A study of the persistent homology pipeline in data analysis

Contents

1 Introduction

Topological Data Analysis (TDA) emerged approximately three decades ago as a fusion of algebraic topology and data analysis, and aimed to rigorously analyse the "shape" of data [Was16]. Since then, it has been an epicentre of computational breakthroughs and researchers from disciplines as varied as computer science, applied mathematics, biology, or sociology have been attracted by the prospect of further groundbreaking advances in the theoretical and practical aspects of TDA. Recently, a subbranch of TDA called Persistent Homology (PH), which allows for data simplification and structure characterisation, has been used in conjunction with modern machine learning (ML) algorithms to produce very successful results [PXL18]. This success is being seen in an increasingly diverse set of fields, including protein analysis [XW14], drug design [CMW18], shape analysis and recognition [Rei+14], natural-language analysis [Zhu13], the study of financial markets [Lei+08], and many more. Therefore, TDA is extremely exciting not only because it provides an innovative perspective on data analysis, by incorporating tools from radically different branches of mathematics, but also because it has been proven to offer solutions and insights on diverse research areas outside of mathematics.

The aim of this essay is to describe the typical PH pipeline and how it can be used with machine learning classification algorithms. In chapter 2 we describe the general PH pipeline, in which data is transformed into geometric simplices which are then studied through the algebraic topology lens to produce a topological descriptor of the data. More concretely, data such as point cloud data or image data is transformed into a sequence of nested simplicial complexes. The specific construction of such a sequence depends on the nature of the data being used. For point cloud data, for example, we can form a simplicial complex by taking the data points as the zero-skeleton and building the complex combinatorially by considering the distance between the data points to form the higher order skeletons. Then, by studying the relationship between the homology groups of the different complexes in the sequence we produce the so-called persistent homology of the sequence, which is a topological descriptor of the data which measures how the generators of the homology groups persist through the sequence of complexes. Concretely, the persistent homology of a sequence of complexes is a multiset in $\mathbb{R}^2$, which we call a barcode and lies inside a natural metric space. In chapter 3 we describe the general framework to vectorise such topological descriptors so that they can be inputted into ML algorithms. This implies embedding the space of such topological descriptors into Hilbert spaces, which are the natural spaces in which machine learning algorithms typically work. In chapter 4 we describe an alternative approach to the vectorisations studied in chapter 3, namely endowing the space of topological descriptors with a similarity function (a kernel) to be able to make use of kernel methods, a subclass of ML methods, directly on the topological descriptors. This removes the necessity of finding an appropriate embedding into a Hilbert space, but the kernel must satisfactory discern different data points. Finally, in chapter 5 we move from theory to practice by studying how the methods presented in the two previous chapters perform on different data sets.

Currently, the theoretical framework of PH is well-established (see chapter 2) and a vast amount of research is being conducted to propose better vectorisations and kernels. However, the benchmarking being conducted is insufficient and the lack of accessibility to such methods makes it very difficult for them to become popular. In the past five years, there has been an increasing effort to make these methods more accessible to the public and to produce a consistent benchmarking to describe the alternatives available, see [Ali+22], [PXL18] or [BPP21] for example. However, the comparison and study of such methods is

very incomplete. There has been no systematic analysis of the different vectorisation and kernel methods. In particular there are 3 main problems:

1. All the studies tend to implement the algorithms on the same 5 data sets, and some of the vectorisation methods seem to have been specifically tuned to give high accuracies on these data sets. Therefore, accuracy figures are not representative of the real utility of these methods.

2. There is minimal work among those carrying out the benchmarking regarding how to select a method for a specific data type. This is more generally a problem with modern ML. Given a specific problem, which algorithm should be used? There are some faint general guidelines in ML but these become even less relevant when it comes to persistent homology based machine learning.

3. When the methods are tested on data, there is very little analysis conducted apart from accuracy figures and error types. Studies seem to prioritise the finding of the method which yields the highest accuracy with little regard to the mechanisms which could explain the results.

Due to the limitations of this project, we can not tackle the first two problems. Nevertheless, throughout the essay, and specifically in chapter 5, we will try to investigate the third problem. Hopefully, it will be clear at the end of the essay that PH is an extremely exciting and fast-moving field of modern applied mathematics which has a lot of unsolved questions, but which is also very promising.

2 The Persistent Homology Pipeline

In this section, we discuss the persistent homology pipeline; that is, the process data undergoes until is ready to be inputted into machine learning algorithms. The first step is to convert the data (digital image, point cloud data, networks, time series...) into a filtered complex. Intuitively, this can be viewed as a nested sequence of geometric complexes which approximate the structure of the data. The exact construction of such a filtered complex depends on the nature of data, and even then several constructions are available. This is the content of subsection 2.1.

Once we have produced a filtered complex, topological information about the data is extracted by computing the so-called *persistent homology*, using tools from algebraic topology, which is in correspondence with a topological summary called a *barcode* or *persistence diagram*. These are central objects in the study of TDA. The brilliance of such a construction is that they are stable under small perturbations of the initial data; that is, similar data sets should give similar barcodes. This is particularly insightful in classification tasks as data points coming from the same class should be similar, so barcodes can be used for such classification tasks. The computation of persistent homology, its relation to the barcode and an explanation of how these objects provide an insight into the geometry and structure of the data is explained in subsection 2.2. Subsequently, in subsection 2.3 we also formalise the notion of similarity in context of data and barcodes by presenting metrics with which the space of barcodes can be endowed.

Finally, in subsection 2.4 we explain how the barcode can be used to analyse the data. In essence, topological features of the data can be directly inferred from the barcode but it is much more common to use vectorisation or kernel methods in conjunction with machine learning algorithms.

2.1 Filtered Complexes from Data

The aim of this section is to explain how we can extract a *filtered complex* from data. This filtered complex is able to encapsulate the topological information intrinsic to the data which is then further processed into a *barcode*. The construction of such a barcode and its relation to the topological information of the data will be explored in the subsequent section however.

Let $\bar{\mathbb{R}}$ be the set of extended reals with the canonical partial order and fix a chain complex (C, d). Consider the set $\{F_r(C)\}_{r\in\bar{\mathbb{R}}}$ of subcomplexes of (C, d) with the property that $F_r(C) \subseteq F_{r'}(C)$ if and only if $r \leq r' \in \bar{\mathbb{R}}$. This is called an $\bar{\mathbb{R}}$-*filtration* and together with the original chain complex we call this an $\bar{\mathbb{R}}$-*filtered chain complex*. From now on we will simply call this a filtered complex. [BK07]

2.1.1 Point Cloud Data and Vietoris-Rips Filtration

Let S be a set and d a measure of distance between the elements of S. We should think of S as a sample taken from an object $\mathbb{X}$ whose shape or homological features we wish to study. We call such a pair (S, d) *point cloud data*. For example, we could get a sample of points from the 2-sphere $S^2 \subset \mathbb{R}^3$ and endow these points with the metric coming from $\mathbb{R}^3$. This is the essence of a classical problem in TDA: how can we infer topological properties of an object if we only have sample points from such an object? [Per18]

A popular strategy is to build a simplicial complex which approximates the underlying structure of $\mathbb{X}$, the so-called *Vietoris-Rips Complex*. For a fixed $\epsilon \geq 0$, consider the set

$$R_\epsilon = \{\{x_1, \ldots, x_n\} \subset S : d(x_i, x_j) \leq \epsilon \text{ for all } 1 \leq i, j \leq n, n \in \mathbb{N}\}.$$

Then the Vietoris-Rips Complex $R_\epsilon(S)$ is the simplicial complex formed by vertices $\{x_0\}$, edges $\{x_0, x_1\}$, triangles $\{x_0, x_1, x_2\}$ and so on. Notice that if $\epsilon_1 \leq \epsilon_2$ then $R_{\epsilon_1}(S) \subseteq R_{\epsilon_2}(S)$. With this in mind, we obtain a filtered chain complex $R(S) = \{R_\epsilon(S)\}_{\epsilon\in\mathbb{R}}$, where we take $R_\epsilon(S) = \emptyset$ for $\epsilon < 0$.

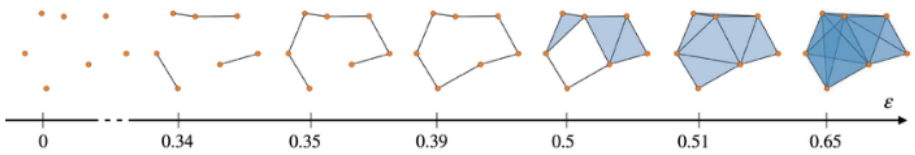

Figure 1: Nested VR-complexes for point cloud data in $\mathbb{R}^2$ with the canonical metric. Taken from [Gue+21]

A very similar construction to the above is the *Čech complex*. Consider the same setup as above and for a given $\epsilon > 0$ define $S_\epsilon = \cup_{x_i\in S} B(x_i, \epsilon)$. Then consider the simplicial complex with k-simplices spanned by $\{x_{i_1}, \ldots, x_{i_k}\}$ if and only if

$$\cap_{1\leq j\leq k} B(x_{i_j}, \epsilon) \neq \emptyset.$$

This is called the *nerve* of S_ϵ or the Čech complex of S at scale ϵ, which we denote by $\check{C}_\epsilon(S)$. Again, the set $\check{C}(S) = \{\check{C}_r(S)\}_{r\in\bar{\mathbb{R}}}$ forms a filtered complex [Ott+17]. The Nerve lemma ensures that $\check{C}_\epsilon(S)$ is homotopy equivalent to S_ϵ, which implies that these two spaces have identical homology groups [Ghr14]. Therefore, intuitively the Čech complex seems an

appropriate simplicial approximation of the space $\mathbb{X}$. For a more rigorous confirmation of this intuition refer to chapter 2 of [Ghr14].

The problem with the construction of the Čech complex is that it is computationally very expensive, so it is common to use the Vietoris-Rips complex as an alternative. On the one hand, in the Vietoris-Rips complex we only need to store the pairwise distances between the vertices to be able to reconstruct the complex. On the other hand, this is obviously not true for the Čech complex: the 1-skeleton alone cannot encode all the information about the complex. In addition to this, the computational complexity of constructing the k-skeleton in the Čech complex is higher than that of the VR-complex. In the latter case, for a specific ϵ, the VR-complex is equivalent to the clique complex [1] of the graph with edges joining all of those vertices which are at a distance less than ϵ from each other. Notice that this is only linearly dependent on the underlying dimension of the points. However, to check whether a specific set of k vertices forms part of the Čech complex, we need to check for the intersection of k balls in, normally, high dimensions. This is computationally equivalent to the bounding sphere problem, which runs in $O(k)$ time with an exponential dependence on the dimension of the underlying metric space [Cha18]. In any case, a substantial amount of research has been carried out to optimise the computation of VR-complexes and papers like [Zom10] present algorithms which are shown to be empirically more efficient than the Čech counterparts.

The question is now whether there is a theoretical basis to assert the the VR-complex is topologically similar to the underlying space $\mathbb{X}$. Chambers et al., [Chm+10], show that introducing a small amount of perturbation in a VR-complex can lead to a 'quasi'-Vietoris-Rips complex of arbitrary fundamental group. Moreover, it is possible to obtain high dimensional complexes from an underlying low dimensional space leading to the possibility of non-trivial homology groups in high dimensions. Indeed, it is possible to arrange points in $\mathbb{R}^2$ to form a VR-complex homotopy equivalent to $\mathbb{S}^k$ for any $k \in \mathbb{N}$, [Ghr14]. Despite the apparent lack of stability of the VR-complex, there are two theoretical pillars which serve as justification for the use of a VR-complex as a simplicial approximation.

The first is that the topological noise introduced by perturbations can be mitagated by looking at VR-complexes of different scales ϵ. Indeed, this is the purpose of *persistent homology*, which will be introduced in the next subsection. The second ground for using VR-complexes is that they provide good approximations for Čech complexes, indeed consider the following lemma.

Lemma 2.1 (de Silva [SG07]). *Let $\mathbb{X}$ and S be defined as above. Then, for any $\epsilon > 0$, there is a chain of inclusion maps*

$$R_\epsilon(S) \hookrightarrow \check{C}_{\sqrt{2}\epsilon}(S) \hookrightarrow R_{\sqrt{2}\epsilon}$$

Intuitively, this lemma shows that we can approximate the Čech complex by a VR-complex. In the next subsection, once we describe the construction of persistent homology, we will explain more rigorously why this lemma provides a theoretical basis to assert that Čech complexes are topologically similar to VR-complexes.

2.1.2 Image Data, Lower-Star Filtration and Cubical Filtration

Given a general topological space $\mathbb{X}$ and a scalar-function $f : \mathbb{X} \to \mathbb{R}$, we can create a sequence of sublevels sets $\mathbb{X}_a = f^{-1}((-\infty, a])$ for $a \in \mathbb{R}$. This is called the *sublevel set*

[1] This is defined in subsection 2.3, but in essence a set of k vertices forms a k-simplex if the pairwise distance of all these vertices does not exceed ϵ.

filtration. We can apply a similar idea to images to get a filtered simplicial complex.

A two-dimensional image is a real-valued function over a regular grid $\mathbb{X} = \{(i,j) \in \mathbb{Z}^2 : 0 \leq i,j \leq N\}$. Call this $\phi : \mathbb{X} \to \mathbb{R}$. For example, a grey-scale image is a function $\phi : \mathbb{X} \to [0,255]$ where the number corresponds to how dark the pixel is. In order to form a filtered simplicial complex out of the image we first need to triangulate the grid. A conventional approach is via a Freudenthal triangulation of the plane (see lemma 1 in [Kuh60]) from which we obtain a simplicial complex $K = K(\phi)$. Once this is done, we can extend ϕ to K by linear interpolation on each simplex using the vertices conforming it. It can be shown that this produces a simplicial complex which is homotopy equivalent to that obtained by taking $K_a = \{\sigma \in K : \max_{v \in \sigma} \phi(v) \leq a\}$. This is called the *lower-star filtration* of K.

Notice that this procedure applies to any simplicial complex K which has been endowed with some function $f : V \to \mathbb{R}$, where $V = K^0$ is the set of vertices (or 0-skeleton) of K. Sometimes the pixel intensity function proposed in the paragraph above is not enough to produce filtered complexes which will be distinguishable by topological methods. Indeed, consider the renowned MNIST database consisting of hand-written digits, conventionally used in machine learning algorithms. Using topological methods and the above lower-star filtration, it would be very hard to distinguish a 6 from a 9: they both have 1 connected component and 1 hole!

Figure 2: A sample of a 6 and a 9 taken from the MNIST database.

To circumvent this problem and similar ones, we have to refine the scalar function to encode some information about the geometry of the data. Adcock et al. [ACC16] offer a lower-star filtration based on "sweeping functions" where pixels are converted into a binary response and the intensity of the response depends on the position of the pixel. In particular, all white pixels get mapped to 0 and any non-white pixel gets mapped to a value depending solely on their position in the grid. For a "down to up sweep" function we could number the rows in the grid from 1 to n and produce a function which sends all the non-white pixels in row i to $i \in \mathbb{R}$. They implement four sweeping functions to obtain 4 different filtrations which they then analyse using conventional machine learning and topological methods. It might be tempting to conclude from the above discussion that TDA is not as powerful as claimed, since alone it is not able to distinguish objects like a 6 and a 9, which seem blatantly different to the human eye. How can TDA then distinguish more complex structures like large chemical molecules if it fails on such a comparatively simple task? This conclusion, however, is misleading for several reasons. Despite the fact that TDA can be used alone to distinguish complicated objects such as proteins, see [XW14], we should not in general assess TDA based on its performance as a stand alone tool. TDA becomes powerful when used in conjunction with other statistical methods and incorporated into existing pipelines for data analysis. For instance, even if PH cannot distinguish between a 6 and 9, PH can be

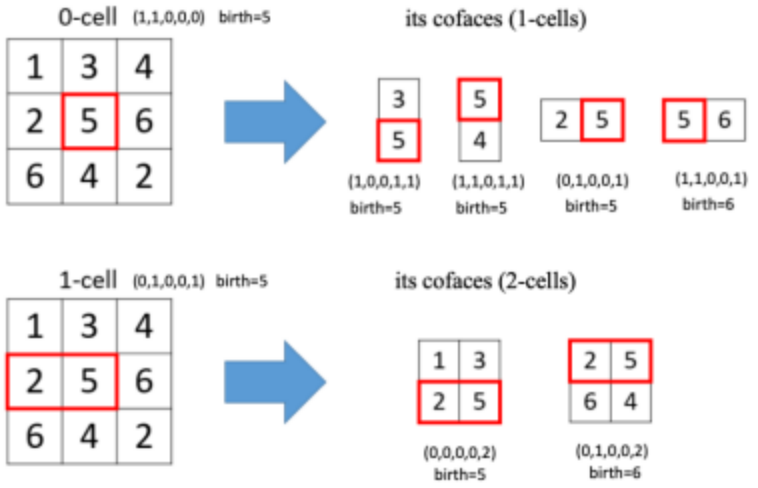

Figure 3: A 3 by 3 grid with numbers on each pixel representing the value of ϕ on it. On red we observe an example of a 0- and 1-cell and their cofaces together with their birth-time.

incorporated into traditional machine learning pipelines for MNIST classification and can produce results comparable to that of state-of-the-art algorithms using neural networks.

An alternative to the above procedure exists to produce a filtered complex called the *filtered cubical complex*. This is widely used in applications since a lot of efficient algorithms like Ripser [TSB18] make use of this instead of other traditional simplicial complexes. As before, suppose we are given a grid $\mathbb{X}$ and an image $\phi : \mathbb{X} \to \mathbb{R}$. With this information we can produce a cubical complex $K(\phi)$ consisting of *cubical cells*. These are identified with a tuple (x, y, d, m) where x, y denote the location of the cell, $d \in \{0, 1, 2\}$ the dimension of the cell and m the type of cell. The 0-cells, that is $d = 0$, only come in one type, $m = 0$, and so they are simply the pixels of the image. Their birth-time is given by $\phi(x, y)$ so they only become part of the complex $K(\phi)^i$ when $i \geq \phi(x, y)$. Then 1-cells, that is $d = 1$, come in two types. For $m = 0$, $(x, y, 1, 0)$ consists of the pair of pixels (x, y) and $(x, y + 1)$ whereas the 1-cell $(x, y, 1, 1)$ of type $m = 1$ consists of the pair of pixels (x, y) and $(x+1, y)$. In a similar manner we construct 2-cells. Cells are killed when they merge into higher-dimension cells. The diagram in figure 3 taken from [KSA20] helps with the visualisation of the construction described.

2.1.3 Other Filtrations and Practicalities

The above constructions present some of the most widely used filtrations in TDA at the time of writing. Nevertheless, there exists a wide range of alternatives used in the theoretical branch of TDA as well as other filtrations appropriate for other data types.

For instance, networks are commonly used in data analysis to study complex interactions between agents. An undirected graph is already a 1-dimensional simplicial complex, and if the graph is weighted we can create the obvious filtered complex. However, this only produces complexes of dimension up to 1 so if we wish to obtain more refined information we should be able to extend this to higher dimensional complexes. The simplest method is the

weight rank clique filtration (WRCF). This produces a $k-$simplex with vertices $\{v_0, \ldots, v_k\}$ if these vertices form a *clique*, that is, if they form a complete subgraph. For a more detailed account of the WRCF, its applications and further discussion of TDA used in network theory refer to [Pet+13].

Another simple method to study connected networks is to map the vertices of the graph to a finite metric space. Here, the distance between two vertices is just the length of the shortest path between the vertices in the graph (notice how here we require the graph to be connected).There are other ways of creating filtered complexes from networks and, more generally, of using TDA methods on them. For an account of this refer to [HMR09].

Other data types like time series [Ma20] or 3D volume data [KSA20] can also be endowed with filtrations, but we shall not pursue this topic any further. In addition to this, if we go back to the most basic data set, the point cloud data, the literature also has a compendium of filtered complexes with which we can endow the data. Concerns such as the computational time to construct the complex or its topological stability under perturbations plays an important role in the theoretical development of such complexes. Again, we will not investigate this subject any further but the interested reader should consult [Ott+17] for a comprehensive discussion of these matters.

2.2 Algebraic Construction

In this section, we will introduce the concept of persistent homology and explain why it encapsulates the topological information of the data being analysed. We will also explain why the persistent homologies of an object are in bijection with the barcodes associated with them.

Let $K = \{K^i\}_{i\in\bar{\mathbb{R}}}$ be a filtered complex as defined in section 2.1. Recall that $K^i \subseteq K^{i+p}$ via an inclusion map $f_{i,p} : K^i \hookrightarrow K^{i+p}$. To simplify our definitions we will restrict our filtered complex by considering a countable number of complexes which we will index by $\mathbb{N}$.

Definition 2.2. The *p-persistent k^{th} homology groups* are given by $\mathrm{im} f^k_{i,p}(K)$, where $f^k_{i,p} : H_k(K^i) \to H_k(K^{i+p})$ is the homomorphism induced in homology by the inclusion map, for each i. Moreover, if γ is a class in $H_k(K^i)$ we say that it is *born* at K^i if $\gamma \notin H_k(K^{i-1})$. Similarly, a class γ born in K^i *dies* entering K_j if $f^k_{i,j-1}(\gamma) \in H_k(K^{j-1})$ but $f^k_{i,j}(\gamma) \notin H_k(K^j)$. Finally, the *persistence* or *lifetime* of such a class is $j - i$ [EL10].

Intuitively, if we take our filtered complex from a Čech or Vietoris-Rips complex coming from point cloud data, the notion of persistence is of geometric importance because it reveals the topological features which persist during a range of values of ϵ. For example, the p-persistent first homology group describes the holes which remain in our filtration and have a lifetime of p. This uncovers some of the geometry of the underlying shape: indeed, long lived k^{th}-homology classes are an indication of the existence of k-dimensional holes in the underlying space.

Moreover, the notion of persistent homology finally elucidates why the VR-complex serves as a good approximation for a Čech complex. By lemma 2.1, we observe that if $j/i \geq \sqrt{2}$ then $\mathrm{im} f^k_{i,j-i}(R(S)) \neq 0$ implies that $H_k(\check{C}_{\sqrt{2}i}(S)) \neq 0$. Therefore, holes in the Čech complex can be inferred from the persistent homology of the VR-complex.

We now provide some algebraic background to build up to the correspondence between persistent homology and barcodes. See [ZC05] for a more detailed account.

Definition 2.3. A *persistence complex* K is a family of chain complexes $\{K^i_*\}_{i\geq 0}$ over a ring R, together with chain maps $f^i : K^i_* \to K^{i+1}_*$.

Our canonical example will always be a filtered complex coming from data. The inclusion maps form chain maps as they commute with the canonical boundary maps of the simplicial complexes.

Definition 2.4. A *persistence module* M is a family of $R-$modules M^i, together with homomorphisms $\phi^i : M^i \to M^{i+1}$.

Again, the homology of a persistence complex is a persistence module, where ϕ^i is again the homomorphism in homology induced by the successive inclusion maps. Moreover, we will require some sort of constrain on the class of persistence complexes we are working with if we aspire to get any classification theorem. Our constrains should be sufficiently general to allow us to work with a wide range of complexes arising in data but sufficiently strict to allow for a classification theorem. A good balance seems to be persistence complexes and modules of finite type:

Definition 2.5. A persistence complex $\{K^i_*, f^i\}$ (respectively persistence module $\{M^i, \phi^i\}$) is of *finite type* if

1. Each component complex (module) is a finitely generated $R-$module,
2. The maps f^i (respectively, ϕ^i) are isomorphisms for $i \geq n$ for some $n \in \mathbb{N}$.

Notice that the complexes explored in subsection 2.1 arising from data are *finite* simplicial complexes, which generate complexes of finite type whose homology are persistence modules of finite type too. Indeed, eventually all the homology groups will be 0 so we get isomorphisms between them.

With this setup in mind, we wish to endow our persistence modules with a graded $R[t]-$module structure. Let $\{M^i, \phi^i\}_{i\geq 0}$ as above be a persistence module over a ring R. Then, we can endow

$$\alpha(M) = \bigoplus_{i=0}^{\infty} M^i$$

with the $R-$structure coming from M and the $t-$action given by

$$t \cdot (m^0, m^1, m^2, \ldots) = (0, \phi^0(m^0), \phi^1(m^1), \phi^2(m^2), \ldots).$$

Theorem 2.6 (Correspondence Theorem I)**.** *The correspondence α defines an equivalence of categories between the category of persistence modules of finite type over R and the category of finitely generated non-negatively graded modules over $R[t]$ [ZC05].*

Notice that this reduces the classification of persistence modules of finite type to finitely generated non-negatively graded modules over $R[t]$. Notice that for rings like $R = \mathbb{Z}$ which are not fields, theorem 2.6 suggests that there is no trivial classification theorems as we know from commutative algebra that the classification of modules over $\mathbb{Z}[t]$ is a very complicated matter. On the other hand, when $R = \mathbb{F}$ is a field we know that $\mathbb{F}[t]$ is a PID and hence we do have a classification theorem for $\mathbb{F}[t]-$modules which can be used to our advantage:

Theorem 2.7 (Structure theorem)**.** *If D is a PID, then every finitely generated $D-$module M is isomorphic to a unique $D-$module of the form*

$$D^m \oplus \left(\bigoplus_{i=1}^{n} D/d_i D \right),$$

where $d_i \in D$ with $d_i \mid d_{i+1}$ and $n, m \in \mathbb{N}$. Similarly, every graded module M over a graded PID D decomposes uniquely into the form

$$\left(\bigoplus_{i=1}^{n} \nabla^{\alpha_i} D\right) \oplus \left(\bigoplus_{j=1}^{m} \nabla^{\gamma_j} D/d_j D\right),$$

where $d_j \in D$ are homogeneous elements with $d_j \mid d_{j+1}$, $\alpha_i, \gamma_j \in \mathbb{Z}$ and ∇^{α} is the $\alpha-$shift upwards in grading.

In terms of our canonical persistence module, the PID is $\mathbb{F}[t]$ and the shifting operator ∇ is precisely $t \in \mathbb{F}[t]$. Hence, for a given filtered complex coming from data the correspondence and the structure theorem gives us a canonical way of representing the persistence module arising from it.

However, we wish to refine this to get a more intuitive representation of the module. To do so, notice that $\{n, m, d_i, \alpha_i, \gamma_j\}$ fully parametrise the module above. With this in mind, we create a bijection $\mathcal{Q}$ which sends a graded $\mathbb{F}[t]-$module to a set $\mathcal{S}$ of $\mathcal{P}-$intervals, where a $\mathcal{P}-$interval is simply an ordered pair (i, j) with $0 \leq i < j \in \mathbb{Z} \cup \{\infty\}$. We define $\mathcal{Q}(i, j) = t^i \cdot \mathbb{F}[t]/(t^{j-i})$ and $\mathcal{Q}(i, +\infty) = t^i \cdot \mathbb{F}[t]$. For a set of $\mathcal{P}-$intervals $\mathcal{S} = \{(i_1, j_1), \ldots, (i_n, j_n)\}$ we define $\mathcal{Q}(\mathcal{S}) = \bigoplus_{k=1}^{n} \mathcal{Q}(i_k, j_k)$. With these definitions, we can reformulate the correspondence theorem as follows.

Corollary 2.8 (Correspondence Theorem II)**.** *The map $\mathcal{S} \to \mathcal{Q}(\mathcal{S})$ defines a bijection between the finite sets of $\mathcal{P}-$intervals and the finitely generated graded modules over $\mathbb{F}[t]$. Hence, there exists a correspondence between the classes of persistence modules of finite types over $\mathbb{F}$ and the finite sets of $\mathcal{P}-$intervals.*

Let us summarise the above construction now. Given some data S we produce a filtered complex $K = \{K^i\}_{i \geq 0}$, which can be viewed as a persistence complex, with inclusion maps between the chain complexes acting as chain maps. Fix now $\mathrm{k} \in \mathbb{N}$ and consider the k^{th}-homology modules for each complex K^i. These form a persistence module M with maps between the homology modules induced by the chain maps.

Definition 2.9. With the above setup, we call this the *persistent homology* of K, and we denote it

$$\mathcal{PH}(K) = \bigoplus_{i \geq 0} H_k(K^i).$$

The persistence complex and module are both of finite type in general so we are in a position to use the correspondence theorems, which tell us that

$$\mathcal{PH}(K) \cong \left(\bigoplus_{i=1}^{n} t^{\alpha_i} \cdot \mathbb{F}[t]\right) \oplus \left(\bigoplus_{j=1}^{m} t^{\gamma_j} \cdot \mathbb{F}[t]/t^{\gamma_j + m_j}\mathbb{F}[t]\right).$$

Therefore, the persistent homology is fully parametrised by a finite set of $\mathcal{P}-$intervals. We call such a set the *barcode* or *persistence diagram* (PD) of the filtered complex K [EL10], and we denote it by $\mathrm{Dgm}_k(K)$. When the filtration comes from a level-set function we sometimes denote it $\mathrm{Dgm}_k(f)$ instead. Strictly speaking a barcode is the collection of intervals (see figure 4), whilst the persistence diagram is the multiset consisting of the endpoints of these intervals. In this essay we will use the concepts interchangeably.

The above isomorphism has a natural interpretation. The torsionless elements of the module correspond to the homology classes which are born at times α_i and persist throughout the filtration. On the other hand, the elements with torsion correspond to those homology classes which are born at time γ_j and die at time $\gamma_j + m_j$. Hence, a barcode encodes the lifetime of the different homology classes which appear during the filtration.

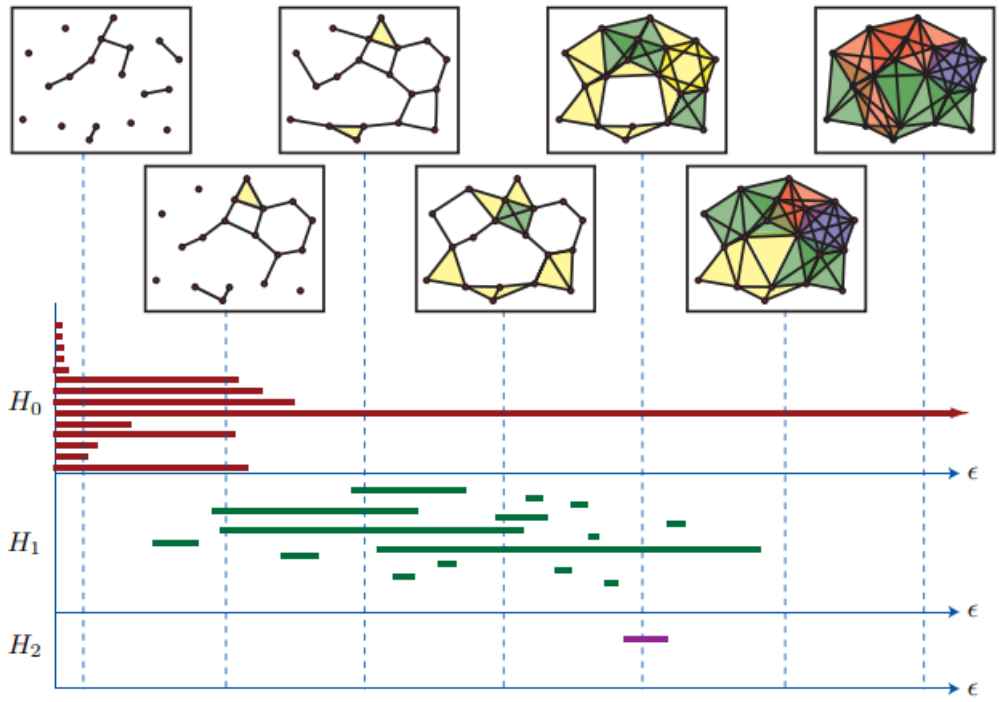

Figure 4: A visual representation of the barcodes for the first three persistent homologies of a filtered complexes arising from point cloud data. Taken from [Ghr08]

We should think of barcodes as analogous to the Betti numbers $\mu^k = \text{rank} H^k$. Indeed, A. Zomorodian and G. Carlsson prove in [ZC05] the following theorem which arises from the above Correspondence theorem.

Theorem 2.10 (Barcode Theorem)**.** *Let K be a persistence complex as above. Then, the rank of the p-persistent k^{th}-homology group $imf^k_{i,p}(K)$ is equal to the number of barcodes of $Dgm_k(K)$ spanning the interval $[i, i+p]$. In particular, $H_k(K^i)$ is equal to the number of intervals containing i.*

The Barcode theorem tells us that barcodes completely determine the Betti numbers of the persistent homology of our filtration. Although in general Betti numbers do not completely describe the homology groups they come from, when we work over a field $\mathbb{F}$ they do. Indeed, fields have no torsion elements so the classification theorem tells us that $\mathbb{F}-$modules are free. In other words, they are simply the free vector space $\mathbb{F}^m$ and hence their rank completely determines the module.

For this reason, in most applications we use persistent homology with field coefficients. In this setup, barcodes provide a complete description of the persistent homology. On the

other hand, we loose the finer geometric structure that would arise from the torsion elements if we were working with PIDs like $\mathbb{Z}$. Nevertheless, since in most applications we are working with noisy data, we want to be able to filter this out so persistent homology over a field provides an efficient tool. In addition, it has been shown that persistent homology is able to detect finer geometric structure. P.Bubenik et al. show in [Bub+20] that persistent homology is able to detect curvature. They sampled from objects of different curvatures but with identical homologies, and showed that the persistent homology of the sampled data is enough to distinguish them. Therefore, it seems reasonable to conclude that the persistent homology of the filtrations arising from data, even if it is over a field, provides a suitable encoder of topological information.

2.3 Metrics and Stability

In the previous sections we showed how to obtain a topological summary of some data in the form of a finite multiset. We called this a barcode or persistence diagram (PD). However, in order for this to be of any use, PDs arrising from similar data must be similar too. The aim of this section is to formalise the concept of similarity and to explain some stability results; that is, theorems showing that small perturbations of the data lead to small perturbations of the PD. We will follow the book by Edelsbrunner et al. [EL10].

For the sake of simplicity let us focus on sublevel filtrations introduced in section 2.1. Let $\mathbb{X}$ be a triangulable topological space and $f : \mathbb{X} \to \mathbb{R}$ continuous. Recall that the subcomplexes $\mathbb{X}_a = f^{-1}((-\infty, a])$ form a filtered complex. Recall too that the images of $f^k_{i,p} : H_k(\mathbb{X}_i) \to H_k(\mathbb{X}_{i+p})$ for varying i forms the $p-$persistent k^{th}-homology group. Notice that if the topology of the simplices does not change for a range of values of p then our map is an isomorphism. This motivates the definition of a *homological critical value* $a \in \mathbb{R}$, which are those values for which there does not exist an $\epsilon > 0$ for which $f^k_{i-\epsilon,p+2\epsilon}$ is an isomorphism for each dimension k. Finally, this gives rise to a class of functions whose filtrations are appropriately stable.

Definition 2.11. Following the above notation, a function $f : \mathbb{X} \to \mathbb{R}$ is *tame* if it has only finitely many homological critical values and all the homology groups of all sublevel sets have finite rank.

As a remark, it is important to realise that in practice this is not a constrain to the constructions we are working with. Indeed, finite point cloud data or finite images give rise to finite simplicial complexes. For VR-complexes, we can construct a tame f on the geometric realisation of the complex for a sufficiently large time such that the level sets correspond to all the complexes which appear in the filtration. Therefore, all the results from this section can be applied to the described complexes. We now wish to compare complexes $K(f), K(g)$ arising from two tame functions f, g. In order to do so we need to explore metrics on PDs.

Let X, Y the two persistence diagrams, on which we add infinitely many points on the diagonal, each with infinite multiplicity. In this scenario, we are viewing the persistence diagrams as points on the plane. This is a necessary requirement for the definition of the metric.

Definition 2.12. The *bottleneck distance* between the diagrams X, Y is given by

$$W_\infty(X, Y) = \inf_{\eta: X \to Y} \sup_{x \in X} \|x - \eta(x)\|_\infty,$$

where the infimum is taken over all bijections $\eta : X \to Y$, and for $x = (x_1, x_2)$ and $y = (y_1, y_2)$ we define $\|x - y\|_\infty = \max\{|x_1 - y_1|, |x_2 - y_2|\}$.

This definition reveals the utility of adding points in the diagonal. These points do not change the persistence module but in order to produce a bijection we need sets of equal cardinality. Moreover, when there is an unequal number of points on the PDs, for those points without a pair we compute their distance (in the above $\|\cdot\|_\infty$ sense) to the diagonal. Since points further away from the diagonal have longer lifetimes, this seems to be an appropriate way of weighting unpaired elements of a PD. In addition to this, it is clear that $W_\infty(X, Y) = 0$ if and only if $X = Y$, $W_\infty(X, Y) = W_\infty(Y, X)$ and $W_\infty(X, Z) \leq W_\infty(X, Y) + W_\infty(Y, Z)$. Therefore, W_∞ satisfies the axioms of a metric so the space of persistence diagrams forms a metric space. Now we are ready to state our first stability result.

Definition 2.13 (Stability Theorem for Tame Functions)**.** Let $\mathbb{X}$ be a triangulable topological space and $f, g : \mathbb{X} \to \mathbb{R}$ two tame functions which produces peristence diagrams $\mathrm{Dgm}_k(f)$ and $\mathrm{Dgm}_k(g)$ for each dimension k. Then we have

$$W_\infty(\mathrm{Dgm}_k(f), \mathrm{Dgm}_k(g)) \leq \|f - g\|_\infty.$$

Despite the utility of the stability theorem, the bottleneck distance is insensitive to changes in the bijection beyond the pair of points further away from each other. This motivates the definition for another metric which takes this into account.

Definition 2.14. The *degree q Wasserstein distance* between the persistence diagrams X, Y is given by

$$W_q(X, Y) = \left[\inf_{\eta : X \to Y} \sum_{x \in X} \|x - \eta(x)\|_\infty^q \right]^{1/q}.$$

As suggested by the notation, the bottleneck distance is the limit of the Wasserstein distance as $q \to \infty$. Similarly, the Wasserstein distance is a metric with which we can endow the space of persistence diagrams. Notice, however, that the previous stability theorem does not hold for Wasserstein distances. Indeed, for any tame function $f : \mathbb{R} \to \mathbb{R}$ we can find a function $g : \mathbb{R} \to \mathbb{R}$ which is in the $\epsilon-$neighbourhood of f but with a large total variation in comparison to f. Nevertheless, there is a similar stability theorem which we sketch below. For a more concise statement and proof refer to [EL10].

Theorem 2.15 (Stability Theorem for Lipschitz Functions)**.** *Let $f, g : \mathbb{X} \to \mathbb{R}$ be tame Lipschitz functions on a triangulable, metric space with some extra structure. Then, there exists constants $C, p \geq 1$ such that*

$$W_q(Dgm_k(f), Dgm_k(g)) \leq C\|f - g\|_\infty^{1-p/q},$$

for all $q \geq p$.

These two metrics and stability theorems form the theoretical basis of much of modern persistent homology theory. It is one of the pillars of why current topological analyses of data via persistent homology works! In addition to this, it has been shown in [Coh+10] that the metric space is complete and separable, and thus provides a suitable setting for statistics and probability. However, despite these good news, the machinery we can use on

the space of barcodes is limited and some further processing needs to be done in order to apply a wider range of statistical tools. Indeed, even the concept of a Fréchet mean barcode is not well defined as the geodesics of this space are not unique. In the next subsection we will look at the main ways of processing or analysing barcodes.

2.4 The End of the Pipeline

In this subsection we briefly cover the main ways in which barcodes are used to analyse data. The content of the next two chapters is to investigate more rigorously some of the methods described here.

Firstly, a minority of the data analysis done presently is performed directly on the barcodes arising from data. For instance, Xia et al. study in [XW14] the structure and flexibility of proteins by analysing the lifetimes of the barcodes of data arising from a slicing of the proteins. In addition, some statistical inference including hypothesis testing can be carried out on barcodes, but again this is very much in the minority of studies being carried out at the moment.

In most applications, we utilise persistent homology in conjunction with machine learning methods. To do so, in many cases we require to be working in a Hilbert space, or at least to have a space in which there exists an operation which mimics that of an inner product. We can do so in two main ways: we can map the space of barcodes into a Hilbert space via a *vectorisation method* or we can endow the space of barcodes with a *kernel*.

By vectorisation method, we mean the process in which barcodes are mapped to vectors in a Hilbert space. In doing so we need to establish some further stability results to ensure that similar barcodes are mapped to similar vectors. These notions are easily formalised with the use of metrics on each space. The conundrum of vectorisation methods is to devise maps which preserve the essential characteristics of the barcode. This will be discussed in detail in the next chapter.

On the other hand we can endow the space of barcodes with a kernel, which is a generalisation of a scalar product. In essence, a kernel on a space $\mathbb{X}$ is a map $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ such that there exists a Hilbert space $\mathcal{H}_k$ and a feature map $\phi : \mathbb{X} \to \mathcal{H}_k$ with the property that $k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$, where $\langle \cdot, \cdot \rangle$ is the inner product of the Hilbert space $\mathcal{H}_k$ [2]. These kernels allow us to exploit the *kernel trick* in statistical learning, that is, to implement machine learning methods like SVM or PCA without needing the space of barcodes to be an inner product space.

All of these matters are discussed in the next two chapters, where we will explore specific vectorisations and kernels implemented in the state-of-the-art research conducted in the last 10 years.

3 Vectorisation Methods

Barcodes are relatively simple combinatorial objects, and as such they are remarkably easy to vectorise. In the last decade, many vectorisation methods have been proposed, with varying levels of complexity and success. This poses the question as to what vectorizations are appropriate for each type of data. There are some attempts of benchmarking in the

[2] Technically, a kernel is simply a symmetric function $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$. However, in this essay we will work with positive definite kernels, that is, maps $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ such that $(K(x_i, x_j))_{1 \leq i,j \leq n}$ is positive definite for any $x_i \in \mathbb{X}, n \in \mathbb{N}$. Then, in this context Mercer's theorem provides an equivalence with the definition used in the essay.

recent literature (see [PXL18], [Ali+22] or [BPP21]), but these tend to be unsatisfactory. Current studies tends to use the same 5-6 databases, often recycling the barcodes produced by other researchers. Whilst this is not a problem per se, it limits the comparative power of the studies and does not allow for generalisations. The aim of this chapter is explore a variety of vectorisations methods and give theoretical justifications as to why they encapsulate the topological information encoded in the barcodes.

In this section, we will consider a general barcode $\mathcal{B}$ consisting of intervals $[p, q]$, and most of the time we will need some finiteness assumptions to proceed. These include having a finite number of intervals and substituting intervals $[p, \infty)$ by $[p, M]$ for a sufficiently large M.

3.1 Statistical Vectorisations (Persistence Statistics)

Statistical vectorisations is a class of vectorisation in which statistical properties of the barcode are summarised. There exists a considerable amount of variants but we will only present one. Some variants and the definition of the vectorisation below are presented in [Ali+22], but the entropy part of the definition was originally defined in [Chi+15].

Definition 3.1. The *persistence statistics* of a barcode $\mathcal{B}$ is a vector with entries:

1. The mean, standard deviation, median, interquantile range, full range, the 10^{th}, 25^{th}, 75^{th} and 90^{th} percentiles of the births p, the deaths q, the midpoints $\frac{p+q}{2}$ and the lifetime $q - p$ of all the intervals in the barcode, counted with multiplicity;

2. the total number of intervals counted with multiplicity, and

3. the *entropy* of $\mathcal{B}$, defined as

$$S(\mathcal{B}) = -\sum_{[p,q]\in\mathcal{B}} \mu_{p,q} \cdot \left(\frac{q-p}{L}\right) \cdot \log\left(\frac{q-p}{L}\right),$$

   where $\mu_{p,q}$ is the multiplicity of the interval $[p, q]$ and

$$L = \sum_{[p,q]\in\mathcal{B}} \mu_{p,q} \cdot (q - p)$$

Despite the simplicity of the vectorisation, the persistence statistics manages to encapsulate the vital information of the barcode. This is patent in the fact that this vectorisation method performs consistently extremely well with a variety of data [Ali+22]. The entropy feature, motivated by the well-known Shannon entropy in information theory, was developed in [Chi+15] as a way of simplifying barcodes. Intuitively, this feature measures how different the lengths of the intervals are. For example, a barcode with uniform lengths will have a small entropy. In addition, notice that short-lived features have a small impact on the entropy of the barcode. This is useful since short-lived intervals are associated to noise produced by the filtration or uncertainty in the data, so in most cases it does not carry topologically useful information. Finally, the weighting L serves as a scaling parameter which allows the entropy to be a comparable quantity between barcodes. Otherwise, filtrations with a longer lifetime would lead to more negative entropies, independently of the ratio between the lengths of the intervals and the total length of the filtration.

However, on the whole, it seems that the theoretical reasoning given by the proponents of such vectorisation methods is rather unsatisfactory. Intuitively, statistical summaries should provide a good descriptor of the barcode, however the mapping of the barcodes into the Euclidean space is not an embedding and it is not stable. In other words, barcodes close with respect to the canonical metrics may not produce vectors which are close under the Euclidean metric. Despite this, it is worth restating that, empirically, persistence statistics do seem to provide reliable vectorisations: in fact they are among those with a highest accuracy in the three comparative studies mentioned in the introduction.

3.2 Algebraic Vectorisations (Tropical Coordinates)

In this essay we follow the terminology of [Ali+22], so by algebraic vectorisations we mean those which generate polynomials with the endpoints of the intervals in a barcode as variables. Again, many algebraic vectorisations have been proposed but we will only focus on the following, which was put forward in [Kal18].

Definition 3.2. For a given barcode $\mathcal{B}$ consisting of n intervals $[p_i, q_i]$ with multiplicity, extract a set of variables $\{x_1, y_1, \ldots, x_n, y_n\}$ where $x_i = p_i$ and $y_i = q_i - p_i$. Then, a *tropical coordinate function* F for $\mathcal{B}$ is a function with variables $\{x_1, y_1, \ldots, x_n, y_n\}$ which is both tropical and symmetric in the following sense:

1. Tropical: the function F can be expressed using only the operations $\max, \min, +, -$ on the variables $\{x_1, y_1, \ldots, x_n, y_n\}$.
2. Symmetric: any permutation of $\{1, \ldots, n\}$ when applied to both $\{x_i\}$ and $\{y_i\}$ leaves F unchanged.

Examples of such tropical coordinates include:

$$F_1 = \max_i y_i \qquad F_2 = \max_{i<j}(y_i + y_j)$$

$$F_3 = \max_{i<j<k}(y_i + y_j + y_k) \qquad F_4 = \max_{i<j<k<l}(y_i + y_j + y_k + y_l)$$

$$F_5 = \sum_i y_i \qquad F_6 = \sum_i \min(ry_i, x_i),$$

for a given parameter $r \in \mathbb{R}$. We can also have more complicated expressions like

$$F_7 = \sum_j \left[\max_i(\min(ry_i, x_i) + y_i) - (\min(ry_i, x_i) + y_i)\right].$$

These seven tropical functions were used in [Kal18] to classify the MNIST dataset with parameter $r = 28$.

The construction of such polynomial vectorisations first appeared in [ACC16], as the authors realised that a barcode can be specified by a vector $\{x_1, y_1, \ldots, x_n, y_n\}$. However, this representation is not unique. Indeed we may permute the pairs $\{x_i, y_i\}$ and additions of pairs of the form $\{x, 0\}$ do not change the underlying barcode. A. Adcock et al. exploited these properties and results about multi-symmetric functions to vectorise these barcodes into polynomials. They then studied the corresponding ring of functions and showed that the family of functions can distinguish barcodes. However, once again the construction was not stable: small fluctuations in the barcodes could lead to big disparities in the polynomials. S.Kališnik then used the ideas presented in the above paper to produce these tropical

coordinates. In [Kal18] they prove that the semiring of rational tropical polynomials can separate points in the barcode space and is stable with respect to both the Bottleneck and Wasserstein metrics. Therefore tropical coordinates seem to provide a theoretically sound choice of vectorisation. The problem arises with the fact that it is hard to characterise all rational tropical functions, and even when suitably chosen subsemirings are found which can still separate the space of barcodes, these are countably infinite sets. Therefore, for computational purposes we must choose a subset of these and hope that they serve for classification purposes, but once again the theoretical foundations for such a choice are scarce. We must turn to empirical techniques such as cross-validation and grid searches to obtain parameters like r above and to decide what coordinates to choose. When dealing with barcodes with finitely many intervals, it is not hard to produce finitely many functions which separate them, but even then the vectors produced might be very high dimensional. In [Kal18], the authors hope to automate the feature-selection process using machine learning methods such as Lasso. A complementation with theoretical results which ensure that the vectorisation encodes sufficient information about the barcode would be a correct approach to yield a robust vectorisation method.

3.3 Curve Vectorisations (Persistence Landscape)

Curve vectorisations consists of those vectorisations in which we encode the barcode into a curve; that is, a piecewise continuous map from $\mathbb{R}$ to a convenient vector space. Vectors can then be extracted from such curves by sampling the curve at finite subsets of $\mathbb{R}$. The simplest curve vectorisation is the following:

Definition 3.3. The *Betti curve* of a barcode $\mathcal{B}$ is a function $\beta(\mathcal{B}) : \mathbb{R} \to \mathbb{R}$ given by

$$\beta(\mathcal{B})(t) = \sum_{[p,q]} \mu_{p,q} \cdot \mathbb{1}_{\{p \leq t \leq q\}}(t),$$

where $\mu_{p,q}$ is the multiplicity of the interval $[p, q]$.

Therefore, this function counts the number of intervals which contain t. By the Barcode Theorem, the Betti curve is encoding the homology of the t^{th} complex in the filtration. Nevertheless, once again we obtain a vectorisation which lacks stability: for a given barcode we can add lots of very small intervals which will be inconsequential with respect of the bottleneck distance but which will produce a very different Betti curve. In order to rectify this, instead of just seeing whether $t \in [p, q] \in \mathcal{B}$, we can check whether a larger subinterval $[t - s, t + s]$ is contained in $[p, q]$. This minimises perturbations of topological noise as described above, and yields one of the oldest and most studied vectorisations, which we introduce now.

Definition 3.4. The *persistence landscape* of a barcode $\mathcal{B}$ is a sequence of curves $\Lambda = \{\Lambda_i^{\mathcal{B}} : \mathbb{R} \to [-\infty, \infty] | i \in \mathbb{N}\}$, given by

$$\Lambda_i^{\mathcal{B}}(t) = \sup \left\{ s \geq 0 \middle| \left(\sum_{[p,q] \in \mathcal{B}} \mathbb{1}_{[t-s,t+s] \subset [p,q]} \cdot \mu_{p,q} \right) \geq i \right\},$$

where by convention the supremum over the empty set is 0.

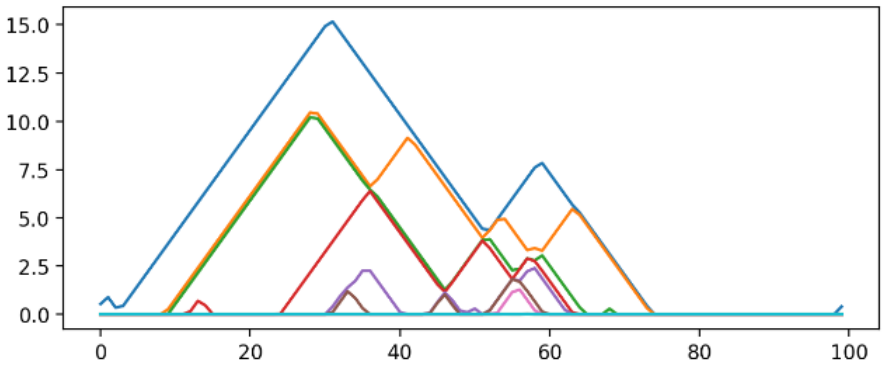

Figure 5: Persistence landscape from the 0-dimensional barcode of a sample image from the Shrec14 database. Made using the app created in [Ali+22]. We limit the number of functions shown to 10.

We can view the persistence landscape as a function $\Lambda : \mathbb{R} \times \mathbb{N} \to [-\infty, \infty]$. Endowing $\mathbb{R} \times \mathbb{N}$ with the product measure of the Lebesgue measure and the counting measure, we can define an L_p-norm on the space of measurable functions $f : \mathbb{R} \times \mathbb{N} \to [-\infty, \infty]$ with respect to this measure:

$$\|\Lambda\|_p^p = \sum_{k=1}^{\infty} \|\Lambda_k\|_p^p = \sum_{k=1}^{\infty} \left(\int |\Lambda_k|^p d\mu \right).$$

We will call such a space of functions $\mathcal{L}^p(\mathcal{S})$, which is a separable Banach space and in the case of $p = 2$ a Hilbert space. For persistence landscapes Λ coming from barcodes with a finite number of intervals we obtain $\Lambda \in \mathcal{L}^p(\mathcal{S})$. We can assume this to be the case in general as all the data sets we look at will produce these finite barcodes. This is great news because now we can perform our classical statistical tests with persistence landscapes: means and other statistical quantities are well defined in this space unlike with the space of barcodes. Much of the machinery developed in probability and statistics can be applied to persistence landscapes, for instance there exists a Strong Law of Large Numbers and a Central Limit Theorem for persistence landscapes. This allows us to apply standard statistical inference procedures, create confidence intervals and so on. Read [Bub15] for a detailed treatment of the above.

Another useful property of the persistence landscape is its stability, which can be summarised in the following theorem from [Bub15].

Theorem 3.5. *Suppose that we have $f, g : \mathbb{X} \to \mathbb{R}$ producing two persistence diagrams $Dgm_k(f)$ and $Dgm_k(g)$ for some $k \in \mathbb{N}$. These in turn produce two persistence landscapes Λ^f and Λ^g. Then,*

$$\|\Lambda^f - \Lambda^g\|_\infty \leq \|f - g\|_\infty$$
$$\|\Lambda^f - \Lambda^g\|_\infty \leq W_\infty(Dgm_k(f), Dgm_k(g)).$$

Notice that this holds for all functions, not only tame functions. Similar results for stability using Lipschitz functions and the Wasserstein distance exist. Again, refer to [Bub15] for a more extensive treatment.

4 Kernel Methods

In machine learning, we sometimes require the feature vectors to be in an inner product space because the algorithms require the use of an inner product. Sometimes, it is enough to have a similarity function between the points in the training set so that we do not require an embedding of the set of feature vectors into an inner product space. These methods are called *kernel methods* and the similarity functions are the *kernels* we introduced at the end of chapter 2. We recall that a kernel on a space $\mathbb{X}$, which we should think of as the space of barcodes, is a map $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ such that there exists a Hilbert space $\mathcal{H}_k$ and a feature map $\phi : \mathbb{X} \to \mathcal{H}_k$ with the property that $k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$, where $\langle \cdot, \cdot \rangle$ is the inner product of the Hilbert space $\mathcal{H}_k$. This is very powerful because it enables us to work in a high-dimensional Hilbert space without requiring to compute the coordinates under ϕ of our barcodes.

In addition, such a kernel induces a metric on the space of barcodes via

$$d_k(x, y) = [k(x, x) + k(y, y) - 2k(x, y)]^{1/2},$$

which is precisely the distance $\|\phi(x) - \phi(y)\|_{\mathcal{H}_k}$ in the feature space. Stable kernels with respect to the canonical metrics, that is kernels such that $d_k(x, y) \leq Cd(x, y)$ where d is the Wasserstein or Bottleneck metric, are desirable for classifications tasks. This is because algorithms such as SVM separate data points via hyperplanes which are robust with respect to perturbations in the canonical metrics. Hence, stable metrics ensure that the hyperplane continues to be robust with respect to the new metric.

The question becomes how to produce such stable kernels from the space of barcodes. A long list of such kernels, with varying degrees of success, has been produced in the last decade, a subset of which are summarised in [PXL18]. In general, given a function $f : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ and a parameter $\sigma > 0$ we can construct a kernel k_σ via

$$k_\sigma(x, y) = \exp\left(-\frac{f(x, y)}{2\sigma^2}\right).$$

However, as pointed out in [BCR84], this will only yield a valid positive definite kernel for any $\sigma > 0$ if and only if f is negative semi-definite, that is, if for all $n \in \mathbb{N}, a_1, \ldots, a_n \in \mathbb{R}$ with $\sum a_i = 0$ we have $\sum_{i,j} a_i a_j f(x_i, x_j) \leq 0$. Unfortunately, in [Rei+14] it is shown experimentally that this condition is not satisfied by any of the canonical metrics. There are other ways of creating kernels from metrics, but it remains an open question whether constructions arising from the canonical metrics lead to valid kernels that can be employed in machine learning. However, we now present a variation of the 1-Wasserstein metric which does lead to a valid kernel.

Definition 4.1. Given $\theta \in \mathbb{S}^1 \subset \mathbb{R}^2$, let $L(\theta) = \{r\theta | r \in \mathbb{R}\}$. Also, let $\pi_\theta : \mathbb{R}^2 \to L(\theta)$ be the orthogonal projection onto the line $L(\theta)$ and π_Δ the projection onto the diagonal. Moreover, let Dgm_1, Dgm_2 be two persistence diagrams, and associated to them define

$$\mu_i^\theta = \sum_{x \in \mathrm{Dgm}_i} \delta_{\pi_\theta(x)}, \ \mu_{i,\Delta}^\theta = \sum_{x \in \mathrm{Dgm}_i} \delta_{\pi_\theta(x) \circ \pi_\Delta(x)} \text{ for } i = 1, 2.$$

Then, define the *Sliced Wasserstein distance* as

$$d_{SW}(\mathrm{Dgm}_1, \mathrm{Dgm}_2) = \frac{1}{2\pi} \int_{\mathbb{S}^1} \mathcal{W}(\mu_1^\theta + \mu_{2,\Delta}^\theta, \mu_2^\theta + \mu_{1,\Delta}^\theta) d\theta,$$

where, in general, given two measures μ, ν on $\mathbb{R}$ with $\mu(\mathbb{R}) = \nu(\mathbb{R})$ we define $\Pi(\mu, \nu)$ to be the set of measures in $\mathbb{R}^2$ with marginals μ, ν and set

$$\mathcal{W}(\mu, \nu) = \inf_{P \in \Pi(\mu,\nu)} \int \int_{\mathbb{R}^2} |x - y| P(dx, dy).$$

Lemma 3.2 in [CCO17] then shows that this metric is negative semi-definite when restricted to the space of bounded finite persistence diagrams, so it produces a valid kernel called the *Sliced-Wasserstein kernel* which we denote k^{SW}. They also show that this metric is, not only stable with respect to the 1-Wasserstein distance, but also equivalent to it.

But, what is the intuition behind this metric? Well, if $\mu = \sum_{i=1}^{n} \delta_{x_i}$ and $\nu = \sum_{i=1}^{n} \delta_{y_i}$ with $x_1 \leq \ldots \leq x_n$ and $y_1 \leq \ldots \leq y_n$ then $\mathcal{W}(\mu, \nu) = \sum_{i=1}^{n} |x_i - y_i|$. Therefore, the sliced-Wasserstein distance is projecting points in the persistence diagrams onto a line $L(\theta)$, where $\mathcal{W}$ is then computed. We integrate over all lines to form an average. Intuitively, instead of minimising the Euclidean distance between points from two diagrams, we compute the distance between points if only allow movements at an angle θ and then form an average. In practice, when we compute the sliced-Wasserstein metric we need to discretise the integral in the usual way, so we uniformly divide $\mathbb{S}^1$ and proceed as explained above.

In order to explore some of the properties of such a kernel, let us introduce a second one to investigate the differences. Recall that the persistence landscape for $p = 2$ embeds the space of persistence diagrams in a Hilbert space $\mathcal{L}^2(\mathcal{S})$. We will call this embedding the feature map $\phi^L : \mathbb{X} \to \mathcal{L}^2(\mathcal{S})$ which produces a kernel k^L which we call the *persistence landscape kernel*. Bubenik shows in [Bub15] that the metric induced by such a feature map is stable, indeed it is shown that for two persistence diagrams Dgm_1, Dgm_2 we have

$$\begin{aligned} d_L(\mathrm{Dgm}_1, \mathrm{Dgm}_2) &:= \|\phi^L(\mathrm{Dgm}_1) - \phi^L(\mathrm{Dgm}_2)\| \\ &\leq \inf_{\gamma} \left(\sum_{u \in \mathrm{Dgm}_1} p(u) \|u - \gamma(u)\|_\infty^2 + \frac{2}{3} \|u - \gamma(u)\|_\infty^3 \right)^{\frac{1}{2}}, \end{aligned}$$

where $p(u) = j - i$ is lifetime of an element $u = (i, j)$ and γ ranges over all bijection from Dgm_1 to Dgm_2.

Notice that the persistence landscape metric weights considerably more points with a high persistence, in consonance with the idea that high persistence points do not correspond to topological noise, but rather actual topological features of the data. Hence it makes sense to have a high distance between diagrams with a different number of points with a high persistence. However, Bauer et al. propose a thought experiment in [Rei+14] which shows that this might be misleading. Consider two diagrams $\mathrm{Dgm}_1 = \{-\lambda, \lambda\}$ and $\mathrm{Dgm}_2 = \{-\lambda + 1, \lambda + 1\}$ for some $\lambda \in \mathbb{R}_{\geq 0}$. Then the Euclidean distance between the points is 1, but the persistence of the points is 2λ. Therefore, whilst $d_{SW}(\mathrm{Dgm}_1, \mathrm{Dgm}_2)$ and $W_q(\mathrm{Dgm}_1, \mathrm{Dgm}_2)$ are bounded as $\lambda \to \infty$, $d_L(\mathrm{Dgm}_1, \mathrm{Dgm}_2)$ grows as $\lambda \to \infty$ in the order of $\sqrt{\lambda}$. We will discuss these matters further when the kernels are compared against data in chapter 5.

5 Comparison of Vectorisation and Kernel Methods

The aim of this section is to gain an insight into how some of the vectorisation and kernel methods presented in the essay compare on some simple classification tasks. We first explain what data sets and machine learning algorithms will be used for this purpose. Then

we exhibit and analyse original results based on the classification of these data sets on code written on python. Whilst doing the analysis, we have tried to formulate a series of hypothesis based on the theory and then we have tested it on our data sets. In light of the results of the experiments conducted, we try to explain how this fits into our understanding of the mathematics behind and compare our hypothesis with the real performance.

5.1 Data sets and Machine Learning Algorithms

The first data set we will analyse is what we will call *synthetic data*. This is point cloud data arising from five different shapes, namely a circle of radius 1, a 2-sphere of radius 1, a cube of side length 1, a 2 dimensional torus of inner radius 1 and outer radius 2, and a "cylinder without lids" with a base circle of radius 1 and a height of 2 . We have limited the sample of points taken from each shape to 100 and we have included Gaussian noise when necessary. The aim of using such a data set is to have the ability to conduct relatively quick analyses on the methods employed that would be very hard to do on more complex data sets due to the computational time.

On the other hand, we will perform some analysis on the MNIST data set mentioned in subsection 2.1. This is supposed to provide an opportunity to apply the methods on a more complex data set. Little exploratory analysis can be conducted on the data set, however, due to limitations in the computational time needed to perform the classification tasks. Efforts have been made to optimise the algorithms implemented but TDA algorithms tend to be computationally expensive, which limits the amount of empirical analysis we can conduct. A word of warning: the accuracy figures presented below may not correspond precisely to those presented in other papers do the inevitable differences in the programming and parameter selection. This means that it is possible for the accuracy of some methods to increase more than others after a more appropriate preprocessing of the data or a more systematic parameter selection, but such comparisons would be beyond the scope of the essay.

Before we move to the ML methods: why have we specifically chosen the above data sets? As was mentioned above, the aim of the synthetic data set was to produce a data set which could be coded quickly and which allowed us to test the vectorisation and kernel methods. It gave us an indication of how the computational times of the different methods compared, but it also allowed us to optimise the algorithms as we were able to run the program a substantial amount of times. However, this still leaves the question as to why we chose those specific shapes. On one hand, we wanted to explore whether shapes which were topologically different (that is, with different homology groups) could be distinguished. This lead us to choosing the circle, sphere, torus and cube. We also included the cylinder without lids, which is homotopy equivalent to the circle, to test whether shapes with identical homology groups but with a different spatial configuration can be discerned by TDA. In those cases where we could choose between homotopy equivalent shapes, such as the ball in three dimensions and the cube, we chose that which was more computationally efficient. We also considered shapes which were easy to visualise to facilitate the interpretation of the results. In addition to the synthetic data, we wanted to test the pipeline on a more complicated data set and we considered the MNIST data set to be an appropriate choice. Firstly, we thought it would be interesting to use image data in order to explore the preprocessing that needs to be undertaken when using TDA. MNIST seems an optimal choice, since it is a widely available data set, the preprocessing is well-understood and it produces images which are still topologically interpretative. That is, when we change the pixel intensity so that it is

location dependant, we are forcing loops which appear in different parts of the image to appear at different times in the filtration. This produces PDs corresponding to a 6 and a 9 which are distinguishable, and we can precisely say why they are distinguishable: the interval in the barcode corresponding to the loop will be shifted as it will enter the filtration at different points in time. This interpretability is much more difficult to obtain with other more complicated image data sets, as it is not always clear what preprocessing needs to be done to ensure that TDA becomes an efficient tool. Finally, whilst the MNIST data set is undoubtedly more complex than the synthetic data set, it is still manageable in the sense that computations can be carried out on a normal laptop. Hence, it seems that it is an appropriate balance between complexity and the computational time needed to conduct the experiments.

Let us continue now by looking at the two machine learning classification algorithms which have been used: the k-nearest neighbour (kNN) classifier and the support vector machine (SVM). The following account is largely based on the classical reference [HTF08], where we also direct the reader who wishes to find a more detailed exposition of kNN and SVM. In general, classification algorithms work as follows: suppose we are given a training set $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, where $y_i \in \{1, \ldots, C\}$ is the class label of the data point x_i. Now suppose we are given a new data point $x = x_{n+1}$. Then a classifier tries to predict the class of this new data point, based on the information given by the training set. In the synthetic data set, we have $C = 5$ and each class corresponds to a different shape. Moreover, x_i is the PD arising from a random sample taken from the shape corresponding to class y_i. When we are using a vectorisation method we are first transforming the PD into a vector in $\mathbb{R}^p$, and then we input this into the classification algorithm.

Let us first look at the kNN classifier. In essence, we fix a parameter $k \in \mathbb{N}$ (this is where the k in kNN comes from!) and we assign to x the "most popular" class between its k closest points. More precisely, order the points in the training set $x_{i_1}, \ldots, x_{i_n}$ so that

$$\|x_{i_1} - x\| \leq \|x_{i_2} - x\| \leq \ldots \leq \|x_{i_n} - x\|.$$

If more than one point is at the same distance to x as another, pick an ordering of those points at random. Then, we define $\mathcal{N}_k(x) = \{x_{i_1}, \ldots, x_{i_k}\}$, which we call the k-nearest neighbours of x for obvious reasons. Then, the kNN algorithm assigns to x the class

$$\hat{C}^{k\mathrm{NN}}(x) = \underset{c \in \{1,\ldots,C\}}{\arg\max} \sum_{x_i \in \mathcal{N}_k(x)} \mathbb{1}_{\{y_i = c\}}.$$

If the sum corresponding to multiple classes are equal, we pick one of the classes at random. Notice that parameter selection is needed for k and that a small k will yield a low bias but a high variance whilst a larger k will achieve the opposite. This leads to the famous *bias-variance trade-off* and an appropriate parameter selection should ensure that our model does not overfit or underfit the data.

In addition to this, we have chosen a Support Vector Machine algorithm (SVM) for the kernel methods. Let us start by considering a classification task on 2 classes, in which case we write $y_i \in \{-1, 1\}$ and for simplicity let us assume that $x_i \in \mathbb{R}^p$. The data is called *separable* if there exists a hyperplane $H = \{x \in \mathbb{R}^p : x^T\beta + \beta_0 = 0\}$ such that points on each side belong to a different class. This is equivalent to asking for $y_i(x_i^T\beta + \beta_0) > 0$ for all i. If this holds, it seems reasonable to find a hyperplane which maximise the distance between it and its closest point, which we call the *support vectors*. This precisely corresponds to the

optimisation problem

$$\sup_{\beta_0\in\mathbb{R},\beta\in\mathbb{R}^p} M \text{ subject to } \frac{y_i(x_i^T\beta+\beta_0)}{\|\beta\|}\geq M, i=1,\dots,n,$$

since $(x^T\beta+\beta_0)/\|\beta\|$ corresponds to the signed distance between the point x and the hyperplane H. Notice too that any solution $(\hat{\beta},\hat{\beta_0})$ of the above optimisation problem remains a solution after scalar multiplication, so we can always obtain $\|\beta\|=1/M$. Therefore, the above optimisation problem is equivalent to

$$\min_{\beta_0\in\mathbb{R},\beta\in\mathbb{R}^p} \|\beta\|^2 \text{ subject to } y_i(x_i^T\beta+\beta_0)\geq 1, i=1,\dots,n.$$

This is a quadratic convex optimisation problem with linear inequality constraints so we can efficiently find the unique solution $(\hat{\beta},\hat{\beta_0})$ if the data is separable. This produces a classifier

$$\hat{C}^{\text{hSVM}}(x)=\text{sign}(x^T\hat{\beta}+\hat{\beta_0})$$

called the *hard-margin support vector classifier*. Of course, in applications it is rare to find separable data sets so we have to adapt the above. To do this, we replace the constraint $y_i(x_i^T\beta+\beta_0)\geq 1$ by the penalty $\max\{1-y_i(x_i^T\beta+\beta_0),0\}=L(y_i,x_i^T\beta+\beta_0)$, where $L(y,y')=\max\{0,1-yy'\}$ is called the *hinge loss*. Hence, the optimisation problem becomes

$$(\hat{\beta_0},\hat{\beta})\in\underset{\beta_0\in\mathbb{R},\beta\in\mathbb{R}^p}{\arg\min}\left(\|\beta\|^2+C\sum_{i=1}^n L(y_i,x_i^T\beta+\beta_0)\right),$$

where $C\geq 0$ is a cost parameter. This produces another classifier

$$\hat{C}^{\text{sSVM}}(x)=\text{sign}(x^T\hat{\beta}+\hat{\beta_0})$$

called the *soft-margin support vector classifier*. Again, the parameter C is related to the bias-variance trade-off of the classifier, as a large C will force the classifier to minimise misclassifications as the support vectors will gain weight in the optimisation. This leads to a lower bias but higher variance as small variations in the points closest to the hyperplane will make it fluctuate significantly. The opposite is true for small values of C.

Notice that the classifier above only depends on inner products, as it only contains terms like $x_i^T\beta$. As was mentioned in chapter 4, such a classifier can be used on feature spaces endowed with a kernel, even if these are not Hilbert spaces. Indeed, suppose that $(x_1,\dots,x_n)$ are the features of a training set which lie on the space of persistence diagrams $\mathbb{X}$, and again $y_i\in\{-1,1\}$. Let k be a kernel on this space, as described in chapter 4, and let $M=(k(x_i,x_j))_{1\leq i,j\leq n}$ be the corresponding Gram matrix. Then the Representer Theorem gives the *support vector machine classifier*

$$\hat{C}^{\text{SVM}}(x)=\text{sign}(\sum_{i=1}^n\hat{\beta}_i k(x_i,x)+\hat{\beta}_0),$$

where

$$(\hat{\beta_0},\hat{\beta})\underset{\beta_0\in\mathbb{R},\beta\in\mathbb{R}^n}{\arg\min}\in\left\{C\sum_{i=1}^n L(y_i,(M\beta)_i+\beta_0)+\beta^T M\beta\right\}.$$

Again, $C \geq 0$ is the cost parameter. If $\phi : \mathbb{X} \to \mathcal{H}$ is the feature map corresponding to the kernel function, then the representer theorem allows a way of conducting SVM on the feature space without actually having to compute ϕ. Kernels are also used when we require non-linear decision boundaries, even if the original feature space is already embedded in a Hilbert space. Finally, this is can be extended to multi-class classification by considering a one-versus-one SVM classification, and proceeding with a maxs-wins voting strategy.

Let us now briefly explain what are the concise computations the algorithm has conducted. First of all, in our program we have used the RIPSER software, [TSB18], to produce the PDs of both data sets, although with the MNIST data set we have conducted a similar preprocessing to [Kal18], as discussed in subsection 2.1. In practice, for point cloud data RIPSER produces a VR-complex and uses cohomology for persistence calculation. Cohomology calculations turn out to be more efficient and via some duality arguments it can produce the persistent homology we require for the PD. The specifics will not be pursued, but RIPSER is the-state-of-the-art software for the computation of PDs and the rest of the software available is substantially less efficient [Ott+17]. On the other hand, the MNIST data is transformed so that the pixel intensity is location-dependant, and then it is inputted into RIPSER. This produces a lower-star filtration and then computes the PDs with the same techniques as for point cloud data.

Once the PDs are constructed, we test the vectorisation and kernel methods on them separately. Whilst the vectorisation methods can be used in conjunction with both kNN and SVM, the kernel methods are limited to SVM. The precise configurations of the experiments carried out will be explained in the next subsection. Let us, however, briefly discuss how we proceeded with the parameter selection. In most classifiers, as is the case of kNN and SVM classifiers, there are some *regularisation hyperparameters* which control the learning of the classifier. These are related to the above concept of bias-variance trade-off and an appropriate selection is necessary to ensure that our model does not overfit the data nor is it too rigid to adapt well to new data points incorporated into the model. Methods such as grid searches, random searches or Bayesian optimisation searches are customary for parameter tuning and their suitability varies from problem to problem. In our case, since we are only tuning a single parameter for each classifier, a computationally expensive tuning such as a grid search would not be inappropriate. Indeed, this is a common approach in the current literature [PXL18], [BPP21]. In our case, this extra layer of computations further slowed down the whole pipeline so we conducted some exploratory analysis on each experiment to find a suitable parameter for k, C in kNN and SVM respectively. Of course, this does not ensure that the chosen parameters produce classifiers with the highest accuracy among all possible classifiers. There exists a possibility that a more appropriate parameter tuning for each specific vectorisation and kernel used will yield results in which some of the methods are comparatively better than what is presented in the analysis in section 5.2. A more systematic and thorough analysis would have to be conducted to discard this possibility, but we deem it unlikely since there is an appreciable disparity between the PH methods used. This will be discussed further in the next section.

Finally, we would like to mention that to ensure that the results are reliable we have conducted cross-validation on all of the experiments. This has typically involved a 5-fold cross validation. This involves partitioning the training set S into five approximately equal subsets $\{S_1, S_2, S_3, S_4, S_5\}$, training a classifier on $S \backslash S_i$ and testing it on S_i. The mean accuracy of the five classifiers on their testing sets is then an approximation of the true accuracy of the model.

5.2 Analysis

Let us compare first the vectorisation methods. When we produce a vector, there are some choices to be made. Each vectorisation method gets as an input a persistence diagram of a given homological degree and outputs a vector. These are then concatenated to give the resulting vector which we input into the algorithms. Hence, a natural question to ask is how many of such vectors such we concatenate. Practitioners tend to use persistence diagrams of degrees 0 and 1 in most cases, but is this reasonable? Consider the homology groups with coefficients in $\mathbb{Z}$ of the shapes used in the synthetic data set:

$$H_k(\mathbb{S}^2;\mathbb{Z}) \cong \begin{cases} \mathbb{Z} & \text{if } k = 0, 2 \\ 0 & \text{otherwise} \end{cases}, \; H_k(\text{Cylinder};\mathbb{Z}) = H_k(\mathbb{S}^1;\mathbb{Z}) \cong \begin{cases} \mathbb{Z} & \text{if } k = 0, 1 \\ 0 & \text{otherwise} \end{cases},$$

$$H_k(\mathbb{T}^2;\mathbb{Z}) \cong \begin{cases} \mathbb{Z} & \text{if } k = 0, 2 \\ \mathbb{Z}^2 & \text{if } k = 1 \\ 0 & \text{otherwise} \end{cases}, \; H_k(\text{Cube};\mathbb{Z}) \cong \begin{cases} \mathbb{Z} & \text{if } k = 0 \\ 0 & \text{otherwise} \end{cases}.$$

In our case it seams reasonable to bound the degree of the persistence diagrams used by 2 since higher order homology groups do not reveal more information that allows us to discern the shapes being studied. Of course, we are not actually computing the homology of these shapes, but rather the persistent homology of a filtered complex arising from a subsets of points taken from these shapes. In addition, noise has been added to the points selected. Therefore, it is plausible to believe that higher order persistent homologies might encode some useful information. We will try to show throughout this section that, at least in the case of the synthetic data set, this is not the case. It is true that noise might incorporate features that could appear in high-dimensional PDs, but these will be short-lived and vectorisation and kernel methods tend to give little weight to short-lived features precisely because of this. In most cases, although not always, short-lived features in PDs are the product of noise and most PH algorithms tend to focus on the long-lived features which correspond to true topological features of the data. Therefore, information encoded in high-dimensional PDs arising from noise would be in any case neglected by most PH-based algorithms. Furthermore, it a well known result from algebraic topology that, under some mild assumptions, the k$^{\text{th}}$-homology of a complex $\mathbb{X}$ only depends on the k+1 skeleton, $\mathbb{X}^{k+1}$, and a complex equal to its k skeleton has trivial homology groups in degrees strictly higher than k. This means that in practical applications where we are considering networks, proteins or shape analysis, the low-dimensional nature of the underlying spaces supports the choice of considering low-dimensional PDs only. Of course, this is not always the case as filtrations may encode information about the underlying shapes and in some applications it is useful to study higher dimensional PDs. In any case, for this data set it is reasonable to bound the degree by 2.

The question now becomes what degrees to use inside the subset $\{0, 1, 2\}$. On one hand, the sphere and the cube can only be distinguished by the second homology groups. However, adding an extra component to the vector increases its dimension, and in some cases dramatically. Indeed, whilst each PD gives a 7-dimensional and a 38-dimensional vector for the tropical coordinates and statistical persistence vectorisations, for the persistence landscape it is customary to produce vectors of dimension in the order of 100. As was mentioned in the subsection 3.3, the persistence landscape produces a sequence of curves which have to be discretised individually and concatenated into a vector. Practitioners tend to optimise the number of curves chosen and how finely to make the grid used to

discretise the curves. In any case, a possible reason for the restriction of vectorisations to PDs of dimensions 0 and 1 (or even just 0) is the well known *curse of dimensionality*. If the vectorisation produces vectors of very high dimensions, then the performance of such methods might drop dramatically in comparison to those with lower dimensional vectors. Another simpler reason for choosing to vectorise only PDs of dimensions 0 and 1 is the computational time of including more dimensions, which could produce better results but at the high expense of having long running times.

To test these hypotheses, we conducted an experiment on the synthetic data set comparing the classification accuracy of 9 classifiers arising from a kNN algorithm in conjunction with one of the three vectorisation methods, concatenating PDs of degrees 0, 0 and 1 or 0,1 and 2. We are comparing how the accuracy varies as the training size N varies from 50 to 550, and all data points have a noise level of 0.2. This means that we have added a random Gaussian vector with a covariance matrix with 0.2 as diagonal elements. Moreover, after some exploratory analysis we have fixed the hyperparameter of the kNN classifier to $k = 3$, which is relatively common. Figures 6 and 7 present the results from the experiment.

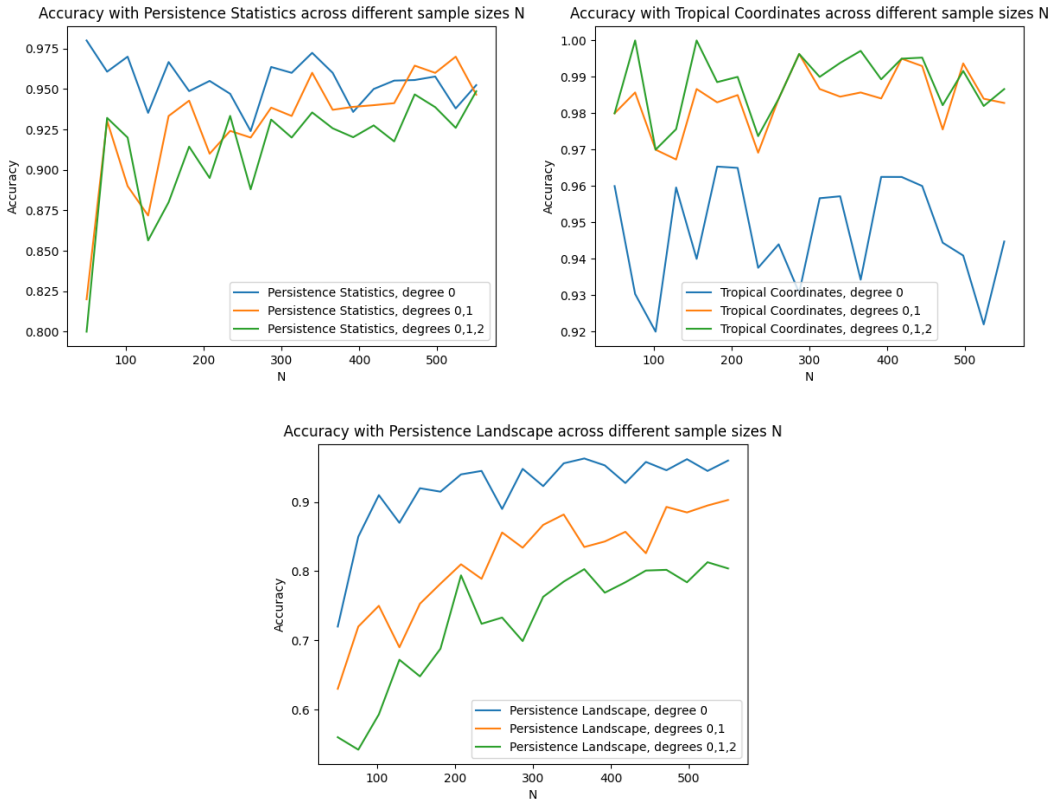

Figure 6: Each vectorisation has a line representing the vector created by concatenating the vectors arising from the vectorisation of PDs of increasing degree. Varying sample size and fixed noise level of 0.2.

From the diagrams we observe that the dimensionality of the vectors plays an impor-

Method	Tropical Coordinates			Persistence Statistics			Persistence Landscape		
Degrees	0	0,1	0,1,2	0	0,1	0,1,2	0	0,1	0,1,2
Mean Accuracy	0.947	0.983	0.988	0.954	0.929	0.913	0.920	0.815	0.728
Standard Deviation	0.014	0.008	0.008	0.013	0.034	0.034	0.055	0.073	0.083

Figure 7: Table including a basic statistical summary of the graphs presented in figure 6.

tant role, which is to be expected as kNN is very sensitive to increases in dimensionality [Bey+98]. Indeed, as we increase the sample size, not all vectorisations respond in the same way. Whilst low dimensional vectorisations like the tropical coordinates have a low standard deviation, indicating that increases in the sample size do not change significantly the accuracy of the algorithm, the complete opposite is true for the persistence landscapes. Indeed, the high standard deviation in the accuracy across all three curves in the persistence landscapes suggests that the high dimensionality of the vectorisation is what hinders its accuracy in this classification task. Increasing the sample size improves the accuracy of the persistence landscape vectors in the three cases, but they do not seem to convergence even at high sample sizes. All the discriminative power gained by the additional topological information of concatenating vectors corresponding to higher degree PDs is eclipsed by the extra dimensionality.

On the other hand, it is very clear from the tropical coordinates graph that including information about the degree 1 PD does improve substantially the accuracy rates. However, the same can not be concluded about the degree 2 PD: whilst including the vector corresponding to the degree 1 PD increase the accuracy by nearly 4%, including that corresponding to the degree 2 PD only increases the accuracy by 0.5%.

Finally, the persistence statistics graph lies in between. As the sample size increases we observe a convergence in the accuracy rates, but for small samples sizes there is a significant difference between them.

One of the main aims of PH is to handle high dimensional data sets by extracting general geometric information and disregarding the noise. However, vectorisations like the persistence landscape, although having a solid theoretical foundation, do not serve as a regularisation method in itself. As such, it might be interesting to perform additional regularisation methods like lasso or PCA as considered in [PXL18]. The problem is that kNN is a non-parametric algorithm and regularisation methods such as lasso requires the algorithm to be based on an optimisation problem. On the other hand, recent studies like [Zha22] show that extra regularisation via PCA together with some improved storage structures can significantly increase the efficiency and accuracy rates of kNN algorithms. Therefore, if the proponents of persistence landscapes want to incorporate this tool in a ML pipeline they will need to use a combination of regularisation methods which reduce the dimensionality of the vectors and is able to adapt to the ML algorithm used at the end of the pipeline.

It is also interesting to observe that even the 0 degree vectorisations perform surprisingly well. It is surprising because the degree 0 should encode information corresponding to the

zero homology of the filtrations produced by the point cloud data, but the shapes being analysed are all connected so they have the same zero homology. However, the 0 degree PD encodes more information than this. Indeed, the persistent homology encodes how the 0^{th}-homology changes as the filtration evolves, so it also measures how far apart the sample points are. This leads to the natural question as to, for example, whether vectorising *only* the degree 2 PD will lead to vectorisations with similar accuracy rates.

Hence, we then conducted the experiment of testing the tropical coordinates solely on the degree 2 PD. The set up is very similar to the previous experiment, all data points have a Gaussian noise of magnitude 0.2 and the size of the training set varies from 10 to 110. From the graph in figure 8, we observe that once the curve stabilises, the accuracy rate is approximately 68% for tropical coordinates arising from degree 2 PDs (and a similar trend is observed for the other vectorisations). There is a high variation in the accuracy rates, even after cross-validation, which is probably due to the inherent nature of the PH-pipeline. High accuracies, such as that observed at around 50, are probably due to the volatility of the model and should be interpreted as natural fluctuations. All the experiments conducted present these highly fluctuating characteristics. The results reinforce the notion that higher dimensional PDs only provide valuable information when used in conjunction with its lower dimensional counterparts. It is also interesting to observe that the accuracy increases and then stabilises. This was not observed in the degree 0 counterpart, which seems logical as increasing the sample sizes should specially help high-dimensional vectors but not as much vectorisations like the tropical coordinates. We believe that a reason for this is that whilst degree 0 PDs tend to have many elements, the same is not true with degree 2 PDs. Indeed in many cases the intervals are short-lived and they are scarce in general. This is possibly because only two of the shapes have non-zero degree 2 homology. Therefore we expect that a lot of the vectors produced by the degree 2 PD will be scarce, and only occasionally it will produce vectors with a higher norm. Hence, we expect to need a substantial amount of sample points to have a significant number of non-scarce vectors which allows for a more accurate classification. Therefore, by itself it does not provide a good method to perform classification tasks, but as it is seen by the results of the tropical coordinates it does provide additional information to the degree 0 and 1 PDs which is useful.

In conclusion, for this specific data set it is reasonable to conclude that vectorising degree 0 and 1 PDs only is optimal when the data set is sufficiently large. In addition to the accuracy rates, the computational times increases dramatically when we incorporate PDs of dimension greater than 1. Indeed, incorporation of more PDs adds complexity to every point of the pipeline and this is possibly a very, if not the most, significant deterrent for using degree 2 PDs. Therefore, for small and basic data sets such as this synthetic data it is recommended to employ those vectorisations with the smallest dimensions. Differences in the topological information encoded in each vectorisation is largely eclipsed by the dimensionality of the features. Of course, this may not apply to more complex data sets.

Another interesting approach would be to increase substantially the noise of the data sets to observe the types of errors the classifier incurs in. Below we have tested the persistence statistics of degrees 0 and 1 across data sets with varying amounts of noise. Now the sample size is fixed to 100 and the noise levels varies from 0 to 1, whereas previously this was fixed to 0.2. Noise has been added to all shapes but the torus. As can be observed, the classifier starts with a perfect classification accuracy, but this starts descending until it stabilises around 80%. This could be due to homogeneous errors across the different classes, but the fact that it stabilises and that this occurs around 80% suggests that it is actually mistaking two classes. Indeed, all the errors attributed on five consecutive runs of

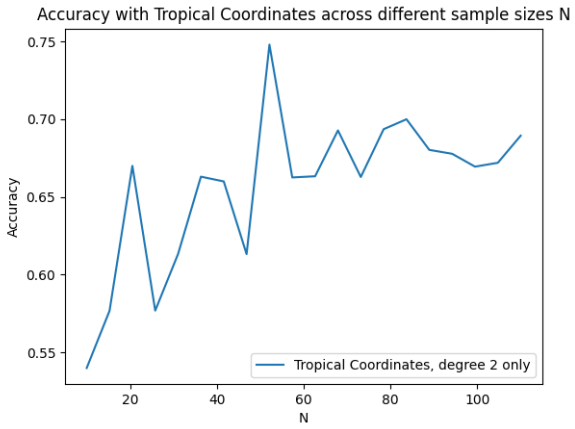

Figure 8: Restriction of tropical coordination to those produced by the PD of degree 2. Varying sample size and fixed noise level of 0.2

the program are due to misclassifications of a torus by a cylinder and vice versa. This should be expected as a cylinder with Gaussian noise added resembles topologically a 1-torus. This is a further reinforcement that the algorithm is not simply considering the 0 degree PD, as mean distances become more homogeneous when we increase the noise but classification of shapes with distinct topological features is still carried out successfully.

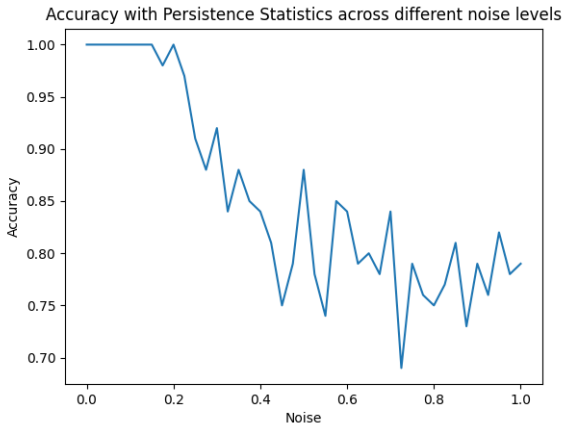

Figure 9: Accuracy of a kNN classifier with persistence statistics of degree 0 and 1 as inputs. Sample size of 100 and varying noise levels.

We now turn to investigating the use of kernels in the synthetic data set. Due to constrains in the computational time we restrict our investigation to accuracy rates for data sets of size 75 with different noise levels varying from 0 to 0.5. Notice that kernels utilise a single PD per data point as it is not possible to superpose PDs of different degrees. We have chosen to use PDs of degree 1.

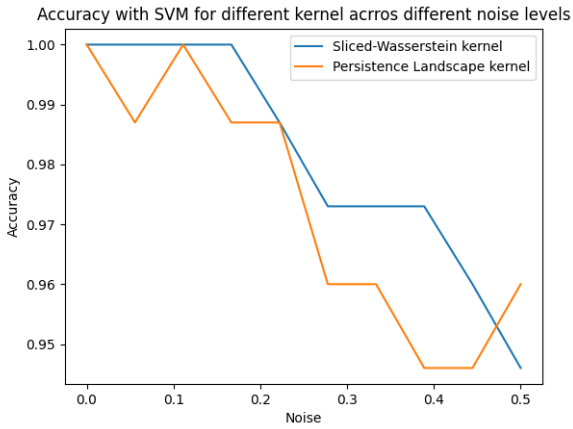

Figure 10: Accuracy for an SVM classifier with different kernels. Sample size of 75 and varying noise levels.

A few things are very surprising about the above results. First of all the accuracy rates are very high for such a relatively small data set of 75 samples: each class contains only 15 samples! Moreover, the accuracy seems to be very high even with high noise levels. This was not observed in the vectorisation methods, and we explained that the underlying reason seemed to be the topological similarity between the cylinder and the torus. However, both of the kernels seem to produce a classifier that is able to distinguish between these objects. Again, this is probably because the persistent homology encodes more topological information than the homology of the underlying space, but it is difficult to know what exact topological information is being used to distinguished the shapes. In addition, recall that these kernels were used on the space of degree 1 PDs, which makes the accuracy rates even more surprising. It seems that the kernel methods produce similar if not better classifiers on same data sets, but this comes at the cost of high computational times. We will not discuss the origins of such disparities between computational times, but notice that in the kNN algorithm we compute the Euclidean distance between pairs of vectors, whilst in the SVM algorithm the inner product between PDs arising from the kernels has a much higher complexity. Of course, the vectorisation methods have some extra complexity arising from the vectorisation but notice that the persistence statistics and the tropical coordinates only have linear complexity. In a simple classification task of the synthetic data, with 100 samples and 5-fold cross-validation, the kNN algorithm with these two vectorisations has a computational time of about 2 minutes, whilst that of the persistence landscape is of 4. Similarly, SVM takes about 5 minutes on the same classification task with either kernel. This means that when we conduct experiments where several dozen of runs are made, the disparity in the computational time is very noticeable. Finally, it is interesting to note that the persistence landscape kernel SVM performs considerably better than its vector counterpart, reinforcing our hypothesis that the persistence landscape is a good encoder of topological information and that the main flaw of the vectorisation is the dimensionality of the vectors produced.

Finally, let us consider the MNIST data set. Whilst before we were considering sample sizes of around 2 orders of magnitudes, we will now consider a sample size of 5,000 for the

vectorisations and subsets of 300 for the kernel methods. Therefore, once again we limit our analysis to a classification procedure due to computational constrains. However, it is reasonable to ask whether smaller data sets could be used. After all, the MNIST data base is well known for being a relatively simple data set compared to the complex structures normally considered in ML. This is true for conventional ML methods, but when we use PH, as we pointed out in section 2.1, we need to produce four different filtrations for each image in order to distinguish topologically similar digits. This is in essence multiplying by 4 the dimensions of our vectors so this increase in dimensionality has to be compensated by the use of a large data set. Moreover, kNN does not perform well with such high dimensional vectors so all tests have been performed on SVM. In the case of the vectorisation methods we have used a Gaussian kernel, which seems to produce the best results.

	Tropical Coordinates	Persistence Statistics	Persistence Landscape Kernel	Sliced Wasserstein Kernel
Accuracy	0.893	0.960	0.854	0.838
Standard Deviation	0.023	0.017	0.032	0.028

Figure 11: Accuracy and standard deviation of classification task with SVM performed on the MNIST data set.

In this data set, the persistence statistics seems to be a superior method to the rest, in consonance with the results of comparative papers like [Ali+22]. It is worth mentioning, however, that it is possible to introduce additional methods to the pipeline (such as extra regularisation) which might sensibly increase the accuracy of the other methods. These results should only be indicative of the raw power of the methods on relatively simple data sets. Nevertheless, is it true that the superiority of the persistence statistics, its small complexity and its intuitiveness makes it very susceptible of becoming part of the standard TDA toolbox. Notice too that the specific tropical coordinates used in this paper were chosen from [Kal18], which also examines the MNIST data set and hence the polynomials and parameter r have probably been tuned to perform specifically well on this data set. Again this suggests that, whilst tropical coordinates do perform very well in comparison to other methods, their advocates should work on automating the selection of specific polynomials given a data set to ensure that such a high performance is observed on new data sets too. On the other hand, the kernel methods perform slightly worst but we are working with substantially smaller data sets. It is tempting to compare the four methods across samples of the same size, but an inherit disadvantage of the kernel methods is their time complexity. Therefore, we have chosen sample sizes with similar computational times to ensure that we factor in the computational costs of each method and compare them holistically. Therefore, the inferiority of the accuracy figures do not necessarily reflect an inferior ability of the kernel methods to extract the topological information, it is also a reflection of their time complexity.

In conclusion, whilst all vectorisations and kernel methods perform at high accuracy rates, the persistence statistics method outperforms the rest. Whilst this might not be true for smaller, simpler data sets like the synthetic data presented, it is the case with MNIST.

A similar superiority is observed across other canonical data sets employed in ML [Ali+22].

6 Conclusion

PH is a vibrant, active field of study in applied mathematics which has caught the attention of many researchers across different disciplines. However, at the time of writing, there does not exists concise doctrines to guide potential scientists who may wish to incorporate elements of PH into their data analysis pipelines. There is a lack of theoretical foundations despite the extraordinary empirical advances of TDA in the past decade. In this essay we hope to have rigorously explained the mathematics behind the PH pipeline, in order to better understand the successes (and failures) of PH in specific circumstances.

In this regard, we hope to observe two main developments in the field in the coming years. On one hand, we hope that more systematic benchmarkings are produced which pay special attention to the mechanisms underlying the pipeline to create better diagnosis of how to build upon the current methods. This does not only encompass adding off-the-shelf statistical methods such as regularisation to the end of the pipeline, but also trying to understand the actual topological information encoded in the vectorisations. In addition, we also hope to see attempts such as that of [Ali+22] to facilitate the use of PH methods to a greater community of scientists. User friendly applications or accessible libraries which agglutinate existing code from different parts of the pipeline are steps in the correct direction. These types of automatisations are needed if we wish for TDA to become a mainstream branch of ML.

References

[Kuh60] Harold W. Kuhn. "Some Combinatorial Lemmas in Topology". In: *IBM Journal of Research and Development* 4.5 (1960), pp. 518–524. DOI: 10.1147/rd.45.0518.

[BCR84] Christian Berg, Jens Peter Reus Christensen, and Paul Ressel. *Harmonic Analysis on Semigroups: Theory of Positive Definite and Related Functions*. Ed. by Springer. 1984.

[Bey+98] Kevin Beyer et al. *When is the nearest neighbor meaningful?* Tech. rep. University of Wisconsin Madisson, CS Dept., 1998.

[ZC05] Afra Zomorodian and Gunnar Carlsson. "Computing Persistent Homology". In: *Discrete Computational Geometry* 33 (2005), pp. 249–274. DOI: https://doi.org/10.1007/s00454-004-1146-y.

[BK07] Peter Bubenik and Peter T. Kim. "A statistical approach to persistent homology". In: *Homology, Homotopy and Applications* 9.2 (2007), pp. 337–362. DOI: 10.4310/hha.2007.v9.n2.a12.

[SG07] Vin de Silva and Robert Ghrist. "Coverage in sensor networks via persistent homology". In: *Algebraic and Geometric Topology* 7 (2007), pp. 339–358. DOI: 10.2140/agt.2007.7.339.

[Ghr08] Robert Ghrist. "Barcodes: the persistent topology of data". In: *Bulletin of the American Mathematical Society* 45.1 (2008), pp. 61–75.

[HTF08] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning. Data Mining, Inference, and Prediction*. Ed. by Springer. 2nd ed. 2008.

[Lei+08] Gregory Leibon et al. "Topological structures in the equities market network". In: *Proceedings of the National Academy of Sciences* 105.52 (2008), pp. 20589–20594. DOI: 10.1073/pnas.0802806106.

[HMR09] Danijela Horak, Slobodan Maletić, and Milan Rajković. "Persistent homology of complex networks". In: *Journal of Statistical Mechanics: Theory and Experiment* 2009.03 (2009), P03034. DOI: 10.1088/1742-5468/2009/03/p03034.

[Chm+10] Erin W. Chmabers et al. "Vietoris-Rips Complexes of Planar Point Sets". In: *Discrete & Computational Geometry* 44 (2010), pp. 75–90. DOI: https://doi.org/10.1007/s00454-009-9209-8.

[Coh+10] David Cohen-Steiner et al. "Lipschitz functions have l_p-stable persistence". In: *Foundations of Computational Mathematics* 10 (2010), pp. 127–139.

[EL10] Herbert Edelsbrunner and John L.Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2010.

[Zom10] Afra Zomorodian. "Fast construction of the Vietoris-Rips complex". In: *Computers & Graphics* 34.3 (2010), pp. 263–271. DOI: https://doi.org/10.1016/j.cag.2010.03.007.

[Pet+13] Giovanni Petri et al. "Topological Strata of Weighted Complex Networks". In: *PLoS ONE* 8.6 (2013). DOI: https://doi.org/10.1371/journal.pone.0066506.

[Zhu13] Xiaojin Zhu. "Persistent homology: An introduction and a new text representation for natural language processing". In: Aug. 2013, pp. 1953–1959.

[Ghr14] Robert Ghrist. *Elementary Applied Topology*. 1st ed. 2014.

[Rei+14] Jan Reininghaus et al. *A Stable Multi-Scale Kernel for Topological Machine Learning*. 2014. DOI: 10.48550/ARXIV.1412.6821.

[XW14] Kelin Xia and Guo-Wei Wei. "Persistent homology analysis of protein structure, flexibility, and folding". In: *International Journal for Numerical Methods in Biomedical Engineering* 30.8 (June 2014), pp. 814–844. DOI: 10.1002/cnm.2655.

[Bub15] Peter Bubenik. "Statistical Topological Data Analysis using Persistence Landscapes". In: *Journal of Machine Learning Research* 16 (2015), pp. 77–102.

[Chi+15] Harish Chintajunta et al. "An entropy-based persistence barcode". In: *Pattern Recognition* 2 (2015), pp. 391–401.

[ACC16] Aaron Adcock, Erik Carlsson, and Gunnar Carlsson. "The Ring of Algebraic Functions on Persistence Bar Codes". In: *Homology, Homotopy and Applications* 18.1 (2016), pp. 381–402.

[Was16] Larry Wasserman. *Topological Data Analysis*. 2016. DOI: 10.48550/ARXIV.1609.08227.

[CCO17] Mathieu Carrière, Marco Cuturi, and Steve Oudot. "Sliced Wasserstein Kernel for Persistence Diagrams". In: *CoRR* (2017).

[Ott+17] Nina Otter et al. “A roadmap for the computation of persistent homology”. In: *EPJ Data Science* 6.1 (2017). DOI: 10.1140/epjds/s13688-017-0109-5.

[CMW18] Zixuan Cang, Lin Mu, and Guo-Wei Wei. “Representability of algebraic topology for biomolecules in machine learning based scoring and virtual screening”. In: *PLOS Computational Biology* 14 (2018), pp. 1–44. DOI: 10.1371/journal.pcbi.1005929.

[Cha18] Timothy Moon Yew Chan. “Improved Deterministic Algorithms for Linear Programming in Low Dimensions”. In: 14.3 (2018). DOI: 10.1145/3155312.

[Kal18] Sara Kališnik. “Tropical Coordinates on the Space of Persistence Barcodes”. In: *Foundations of Computational Mathematics* 19.1 (2018), pp. 101–129. DOI: 10.1007/s10208-018-9379-y.

[Per18] Jose A. Perea. *A Brief History of Persistence*. 2018.

[PXL18] Chi Seng Pun, Kelin Xia, and Si Xian Lee. *Persistent-Homology-based Machine Learning and its Applications – A Survey*. 2018. DOI: 10.48550/ARXIV.1811.00252.

[TSB18] Christopher Tralie, Nathaniel Saul, and Rann Bar-On. “Ripser.py: A Lean Persistent Homology Library for Python”. In: *The Journal of Open Source Software* 3.29 (2018), p. 925. DOI: 10.21105/joss.00925.

[Bub+20] Peter Bubenik et al. “Persistent homology detects curvature”. In: *Inverse Problems* 36.2 (2020), p. 025008. DOI: 10.1088/1361-6420/ab4ac0.

[KSA20] Shizuo Kaji, Takeki Sudo, and Kazushi Ahara. “Cubical Ripser: Software for Computing Persistent Homology of Image and Volume Data”. In: (2020).

[Ma20] Gang Ma. “Using Topological Data Analysis to Process Time-series Data: A Persistent Homology Way”. In: *Journal of Physics: Conference Series* 1550.032082 (2020). DOI: 10.1088/1742-6596/1550/3/032082.

[BPP21] Danielle Barnes, Luis Polanco, and Jose Perea. “A Comparative Study of Machine Learning Methods for Persistence Diagrams”. In: *Frontiers in Artificial Intelligence* 4 (July 2021).

[Gue+21] Marco Guerra et al. “Homological scaffold via minimal homology bases”. In: *Scientific Reports* 11 (Mar. 2021), p. 5355. DOI: 10.1038/s41598-021-84486-1.

[Ali+22] Dashti Ali et al. “A Survey of Vectorization Methods in Topological Data Analysis”. In: *arXiv preprint arXiv:2212.09703* (2022).

[Zha22] Haoyu Zhai. “Improving KNN Algorithm Efficiency Based on PCA and KD-tree”. In: *2022 International Conference on Machine Learning and Knowledge Engineering (MLKE)*. 2022, pp. 83–87. DOI: 10.1109/MLKE55170.2022.00021.