor. The third approach uses the coproduct of simplicial complexes to produce different global views of a space at each level of depth in a topological hierarchy.

Let Y be any group in a topological hierarchy $\mathcal{X}$ with covers $\mathcal{U}_*$ and nerve-like map F, and set $Y_1 = Y$. Define inductively $Y_{i+1} = p(Y_i)$. This produces a sequence $Y = Y_1, \ldots, Y_i, \ldots, Y_m = X$, where X is the root node of the hierarchy. The associated simplicial complexes $K_i = F(\mathcal{U}_{Y_i})$ form a sequence of complexes joined by simplicial maps. Applying homology with $\mathbb{Z}_2$ coefficients results in a sequence:

$$H_p(K_1) \to H_p(K_2) \to \cdots \to H_p(K_m)$$

of vector spaces $H_p(K_i)$ connected by linear transformations. This describe a persistence module on the diagram of simplicial complexes. One can interpret the persistent features of this module as those that are present at different scales of looking at the space; a persistent feature is one that is present at a "zoomed-in" level on a subspace, and remains as points are added back into the space. For example, a feature in H_1 would be a hole surrounded by connected points. If this feature persists, it means that it is not filled in as points are added back into the space. Similarly, features in H_0 are connected components of the space. A persistent feature here would be such a component that is not joined with another as points are added back in. One can also look at it the other direction and ask which features present

in X persist when points are removed.

Definition 51 (Ancestor diagram)**.** Let

$$X_1 \xrightarrow{f_1} \dots \xrightarrow{f_{s-2}} X_{s-1} \xrightarrow{f_{s-1}} X_s \xleftarrow{g_s} X_{s+1} \xleftarrow{g_{s+1}} \dots \xleftarrow{g_{m-1}} X_m$$

be a diagram of sets X_i with maps $f_i : X_i \to X_{i+1}$ for $1 \leq i < s$ and maps $g_i : X_{i+1} \to X_i$ for $s \leq i < m$. This diagram is an *ancestor diagram*. When the sets are vector spaces and the maps linear transformations, the sequence is called an *ancestor module*.

Note that an ancestor module is a zigzag module of length m of type $ff \dots fgg \dots g$, where there are $s-1$ forward maps f_i and $m-s$ backward maps g_i. Now assume we have a topological hierarchy $\mathcal{X}$ and consider any two groups Y and Y'. Let Z be the common ancestor of Y and Y', so that we obtain a sequence $Y = Y_1, Y_2, \dots, Y_s = Z, \dots, Y_m = Y'$, where $s \leq m$, $Y_{i+1} = p(Y_i)$ for $i < s$, and $Y_i = p(Y_{i+1})$ for $i >= s$. Applying homology to the associated simplicial complexes $K_i = F(\mathcal{U}_{Y_i})$ gives a sequence

$$H(K_1) \to \dots \to H(K_s) \leftarrow \dots \leftarrow H(K_m) \tag{4.2}$$

of vector spaces with linear transformations between them which is an ancestor module. Since this is a zigzag module, the methods of [13] and [12] can be used to compute persistence on it.

The previous notions of persistence on a topological hierarchy all look at local features by comparing subsets of a parent space. The traditional definition of persistent homology looks at homology across different complexes defined on the same space, which identifies global features. Using the notion of a disjoint union, we can recover global persistence from a topological hierarchy. I first define a coproduct for simplicial complexes and simplicial maps, then use that to describe persistence across a topological hierarchy.

Definition 52 (Coproduct of simplicial complexes)**.** Let K and K' be simplicial complexes. The *disjoint union* or *coproduct* of K and K' is a simplicial complex $K \sqcup K'$ defined by:

- The vertex set of $K \sqcup K'$ is $\{(v,0)|v \in V(K)\} \cup \{(v,1)|v \in V(K')\}$
- If $\{v_0, v_1, \ldots, v_k\} \in K$ spans a k-simplex in K, then $\{(v_0,0),(v_1,0),\ldots,(v_k,0)\}$ is a k-simplex in $K \sqcup K'$. Similarly, if $\{v'_0, v'_1, \ldots, v'_k\} \in K'$ spans a k-simplex in K', then $\{(v'_0,1),(v'_1,1),\ldots,(v'_k,1)\}$ is a k-simplex in $K \sqcup K'$. These are the only simplices in the disjoin union of K and K'.

One can view $K \sqcup K'$ as relabeling the vertices of K and K' so the vertex sets are disjoint, then taking the union of the two complexes. This is distinct from the normal sense of union, where if the vertex sets are overlapping, could lead to parts of the two complexes merging. One should note that there are inclusion simplicial maps $i_K : K \to K \sqcup K'$ and $i_{K'} : K' \to K \sqcup K'$ defined on the vertices by $i_K(v) = (v,0)$ and $i_{K'}(v') = (v',1)$.

Given simplicial complexes K, K', and L, let $\phi : K \to L$ and $\phi' : K' \to L$ be simplicial maps. I can define a simplicial map $\phi \sqcup \phi' : K \sqcup K' \to L$ by its action on vertices:

$$(\phi \sqcup \phi')((v,i)) = \begin{cases} \phi(v), \text{ if } i = 0 \\ \phi'(v), \text{ if } i = 1 \end{cases}$$

That this defines a simplicial map is easily seen; it is equivalent to identifying a simplex σ from $K \sqcup K'$ as either being from K or K' then applying ϕ or ϕ' respectively. Finally, if we have simplicial maps $\phi : K \to L$ and $\phi : K' \to L'$, I can define a simplicial map $\phi \sqcup \phi' : K \sqcup K' \to L \sqcup L'$ as follows: I take the insertions $i : L \to L \sqcup L'$ and $i' : L' \to L \sqcup L'$ and note that we have two simplicial maps $i \circ \phi : K \to L \sqcup L'$ and $i' \circ \phi' : K' \to L \sqcup L'$. Then the desired simplicial map is just the coproduct of these, $(i \circ \phi) \sqcup (i' \circ \phi')$.

I will use the notion of coproduct of simplicial complexes and simplicial maps to define global persistent homology on a topological hierarchy; each level of the hierarchy will be associated with a single homology group.

Definition 53 (d-level sets)**.** Let $\mathcal{X}$ be a topological hierarchy with a maximum depth (number of edges from the root X to the farthest leaf node) of D and let $0 \leq d \leq D$. By $X(d)$ I mean the collection of groups at that level, where $X(0) = \{X\}$. Note that $\bigcup_{Y \in X(d)} Y = X$. I call $X(d)$ the *d-level sets*. Each $X(d)$ represents a partition of X, where the larger d is the finer-grained the partition is.

Associated with each depth $0 \leq d \leq D$ I will define a simplicial complex $K(d)$, given by the disjoint union of the associated complexes for each group at that depth:

$$K(d) = \coprod_{Y \in X(d)} F(\mathcal{U}_Y)$$

For $0 < d \leq D$ I will have simplicial maps $\psi_d : K(d) \to K(d-1)$ given by the coproduct of the individual maps $i_Y \circ \phi_Y : F(\mathcal{U}_Y) \to K(d-1)$, where i_Y is the inclusion $\mathcal{U}_{p(Y)} \to K(d-1)$:

$$\psi_d = \coprod_{Y \in X(d)} (i_Y \circ \phi_Y)$$

From the previous definition we have a sequence of simplicial maps:

$$K(D) \xrightarrow{\psi_D} K(D-1) \xrightarrow{\psi_{D-1}} \dots \xrightarrow{\psi_2} K(1) \xrightarrow{\psi_1} K(0)$$

Passing to homology gives a persistence module:

$$H_k(K(D)) \to H_k(K(D-1)) \to \cdots \to H_k(K(1)) \to H_k(K(0))$$

for each non-negative integer k. Since the union of groups at each depth in the hierarchy is X, each simplicial complex $K(d)$ covers the entire space X. A feature in this persistence module is one represented in the original space X that persists at higher levels of decomposition - i.e., the feature exists starting from $K(0)$ down to some deep $K(d)$. This notion of persistence is closest to the original interpretation of persistent homology: a feature is persistent if it is present when viewing the space at different scales.

4.4 Clustering and Evaluation

In order to evaluate a topological hierarchy as a clustering, it is necessary to obtain a flat clustering from one. Unlike hierarchical clustering, where the dendrogram is a binary tree and for a given parent cluster, there is only a single distance between the two children clusters A and B, a node in a topological hierarchy can have multiple children. If $Y \in \mathcal{X}$ is a group in a topological hierarchy, then there will be $|c(Y)|\left(|c(Y)|-1\right)/2$ distances among its children groups. These must be combined into a single distance in some way to allow treating the hierarchy as a dendrogram which can then be cut to obtain a flat clustering.

For computing the distance between two children groups $A, B \in c(Y)$, I can use the same linkage criteria as hierarchical clustering. Then, to combine these distances to obtain a single distance for Y, I identify three possible criteria:

- (minimum) $d(Y) = \min\{d(A,B)|A,B \in c(Y), A \neq B\}$
- (average) $d(Y) = \frac{1}{|c(Y)|(|c(Y)|-1)} \sum_{A \in c(Y)} \sum_{B \in c(Y)} d(A,B)$
- (maximum) $d(Y) = \max\{d(A,B)|A,B \in c(Y), A \neq B\}$

Using this approach, we can assign a distance to every group in the topological hierarchy; leaf nodes are assigned a distance of zero. By choosing a threshold distance t, we can cut the hierarchy, removing all groups Y with distance $d(Y) < t$. The leaf nodes of

the resulting tree are then the partitions of the flat cluster. This gives a family of clustering functions $f_t : X \to F(X, s(t))$ where the number of clusters s is now a function of the distance threshold t.

I conclude with a discussion of measures for evaluating a topological hierarchy as a clustering. I will focus on supervised information-theoretic measures that compare a given clustering to a true clustering or set of labels; see [94] for a deeper discussion. The first measure is the Rand index (RI), typically used in a normalized form called the Adjusted Rand index (ARI) [51]. Let X be a point-set of N points, and $U = \{U_1, \ldots, U_r\}$ and $V = \{V_1, \ldots, V_s\}$ be two partitions of X. Construct a matrix $(n_{ij}) \in \mathbb{N}^{r \times s}$ known as a *contingency table*, where n_{ij} is the number of points in X that are present in both U_i and V_j, i.e. $n_{ij} = |U_i \cap V_j|$. Furthermore, consider the $\binom{N}{2}$ possible pairs of points in X and how they are distributed in U and V, constructing the following four statistics:

- N_{00}: The count of pairs that are in different clusters in both U and V
- N_{10}: The count of pairs that are in the same cluster in U but different clusters in V
- N_{01}: The count of pairs that are in different clusters in U but the same cluster in V
- N_{11}: The count of pairs that are in the same cluster in both U and V

Then the Rand index is $\mathrm{RI}(U, V) = (N_{00} + N_{11})/\binom{N}{2}$, and the Adjusted Rand index is

$$\frac{2(N_{00}N_{11} - N_{01}N_{10})}{(N_{00} + N_{01})(N_{01} + N_{11}) + (N_{00} + N_{10})(N_{10} + N_{11})}$$

The ARI takes on values in the range $[0, 1]$, with higher values indicating greater similarity of the partitions. In our case, one partition is the clustering I compute, and the other is the true partition of the data or some other set of ground-truth labels, so ARI values close to 1 indicate better performance.

The other similarity measure I consider is mutual information. Given clusterings U

and V as described above, their mutual information is given by

$$\mathrm{MI}(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N} \log \frac{N|U_i \cap V_j|}{|U_i||V_j|}$$

In order to adjust for chance, there is a family of adjusted mutual information (AMI) scores given by

$$\mathrm{AMI}(U,V) = \frac{\mathrm{MI}(U,V) - \mathrm{E}\left[\mathrm{MI}(U,V)\right]}{\mathrm{avg}(H(U),H(V)) - \mathrm{E}\left[\mathrm{MI}(U,V)\right]}$$

where avg is one of the following functions: maximum, minimum, geometric mean, or arithmetic mean and

$$H(U) = \sum_{i=1}^{|U|} \frac{|U_i|}{N} \log \frac{|U_i|}{N}$$

is the entropy.

To apply these measures to a topological hierarchy $\mathcal{X}$ built on some point-set X, I assume that there is some true clustering of X, which I will denote U. Then, for a choice of criterion for combining child group distances and a given cut of the hierarchy at a distance t we obtain a flat clustering $V[t]$ of X from the hierarchy. In order to compare this to a hierarchical clustering approach, one can cut the dendrogram and the topological hierarchy at several distance $0 \leq t \leq t_{max}$ with the max distance decided by X and the metric on it. Assume we're using AMI as the clustering score. Then for each value of t, there is a score $\mathrm{AMI}(U, V[t])$ for the topological hierarchy and a score $\mathrm{AMI}(U, W[t])$ for the hierarchical clustering. A simple way to compare the results would be to compare the maximum scores for each. Alternatively, one could compare the value $\int_0^{t_{max}} \mathrm{AMI}(U, V[t])\mathrm{d}t$ to the equivalent one for the hierarchical clustering, similar to the concept of area under the curve (AUC) for evaluating supervised learning algorithms.

As an example of these metrics on real-world data, I used the FICO Explainable Ma-

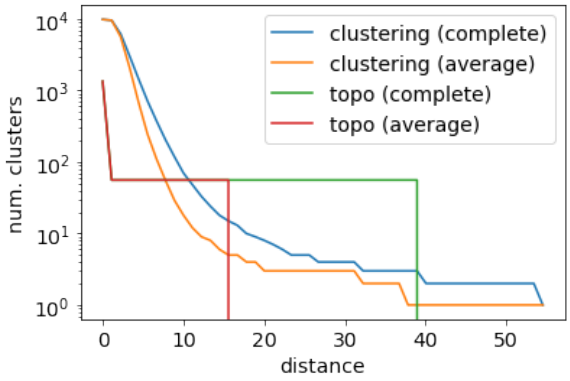
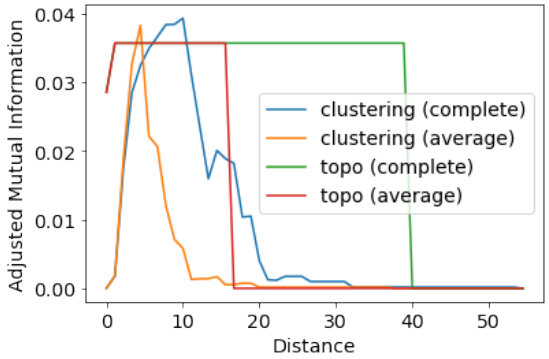


Figure 4.4: Distance vs. number of clusters on the FICO dataset.

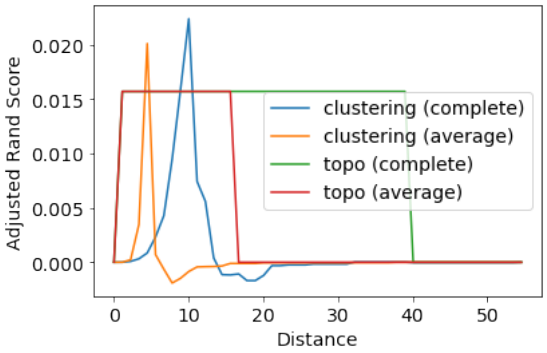
chine Learning Challenge dataset[1]. To evaluate topological hierarchies qualitatively, I produced plots of distance vs. score for AMI and adjusted Rand score. I did minimal preprocessing to the data, replacing categorical values with numeric ones and removing the target RiskPerformance field and external risk estimate. Then I scaled the data so each column had a standard normal distribution. All experiments used a Euclidean metric. Figure 4.4 shows the number of obtained clusters versus the distance the dendrograms are cut at. The legend "topo" refers to the topological hierarchy and "clustering" to the dendrogram produced by hierarchical clustering. Complete and average linkage refer to the usual hierarchical clustering approaches. "Complete" in the context of a topological hierarchy refers to the maximum criterion used to assign a single distance to a group with more than two children. A characteristic of the topological hierarchy is maintaining a nearly constant number of clusters over a range of distances.

Plots of AMI scores and adjusted Rand scores are given in Figure 4.5. Higher values of the scores represent better clusterings in terms of the clusters tending to contain points of one label. The optimal distances to cut the dendrograms or hierarchies to obtain a flat clustering are those which provide the maximal AMI or Rand score. Complete and average linkage clustering have similar results, with complete linkage having slightly better scores. Single linkage is excluded from the plots to avoid clutter, since it had the worst performance

[1] Available at https://community.fico.com/s/explainable-machine-learning-challenge.



(a) Comparison of Adjusted Mutual Information (AMI) scores.



(b) Comparison of Adjusted Rand scores.

Figure 4.5: Comparison of clustering scores on the FICO dataset. The lines labeled "topo" are those for the topological hierarchies.

of the hierarchical clustering techniques.

These results show that topological hierarchies can be competitive with hierarchical clustering when evaluated as a clustering algorithm. Traditional clustering proves better in terms of having the best score at a single distance, but topological hierarchies show their best scores for a range of distances. Note that unlike with hierarchical clustering, there is no significant difference between the average and complete criteria for topological hierarchies at lower distances, and at higher distances the differences are only in a "jump" where the bulk of clusters are included in the resulting flat clustering.

4.5 Analysis and Optimization

This section describes the computational aspects of topological hierarchies, and how they work on a finite point cloud set $S = \{x_1, \ldots, x_n\}$ rather than an arbitrary space X. There are some optimizations given which apply in certain situations, such as using a specific open cover or using MAPPER. Then, I analyze the time complexity of the MAPPER algorithm and use the results to come up with worst-case estimates for building a topological hierarchy with MAPPER.

First, I cover some potential optimizations in the pipeline of simplicial constructions and hierarchies.

- *Tracking membership in open covers.* This applies when constructing an increasing hierarchy (with either MAPPER or Rips-like constructions) using the simplicial complexes to partition the topological space and its subspaces. From the largest resolution ϵ_0, determine the membership of points x_i in the open sets of the cover $\mathcal{U}_{\epsilon_0}$. When a split occurs, since the same open cover is used over a smaller domain, one only needs to remove the points not in each new group from this list of memberships. However, in the case of a resolution decrease, the entire membership may need to be recomputed as new open sets can emerge from splitting a larger open set in the higher resolution cover.

- *Tracking membership with common covers.* A common class of open covers are open balls under a metric d, usually the Euclidean metric or some form of p-norm based metric. This includes generalized intervals, which are open balls under the $p = \infty$ metric (maximum metric). I assume that the centers of the open balls are fixed, and the resolution is identified with the radii of the balls. To track membership in the open covers as the resolution is varied, one only needs to precompute the distances of each point x_i from the centers of each open ball. Then, checking membership is as simple as checking if the distance is greater than the resolution.

- *Precomputing Distance Matrices.* Most clustering algorithms are designed to use a distance metric to determine nearness of points. When building a topological hierarchy based on a clustering or using MAPPER, the distance between the same pair of points may be needed several times. Therefore, it makes sense to precompute these distances and store them in a distance matrix. If the dataset is too large to do so, one can still precompute distances for groups in the hierarchy below a certain size, as these distances can be reused when constructing children groups. Rather than using

the full $|X| \times |X|$ symmetric distance matrix, a condensed vector representation of $\binom{|X|}{2}$ entries can be used, as is done in the SciPy library [95]. Either representation results in a $O(n^2)$ memory usage, which can become large for larger datasets.

- *MAPPER Recomputation.* If MAPPER is used as the nerve-like map in the topological hierarchy, one can optimize the recomputation of the pullback covers for children groups. One only needs to check which open sets in the filter space have points removed, and re-run clustering on the inverse images of those sets, rather than clustering on all the inverse images. If only a few points are removed or a large number of points are removed from one bin in the filter space, this can greatly improve the efficiency.

- *Precomputing Filter values.* Similar to the distance matrices, filter values can be precomputed for some or all points in the set X. Since the filter is a function $f : X \to Z$, storing filter values is only an $O(n)$ memory cost, as opposed to the $O(n^2)$ cost for storing distance matrices. However, since the filter values are not used in the clustering steps of MAPPER, caching them will not provide as much of an improvement to performance.

4.5.1 Complexity of the THD Algorithm

This section describes the time complexity of the THD algorithm in terms of its sub-algorithms, particularly MAPPER.

MAPPER

First, we must consider the time complexity of MAPPER. I break it into three steps: the filter, the clustering, and the construction of the simplicial complex. I will assume we are working with a d-dimensional dataset of n rows, with a filter function $f : \mathbb{R}^d \to \mathbb{R}^{d'}$ that reduces the dimensionality to d'. Furthermore, I denote the number of open sets in the

cover of $\mathbb{R}^{d'}$ by k. In general I assume $d' < d << n$, i.e. the dimensionality is always much lower than the number of points.

The time complexity of the filter will depend on the details of the function one uses for it. The simplest filters, such as the identity, or taking one or more components of each point, will be $O(1)$. Dimensionality reduction and manifold learning algorithms will tend to be $O(n \log n)$ to $O(n^2)$, as the following non-exhaustive list describes:

- *PCA*. Many algorithms for computing principal components exist. Matrix-based methods are the oldest, with SVD or Householder-QR techniques having a complexity of $O(nd^2 + d^3)$ and the power method $O(nd^2 + d'd^2)$ [72]. Iterative approaches such as expectation maximization (EM) are of the order $O(d'dn)$, but depend on other parameters such as a learning rate [78].

- *Multi-dimensional Scaling (MDS)* [55, 57, 90]. Classical MDS is $O(n^3)$ due to the eigenvalue calculation. As with PCA, faster iterative approaches are known, with time complexities varying from $O(n\sqrt{n})$ to $O(n^2)$.

- *t-SNE* [93]. Its complexity is originally $O(n^2)$, not considering the dimensionality of the data. A variant of the Barnes-Hut algorithm and the dual-tree algorithm improves this to $O(n \log n)$ [91, 92].

- *UMAP* [65]. The nearest neighbor search phase determines the worst-case time complexity, which is $O(dn^2)$. Empirical analysis by the original author hints that the complexity scales like $O(dn^{1.14})$, and could be treated as $O(n \log n)$ for practical purposes [64].

Hereafter, $O(F(n, d, d'))$ indicates the time complexity of an arbitrary filter. I will consider $O(dn^2)$ as the "worst case" and $O(dn \log n)$ as the "best case" for the choice of filter.

After the filter step, we must approximate the pullback cover by repeated application of a clustering algorithm. The first step is to identify membership of points in open sets in

the filter space. This will be $O(knd')$ in general, as for each open set, we must iterate over the points and determine if they are in the set or not based on the individual components, assuming a generalized interval or d'-ball is used for the open sets. Then, we cluster the points in the inverse image of each open set, which will be $O(kC(n,d))$ where $C(n,d)$ is the worst-case time complexity of the clustering algorithm used.

- *Hierachical Clustering*. Sequential, agglomerative, hierarchical nonoverlapping (SAHN) clustering methods require at worst $O(n^3)$ time, with a heap-based optimization giving $O(n^2 \log n)$ time [21]. For single-linkage and complete-linkage, the SLINK and CLINK algorithms give $O(n^2)$ performance [83, 22].

- *DBSCAN* [30]. The worst case time complexity of DBSCAN is $O(n^2)$ [3], and an improved version of the algorithm boasts $O(n \log n)$ performance [41]. If inaccuracy is allowed, it is possible to reduce this to $O(n)$, and a hash table based algorithm exists with this complexity [35, 98].

- *OPTICS* [5, 81]. In its base form, OPTICS would be $O(n^2)$. A tree-based index structure allows for $O(n \log n)$ performance. If objects are arranged in a grid, this knowledge can be used for $O(n)$ performance, but this is not a common property of point-cloud datasets.

The final step is constructing the simplicial complex. This is determined by the degree of simplices desired in the final MAPPER complex; in general, one must compare at least $\binom{k}{s+1} = O(k^{s+1})$ clusters for overlap to build a simplicial complex with s-simplices. For building a THD, only 1-simplices are required, so the dependence on k is quadratic. There is also a dependence on the size of clusters, as one needs to check each pair of points between two clusters to determine overlap. For the 1-simplex case, this is $O(n^2k^2)$ in the worst case, but it will be much better in general as clusters will only contain a very small subset of the dataset.

The total time complexity of MAPPER is the sum of that of the three steps:

$$O\left(F(n, d, d') + kC(n, d) + n^2k^2\right)$$

In the worst case but assuming the optimized versions of the algorithms discussed above are used, this is $O\left(dn^2 + kn^2 + n^2k^2\right)$. This is quadratic in the dataset size, quadratic in the number of sets in the cover, and linear in the data dimension. The absolutely best case requires specific algorithms: iterative PCA as filter and pDBSCAN as clustering. In this case, the time complexity is $O(d'dn + kdn \log n + n^2k^2)$. Despite the simplicial complex term being quadratic, it is the least significant term for smaller datasets and lower numbers of open sets, as the filter and clustering algorithms will have a lot of overhead. This term would only become important as the dataset reaches very large sizes, on the order of 10^5 or so.

THD

There will be two factors that contribute the most to the time complexity of the entire THD algorithm: the number of times MAPPER is run, and the size of the groups each MAPPER is run on. The group threshold, which will end recursive application of MAPPER to child groups below a certain size, controls both; larger thresholds will reduce the number of MAPPER runs, and the minimum size of a group MAPPER will be run on is bounded by the threshold. The following assumptions are made for the time complexity analysis of THD:

1. The group threshold is 1, so that every leaf node in the resulting tree will be a group with one point, on which MAPPER is not run.

2. A split is observed for every group with more than one point.

There are two edge cases of interest; in the first, every group is split into two children,

one with one point, and the other with the remaining points. This results in a tree with $n-1$ total MAPPER groups, and n non-MAPPER groups, each branching from a MAPPER group; the final MAPPER group has two points and two children of one point, unlike the others. If we ignore the dependence of MAPPER's time complexity on dimensionality and number of sets in the cover, we can denote it $M(n)$ and then the total time complexity to build this THD is

$$\sum_{i=1}^{n-1} kM(n) \approx k \sum_{i=2}^{n} i^2 = k\left(\frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6} - 1\right) = O(kn^3)$$

The next edge case is when there is an even split at each group; I will assume a binary split for simplicity. Then the first group of n points is split into two groups of $n/2$ points each, and so on, until all that remains are n leaf nodes of 1 point on which MAPPER is not run. At each depth t of the tree, there will be approximately 2^t groups of $\frac{n}{2^t}$ points each. The max depth is around $D = \lfloor \log_2 n \rfloor$. Then the total time to build this tree is

$$k \sum_{t=0}^{D} 2^t M\left(\frac{n}{2^t}\right) \approx k \sum_{t=0}^{D} \frac{n^2}{2^t} = kn^2(2n-1) = O(kn^3)$$

Finally, consider the impact the optimizations described at the start of the section have on the time complexity. Precomputing filter values means that they only contribute a single $F(n, d, d')$ time to the THD, and this filter term does not come into play past the first MAPPER computation. Similarly, precomputing distance matrices means that the full clustering time $C(n, d)$ only applies once, but further runs of the clustering algorithm will be at least linear, since the clustering must still be done. Therefore, there is a clustering step that is at least linear in the number of points of a group for each MAPPER.

Remarks on Space Complexity

- *Original Data.* The dataset takes $O(nd)$ storage in an uncompressed form.

- *Filter Values.* These will be $O(nd')$, always smaller than the original dataset. These are the most efficient to precompute, as they are only linear in the number of rows.

- *Distances.* The full distance matrix is $O(n^2)$ storage. Since it's symmetric, a reduced form can be used which is still $O(n^2)$ but stores less than half the number of values as the full matrix. The quadratic dependence means that for larger datasets precomputing all of the distances is not feasible.

- *Simplicial Complexes.* When every open set in the cover has at least one point inside, the simplicial complex will need at least $O(k)$ storage, as each open set in the cover will correspond to one or more 0-simplices. 1-simplices will add at most $O(k^2)$ storage on top of this for a fully connected complex.

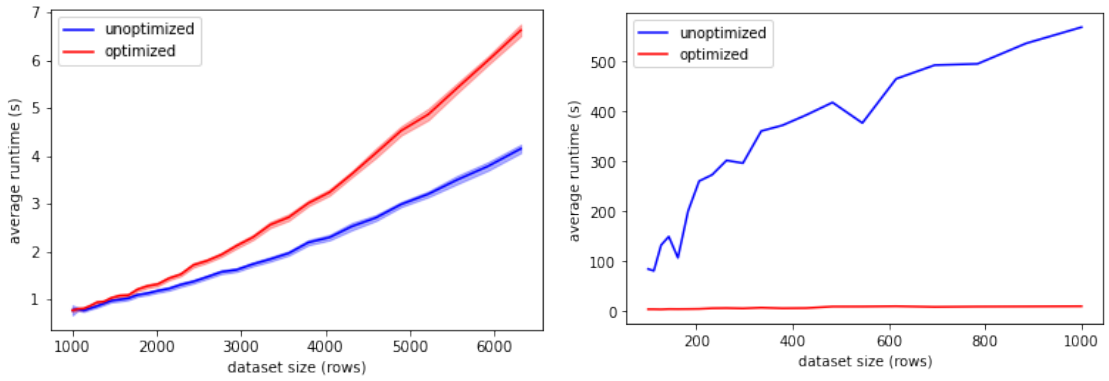
- *Topological Hierarchy.* A full hierarchy without a group threshold will require at least $O(n)$ storage for the leaf nodes. This is not an issue in practice as thresholds are used, and the size of the tree can be kept manageable.

Implementation of a Python library and Dashboard Application for Data Analysis with Topological Hierarchies

This section describes a Python library and application that implements MAPPER and the THD algorithm described in Section 4. Python 3 was used, with the majority of testing done on Python 3.7.2. NumPy provides the base multi-dimensional array structure used to hold data in memory [45]. The clustering step of the MAPPER implementation makes use of the hierarchical clustering techniques implemented by SciPy [95]. These were preferred over the scikit-learn implementations for their efficiency. The API is flexible enough to support custom clustering functions, as long as a method for determining the number of clusters automatically exists for the algorithm.

Several filter functions are supported. Simple ones such as the identity filter, single or multiple components of the original data, and eccentricity are built-in. Any Python class implementing the scikit-learn [74] API for dimensionality reduction can be used as a filter. The SciPy implementation of PCA is also supported as a filter, once again preferred over scikit-learn for its efficiency. Some other common filters that use the scikit-learn API and are thus automatically compatible are t-SNE [61] and UMAP [65].

The implementation of MAPPER supports simplices of arbitrary order. Simplicial complexes are implemented as simplex trees, a trie which stores and retrieves a simplicial complex efficiently [10]. Of interest is the time complexity of common operations for constructing MAPPER and THDs; to add a 1-simplex and its subfaces (the two vertices) takes $O(D_m)$, where D_m is the maximum number of operations to perform a search in a dictionary whose size is at most the highest degree of a node in the simplicial complex. In the worst case D_m will be $O(\log n)$, with the maximal degree $O(n)$. Since inserting a simplex is the main operation needed to build a complex for MAPPER, this shows the efficiency of using simplex trees for constructing and storing complexes in the THD algorithm.



(a) Runtimes with PCA as filter function.

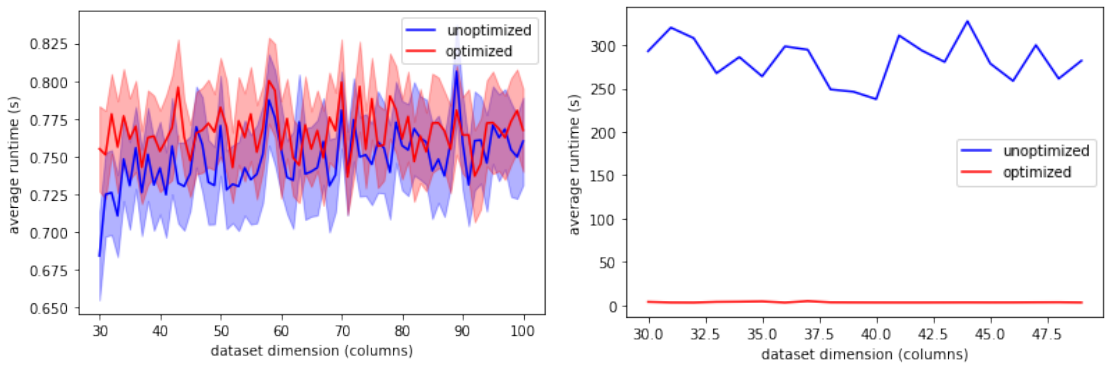
(b) Runtimes with UMAP as filter function.

Figure 5.1: Plot comparing runtime of unoptimized and optimized THD implementations with varying dataset size.

5.1 Performance Tests

All tests were run on a PC with Windows 10, a 3.5 GHz AMD Ryzen 3 2200G CPU, and 16 GB of RAM. To get an accurate estimate of the time taken to run tasks, each one was run several times, taking the mean runtime, and a 99% confidence interval of the times was computed. Tests were done with an unoptimized THD, which doesn't precompute anything, and an optimized THD, which precomputes filter values and distance matrices, passing these to the algorithms which need them. Two filter functions were considered: PCA, which is very efficient to compute, and UMAP, which has significant overhead. For the experiments with UMAP, it was not possible to run the THD several times due to the time it took, so the figures only show values for one THD run and no confidence intervals.

The first performance test was to determine how THD scales with the number of points (rows, features) in the dataset. I used 100-dimensional data drawn from a uniform distribution in $[0, 1)$, with generated datasets having anywhere from 1 to 6000 rows. The filter function is two-component PCA, clustering is single-linkage, and the open covers all have 15 sets. A plot showing the average runtime of THD against the number of rows is given in Figure 5.1a. This figure shows the time complexity to be empirically polynomial in the dataset size, matching the discussion of Section 4.5.1. Surprisingly, the optimized ap-



(a) Runtimes with PCA as filter function.

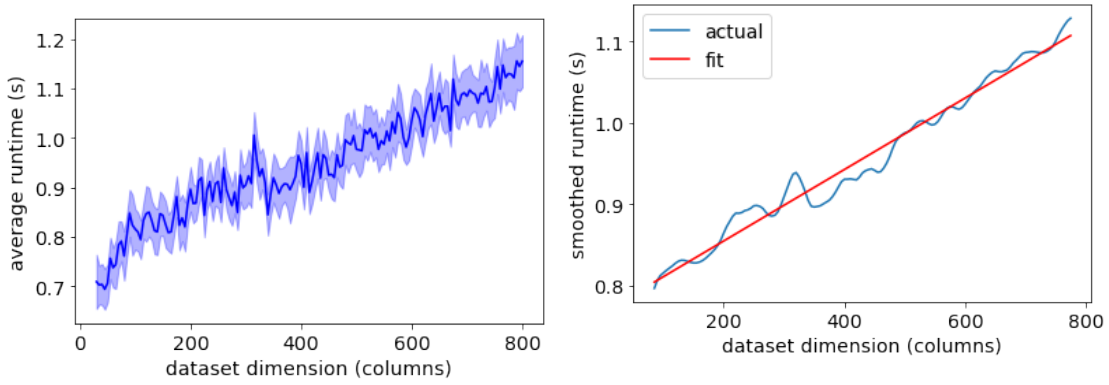
(b) Runtimes with UMAP as filter function.

Figure 5.2: Plot comparing runtime of unoptimized and optimized THD implementations with varying dataset dimension.

proach is slower than the naive unoptimized one. This is due to efficiency of computing the PCA filter, as well as how the clusterings are handled in the MAPPER steps. Clustering is always done on the pre-image of some subset of the filter space, so as long as there are multiple open sets, each clustering will always be on a proper subset of the data, and all the distances will not be needed at once. Therefore, computing all these distances ahead of time actually adds more overhead than time is saved by precomputing the distances.

Now consider the performance of the THD algorithm with UMAP as the filter, with results given in Figure 5.1b, where smaller dataset sizes are used due to the time taken to run the experiments. Note that the unoptimized times are now almost 100 times as large, showing the greater overhead of UMAP. Precomputing the filter values provides a huge boost to performance then, allowing the optimized approach to perform nearly as well as with PCA – there is still the overhead of the single run of UMAP versus the single run of PCA, but once that's computed, the time to run the THD algorithm will be comparable. Therefore, for "slow" filters such as UMAP and tSNE, it is more efficient to precompute filter values, while for "fast" filters such as PCA and the identity, it is more efficient to not precompute them.

The second test looked at datasets of increasing dimension, from 30 features to 100.



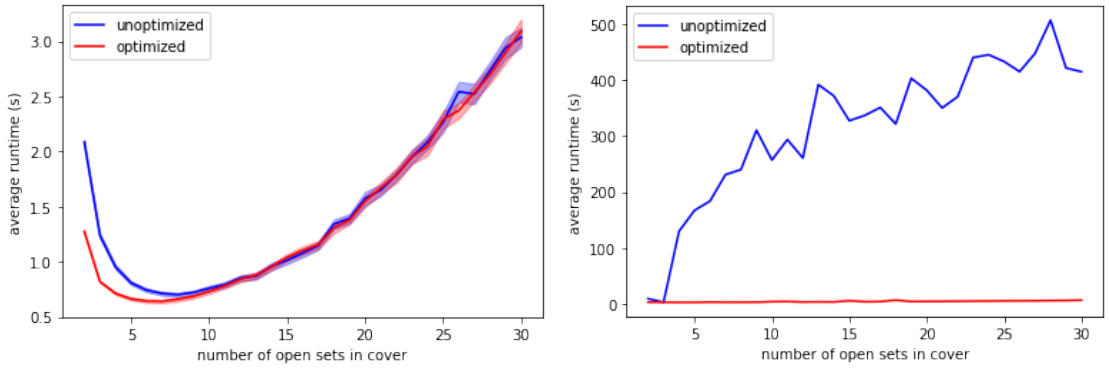
(a) No smoothing applied, showing average increase in runtime.

(b) Runtimes smoothed with convolution by a small window, compared with a linear regression fit.

Figure 5.3: Runtime of the unoptimized approach for varying dataset dimension at higher dimensions.

The dataset size was fixed at 1000 rows for all dimensions, and a PCA filter with 2 components was used. The results are shown in Figure 5.2a. Both approaches show extreme variations over the dataset sizes, making interpreting the results difficult. The unoptimized approach appears to have a slight increase in average runtime. This is the linear dependence found in the analysis in Section 4.5.1. Note that the average runtimes in this experiment are all quite small, less than a second, while for the previous one they ranged from a second to 7 seconds. This shows the impact dimensionality has on runtime is small compared to the number of points. With UMAP as a filter, similar results are obtained to the first experiment, where the optimized approach is now much more efficient. Notice that the unoptimized approach behaves as with PCA, showing no obvious dependence on dimensionality.

In order to identify the linear relation between dataset dimension and runtime, it was necessary to look at datasets with dimension higher than 100. The plots for the unoptimized approach with up to 800 columns are shown in Figure 5.3. The unsmoothed plot in Figure 5.3a shows a trend in increasing runtime, but it is hard to identify its linearity. When smoothed by convolution with a small window (with a width of around 15), the trend be-



(a) Plot comparing runtime of unoptimized and optimized THD implementations with varying number of open sets.

(b) Plot comparing runtime of unoptimized and optimized THD implementations with varying number of open sets when the filter is UMAP.

Figure 5.4: Plot comparing runtime of unoptimized and optimized THD implementations with varying open cover size.

comes more evident. This is shown, along with a linear regression fit, in Figure 5.3b. The theoretical prediction of a linear dependence of runtime on dataset dimension is shown empirically.

The final test looked at the impact varying the number of open sets in the cover had on the runtime. Recall that this determined the number of clusterings needed for each MAPPER run, and also appeared as a quadratic term for building the 1-skeleton of the simplicial complex. In this test, datasets of 1000 rows and 100 dimensions were generated, and covers with 2 to 31 open sets were used. The results are in Figure 5.4a. For a very low number of open sets, around 2 to 10, the runtime actually *decreases* as the number of open sets increases. This is due to open sets being larger when there are less of them, so they each contain a large number of points. Therefore, each clustering on average is done on larger subsets of the data, and this ends up dominating the total runtime. When there are more open sets so that each one contains fewer points, the individual clusterings end up contributing less to the time, and instead the total number of clusterings ends up dominating. The optimized approach is actually faster here, as for large clusterings, precomputing the distances is more efficient than computing them as needed while doing the clustering.

For larger numbers of open sets, I recover the polynomial time shown in Section 4.5.1. In this case, the time is dominated by the combinatorial aspects of the algorithm – the number of clusterings run is important, even while an individual clustering contributes little to the time. Also, having a large number of open sets means needing to check a larger number of possible overlaps when building the simplicial complex. Finally, note that the optimized and unoptimized approaches have little difference in runtime. As with the other two experiments, I obtain a massive gain in performance in using the optimized approach with the UMAP filter.

In conclusion, the number of rows in the data have the largest impact on the runtime, and the empirical results in Figure 5.1a support the polynomial time predicted by the theoretical analysis of MAPPER and THD in Section 4.5.1. The impact of the number of open sets on the runtime is multifaceted, but two regimes can be identified in Figure 5.4a; the first is for few open sets, where the time to run the clustering dominates, and the second is for many open sets, where the quadratic time due to the combinatorics of testing cluster overlap dominates. Therefore, having more open sets can make individual clusterings faster, at the cost of needing to run the clustering algorithm more times. MAPPER and THD scale linearly with the number of dimensions or columns in the data, as can be seen in Figure 5.2, which matches the $O(d)$ complexity of MAPPER. Finally, I observe that for simple filter functions that are quick to compute, such as PCA, the unoptimized approach to computing a THD is more efficient.

5.2 Dashboard

This section describes the implementation and features of an interactive web application dashboard for running MAPPER and the THD algorithm for the purpose of exploratory data analysis. The dashboard uses the Plotly Dash library [76, 50], which runs as an HTTP server using the Flask library. It is accessed through a web browser as a ReactJS app that

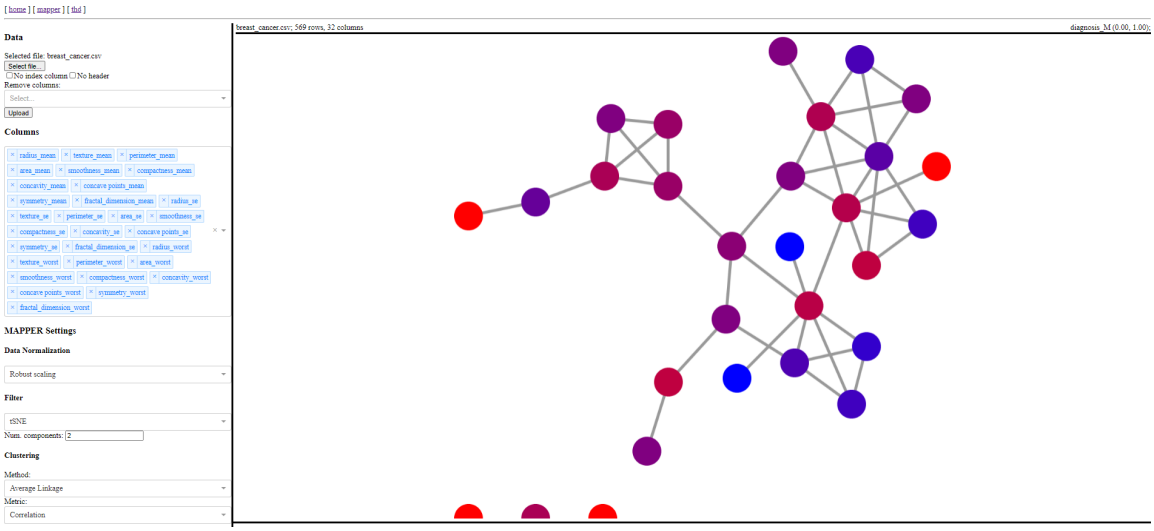


Figure 5.5: The main part of the THD Dashboard. The currently selected simplicial complex (one-skeleton) is shown on the right, while the upload and mapper settings are on the left third of the screen.

communicates with the server. This makes it possible to run the backend on a machine with more RAM and computational power than the machine that accesses the dashboard via HTTP.

The main section of the THD Dashboard is shown in Figure 5.5. There are several areas of the dashboard visible in the screenshot. The menu at the very top contains links to all the separate pages of the dashboard. On the upper left is where the user can upload a dataset, with options for loading depending on the presence of a header or index column. Below that, the user can choose which columns to remove when loading the dataset from a dropdown. Beneath that is a list of columns loaded, and the user can choose which ones to use in the THD. Below that is a section for MAPPER-specific settings. The user can select various data normalizations such as min-max, mean scaling, and others from a dropdown. For the filter function, they can choose between PCA, tSNE, UMAP, and an eccentricity filter. Below that are the settings for the clustering algorithm. The user can choose the type of hierarchical clustering: single-linkage, complete-linkage, or average-linkage, as well as the metric to use for the clustering and the filter (for filters that require a metric).

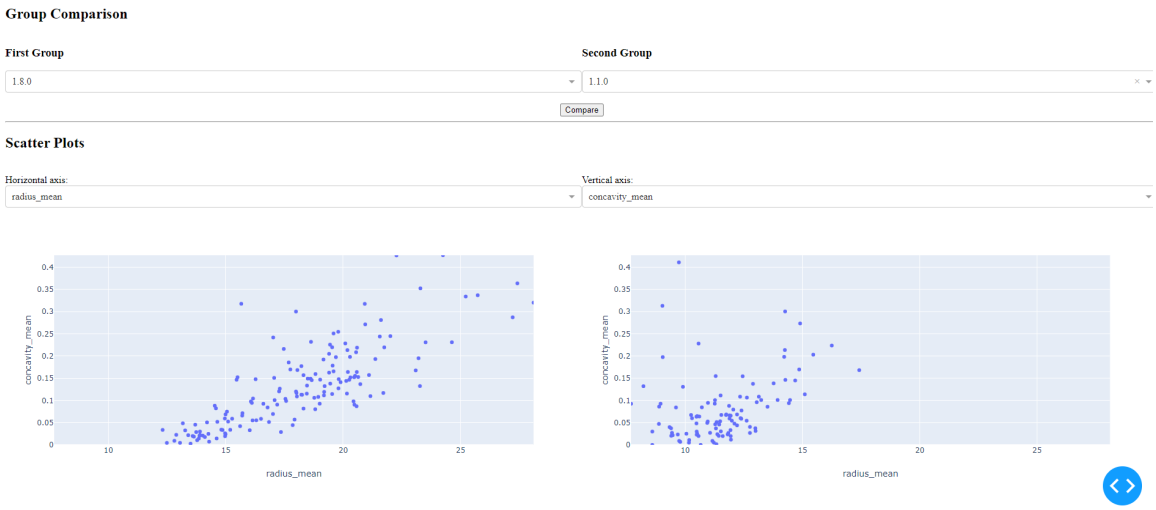


Figure 5.6: The group comparison settings and scatter plots. Two groups (clusters) in the THD can be selected and compared, and the scatter plots show the different distributions of selected features between the two groups.

Figure 5.6 shows the part of the dashboard that allows comparing two groups, as well as scatter plots for immediate comparison of two features in the two selected groups. The user can see the names of the groups by clicking on them in the THD view, and use this to determine which groups they wish to compare. Once two groups are selected, the scatter plots will be populated. The axes of the scatterplots can be selected and they will be updated in real-time when different features or groups are selected. The left scatterplot corresponds to the group chosen in the dropdown on the left side, and the right scatterplot to the other group. The user may also click on the "Compare" button to run a detailed group comparison.

When a group comparison is run, the results will open in a new browser tab or window. Figure 5.7 shows how these results will appear. The top half of the results in the screenshot are summary statistics for each group. The number of rows (points or items) in the group are shown above a table giving summary statistics. For each feature in the dataset, the mean, median, quantile statistics, inter-quartile range (IQR), minimum, and maximum values are listed for both groups. The bottom half of the screenshot shows the results of running a Kolmogorov-Smirnov test (KS test) between the two groups. The results can be ordered

Group Summaries

Group 1 (1.8.0)

151 rows

column	mean	quantile25	median	quantile75	iqr	min	max
radius_mean	18.0764	15.3650	18.2200	20.1700	4.8050	12.3100	28.1100
texture_mean	20.1177	17.3000	19.8300	22.1450	4.8450	10.3800	32.4700
perimeter_mean	118.4666	99.0850	119.6000	132.7000	33.6150	79.1900	188.5000
area_mean	1048.9311	737.2000	1027.0000	1260.5000	523.3000	470.9000	2501.0000
smoothness_mean	0.0953	0.0865	0.0945	0.1036	0.0171	0.0643	0.1447
compactness_mean	0.1163	0.0726	0.1108	0.1445	0.0719	0.0267	0.3114
concavity_mean	0.1301	0.0551	0.1168	0.1735	0.1184	0.0028	0.4268
concave points_mean	0.0772	0.0383	0.0795	0.1043	0.0660	0.0044	0.2012
symmetry_mean	0.1780	0.1581	0.1769	0.1943	0.0363	0.1215	0.2655
fractal_dimension_mean	0.0583	0.0553	0.0570	0.0610	0.0057	0.0500	0.0810

« ‹ 1 / 4 › »

Group 2 (1.1.0)

104 rows

column	mean	quantile25	median	quantile75	iqr	min	max
radius_mean	11.5923	10.4800	11.4700	12.4925	2.0125	7.6910	17.4200
texture_mean	19.1268	17.2825	18.9450	21.0725	3.7900	13.1700	27.6100
perimeter_mean	75.2222	66.7950	73.8700	81.5700	14.7750	48.3400	114.5000
area_mean	422.0846	334.0500	402.2000	478.1500	144.1000	170.4000	948.0000
smoothness_mean	0.0975	0.0879	0.0994	0.1054	0.0175	0.0661	0.1248
compactness_mean	0.1044	0.0730	0.0986	0.1249	0.0519	0.0387	0.2413
concavity_mean	0.0809	0.0292	0.0617	0.1010	0.0718	0.0000	0.4108
concave points_mean	0.0332	0.0179	0.0269	0.0430	0.0251	0.0000	0.0971
symmetry_mean	0.1840	0.1638	0.1819	0.1995	0.0357	0.1203	0.2743
fractal_dimension_mean	0.0671	0.0619	0.0658	0.0697	0.0078	0.0567	0.0930

« ‹ 1 / 4 › »

KS Test Results

column	ks_statistic	p_value	group1_mean	group2_mean
radius_mean	0.8064	0.0000	18.0764	11.5923
texture_mean	0.1403	0.1572	20.1177	19.1268
perimeter_mean	0.7806	0.0000	118.4666	75.2222
area_mean	0.8064	0.0000	1048.9311	422.0846
smoothness_mean	0.1518	0.1028	0.0953	0.0975
compactness_mean	0.1551	0.0905	0.1163	0.1044
concavity_mean	0.3448	0.0000	0.1301	0.0809
concave points_mean	0.4962	0.0000	0.0772	0.0332
symmetry_mean	0.1419	0.1484	0.1780	0.1840
fractal_dimension_mean	0.6122	0.0000	0.0583	0.0671

« ‹ 1 / 4 › »

Figure 5.7: The group comparison results. Summary statistics of the two groups are shown, along with Kolmogorov-Smirnoff test results.

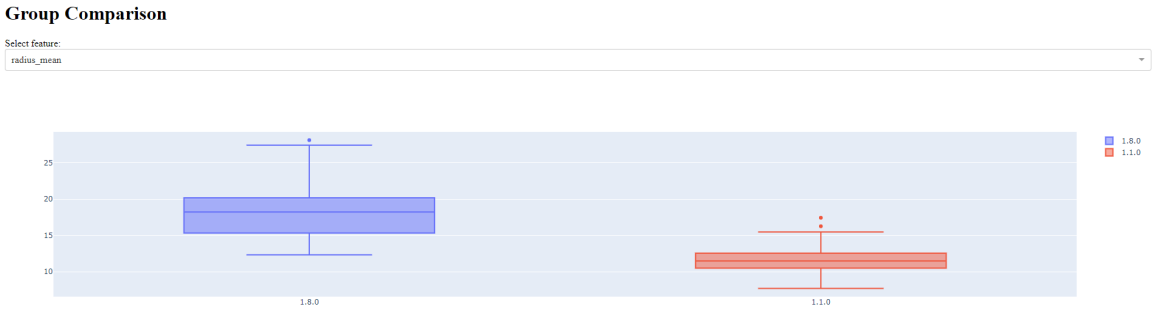


Figure 5.8: The group comparison box and whisker plots. This allows a quick comparison in the distribution of a single feature between the two groups.

by the KS statistic value, listing the most statistically significant features distinguishing the two groups first. The p-value and means of the feature in both groups are included in the KS test table as well.

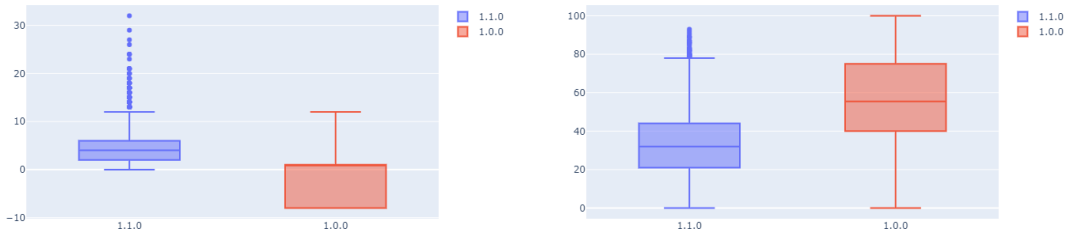
Figure 5.8 shows the box-and-whisker plots on the group comparison page. The user may select a single feature from the dataset, and a box-and-whisker plot for the feature in both groups is generated. This allows for qualitatively determining features that distinguish the two groups, as well as being useful for identifying outliers in either group.

Figure 5.9: THD Tree for the first experiment, colored by RiskPerformance, with Red = Good and Blue = Bad.

5.3 Comparison of Results with Proprietary TDA Tools

In this section, I present a comparison of results obtained from applying the dashboard to the HELOC dataset described in Chapter 6. I used the same methodology which is described in more detail in that chapter: build a THD on the dataset, and for each group look at the most statistically significant features in that group which distinguish the group from the rest of the data. Using this information, I can then build an explanation for why the applicants in the group were or were not able to pay back their loan. The features are compared with the results in Section 6.1.3 obtained from the Ayasdi Platform [7].

For the first experiment, the dashboard settings are as follows. The data is preprocessed by subtracting off the median and dividing by the IQR. The filter function is UMAP with 2 components, using a cosine metric. For clustering, I used average linkage also with a cosine metric. The open cover in the filter space has 50 sets with 75% overlap, and the THD settings are to contract by 20% when no split is observed and to prune groups with less than 200 members. The resulting THD tree is given in Figure 5.9, colored by RiskPerformance.



(a) Number of revolving trades with balance.

(b) Percentage installment trades.

Figure 5.10: Box and whisker plots of NumRevolvingTradesWBalance and PercentInstallTrades for the first split.

I observe that the group on the right to split off first has on average worse RiskPerformance than the rest of the data, while the lowest level groups on the left have good RiskPerformance. My goal is to explain this based on statistically significant features distinguishing the groups, and compare the results with the analysis in Chapter 6 to attempt to replicate some of the results of our study which used proprietary TDA tools.

Next, I analyzed the first split (from the top of the tree) to identify statistically significant features between the two resulting groups. The group on the left has 9,281 points and the one on the right has 582 rows, with respective distributions of good RiskPerformance of 49.29% and 27.49%. The statistically significant features based on a Kolmogorov-Smirnoff test are the number of revolving trades with balance, percentage installment trades, number of satisfactory trades, and percentage of trades with balance among others. Box and whisker plots of two of these features are given in Figure 5.10. Based on these features, I conclude that the smaller group that split off (the one on the right) consists of individuals without credit cards who have a high proportion of installment trades, and more trades with outstanding balances. For example, these may be individuals with many installment loans they are unable to pay back.

Finally, I compared these results with the ones obtained using the proprietary Ayasdi

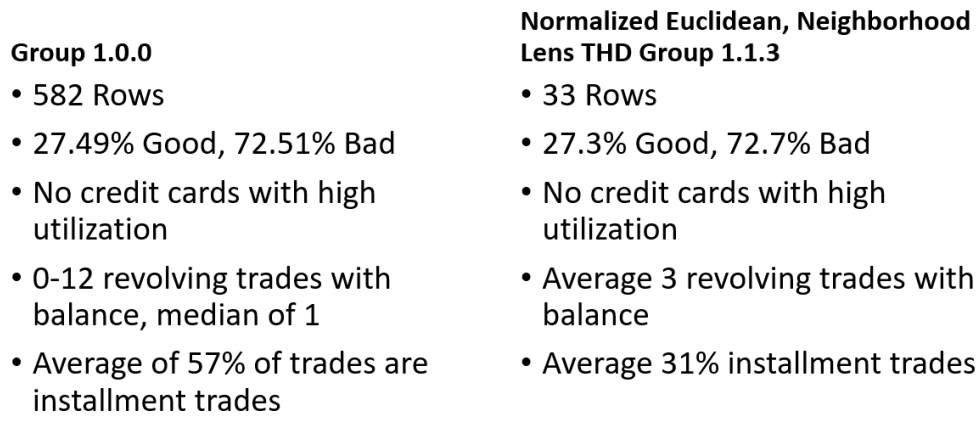


Figure 5.11: Similarities between the first split observed in the experiment with the dashboard and a group obtained by THD with the Ayasdi Platform.

Platform TDA software, described in Chapter 6. The goal was not to exactly replicate the results, but to see if I obtained similar explanations or even groups that have the same kind of characteristics, to see if my implementation of MAPPER and THD "decomposes" data in the same way. Similar features between the group I analyzed in the previous paragraphs and a group from the Ayasdi Platform-based analysis are shown in Figure 5.11. Most interesting is that the group obtained with the dashboard has many more points, while retaining a similar distribution of feature values. Based on this, I conclude that my MAPPER and THD implementation is able to decompose data in a way that segments it based on interesting, statistically meaningful features in a way that is similar to that of proprietary tools.

Applications of Topological Hierarchies to Data Science

6.1 HELOC Applicant Risk Evaluation

Because of heavy regulations in the financial services industry, there are stringent requirements for financial decisions made by algorithm to be explainable. In particular, it is important for credit institutions to be able to explain their decisions to creditors. This presents a challenge to the adoption of artificial intelligence techniques in the industry, as many AI algorithms act as "black boxes" which are difficult for a human to interpret or to explain why the algorithm made a certain decision [58, 25]. Unfortunately, the most powerful machine learning techniques such as deep neural networks are not inherently explainable. The goal of explainable AI (XAI) is to remedy this issue by one of two broad approaches: the first takes existing (unexplainable) algorithms to them explainable; the second develops new, powerful explainable techniques from scratch. We choose the latter approach.

We apply THD to an anonymized dataset of home equity line of credit (HELOC) loan applications made available by FICO [33] and illustrate how it provides insight, based on groups with distinct distributions of data features, into why applicants may be unable to pay back a loan within 90 days, making them risky to loan to. The goal is not to provide an exhaustive explanation for every individual from the dataset. Instead, we will provide illustrative explanations extracted from two THDs and describe how they could be used in an explainable AI approach. Examples include use by a lending institution to explain to a potential creditor why they are denied or granted a home equity line of credit, or to explain a decision made by a black-box supervised machine learning algorithm, simply based on records of past loan performances rather than by training a transparent supervised learning algorithm that requires a historical account of whether customers' past loans were approved or denied.

6.1.1 Glossary of Financial Terminology

This section serves to give definitions for the financial terminology used in the HELOC Dataset.

amortizing loan A loan where the borrower pays a fixed monthly payment of principal and interest to pay off a debt over time.

delinquency A payment received some period of time past its due date. Usually measured in multiples of 30 days.

home equity The difference between the current market value of a home and its purchase price.

home equity line of credit (HELOC) A line of credit typically offered by a bank as a percentage of home equity. That is, it is a line of credit which uses home equity as collateral. Unlike a mortgage, a HELOC is a revolving trade instead of being paid back in installments.

inquiry A line of information that captures when a lending institution has pulled a consumer's credit bureau report in order to make a credit decision.

line of credit An agreement between a financial institution and a creditor that establishes the maximum amount of a loan that the creditor can borrow. This is a type of revolving trade.

revolving trade Any type of credit that a creditor can use multiple times. Example include credit cards and equity lines. Contrast with an amortizing loan, which is a one-time transfer of funds that is repaid in installments.

trade In the context of HELOC, short for *trade line*.

trade line A credit agreement between the consumer and a lending institution (e.g. bank), represented by a separate "line" of information.

6.1.2 Dataset Description

This study uses of the FICO Explainable Machine Learning dataset (hereafter the *HELOC dataset*) made available by FICO [33]. The dataset consists of anonymized home equity line-of-credit (HELOC) loan applications made by homeowners requesting a loan in the range of $5,000 to $150,000. The target (label) feature is called RiskPerformance, and is a categorical value of either ”Bad” if the consumer was more than 90 days past the due date on a payment in the 24 months after their credit account was opened, and ”Good” otherwise. The dataset contains 5,000 ”Good” individuals and 5,459 ”Bad” individuals for a total of 10,459 samples, giving a distribution of 48% ”Good” individuals and 52% ”Bad” individuals.

The following list explains the interpretations of individual features in the dataset:

AverageMInFile The average number of months the individual appears in the file.

ExternalRiskEstimate A consolidated risk estimate from other credit bureaus. Higher values indicate less risk.

MaxDelq2PublicRecLast12M The maximum period of delinquency or presence of ”Derogatory” item in public records for the last 12 months.

MSinceOldestTrade The number of months since the oldest trade opened by the individual.

MSinceMostRecentDelq The number of months since most recent delinquency by the individual.

MSinceMostRecentTradeOpen The number of months since the most recent trade opened by the individual.

NumBank2NatlTradesWHighUtilization The number of credit cards owned by the individual on a consumer credit bureau report carrying a balance that is at 75% of its limit or higher.

NumSatisfactoryTrades The number of credit agreements involving the individual on a consumer credit bureau report with on-time payments (satisfactory payments).

NumTrades60Ever2DerogPubRec The number of trade lines on a credit bureau report that record a payment received 60 days past its due date, added to the number of items considered ”Derogatory” in all Public Records available for the consumer.

NumTrades90Ever2DerogPubRec The number of trade lines on a credit bureau report that record a payment received 90 days past its due date, added to the number of items considered ”Derogatory” in all Public Records available for the consumer.

PercentTradesNeverDelq The percentage of trades which were never delinquent, i.e. loans which were paid back on time.

RiskPerformance Whether the individual paid back the HELOC as negotiated within 12-36 months of the loan.

6.1.3 Results

We did not seek to predict RiskPerformance, but merely explain its value given the other features in an unsupervised way using a THD. To establish the difference between groups, we did a statistical comparison between them using KS-score for continuous variables and a hypergeometric distribution for categorical ones. By doing this for several splits in the THD, we can then track the path of an individual throughout the THD, using the ”choice” of which branch to follow at each split to ”tell a story” about why this person was or was not able to pay back a loan on time. From these branches we were able to extract illustrative explanations for whether individuals in a group were 90 or more days delinquent 24 months after taking out a loan.

Two THDs were computed from the entire dataset, using all features but RiskPerformance and ExternalRiskEstimate. We excluded the ExternalRiskEstimate because it was

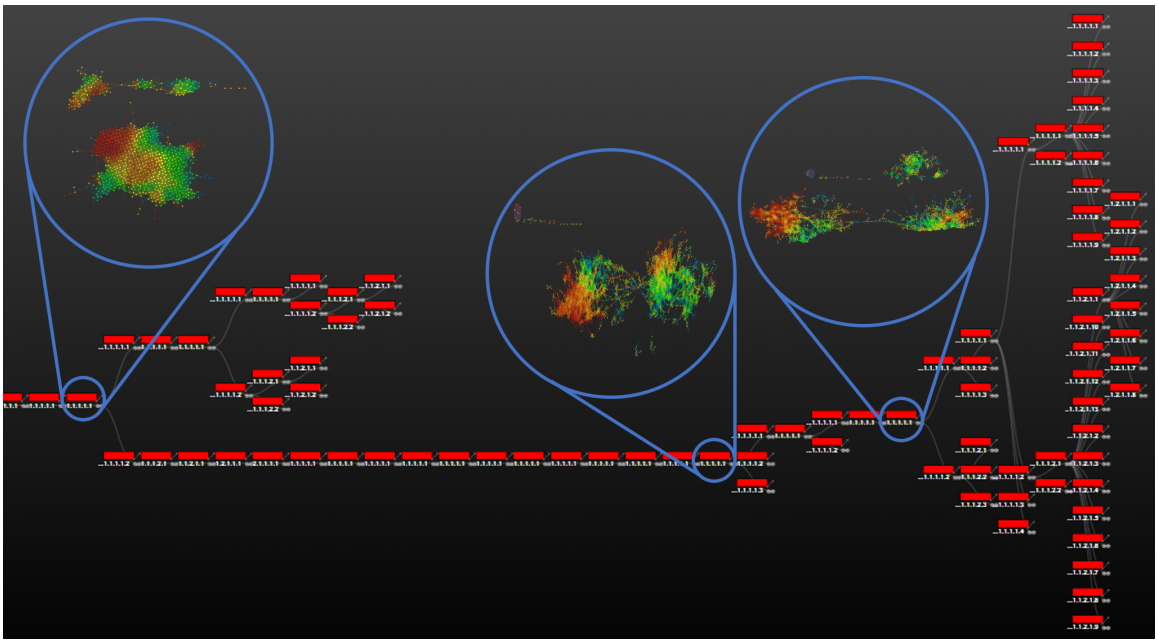


Figure 6.1: THD tree for VNE metric, NHL as filter with networks for selected groups shown

obtained from an outside source and may not be as useful in an explanation. For our initial resolution we always used 1, which gives a topological network with one node containing the entire dataset. This resolution is increased until the first split occurs, and then THD is ran recursively on each branch until no connected components with a number of points above the threshold remain. The *gain* (overlap parameter) was 2.7 for all THDs, and remains fixed throughout the whole process. The split threshold was set to 20 points, i.e. a connected component with 20 data points (not just nodes) would be considered a split.

We used the variance-normalized Euclidean distance as the metric for both THDs, and for filters we built one THD with a *neighborhood* lens (NHL), which is analogous to the first two components of t -SNE [61], and the other with the first two multi-dimensional scaling (MDS) [55, 56] coordinates. The Ayasdi Platform [7] was used to build the topological networks in the THD with MAPPER. The portion of the THD tree for the NHL filter showing splits is given in Figure 6.1. Topological networks for selected nodes are shown as well, colored by RiskPerformance where red means more ”Good” individuals and blue more ”Bad” individuals. The tree for the MDS filter, not shown, exhibits different behavior

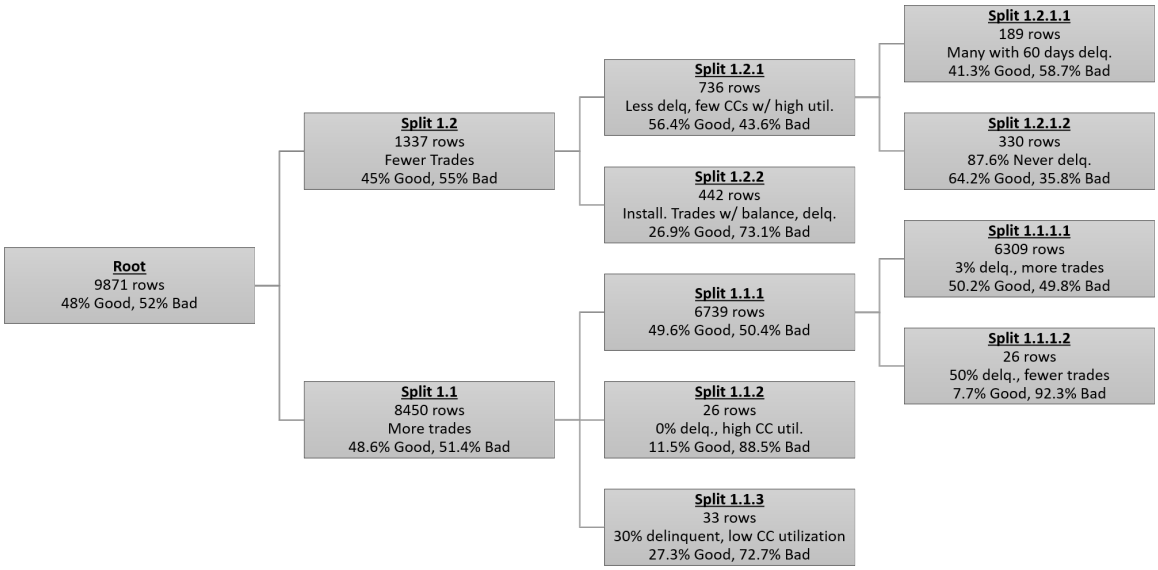


Figure 6.2: Summary of significant features at high-level splits in the THD tree with VNE metric, NHL as filter

in splitting, where there are a lot of small splits until the last few large splits are reached. In the NHL filter THD, there is a significant split early on which is not observed with the MDS filter. In both THDs, there seems to be a large split at the end, with two large groups that are able to be further decomposed.

A simplified view of the NHL THD, showing the early splits, is given in Figure 6.2. Note that the implementation of THD using Ayasdi platform will discord smaller connected components from the topological network that fall beneath a group threshold, so that the total number of points in the children of a group may be less than the number of points in the parent. The root node gives the number of points and RiskPerformance distribution for the entire dataset. At each split, a summary of the most important features distinguishing this group from other groups in the split is given. Finally, the number of points and the distribution of RiskPerformance values in the group is given. This diagram could be extended to include all splits in a THD, and then used to explain an individual's performance based on their path through the THD. For example, we observe that individuals falling in Split

1.1.2 could be turned down for a loan as the group has 88.5% of its members unable to pay back on time. Further investigation reveals that most individuals in the group exhibit high credit card utilization, suggesting an explanation for these individuals. We summarize other explanations are extracted from Figure 6.2:

- (a) Individuals in Split 1.1.2 were unable to pay a loan due to high credit card utilization leading them unable to pay back on time.
- (b) Individuals in Split 1.1.3 were unable to pay a loan due to a past history of delinquency, *despite* low credit card utilization.
- (c) Individuals in Split 1.1.1.2 were unable to pay a loan due to having few trades, meaning they have less of a credit history, but also history of delinquency on past trades. This can make such users riskier to lend to.
- (d) Individuals in Split 1.2.1.2 paid their loan, even though they have a short history and few trades, but have a very low rate of delinquency.

We extracted explanations in a similar way from the MDS THD as well. Here the group names such as ”Split 1.2” refer to groups in the MDS THD (not shown), and not in the NHL THD:

- (e) Individuals in Split 2 would be denied a loan due to having a large number of loans with balance, and a high number of inquiries.
- (f) Individuals in Split 1.2 would be denied a loan due to a history of delinquency over 120 days and a large number of trades with balance.
- (g) Individuals in Split 1.1.1.1.2 would be denied a loan due to a history of delinquency and a high number of revolving trades with balance. This is in spite of the fact that these individuals have a better external risk estimate than individuals in their sister group of the split.

Note in explanation (d) how the THD is able to identify fine grained groups of customers who could be seen as good loan customers, even though they have a shallow credit history. The THD is further able to find a set of customers likely to be poor loan customers by explanation (b), even though they have relatively low credit utilization. These customers may be counterintuitive in nature, in the sense that their features intuitively suggest that the customer should (not) be granted credit.

Predicting the group of a split where a new applicant would fall within the THD would thus provide both a decision and reason for granting or denying the applicant even when applicant features take on surprising or contradictory values, and the explanation can be presented at a finer or coarser grain depending on the split depth an analyst chooses to select an explanation from. Specifically, the denial of a loan can be explained by an applicant that falls into a group with a high percentage (greater than the global average of 52%) of "Bad" RiskPerformance values, where membership in a group is defined by a distribution of feature values that distinguish the group from others at a split in the THD. These explanations are only based on the features of the applicant, and have nothing to do with the past loaning behavior of the organization. Note that using different settings for the THD will result in different explanations, although there are some similarities such as a history of delinquency and a large number of loans with balance correlating with "Bad" RiskPerformance, and hence these individuals would be denied a loan.

It is also instructive to look at individual topological networks where a split occurs. An example for the first split in the NHL THD is given in Figure 6.3. The connected component at the top of the network corresponds to the smaller group labeled "Split 1.2" in Figure 6.2. The further split in this group can already be seen, as there is a vertex in the upper component that can be removed to split it into two more components. "Split 1.2.1," which has few credit cards with high utilization, can be seen as the right component of this upper network, based on the coloring in Figure 6.3b. The topological networks can be used for an even more granular explanation, as we can consider the nodes an individual

belongs to in the network as containing similar individuals. We can also look at the immediate neighbors of these nodes to get a very local group of individuals similar to the one under consideration. This ability to go from a high-level, group-based representation, to individual topological networks, and then to just points from a group of related nodes in a network is a novel and useful feature of THD.

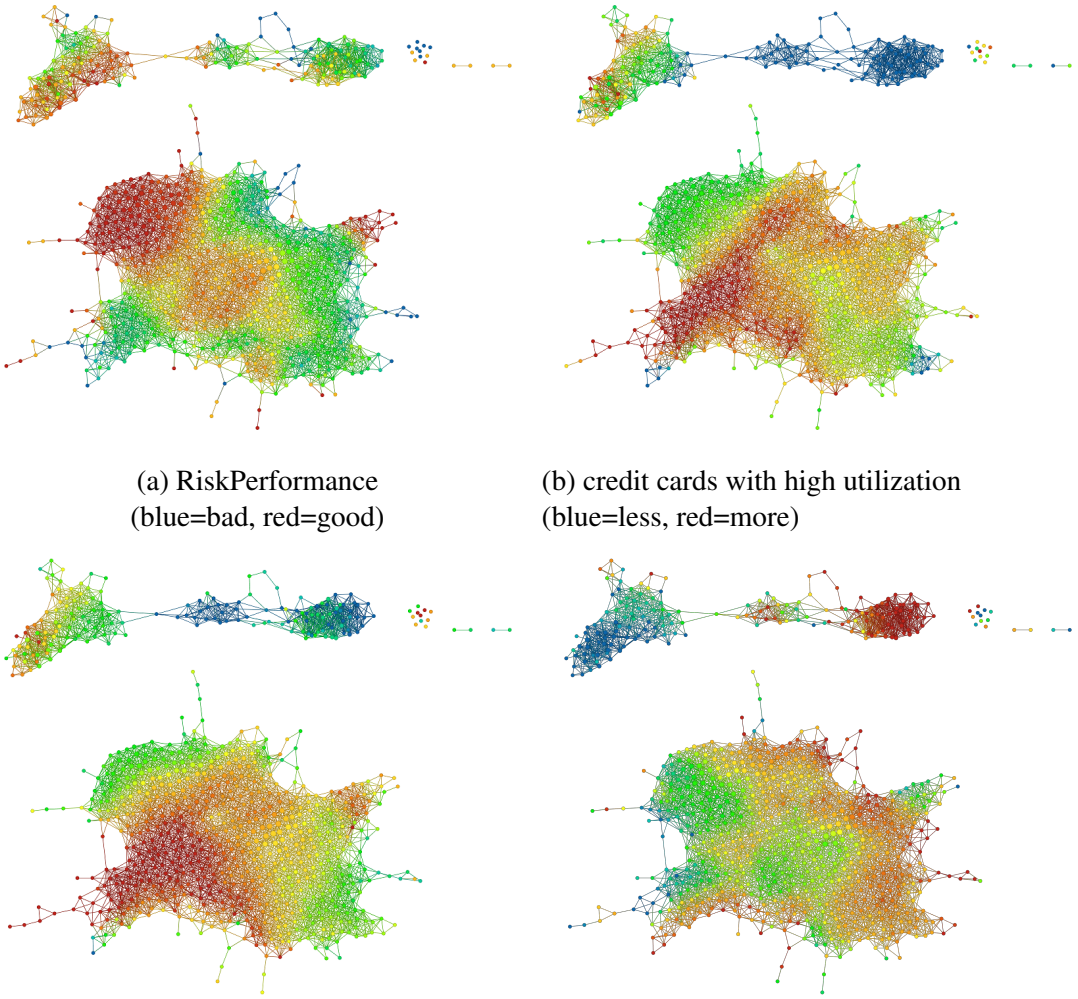
Comparison to transparent supervised models

It should be noted that Figure 6.2 does not appear all that different from a decision tree. Each split in the THD is based on a set of feature properties that differentiate one group from another, which is not unlike a decision tree that makes classification decisions by learning a hierarchy of heuristics to bin data. Moreover, decision trees are inherently transparent in the sence that each path down a tree from root to leaf describes a series of conditions explaining why data is classified.

The key difference between using splits of a THD to provide explanations rather than a decision tree is that THD is an entirely *unsupervised* technique; in constructing a THD the target feature RiskPerformance is never used. A decision tree, in contrast, is a *supervised* approach where the target feature is used directly during learning. When this training data is collected based on credit award decisions made by an organization from the past, the decision tree essentially learns a model describing how and why a firm awards credit to applicants. The learned model thus incorporates any potentially historical biases or priorities of the organization the training data is from. In taking an unsupervised approach, the THD becomes *decoupled* from the organization or institution who issues credit: splits in a THD are based on distinguishing features between sets of past applications conditioned on whether they successfully paid their loan. Thus, the THD can lead to automatic loan decisions based solely on the merits of the applicant, instead of a combination of applicant merit and historical firm behavior. Moreover, a THD is theoretically grounded by exploiting the shape and structure of the underlying manifold of data about applicants, which is

more likely to have a shape and structure characteristics across applicants for all forms of credit besides HELOC. Insights from a THD are thus more likely to be transferable across domains (e.g., to support decisions for other lines of credit besides HELOC), compared to decision tree heuristics that are (over)fitted to a single, specific dataset.

We further note that the THD requires no *a priori* information about the meaning or importance of each feature. Since these explanations are independent of any machine learning model used in classification, they could thus be used to supplement and explain decisions made by the algorithm. For example, a linear regression may give larger magnitude to weights that were found to correlate with RiskPerformance in THD groups, such as percentage of trades never delinquent and number of trades with balance. Finally, these explanations could also be used to understand a *misclassification* made by a classifier. The classifier may be weighting the wrong features, i.e. features that correlate with RiskPerformance in a different THD group than the one the point being classified belongs to. Another possibility is that the data point being classified is an outlier - it is in a THD group but has unusual features for that group. THD provides a framework for identifying such points automatically.



(a) RiskPerformance
(blue=bad, red=good)

(b) credit cards with high utilization
(blue=less, red=more)

(c) revolving trades with balance
(blue=less, red=more)

(d) percentage installment trades
(blue=0%, red=100%)

Figure 6.3: Topological Network for the first split in the NHL THD, colored by different features

Conclusion

In this dissertation, I introduced topological hierarchies, the THD algorithm, and looked at applications of them. I began by reviewing the literature, summarizing the relevant papers on TDA that cover MAPPER, persistent homology and persistence diagrams, zigzag persistence, and multiscale MAPPER, as well as reviewing articles on hierarchical clustering and describing applications of TDA to various fields. Next, I gave a detailed introduction to the mathematical background underpinning TDA, MAPPER, and persistent homology. This set the stage for my main contributions in Chapter 4, which introduces topological hierarchies, indexed hierarchies, and the topological hierarchical decomposition algorithm.

I defined a topological hierarchy as a mathematical tree structure, with an associated topological space and simplicial complex at each node, and maps of covers inducing a simplicial map between each child and parent node via a nerve-like map. In an indexed hierarchy, there is a parameter encoding scale or resolution which increases or decreases with depth in the tree. Based on the concept of connected components from graph theory, I described how to partition a topological space based on the simplicial complex asscoiated with it. By recursively applying this process, one arrives at the topological hierarchical decomposition described in Algorithm 1. Then I defined three ways of computing persistent homology on topological hierarchies: a child-to-ancestor approach, a global approach that creates a single persistence module by taking the disjoint union of simplicial complexes at each level in the hierarchy, and a child-to-child approach that compares two arbitrary nodes through their common ancestor using zigzag persistence.

The last section of Chapter 4 compares topological hierarchies to hierarchical clustering using existing metrics to measure the quality of a clustering. The Rand index and mutual information are chosen as good metrics to evaluate a clustering in a supervised manner (i.e. comparing the clustering with some pre-existing labels). Looking at the results of hierarchical clustering and THD on the FICO dataset, I observe similar performance in terms of the two metrics. Finally, I finish the chapter by looking at the performance of

MAPPER and THD, including a time complexity analysis, potential optimizations to the algorithm, and a short discussion of the space complexity of the algorithm. I show that in general MAPPER will be $O(n^2)$ and THD will have a worst case of $O(n^3)$, where n is the number of rows or points in the data. A rough analysis of space complexity indicates that if entire distance matrices are computed, this will be an $O(n^2)$ storage cost, more than any other part of the algorithm outside of filter functions that may use more storage temporarily.

Chapter 5 describes the practical side of THD, introducing an implementation of MAPPER and THD in Python 3 and a dashboard-styled web application for interactive data analysis using them. I give the results of several performance tests using the Python implementation, using a PCA and UMAP filter, and exhibit how well it handles data of various sizes, dimensionality, and how it scales with increasing number of sets in the open cover. These results are compared with the theoretical analysis of the preceding chapter, and I recover the predicted $O(n^2)$ complexity in the number of rows and open sets, and the $O(n)$ complexity in the number of dimensions. The results also show that it may not be worth precomputing distances and filter values for simple filters such as PCA, while for heavier filters like UMAP and tSNE it provides a huge improvement in performance at the cost of a larger storage overhead.

The next third of Chapter 5 introduced the dashboard web application, which is built on the Python Plotly Dash library. After describing the implementation environment, I run through its features and how they access the MAPPER and THD functionality of the Python library. The final third of the chapter covers the reproduction of the results of Chapter 6 using the dashboard, namely the analysis of the FICO dataset.

Finally, Chapter 6 introduced the application of THD to the FICO HELOC dataset. After describing needed financial terminology and describing the dataset, I presented the results of an exploratory analysis of the dataset using THD, with the goal of explaining the performance of individuals and groups of individuals in whether they were able to pay back a HELOC loan within the designated time. Examining the splits in a THD, we

identify statistically significant features in the parent and children groups that explain why the split occurred and why the members of each group, on average, were or were not able to pay back their HELOC loans. We compared this unsupervised, semi-qualitative approach to quantitative supervised models, and explain how it could potentially reduce bias in an automated system for determining whether to approve a loan applicant.

There are some cases in which TDA and THD in particular may not be applicable, or may have diminished results. In particular, clustering and other metric-based algorithms may not handle categorical or discrete data with few possible values well. One solution would be to use metrics designed for such data, but it is often the case that a dataset has both continuous and discrete or categorical fields. Traditional clustering algorithms assume that all columns use the same distance metric, but it may be possible to extend them to allow for multiple metrics, for example by summing together the independent distances for each subset of columns under the same metric. However, filter functions that use dimensionality reduction rely on a single metric, and it's not clear how techniques such as t-SNE or UMAP could be generalized. A temporary solution may be to apply the dimensionality reduction independently to each subset of columns under a same metric, and then concatenate these results.

Extremely large datasets are also not suited for THD, due to the $O(n^3)$ time complexity and prohibitive memory requirements to store distance matrices. While the memory usage could be reduced through optimization, the time complexity is due to the many calls to clustering or dimensionality reduction algorithms needed in THD, and these are not so easily reduced. A final situation where THD may not be immediately applicable is to supervised learning tasks. Building a topological hierarchy is fundamentally an unsupervised process, and it is not straightforward to incorporate supervision into it.

7.1 Future Work

This section discusses future avenues of research for topological hierarchies and THD which seem promising. One could look into the usage of other clustering algorithms within the implementation of MAPPER, especially density based clustering algorithms such as DBSCAN. This would allow MAPPER to scale to larger datasets, as DBSCAN lacks the $O(n^2)$ memory usage of hierarchical clustering algorithms when they compute distance matrices. It would be useful to understand how these improvements on space-complexity would propagate to the space complexity of THD, as well as whether density-based clustering provides better topological hierarchies under a suitable metric such as a clustering score like AMI.

A promising application of THD and MAPPER is to high-dimensional datasets with a small number of points. Due to the linear time dependence on the number of dimensions the algorithms will scale well to these kinds of datasets. An example of the kind of data that would fit these criteria is satellite imagery and photographs. Image data is well suited to THD since it typically takes only continuous values, or discrete values with a large number of them. Natural language processing (NLP) applications can generate high-dimensional data as well, such as vector space models where each row is a document and the columns are words in the vocabulary.

As mentioned previously, THD is an entirely unsupervised algorithm. However, it may be possible to use a model produced by THD to make predictions in a supervised manner. The obvious approach is to predict where new data would fall in the hierarchy, including its position in the simplicial complexes of the groups it appears in. The neighboring points in the lowest simplicial complex it appears in could then be used to make a prediction, for example by a majority vote of their values for the label to predict. For regression tasks, a mean or median could be taken instead. Such a prediction could be made for each group the point is predicted in, and either the lowest group taken or the predictions combined in some manner. Another way of including supervision into THD would be sim-

ply to use the label or target feature as one of the columns in building the THD. However, in this approach it is not as obvious how to make predictions on new data.

Finally, it remains to be seen how easily THDs can be used to produce explainable models. Explainable artifical intelligence (XAI) is not a rigorously defined term, but is used to refer to models which can explain their decisions in a way easily understood by humans without sacrificing accuracy [42]. In practice, XAI refers to the growing initiative to produce such models [1]. While the analysis of the HELOC dataset in Section 6.1.3 shows the potential of THD to produce low-level explanations, it remains to be seen whether it can be used to build high-level, qualitative explanations of the type sought by XAI. In a topological hierarchy it is possible to examine indiviudal groups through the associated simplicial complexes, identifying relations between individual points through them. This could be extended into something like a decision tree or other easily understood classifier, to produce an explanatory model for why a given point was placed within a certain node in a simplicial complex associated to some group.

Bibliography

[1] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160, 2018.

[2] Cuneyt G Akcora, Yitao Li, Yulia R Gel, and Murat Kantarcioglu. Bitcoinheist: Topological data analysis for ransomware prediction on the bitcoin blockchain. In *Proceedings of the Twenty-ninth International Joint Conference on Artificial Intelligence*, 2020.

[3] Tariq Ali, Sohail Asghar, and Naseer Ahmed Sajid. Critical analysis of dbscan variations. In *2010 International Conference on Information and Emerging Technologies*, pages 1–6. IEEE, 2010.

[4] Khaled Almgren, Minkyu Kim, and Jeongkyu Lee. Extracting knowledge from the geometric shape of social network data using topological data analysis. *Entropy*, 19(7):360, 2017.

[5] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60, 1999.

[6] Aras Asaad and Sabah Jassim. Topological data analysis for image tampering detection. In *International workshop on digital watermarking*, pages 136–146. Springer, 2017.

[7] Ayasdi. Ayasdi platform, 2018.

[8] Shai Ben-David and Ulrike Von Luxburg. Relating clustering stability to properties of cluster boundaries. In *21st Annual Conference on Learning Theory (COLT 2008)*, pages 379–390. Omnipress, 2008.

[9] Alexander Bernstein, Eugeny Burnaev, Maxim Sharaev, Ekaterina Kondrateva, and Oleg Kachan. Topological data analysis in computer vision. In *Twelfth International Conference on Machine Vision (ICMV 2019)*, volume 11433, page 114332H. International Society for Optics and Photonics, 2020.

[10] Jean-Daniel Boissonnat and Clément Maria. The simplex tree: An efficient data structure for general simplicial complexes. *Algorithmica*, 70(3):406–427, 2014.

[11] Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009.

[12] Gunnar Carlsson and Vin De Silva. Zigzag persistence. *Foundations of computational mathematics*, 10(4):367–405, 2010.

[13] Gunnar Carlsson, Vin De Silva, and Dmitriy Morozov. Zigzag persistent homology and real-valued functions. In *Proceedings of the twenty-fifth annual symposium on Computational geometry*, pages 247–256. ACM, 2009.

[14] Gunnar Carlsson, Afra Zomorodian, Anne Collins, and Leonidas J Guibas. Persistence barcodes for shapes. *International Journal of Shape Modeling*, 11(02):149–187, 2005.

[15] Frédéric Chazal, David Cohen-Steiner, Marc Glisse, Leonidas J Guibas, and Steve Y Oudot. Proximity of persistence modules and their diagrams. In *Proceedings of the twenty-fifth annual symposium on Computational geometry*, pages 237–246, 2009.

[16] Frédéric Chazal and Bertrand Michel. An introduction to topological data analysis: fundamental and practical aspects for data scientists. *arXiv preprint arXiv:1710.04019*, 2017.

[17] Yiran Chen and Ismar Volić. Topological data analysis model for the spread of the coronavirus. *Plos one*, 16(8):e0255584, 2021.

[18] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, 2007.

[19] Alex Cole and Gary Shiu. Topological data analysis for the string landscape. *Journal of High Energy Physics*, 2019(3):1–31, 2019.

[20] William Crawley-Boevey. Decomposition of pointwise finite-dimensional persistence modules. *Journal of Algebra and its Applications*, 14(05):1550066, 2015.

[21] William HE Day and Herbert Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification*, 1(1):7–24, 1984.

[22] Daniel Defays. An efficient algorithm for a complete link method. *The Computer Journal*, 20(4):364–366, 1977.

[23] Tamal K Dey, Fengtao Fan, and Yusu Wang. Computing topological persistence for simplicial maps. In *Proceedings of the thirtieth annual symposium on Computational geometry*, page 345. ACM, 2014.

[24] Tamal K Dey, Facundo Mémoli, and Yusu Wang. Multiscale mapper: Topological summarization via codomain covers. In *Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 997–1013. SIAM, 2016.

[25] F. K. Došilović, M. Brčić, and N. Hlupić. Explainable artificial intelligence: A survey. In *2018 41st International Convention on Information and Communica-*

tion Technology, Electronics and Microelectronics (MIPRO), pages 0210–0215, May 2018.

[26] Herbert Edelsbrunner and John Harer. Persistent homology-a survey. *Contemporary mathematics*, 453:257–282, 2008.

[27] Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010.

[28] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. In *Proceedings 41st annual symposium on foundations of computer science*, pages 454–463. IEEE, 2000.

[29] Herbert Edelsbrunner and Dmitriy Morozov. Persistent homology: theory and practice. Technical report, Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States), 2012.

[30] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[31] Davide L. Ferrario and Renzo A. Piccinini. *Simplicial Structures in Topology*, pages 43–97. Springer New York, New York, NY, 2011.

[32] Massimo Ferri. Persistent topology for natural data analysis—a survey. In *Towards integrative machine learning and knowledge extraction*, pages 117–133. Springer, 2017.

[33] FICO. Explainable machine learning challenge, 2018.

[34] Joseph A Gallian. *Contemporary abstract algebra*. Chapman and Hall/CRC, 2021.

[35] Junhao Gan and Yufei Tao. Dbscan revisited: mis-claim, un-fixability, and approximation. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pages 519–530, 2015.

[36] Robert Ghrist. Barcodes: the persistent topology of data. *Bulletin of the American Mathematical Society*, 45(1):61–75, 2008.

[37] 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, 2018.

[38] Anubha Goel, Puneet Pasricha, and Aparna Mehra. Topological data analysis in investment decisions. *Expert Systems with Applications*, 147:113222, 2020.

[39] John C Gower and Gavin JS Ross. Minimum spanning trees and single linkage cluster analysis. *Applied statistics*, pages 54–64, 1969.

[40] Pierre Antoine Grillet. *Abstract algebra*, volume 242. Springer Science & Business Media, 2007.

[41] Ade Gunawan and M de Berg. A faster algorithm for dbscan. *Master's thesis*, 2013.

[42] David Gunning, Mark Stefik, Jaesik Choi, Timothy Miller, Simone Stumpf, and Guang-Zhong Yang. Xai—explainable artificial intelligence. *Science Robotics*, 4(37), 2019.

[43] Hongfeng Guo, Shengxiang Xia, Qiguang An, Xin Zhang, Weihua Sun, and Xinyao Zhao. Empirical study of financial crises based on topological data analysis. *Physica A: Statistical Mechanics and its Applications*, 558:124956, 2020.

[44] Wei Guo and Ashis G Banerjee. Identification of key features using topological data analysis for accurate prediction of manufacturing system outputs. *Journal of Manufacturing Systems*, 43:225–234, 2017.

[45] Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020.

[46] John A Hartigan. Consistency of single linkage for high-density clusters. *Journal of the American Statistical Association*, 76(374):388–394, 1981.

[47] John A Hartigan and Surya Mohanty. The runt test for multimodality. *Journal of Classification*, 9(1):63–70, 1992.

[48] Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

[49] Sven Heydenreich, Benjamin Brück, and Joachim Harnois-Déraps. Persistent homology in cosmic shear: constraining parameters with topological data analysis. *Astronomy & Astrophysics*, 648:A74, 2021.

[50] Shammamah Hossain, C Calloway, D Lippa, D Niederhut, and D Shupe. Visualization of bioinformatics data with dash bio. In *Proceedings of the 18th Python in Science Conference*, pages 126–133, 2019.

[51] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[52] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[53] Michael Kerber. Persistent homology: state of the art and challenges. *Internationale Mathematische Nachrichten*, 231(15-33):1, 2016.

[54] Vi Krishnamurthy. On the number of topologies on a finite set. *The American Mathematical Monthly*, 73(2):154–157, 1966.

[55] Joseph B Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

[56] Joseph B Kruskal. Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29(2):115–129, 1964.

[57] Joseph B Kruskal. *Multidimensional scaling*. Number 11. Sage, 1978.

[58] Devinder Kumar, Graham W. Taylor, and Alexander Wong. Opening the black box of financial AI with clear-trade: A class-enhanced attentive response approach for explaining and visualizing deep learning-driven stock market prediction. *CoRR*, abs/1709.01574, 2017.

[59] Max Z Li, Megan S Ryerson, and Hamsa Balakrishnan. Topological data analysis for aviation applications. *Transportation Research Part E: Logistics and Transportation Review*, 128:149–174, 2019.

[60] Derek Lo and Briton Park. Modeling the spread of the zika virus using topological data analysis. *Plos one*, 13(2):e0192120, 2018.

[61] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[62] T. Soni Madhulatha. An overview on clustering methods. *CoRR*, abs/1205.1117, 2012.

[63] Melissa R McGuirl, Alexandria Volkening, and Björn Sandstede. Topological data analysis of zebrafish patterns. *Proceedings of the National Academy of Sciences*, 117(10):5113–5124, 2020.

[64] Leland McInnes. Answer to github issue "[question] what's the scaling complexity?", November 2017.

[65] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[66] Elizabeth Munch. A user's guide to topological data analysis. *Journal of Learning Analytics*, 4(2):47–61, 2017.

[67] James R Munkres. *Topology*. Prentice Hall, 2000.

[68] Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97, 2012.

[69] Jessica L Nielson, Jesse Paquette, Aiwen W Liu, Cristian F Guandique, C Amy Tovar, Tomoo Inoue, Karen-Amanda Irvine, John C Gensel, Jennifer Kloke, Tanya C Petrossian, et al. Topological data analysis for discovery in preclinical spinal cord injury and traumatic brain injury. *Nature Communications*, 6:8581, 2015.

[70] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1):51–59, 1996.

[71] Małgorzata Olejniczak, Andre Severo Pereira Gomes, and Julien Tierny. A topological data analysis perspective on noncovalent interactions in relativistic calculations. *International Journal of Quantum Chemistry*, 120(8):e26133, 2020.

[72] Matthew Partridge and Rafael Calvo. Fast dimensionality reduction and simple pca. *Intelligent data analysis*, 2(3):292–298, 1997.

[73] Valerio Pascucci, Xavier Tricoche, Hans Hagen, and Julien Tierny. *Topological Methods in Data Analysis and Visualization: Theory, Algorithms, and Applications*. Springer Science & Business Media, 2010.

[74] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011.

[75] Charles C Pinter. *A book of abstract algebra*. Courier Corporation, 2010.

[76] Plotly. Dash user guide, 2021.

[77] Manya Raman-Sundström. A pedagogical history of compactness. *The American Mathematical Monthly*, 122(7):619–635, 2015.

[78] Sam Roweis. Em algorithms for pca and spca. *Advances in neural information processing systems*, pages 626–632, 1998.

[79] Manish Saggar, Olaf Sporns, Javier Gonzalez-Castillo, Peter A Bandettini, Gunnar Carlsson, Gary Glover, and Allan L Reiss. Towards a new approach to reveal dynamical organization of the brain using topological data analysis. *Nature communications*, 9(1):1–14, 2018.

[80] Jörg Sander, Xuejie Qin, Zhiyong Lu, Nan Niu, and Alex Kovarsky. Automatic extraction of clusters from hierarchical clustering representations. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 75–87. Springer, 2003.

[81] Erich Schubert and Michael Gertz. Improving the cluster structure extracted from optics plots. In *LWDA*, pages 318–329, 2018.

[82] Harlan Sexton and Mikael Vejdemo-Johansson. jPlex, December 2008. http://comptop.stanford.edu/programs/jplex/.

[83] Robin Sibson. Slink: an optimally efficient algorithm for the single-link cluster method. *The computer journal*, 16(1):30–34, 1973.

[84] Gurjeet Singh, Facundo Mémoli, and Gunnar E Carlsson. Topological methods for the analysis of high dimensional data sets and 3d object recognition. In *SPBG*, pages 91–100, 2007.

[85] Ann E Sizemore, Jennifer E Phillips-Cremins, Robert Ghrist, and Danielle S Bassett. The importance of the whole: topological data analysis for the network neuroscientist. *Network Neuroscience*, 3(3):656–673, 2019.

[86] Alexander D Smith, Paweł Dłotko, and Victor M Zavala. Topological data analysis: concepts, computation, and applications in chemical engineering. *Computers & Chemical Engineering*, 146:107202, 2021.

[87] Werner Stuetzle. Estimating the cluster tree of a density by analyzing the minimal spanning tree of a sample. *Journal of classification*, 20(1):25–47, 2003.

[88] Dane Taylor, Florian Klimm, Heather A Harrington, Miroslav Kramár, Konstantin Mischaikow, Mason A Porter, and Peter J Mucha. Topological data analysis of contagion maps for examining spreading processes on networks. *Nature communications*, 6(1):1–11, 2015.

[89] Chad M Topaz, Lori Ziegelmeier, and Tom Halverson. Topological data analysis of biological aggregation models. *PloS one*, 10(5):e0126383, 2015.

[90] Warren S Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419, 1952.

[91] Laurens Van Der Maaten. Barnes-hut-sne. *arXiv preprint arXiv:1301.3342*, 2013.

[92] Laurens Van Der Maaten. Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

[93] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[94] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research*, 11:2837–2854, 2010.

[95] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[96] Yuan Wang, Hernando Ombao, and Moo K Chung. Topological data analysis of single-trial electroencephalographic signals. *The annals of applied statistics*, 12(3):1506, 2018.

[97] Larry Wasserman. Topological data analysis. *Annual Review of Statistics and Its Application*, 5:501–532, 2018.

[98] Yi-Pu Wu, Jin-Jiang Guo, and Xue-Jie Zhang. A linear dbscan algorithm based on lsh. In *2007 International Conference on Machine Learning and Cybernetics*, volume 5, pages 2608–2614. IEEE, 2007.

[99] Xin Xu, Jessi Cisewski-Kehe, Sheridan Beckwith Green, and Daisuke Nagai. Finding cosmic voids and filament loops using topological data analysis. *Astronomy and Computing*, 27:34–52, 2019.

[100] Afra Zomorodian. Topological data analysis. *Advances in applied and computational topology*, 70:1–39, 2012.

[101] Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 33(2):249–274, 2005.