Advanced Data Analytics Final Project: Persistent Homology

Matthew LeBar

December 9, 2025

Abstract

This report provides a self-contained presentation of Persistent Homology, a major algorithm in the field of Topological Data Analysis. The theoretical exposition assumes only knowledge of set theory and general mathematical maturity, and connects to an explanation of a major algorithm for computing Persistent Homology. The report also contains notes on an accompanying implementation of the algorithm in Python, which can be found on Github.

1 Introduction

Topological Data Analysis and Persistent Homology

Topological Data Analysis (TDA) is the use of topological tools and techniques to investigate qualitative features of datasets. Persistent Homology (PH) is one such tool, and is considered the main workhorse of Topological Data Analysis [26]. Intuitively, it detects the structures of "holes" in datasets. Methods in TDA often require a high level of mathematical background to understand (specifically, Algebraic Topology), and Persistent Homology is no exception - the reader will not reach a formal definition of Persistent Homology until section 2.5. The problem is particularly severe for TDA because it is not just the algorithms that require mathematical expertise to understand, but the results of the algorithms. The products of TDA methods are typically topological objects, and their interpretation requires familiarity with topology. Thus, TDA cannot just be treated as a set of "black box" algorithms. This creates a high barrier to entry for use of TDA.

Despite this, TDA and PH have had a wide range of successful applications, including in finance [1, 12, 13, 14], nanotechnology [18], molecular science [27], biology [7, 24], signal processing [23], network science [2, 11], materials science [4, 8], and oncology [5, 21]. Many (though not all) of these applications come when there is a clear underlying spatial structure to analyze, rather than just a dataset one is generally curious about. TDA methods are not in competition with e.g. regression. For general overviews, the reader is referred to Otter et. al for PH [22], Pun et al. for PH in Machine Learning [25], Leykam and Angelakis [19] or Zia et. al [28] for TDA in Deep Learning, and Su et al. for TDA beyond PH [26].

Related Work

There are several self-contained presentations of Persistent Homology that do not assume prior background in Algebraic Topology but do not include exposition of any algorithm for computing it such as Kemme and Agyingi [16], Buchet and Hiroaka [4], or Aktas et. al [2]. Zomorodian and Carlsson provide an overview of Persistent Homology and the standard algorithm for computing it in the general case [10], but assume a very high level of background knowledge in abstract algebra. Edelsbrunner et al. gives an exposition of the theory and the algorithm [9], but assume background in abstract algebra, provide very abstract pseudocode implementation of the algorithm, and do not discuss peristence complexes. Otter et al. is a quality beginner-friendly introduction [22] that includes both theoretical background and the algorithm, but use linear algebraic representations of boundaries and homologies, which is not necessary for understanding or computing PH for most applications. Coskunuzer and Akçora provide another beginner-friendly introduction for both the theory and algorithm for PH, but do so specifically for PH for Machine Learning, while this presentation is more general. Finally, Bauer provides an alternative algorithm for PH that is more efficient in cases where it is applicable [3].

This Work

The aim of this report is to provide a self-contained overview of Persistent Homology both conceptually and computationally, with the aim of making Persistent Homology a usable tool for those without prior training in Algebraic Topology. With this in mind, the theory in this report is aimed to give an essential understanding of the simplest and most intuitive case of persistent homology, and the one most suited for data analysis. This approach will capture the information we want for data analysis - holes and loops in the data - while ignoring other details (orientation of loops, non-orientable behavior) that is mathematically interesting but not as essential for most applications. Section 2.7 clarifies the sense in which this is a specific case of Persistent Homology.

2 Background Theory

We start by covering some basic group theory from abstract algebra, then move onto concepts from Algebraic Topology and conclude by defining persistent homology groups.

2.1 Groups

Our presentation of groups loosely follows Judson [15]. A **group** G is a set equipped with an operation $\cdot$ satisfying three axioms:

1. **Associativity** $\forall a, b, c \in G, (a \cdot b) \cdot c = a \cdot (b \cdot c)$
2. **Identity** $\exists e \in G$ such that $\forall a \in G, ea = a = ae$
3. **Inverses** $\forall a \in G, \exists a^{-1} \in G$ such that $aa^{-1} = e = a^{-1}a$

Some canonical examples include: the integers under addition, the rational numbers under addition or multiplication, the integers modulo some number n under addition, and the real numbers under addition or multiplication. The group of integers modulo n under addition is written $\mathbb{Z}/n\mathbb{Z}$ or $\mathbb{Z}_{\ltimes}$. In general, group elements can be thought of as symmetries; Carter provides an excellent intuitive overview [6]. If the operation is also commutative, that is, $\forall a, b \in G, a \cdot b = b \cdot a$, we say G is Abelian. With Abelian groups, we usually consider the operation to be addition and write the identity element as 0, a convention we follow in this report. All the canonical examples given above were Abelian, as will all the groups considered in this report. That said, there are important examples of non-Abelian groups, like invertible square matrices of a certain dimension under multiplication.

We define a **homomorphism** as a function ϕ between two groups $(G, \cdot)$ and $(H, +)$ such that

$$\forall g_1, g_2 \in G, \phi(g_1 \cdot g_2) = \phi(g_1) + \phi(g_2)$$

We can think of this as an operation preserving map. The preimage of the identity element in G is a subgroup of G, called the **kernel** of ϕ and written $\ker \phi$.

Just as we have subsets of sets, we have **subgroups** of groups. A subgroup H of a group G is a subset of the group such that H is a group when the group operation is restricted to it. We write this as $H \leq G$, or if $H \neq G$ $H < G$. A subset H of a group is a subgroup just when three conditions hold:

1. $e \in H$
2. $\forall a, b \in H, ab \in H$
3. $\forall a \in H, a^{-1} \in H$

Note that for any group G, both G and $\{e\}$ will be subgroups.

A subgroup H of G generates what are called **cosets**. The left coset of H with representative g, written gH, is defined as $gh : h \in H$; right cosets just switch the side of g. We will not be dealing with right cosets for the rest of this exposition. Cosets are equivalence classes of elements of g. Note $H = eH$, but none of the other cosets generated by H can be subgroups, since they will lack the identity element. (The representative ”takes the place” of e).

Sometimes, the set of cosets generated by a subgroup N can itself be treated as a group, with the operation defined as $(aN)(bN) = (ab)N$. In order for this to be well defined, we must have that N is a **normal subgroup** of G, written $N \triangleleft G$. N is a normal subgroup iff $\forall g \in G$, $gN = Ng$, or equivalently, $gNg^{-1} = N$. The resulting group of cosets, called a **factor group** or **quotient group**, is written as G/N. A helpful intuition for this construction is that we are "zeroing out" all the elements in N; we are making them equivalent to the identity element. Then the rest of the group is structured relative to this enlarged identity element. In the context of Abelian groups (which again, is the context we will be working for this report), all subgroups are automatically normal (this can be verified very easily), so we will always be able to construct quotient groups from any given subgroup.

The final group-theoretic construct we will make use of is a **free Abelian group**. The exposition here follows Lang [17]. An Abelian group A is free if it has a **basis**. Treating the operation of A as addition, a basis B is a subset of A such that $\forall a \in A$, a can be uniquely expressed as a linear sum:

$$a = \sum_{b_i \in B} c_i b_i$$

where $c_i \in \mathbb{Z}$ and finitely many c_i are non-zero. That is, each element of A can be uniquely expressed as a finite sum of elements of B. The elements of B are called generators and we call $|B|$ the **rank** of A.

A fuller mathematical treatment of PH in general would also need to include an exposition of rings, modules, fields, graded rings, and graded modules. However, this will suffice for our purposes.

2.2 Simplical Complexes

Having completed a very brief overview of key concepts in group theory, we move on to material from Algebraic Topology. Our presentation begins by following Munkres [20]. We will initially consider **simplices** as geometric objects. To do this we first say that a set of points $\{a_0, a_1, \dots a_k\} \subset \mathbb{R}^K$ is **geometrically independent** just when $\sum\limits_{i=0}^{k} t_i = 0$ and $\sum\limits_{i=0}^{k} t_i a_i = 0$ jointly imply $t_0 = t_1 = \cdots = t_k = 0$. Then we define the **k-simplex** σ spanned by $a_0, \dots, a_k$ (which we call its **vertices**) to be all $x \in \mathbb{R}^K$ such that $x = \sum\limits_{i=0}^{k} t_i a_i$ where $\sum\limits_{i=0}^{k} t_i = 1$ (essentially, x must be a weighted average of the vertices). Note a k-simplex has $k+1$ points. Then a 0-simplex is a point, a 1-simplex is a line, a 2-simplex is a (filled in) triangle, and a 3-simplex is a (filled in) tetrahedron (the value of k gives the dimension the simplex must be embedded in). Intuitively, the point of specifying that the points are geometrically independent is to ensure we do not try and construct a triangle out of three collinear points, or the equivalent in a different dimension.

A simplex spanning any subset of $\{a_0, \dots a_k\}$ is called a a **face** of σ. Thus the faces of a 2-simplex (a triangle) would be its edges and vertices. We are interested in collections of simplices called **simplicial complexes**. A collection K of simplices is a simplical complex just when

1. Every face of a simplex in K is also in K
2. The intersection of any two simplices of K is a face of each of them

Essentially a simplicial complex is just a well-structured set of simplices.
The geometric structure is helpful for developing an intuitive understanding of simplicial complexes, and absolutely crucial for understanding what homology groups tell us about a dataset, but it is not analytically necessary for defining or computing persistent homology. From here we follow the presentation in Zomorodian and Carlsson, which deals with what Munkres calls abstract simplicial complexes [20, 29]. We follow Zomorodian and Carlsson by redefining ”simplicial complexes” to refer specifically to these abstract simplicial complexes. Now, a simplicial complex is a set K with a collection $\mathcal{S}$ of subsets of K, which are the simplices of K, such that

1. $\forall \sigma \in \mathcal{S}$, if $\tau \subseteq \sigma$, $\tau \in \mathcal{S}$.
2. $\forall v \in K$, $\{v\} \in \mathcal{S}$

These correspond to the first and second conditions of the geometric simplicial complexes above. (The second one takes some thinking to see as equivalent.) Our vertices are the singleton sets v, σ is a k-simplex just when $|\sigma| = k + 1$, and if $\tau \subseteq \sigma$, we say τ is a face of σ. Thus, all we have done is recharacterize the structure of a simplicial complex set-theoretically rather than geometrically.

2.3 Filtrations and Alpha Complexes

A **filtration** or **filtered complex** is a sequence of nested complexes where

$$\varnothing = K^0 \subseteq K^1 \subseteq \cdots \subseteq K^m = K$$

For any $i \geq m$, we say $K^i = K^m$. Intuitively we can think of this as a complex that is growing (or filling in) over ’time’. Persistent homology will compute the persistence of homologies over that time.
I was originally going to code the filtrations using Vietoris-Rips complexes, which builds a filtration by increasing a threshold value and connecting all simplices all of whose vertices have pairwise distance less than that threshold. However, the computing of pairwise distances leads to combinatorial explosion in trying to construct the complexes; the most popular and efficient implementation of persistent homology for Vietoris-Rips complexes, Ripser, uses cohomology and only implicitly calculates the complex construction [3]. That would require considerably more theoretical overhead, and the construction of the complexes (as opposed to computation of the homology groups) was only

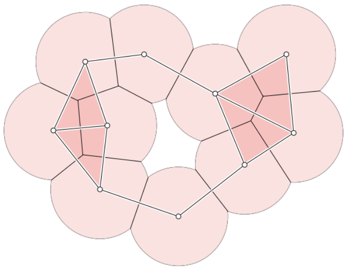

Figure 1: An alpha complex, from Edelsbrunner and Harer [10]

of secondary interest, so I elected to use a Gudhi library to generate my complexes. This also made it easy to verify the correctness of my algorithm, since the Gudhi representation of filtrations comes with a built-in method for computing persistent homology.
Of course, using a library doesn't obviate the time complexity of constructing Vietoris-Rips complexes, so I decided to work with alpha complexes instead. For my exposition of alpha complexes, I follow Edelsbrunner and Harer [10]. Note here we are in a geometric context, but the resulting complex can be represented abstractly. We start by defining Voronoi cells. Given a set $S \subset \mathbb{R}^n$, the **Voronoi cell** for a point $s \in S$ is given by

$$V_s = \{x \in \mathbb{R}^n | \forall t \neq s \in S, ||x - s|| \leq ||x - t||\}$$

Now we define two more constructions on s. Given a non-negative radius $r \in \mathbb{R}$, let $B_s(r) = \{x \in \mathbb{R}^n | \quad ||x-s|| \leq r\}$ (the closed ball of radius r around s), and let $R_s(r) = B_s(r) \cap V_s$. Thus, $R_s(r)$ is the set of points 1) at most r distance from s and 2) closer to s than any other member of S. Thus note the intersection of any two such R_s's would have to be on the boundary of both of them. Then the alpha complex is given by

$$\text{Alpha}(r) = \{\sigma | \bigcap_{s \in \sigma} R_s(r) \neq \varnothing\}$$

Thus, a simplex is a member of the complex just when the boundaries of the R_s's for all its vertices intersect. Intuitively, as we grow r, the balls will expand, and as they start to "push against" each other, creating shared boundaries.See figure 1. Eventually if r increases enough we will get all possible intersections for a given dimension. It is easy to verify that this satisfies our requirements for a simplicial complex.

2.4 Chain Complexes

We return to Zomorodian and Carlsson to unite simplicial complexes with group theory [29]. We will briefly introduce the idea of an **oriented simplex**, which we write as $[\sigma]$, where σ is a simplex. We mark the simplex with opposite orientation to $[\sigma]$ as $-[\sigma]$. The orientation itself is an equivalence class of orderings of the vertices, so we often write an oriented simplex as an ordered list of vertices, i.e. $[\sigma] = [v_0, v_1, \ldots, v_k]$. Concretely, if $[\sigma] = [a, b, c]$ then we also have $[\sigma] = [b, c, a]$ or $[c, a, b]$, but $[\sigma] = -[a, c, b]$. Intuitively we can think of the orientation of a 1-simplex (i.e., a line) as just the direction in which that simplex is traversed. Then the orientation for a 2-simplex (a triangle) can be thought of as just summarizing the orientation of each of its faces (edges), and the orientation of a 0-simplex (a point) tells us whether we think of it as a ”source” or ”sink” node. Similarly, you can think of the orientation of a tetrahedron as just a summary of the orientation of its faces (triangles).

Now the k**th chain group** C_k of a simplicial complex K is the free Abelian group over addition generated by the oriented k-simplices of K. The intuition here becomes somewhat abstract, but when k is say 2, this is the group of all possible combinations (including orientations) of all triangles in the complex. Such combination (or more precisely, formal sum c is called a k-chain, and can be expressed as $c = \sum_i n_i[\sigma_i], \sigma_i \in K$ and $|\sigma_i| = k + 1$ with coefficients $n_i \in \mathbb{Z}$.

We now define the **boundary operator** $\partial_k : C_k \mapsto C_{k-1}$ as a homomorphism where if $[\sigma] = [v_0, v_1, \ldots, v_k]$ then

$$\partial_k[\sigma] = \sum_i (-1)^i [v_0, v_1, \ldots, \hat{v_i}, \ldots, v_k]$$

where $\hat{v_i}$ indicates that v_i is removed from the simplex, and

$$\partial_k \sum_i n_i[\sigma_i] = \sum_i n_i \partial_k n_i[\sigma_i]$$

Thus the boundary operator maps a k-simplex to an alternating formal sum of its $k-1$-simplices, and extends this linearly to the boundary of a formal sum of k-simplices. Intuitively, a tetrahedron is bounded or enclosed by its triangular faces, and a triangle is bounded by its edges.

An important subgroup of the kth chain group is the **cycle group**, defined by $Z_k = \ker \partial_K$. Intuitively, an example of an element of Z_1 would be a loop of edges, with or without the interior (formally, the interior being included would mean that there are triangles tiling that space included in the k-chain), and an example of an element of Z_2 would be a tiling of the surface of a 3d shape by triangles (like a triangulated sphere or a torus), with or without the interior.

A **chain complex** is a series C_* of chain groups connected by boundary operators:

$$\ldots \xrightarrow{\partial_{k+1}} C_k \xrightarrow{\partial_k} C_{k-1} \xrightarrow{\partial_{k-1}} \ldots$$

This construction leads to a crucial observation. Let $[\sigma] = [v_0, v_1, \dots, v_k]$ be a k-simplex. Then

$$\begin{aligned}
\partial_{k-1}\partial_k[\sigma] &= \partial_{k-1}\sum_i (-1)^i [v_0, v_1, \dots, \hat{v_i}, \dots, v_k] \\
&= \sum_i (-1)^i \partial_{k-1}[v_0, v_1, \dots, \hat{v_i}, \dots, v_k] \\
&= \sum_i (-1)^i \left(\sum_{j<i} (-1)^j [v_0, v_1, \dots, \hat{v_j}, \dots \hat{v_i}, \dots, v_k] \right. \\
&\quad \left. + \sum_{j>i} (-1)^{j-1} [v_0, v_1, \dots, \hat{v_i}, \dots \hat{v_j}, \dots, v_k] \right)
\end{aligned}$$

Now take any $j < i$ in our indexing of simplices and consider

$$[v_0, v_1, \dots, \hat{v_j}, \dots \hat{v_i}, \dots, v_k]$$

in the inner sums. Note that it will show up twice, once in the sum generated by i and once in the sum generated by j. Thus we will have

$$(-1)^i(-1)^j[v_0, v_1, \dots, \hat{v_j}, \dots \hat{v_i}, \dots, v_k]$$

and

$$(-1)^{i-1}(-1)^j[v_0, v_1, \dots, \hat{v_j}, \dots \hat{v_i}, \dots, v_k].$$

But $(-1)^{i-1}(-1)^j = -(-1)^i(-1)^j$ so they will cancel out. Since any such simplex can be represented, the sum as a whole will cancel out, so we get $\partial_{k-1}\partial_k[\sigma] = 0$ for any simplex. Since the boundary operators are linear over formal sums of simplices, $\partial_{k-1}\partial_k$ will map any complex to 0. Thus, the boundary of a boundary is always 0. Making this construction work does require the counterintuitive idea that any k-simplex in some sense contains every $k-2$-simplex in it "twice", once with each orientation. Intuitively, once we remove what a boundary encloses, we have something that doesn't enclose anything, i.e., something with no boundary. When we consider the surface of a sphere as enclosing the sphere, it's a boundary, but when we consider it as a space in its own right, it has no volume so encloses nothing. Thus, whether the examples of cycles given above includes the interior or not corresponds to whether they are boundaries or not. It is crucial to note that we cannot tell if a k-chain is a boundary or not just by looking at C_k; you need to look at C_{k+1} and ∂_{k+1}.
It is easy to verify (due to linearity) that $B_k = \operatorname{Im}\partial_{k+1}$ is a group (the boundary group), and the above shows that $B_k \subseteq Z_k$, so we have a subgroup nesting $B_k \trianglelefteq Z_k \trianglelefteq C_k$. See figure 2. (Since the group is Abelian, all subgroups are trivially normal.)

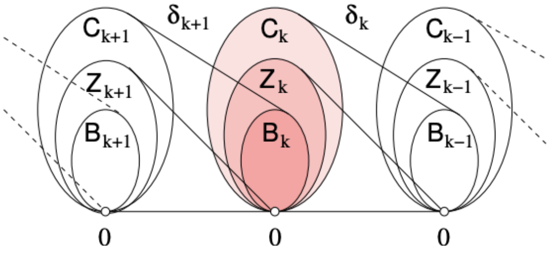

Figure 2: The structure of a chain complex with its boundary operators [29]

2.5 Homology Groups and Persistent Homology Groups

We are now in position to define homology groups. The k-th **homology group** is defined as $H_k = Z_k/B_k$, using the normal group construction defined in 2.1. Essentially we are treating all elements of the boundary group as 0 elements. Thus, the elements of the homology group will be equivalence classes of cycles. Two such equivalent cycles are called *homologous* and differ by a boundary. An element of H_k that is not the identity element is an equivalence class of cycles that are not boundaries, i.e. cycles without their interiors. We are interested in counting the generating elements of the homology group other than boundaries, which we call the **Betti number**, and can define as $\text{rank}Z_k - \text{rank}B_k$.

Now given a point cloud dataset, it is not obvious what simplicial complex to use to analyze it. We can construct any simplices we would like, from 0-simplices containing just individual points to a high dimensional simplex containing all points as vertices. Rather than find a single best complex with which to analyze homology, we create a filtration complex, and see how features persist over the filtration. Transient elements of the homology group are likely to be just noise in the dataset, while ones that span a large time range in the filtration are likely fundamental features of the dataset.

To represent the structure of persistence over chain complexes, we start by considering a given complex K^i of a filtered complex. We can associate it with k-chains C_k^i, boundary operators δ_k^i as outlined in section 3.5, and from there cycle groups Z_k^i, boundary groups B_k^i, and homology groups H_k^i. Now the **p-persistent k-th persistent homology group** of K_i is defined as

$$H_k^{i,p} = Z_k^i/(B_k^{i+p} \cap Z_k^i).$$

Intuitively, this zeroes out all the elements of the cycle group that are boundaries after p timesteps. Note that if cycle is already a boundary after i timesteps, it will still be a boundary after $i + p$ timesteps, as $K_i \subset K_{i+p}$, so we cannot lose the interior the cycle bounds. On the other hand, something is a cycle but not a boundary can become a boundary if its interior is added later. Note that clearly $B_k^{i+p} \leq C_k^{i+p}$ and $Z_k^i \leq C_k^{i+p}$ (since cycles will not disappear over time), and the intersection of two subgroups is always a subgroup, so we must have $B_k^{i+p} \cap Z_k^i \leq Z_k^i$.

2.6 Persistence Complexes

We would like a structure to simultaneous track persistence across chain complexes and boundary structure within complexes; we introduce **persistence complexes** to do so. We define a **persistence complex** $\mathcal{C}$ as a family of chain complexes $\{C_*^i\}_{i\geq 0}$ along with inclusion maps $f^i : C_*^i \mapsto C_*^{i+1}$. We can combine this with the boundary operators in our chain groups to make the following 2d diagram, with the filtration increasing from left to right, and our dimension within a complex decreasing from to bottom:

$$\begin{tikzcd}
{} \arrow[d, "\partial_3"] & {} \arrow[d, "\partial_3"] & {} \arrow[d, "\partial_3"] & \\
C_2^0 \arrow[r, "f^0"] \arrow[d, "\partial_2"] & C_2^1 \arrow[r, "f^1"] \arrow[d, "\partial_2"] & C_2^2 \arrow[r, "f^2"] \arrow[d, "\partial_2"] & \dots \\
C_1^0 \arrow[r, "f^0"] \arrow[d, "\partial_1"] & C_1^1 \arrow[r, "f^1"] \arrow[d, "\partial_1"] & C_1^2 \arrow[r, "f^2"] \arrow[d, "\partial_1"] & \dots \\
C_0^0 \arrow[r, "f^0"] & C_0^1 \arrow[r, "f^1"] & C_0^2 \arrow[r, "f^2"] & \dots
\end{tikzcd}$$

2.7 Simplifying Coefficients

We have all the theoretical background needed to understand persistent homology now. Finally, we need to discuss a simplification common for practical applications of persistent homology. In section 3.4, we defined chain complexes using coefficients from $\mathbb{Z}$. However, we can instead use $\mathbb{Z}_2$, which contains just the elements 0 and 1. Further, its group addition is defined equivalently to XOR, which means we can think of k-chains as sets of simplices. Note also that $-1 \equiv 1 \mod 2$ so our boundary operator can be redescribed without the power of -1, i.e.

$$\partial_k[\sigma] = \sum_i [v_0, v_1, \dots, \hat{v_i}, \dots, v_k].$$

Now the cancellation property that ensured the boundary of a boundary is zero (which is crucial for our definition of homology groups) can be seen just by observing that addition is equivalent to XOR. Thus, this considerably simplifies our analysis and presentation.

This raises two question. First, why bother with $\mathbb{Z}$ at all? While $\mathbb{Z}_2$ suffices for most practical applications, it does miss out on some information. In particular, it does not capture the orientation of simplices (since it does not distinguish between positives and negatives), and it cannot capture what are called non-orientable behaviors, which are important features for some abstract topological spaces. It's not obvious what practical benefit these would be in most cases but it important to recognize the limitations of $\mathbb{Z}_2$. Ultimately, I elected to keep the exposition with $\mathbb{Z}$ in because I do think it is helpful for understanding the fundamental conceptual apparatus.

Second, is this simple enough to remove the discussion of groups from the exposition, resulting in a considerably simpler approach? It is possible that a set-theoretic construal of homology groups is available, but it would need some way to represent the preservation of the XOR structure by the boundary maps, as well as the relation between homology groups, boundary groups, and cycle groups. It is possible that such a presentation is available but it is not obvious that it would be any easier or simpler than the group theoretic presentation. This would be an interesting direction for future research.

3 Algorithm

Before laying out the algorithm, we need a key observation. For any k-simplex σ entering the filtration, it will either 1) "kill" a $k-1$ cycle (i.e., make it a boundary) or 2) "birth" a k-cycle. To see why this is the case, consider $\partial_k \sigma$. Either it is in the boundary of another k-chain in the simplex or it is not. If it is not, then we have killed it by making it a boundary. If it is, then if the k-chain is c, we must have $\partial_k(\sigma + c) = \partial_k \sigma + \partial_k c$. Since addition is equivalent to XOR, we must have $\partial_k(\sigma + c) = 0$, so by definition the new k-chain $\sigma + c$ is a cycle that has just been generated.

The pseudocode is an adaptation from Zomorodian and Carlsson [29]. We have simplified their algorithm presentation, as their version is more general. The generality of their approach requires a matrix algebra approach to the algorithm that we have jettisoned for clarity. We present the pseudocode and then explain the algorithm in depth. The algorithm assumes that the filtration complex is given sorted either by 1) filtration order or 2) dimension, and then within dimension by filtration order.

L_k stores the intervals for homologies of dimension k, and $T[i]$ stores the boundary of the simplex that kills the k-cycle generated by simplex i. Marking a simplex indicates it creates a cycle. Our algorithm calls another algorithm "RemoveDeadBoundaries", which is an adaptation of "RemovePivotRows" in Zomorodian and Carlsson. We also use "FindYoungest" where they use "MaxIndex", which finds the youngest (i.e., most recent to enter the filtration) simplex in a boundary chain. Since simplices of the same dimension are entered in filtration order, this can be done with simple iterative checking, so we do not include pseudocode for it. We treat simplices as sets of vertices and k-chains as sets of simplices in accordance with the discussion in section 2.7. The degree "deg" of a simplex is when it enters the filtration complex.

We start our analysis with RemoveDeadBoundaries. The ultimate goal is to

Algorithm 1 ComputeIntervals(K: filtration complex)

```
for $k = 0$ to $\dim(K)$ do
    $L_k \leftarrow \varnothing$
end for
for $\sigma_j \in K$ do
    $d \leftarrow$ RemoveDeadBoundaries($\sigma_j$)
    if $d = \varnothing$ then
        Mark $\sigma_j$
    else
        $i \leftarrow$ FindYoungest($d$), $k \leftarrow \dim(\sigma_i)$, $T[i] \leftarrow d$
        $L_k$.append($(\deg(\sigma_i), \deg(\sigma_j))$)
    end if
end for
for $\sigma_j \in K$ do
    if $\sigma_j$ is marked and $T[j]$ is empty then
        $k \leftarrow \dim(\sigma_j)$
        $L_k$.append($(\deg(\sigma_i), \infty)$)
    end if
end for
Return $L_k$s
```

Algorithm 2 RemoveDeadBoundaries(σ: k-simplex)

```
$k \leftarrow \dim(\sigma)$, $d \leftarrow \partial_k \sigma$
Remove unmarked elements of $d$
while $d \neq \varnothing$ do
    $i \leftarrow$ FindYoungest($d$)
    if $T[i]$ is empty then
        Break
    end if
    $d \leftarrow d \oplus T[i]$
end while
return $d$
```

check if σ killed a cycle or not. If it did, said cycle would be its boundary, so we examine its boundary. Firstly, we discard any elements of the boundary that did not birth a cycle, as they will not help us tell whether our simplex kills its boundary. (Lemma 4.2 in Zomorodian and Carlsson guarantees that this will not remove any cycles we would otherwise want to check [29].) Then, we work through the elements of the boundary that did generate a cycle, and check to see if that cycle has already been killed. If it has, then we remove all other elements of the boundary that were killed by the same simplex (this is what the XORing with $T[i]$ does). If not, we know the simplex kills its boundary, and we exit the loop. Since we go in reverse filtration order, we know that the element we are examining must have been the one to birth the boundary. If no element of the boundary gets killed, we continue until all elements are XORed out.

With this in mind, the outer ComputeIntervals algorithm is fairly straightforward. We simply examine each simplex in the order it enters the filtration complex and check if it kills something. If it does not, we mark it as having birthed a cycle; otherwise, we record the interval for the cycle it kills. Then we do a second loop over simplices and add intervals going to infinity for any cycles that are birthed but never killed.

4 Implementation

My implementation is in Python, and can be found on Github. It is designed to be used starting with a point cloud represented as a list of lists of real numbers, and by default will compute intervals for H_0, H_1 and H_2 from an alpha complex constructed from the dataset. It also includes methods for generating shapes with which to test persistent homology, and a method for producing barcode diagrams (seen in section 5). See Github for implementation details and more specific usage guidance. My code makes use of the open-source Gudhi TDA library for constructing alpha complexes.

5 Testing and Results

To verify the correctness of my algorithm, I ran some small (∼10 points) datasets and checked to see if the resulting interval sets looked similar to those generated by Gudhi (there are some minor differences in implementation that result in cosmetic differences, but the fundamental structure is the same). I also tested with point clouds sampled from several shapes that I knew would have well defined and easy to verify bases for either H_1 or H_2. For example, to see results for point cloud sampled from five non-overlapping spheres, see figure 3 for Gudhi's barcode diagram and figure 4 for mine. These diagram display the persistence of the 10 longest lasting intervals (I compute the underlying degree differently than they do, so the exact numbers are different, but the basic structure is the same.) All tests verified the correctness of my code (of course, after some debugging).

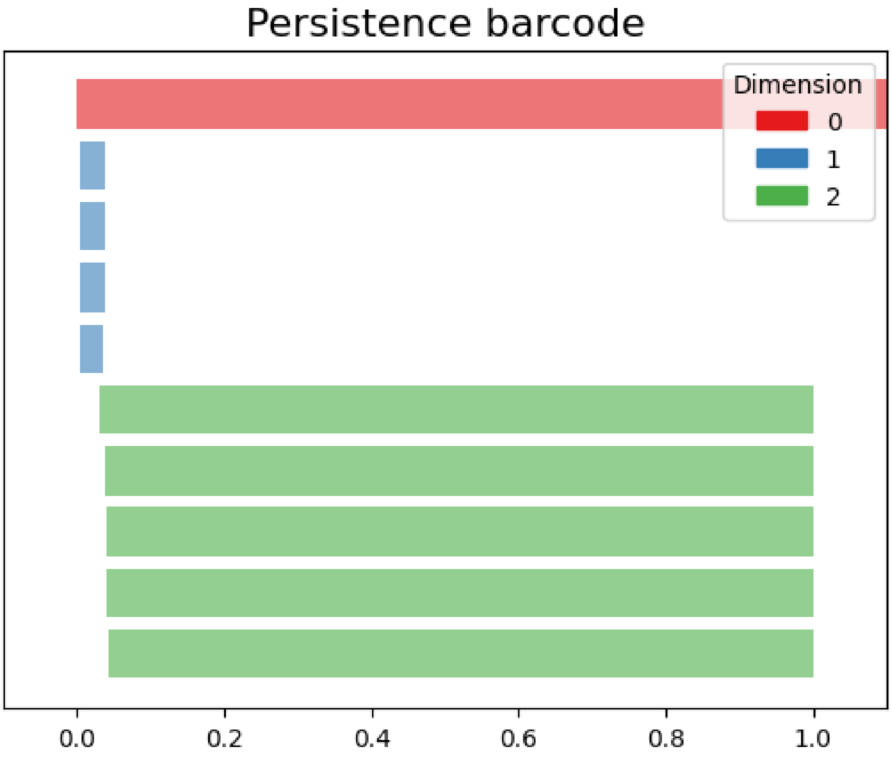

Figure 3: Gudhi Barcode Diagram

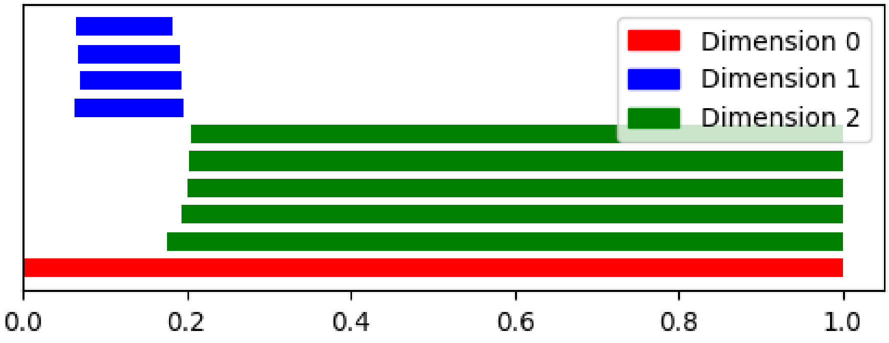

Figure 4: My Barcode Diagram

6 Efforts

The vast majority of my efforts went towards understanding and presenting the theory behind persistent homology. I already knew the theory in sections 2.1 and 2.2 (with the relatively minor exception of Free Abelian Groups), but the meat of the theoretical presentation in 2.3-2.7 was entirely new, and VERY hard to wrap my head around. The use of cancellation through addition essential to defining boundaries was very counterintuitive. Now that I have a sense of the topic and literature, I can see I would have been better served by following a self-contained and intuitive exposition of what Persistent Homology is, then switching to a presentation of the algorithm either in Zomorodian and Carlsson or Edelsbrunner, Letscher, and Zomorodian, but I proceeeded by using Zomorodian and Carlsson as my main resource, which meant trying to work through some very challenging abstract algebra [9, 29]. I was surprised once I had the basic theory in view how simple the algorithm is, both in implementation and intuition. It took ∼80 lines of code to write the core algorithm, and I think my explanation of it is reasonably thorough and accurate.

7 Conclusion and Future Directions

This project provides a self-contained exposition of both the key theoretical background and practical implementation details for Persistent Homology. With some significant time to work through the theory, someone with reasonable algorithmic and mathematical maturity could move from having no understanding of Persistent Homology to being able to code the algorithm presented here themselves.

There are several interesting directions for future work from here. First, while some degree of theoretical complexity is unavoidable, the theoretical exposition here remains formidable (Amitabh and Eric, I apologize for giving you 9 pages of theory to go through in this project - my hope is that the intuition I provide throughout makes it faster to read than a more terse exposition would be). This, along with the observation in section 2.7 that the use of group theory is in some sense quite simple, and in section 6 that the intuition for the algorithm is fairly simple relative to the theoretical background, makes wonder if a simpler presentation of the algorithm is available. Given TDA's notoriously high theoretical barrier to entry, this could provide major benefits for the adoption of Persistent Homology and interest in TDA more broadly.

Along these lines, it would be interesting to see a proof of correctness with only the theoretical background presented here. The basic ideas behind that proof are contained in my exposition of the algorithm, but my argument is somewhat informal, and my appeal to lemma 4.2 from Zomorodian and Carlsson means the proof relies on algebraic concepts more complex than what is presented here. An even further development of course would be to present a proof within a framework that uses even less algebra than the presentation here, if one is possible.

Finally, while I explain what alpha complexes are in section 2.4, I treat the production of alpha complexes from a dataset by the Gudhi library as a black box, and it would be illuminating to include the algorithm for that as well. That said, that is strictly speaking pre-processing for Persistent Homology rather than a part of the algorithm itself, so perhaps it would be more appropriate to include it in something like a comprehensive user manual for Persistent Homology.

References

[1] Samuel W. Akingbade, Marian Gidea, Matteo Manzi, and Vahid Nateghi. Why topological data analysis detects financial bubbles?, 2023.

[2] Mehmet Emin Aktas, Esra Akbas, and Ahmed El Fatmaoui. Persistence homology of networks: Methods and applications, 2019.

[3] Ulrich Bauer. Ripser: efficient computation of vietoris–rips persistence barcodes. *Journal of Applied and Computational Topology*, 5(3):391–423, June 2021.

[4] Mickaël Buchet, Yasuaki Hiraoka, and Ippei Obayashi. *Persistent Homology and Materials Informatics*, pages 75–95. Springer Singapore, Singapore, 2018.

[5] Anuraag Bukkuri, Noemi Andor, and Isabel K. Darcy. Applications of topological data analysis in oncology. *Frontiers in Artificial Intelligence*, 4:659037, 2021.

[6] Nathan Carter. *Visual Group Theory*. Mathematical Association of America, 2009.

[7] Joseph Minhow Chan, Gunnar Carlsson, and Raul Rabadan. Topology of viral evolution. *Proceedings of the National Academy of Sciences*, 110(46):18566–18571, 2013.

[8] Dong Chen, Bingxu Wang, Shunning Li, Wentao Zhang, Kai Yang, Yongli Song, Guo-Wei Wei, and Feng Pan. Superionic ionic conductor discovery via multiscale topological learning, 2024.

[9] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, FOCS '00, page 454, USA, 2000. IEEE Computer Society.

[10] Herbert Edelsbrunner and John Harer. *Computational Topology - an Introduction*. American Mathematical Society, 2010.

[11] Robert Ghrist and Abubakr Muhammad. Coverage and hole-detection in sensor networks via homology. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, IPSN '05, page 34–es. IEEE Press, 2005.

[12] Marian Gidea, Daniel Goldsmith, Yuri Katz, Pablo Roldan, and Yonah Shmalo. Topological recognition of critical transitions in time series of cryptocurrencies, 2018.

[13] Marian Gidea and Yuri Katz. Topological data analysis of financial time series: Landscapes of crashes. *Physica A: Statistical Mechanics and its Applications*, 491:820–834, February 2018.

[14] Ecaterina Guritanu, Enrico Barbierato, and Alice Gatti. Topological machine learning for financial crisis detection: Early warning signals from persistent homology. *Computers*, 14(10), 2025.

[15] Thomas W. Judson. *Abstract Algebra: Theory and Applications*. Self-published, annual edition 2022 edition, 2022. Open source textbook, GNU Free Documentation License.

[16] Aurelie Jodelle Kemme and Collins Amburo Agyingi. Persistent homology: A pedagogical introduction with biological applications, 2025.

[17] Serge Lang. *Algebra*. Graduate Texts in Mathematics. Springer, New York, NY, 3 edition, 2002.

[18] Byeoksong Lee, Mahnmin Choi, Jibin Shin, Hyunwook Ha, Doeun Shim, Sohee Jeong, and Joongoo Kang. Topological machine learning unveils hidden reaction pathways in nanocrystal synthesis. *Journal of the American Chemical Society*, 0(0):null, 0. PMID: 41313665.

[19] Daniel Leykam and Dimitris G. Angelakis. Topological data analysis and machine learning. *Advances in Physics: X*, 8(1), April 2023.

[20] James R. Munkres. *Elements of Algebraic Topology*. Addison Wesley Publishing Company, 1984.

[21] Monica Nicolau, Arnold J. Levine, and Gunnar E. Carlsson. Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proceedings of the National Academy of Sciences*, 108:7265 – 7270, 2011.

[22] Nina Otter, Mason A Porter, Ulrike Tillmann, Peter Grindrod, and Heather A Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science*, 6(1), August 2017.

[23] Jose Perea and John Harer. Sliding windows and persistence: An application of topological methods to signal analysis, 2013.

[24] Jose A. Perea, Andrea Deckard, Susan B. Haase, and John Harer. SW1PerS: Sliding windows and 1-persistence scoring; discovering periodicity in gene expression time series data. *BMC Bioinformatics*, 16:257, 2015.

[25] Chi Seng Pun, Kelin Xia, and Si Xian Lee. Persistent-homology-based machine learning and its applications – a survey, 2018.

[26] Zhe Su, Xiang Liu, Layal Bou Hamdan, Vasileios Maroulas, Jie Wu, Gunnar Carlsson, and Guo-Wei Wei. Topological data analysis and topological deep learning beyond persistent homology – a review, 2025.

[27] JunJie Wee and Jian Jiang. A review of topological data analysis and topological deep learning in molecular sciences. *Journal of Chemical Information and Modeling*, 65(23):12691–12706, 2025. PMID: 41235667.

[28] Ali Zia, Abdelwahed Khamis, James Nichols, Usman Bashir Tayab, Zeeshan Hayder, Vivien Rolland, Eric Stone, and Lars Petersson. Topological deep learning: a review of an emerging paradigm. *Artificial Intelligence Review*, 57(4), February 2024.

[29] Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG '04, page 347–356, New York, NY, USA, 2004. Association for Computing Machinery.