Spring 2022

Does Bias Have Shape? An Examination of the Feasibility of Algorithmic Detection of Unfair Bias Using Topological Data Analysis

Ansel Steven Tessier
Bard College

Does Bias Have Shape? An Examination of the Feasibility of Algorithmic Detection of Unfair Bias Using Topological Data Analysis

A Senior Project submitted to
The Division of Science, Mathematics, and Computing
of
Bard College

by
Ansel Tessier

Annandale-on-Hudson, New York
May, 2022

Abstract

Artificial intelligence and machine learning systems are becoming ever more prevalent; at every turn these systems are asked to make decisions that have lasting impacts on peoples' lives. It is becoming increasingly important that we ensure these systems are making fair and equitable decisions. For decades we have been aware of biased and unfair decision making in many sectors of society. In recent years a growing body of evidence suggests these biases are being captured in data that are then used to build artificial intelligence and machine learning systems, which themselves perpetuate these biases. The question is then, can we detect these biases in the data before it is used to create these systems? In this paper we will be exploring the feasibility and effectiveness of using a technique from topological data analysis to detect unfair bias in a criminal sentencing dataset.

Contents

Dedication

For those who truly believe mathematics, science, and technology should work for the betterment of all, and are willing to demand that they do so.

Acknowledgments

I would like to thank my advisors Dr. Leverson and Dr. McGrail for their unending support in this senior project. I would also like to thank all the classmates that I have had the privilege of learning alongside for the past four years; those who encouraged me to try harder, and showed me the joy of learning for knowledge's sake. As well as the professors of the Mathematics and Computer Science departments who have piqued my curiosity at every turn, and shown me how much there is to know. Finally I would like to acknowledge the inexhaustible love from my family that has never failed to push me forward.

1
Introduction

Artificial intelligence (AI) and Machine Learning (ML) are often championed as some of the most significant technological innovations of the twenty-first century. With ever increasing pools of computing power and a myriad of advanced algorithms, we have both the capacity and motivation to begin offloading significant levels of decision making to computational systems.

Tools from the field of artificial intelligence have already become indispensable across industry and government. (For a brief primer on AI, see the appendix.) There is an implicit logic in the justification of the use of these AI systems. The thinking goes, algorithms and AI systems are mathematical constructs, and as such they are not susceptible to the whims and biases that are so prevalent in human decision making, so they have the capability to be neutral arbiters. However, this reasoning is flawed. In designing and training AI and ML systems, developers will often unintentionally leave the fingerprints of human biases in these systems. To train these systems, developers gather large collections of data that are then used to teach the AI or ML system to determine relevant patterns for the task at hand. This is where the fallacy of AI comes into play. While the AI or ML system will usually find relevant patterns in the data, the system is not aware of the larger context in which the data was collected. As such it is unable to differentiate between true patterns in the data and artifacts introduced due to human biases.

In her 2016 book, *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*, Cathy O'Neil describes how the myths surrounding AI allow its use to go unchecked, and the subsequent dangers this can bring. "... if it becomes clear that automated systems are screwing up on an embarrassing and systematic basis, programmers will go back in and tweak the algorithms. But for the most part, the programs deliver unflinching verdicts, and the human beings employing them can only shrug" [2].

We often celebrate the ability of ML systems to analyze patterns that are often too subtle for humans to pick out, and make predictions and decisions based on these patterns. While this is a powerful tool, it is one that requires oversight. One of the ways in which we can start to ensure that AI and ML systems do not perpetuate and exacerbate the biases held by people is to attempt to detect such biases in datasets before they are used to train these systems.

Given that AI and ML seem to be here to stay, how can we ensure that their use does not further harm disenfranchised and marginalized groups and individuals? One answer is to more thoroughly screen the data that is used to build AI. The terms artificial intelligence and machine learning often go hand in hand with the phrase "big data," which generally refers to the practice of gathering large quantities of data on which to train AI and ML systems. As large datasets are at the core of these systems, there is good reason to turn to the field of topological data analysis for assistance in these issues. Topological data analysis is concerned with the macro scale of a dataset, and is primarily concerned with questions like "what is the underlying shape of this data?". The goal of this work is to determine if the presence of systemic and unfair biases in a dataset is represented in its fundamental underlying topology.

1.1 Defining Discrimination & Bias

Much of the power that predictive models and machine-learning algorithms leverage comes from finding and applying patterns found in historical data. The issue here is that while these systems make unbiased decisions *relative to the data*, there is no guarantee that historical precedent does not contain bias or discriminatory decisions [5]. Machine-learning models are built with the

assumption that the data they will be trained on aptly represents a population, this is not always the case.

In the context of machine learning, non-discrimination can be defined as: (1) people that are similar in terms of non-protected characteristics will receive similar predictions, and (2) differences in predictions across groups of people can only be as large as justified by non-protected characteristics [5]. In this context a protected characteristic is an attribute of a dataset that historically has been the defining factor in unfair treatment.

The problem with the above definition is that it falls to the developers of a system to decide on protected and non-protected characteristics. For example, using geographical location as part of a decision for loan approval, if an area has historically been at an economic disadvantage and people from that area have historically defaulted on loans due to discriminatory practices based on socioeconomic status or race, then using this historical precedent will only serve to exacerbate existing disparities. That is, protected characteristics may be hidden behind unprotected characteristics

To remedy the issue of choosing protected and non-protected characteristics outlined above, the emerging fields of discrimination-aware machine learning and data mining assume that some regulatory body will define these characteristics. So given a set of protected characteristics and groups, the goal is to use them to formulate constraints on ML algorithms to avoid algorithmic discrimination. [5]

Despite a number of discrimination-aware ML models and performance metrics being developed in the past few years, researchers have yet to reach a consensus on how to define a fair predictive model. This makes it difficult to compare the relative effectiveness of various approaches to this problem [5].

The 2015 paper [5] describe several discrimination measures that have been used to analyze the discriminatory decision making of ML systems. These measures can be defined in one of four categories: statistical tests, absolute measures, conditional measures, and structural measures. In statistical testing, discrimination is measured by the results of classical statistical tests, where

the null hypothesis assumes there is no significant difference in treatment between the general population and protected groups. Absolute measure tests seek to capture the magnitude of difference between two groups of people, differentiated by a protected attribute. Given multiple groups of people differentiated by a protected attribute, conditional measure seeks to tell how much of differences in treatment between the groups can be explained by other factors, outside of the protected attribute. Finally, structural measures look to see if each individual in a dataset has been discriminated against.

In this paper, we are concerned with ascertaining the presence of bias in data. This prompts the question; can we define a notion of bias that is quantifiable? This may seem like a trivial question, simply a matter of finding some underlying metric to capture; however giving a rigorous definition for conceptual ideas as nebulous as "bias" and "fairness" that precisely capture the colloquial meaning of these words is a task that has proven quite difficult. In this work we seek to interrogate the hypothesis that bias is present if the underlying topology of a dataset differs significantly when partitioned along a protected attribute.

1.2 Preprocessing

The idea of data preprocessing is common in the fields of data science and machine learning. Ultimately, every dataset comes from collecting information from the real world, a messy and imperfect practice. Data collection is an inherently noisy process due to a number of factors from missing data to human error. In light of this, data preprocessing is a way to clean up this messy data so that it will fit nicely into an algorithm. Preprocessing is a vast collection of techniques and only a handful are ever used on a given dataset. Some of the more common techniques include reformatting data so that all the same information is included, but in a way that is acceptable to an algorithm. Another is normalizing the data, so that each attribute in the dataset has it's original distribution, but all attributes are measured on the same scale. Yet another common technique is to strip each data point that contains a null value (null values usually corresponds to a value that was not collected). Preprocessing can also include

preliminary tests on data to see what values have correlations to desired target values, and removing values from each data point that do not seem to have strong correlations in order to save on computational resources. Therefore, preprocessing is a powerful tool that allows for a given dataset to be used more effectively and in a wider variety of contexts.

Given a dataset that has some bias embedded into it, finding ways to remove the bias from the data before it is used in an AI system is advantageous. The 2011 paper [4] proposes four methods of filtering data in order to address the discrimination-aware classification problem. Once a dataset has been preprocessed in such a way, they claim that that data can be used to train a classifier such that it will not emulate the biases in the original data.

Another reason we consider preprocessing techniques in this paper, is to understand the limits of how one can alter a dataset so that it is better suited for training an AI or ML system. In the case that it is possible to show the existence of a systemic bias in a dataset, we can then ask the question; "can we preprocess this data to eliminate that bias?", a question that will first require us to be able to see such a bias in the dataset with tools from topological data analysis.

1.3 Outline

Before we are able to explore the effectiveness of topological data analysis in detecting bias, we first must understand the theoretical under-pining of these tools. In Chapter 2 we will review some basic ideas from the sub-field of mathematics known as topology, which is generally concerned with the study of shapes and spaces, without concern for geometric properties like distance and orientation. We will then, in Chapter 2.4, move onto a more in-depth look at a particular idea from topology called homology, which is concerned with notions of equivalence of objects based on the number of various dimensional "holes" that they contain. From there we step into the fields of computational topology and topological data analysis by defining one of the principle tools from topological data analysis, persistent homology. In Chapter 3.2 we will examine a tool from classical data analysis; bottleneck matching. This is a tool that we will use for comparing and interpreting the results of persistent homology. After defining the

theoretical concepts, we will quickly discuss the software and tools used for this paper. Finally in Chapter 4 we see these tools in action, and discuss their use in detecting biases in a set of criminal sentencing data.

2
Basic Topology and Homology

The goal of this chapter is to provide a working understanding of homology by giving some intuition, and by stating the formal definitions. In service building this understanding, we will first review some key basics of topology.

2.1 Basic Topology

Topology is one of the main fields of study in modern mathematics. It is the generalization of ideas from geometry, and is generally concerned with the study and classification of complex objects and spaces. In topology there is a notion of "continuous deformation" where one can think of objects being made out of infinitely stretchy rubber. If one is able to stretch one object into the shape of another without tearing it, or gluing parts of the object back onto itself, we consider these objects to be equivalent. The classic introductory example topologists will use to explain this concept is the following: consider a doughnut and a coffee mug (Figure 2.1.1). At first these may seem like completely distinct shapes. Indeed if we were concerned with geometric features like volume and concavity, they would be. However if we imagine these objects to be infinitely stretchy, we can think about squishing most of the mass of the doughnut to one side. Then we can imagine pressing a divot into the outside of the thicker part of the doughnut, and pinch the divot until a well forms. Hence we can form a coffee mug out of a doughnut

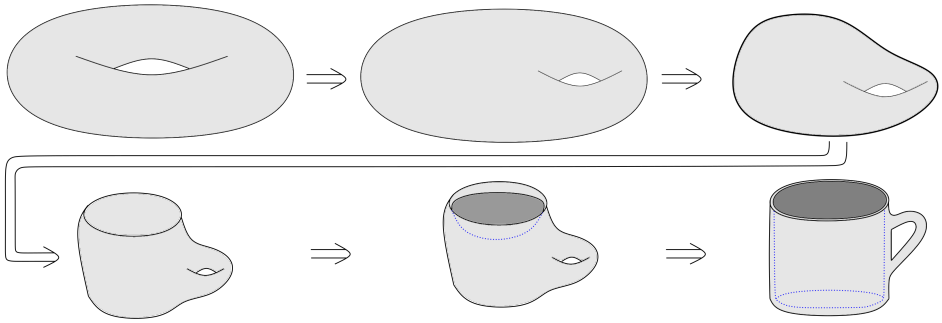


Figure 2.1.1: The doughnut and coffee mug example of continuous deformation.

without needing to tear or glue the material, so we say that these objects are equivalent up to homeomorphism.

Another way we can think about this notion of topological equivalence is to think about a graph, or a collection of nodes connected by a collection of edges. The graph is defined by which nodes are connected and which are not; but is not reliant on the spacial position of any nodes or edges. We can think about taking a graph and moving the nodes around on the page without removing any edges, the resulting graph will be the same. So we would say that the topology of the graph is unaffected by the location of the nodes and edges in space.

2.2 Homeomorphism and Topological Equivalence

When first building intuition about topology, thinking about objects as made of a magical, infinitely stretchy material is helpful for building an understanding of topological equivalence. However, it would be quite difficult to do rigorous mathematics with this notion alone. The rigorous way of understanding this idea is with the notions of continuous deformation and homeomorphism. Informally we can say two objects are equivalent if one can be continuously deformed into the other, or equivalently; there exists a homeomorphism from one object to the other.

Definition 2.2.1. A function $f : X \to Y$ where X and Y are topological spaces is said to be a *homeomorphism* if and only if f is a continuous function and its inverse, f^{-1} is also a continuous function.

Figures 2.1.1 and 2.2.1 are both examples of homeomorphisms. Figure 2.1.1 is a 3-dimensional example, while Figure 2.2.1 is a 2-dimensional example. In each case we can describe a function that maps points in the first object to points in the second, constrained by the conditions of a homeomorphism, however it is often helpful to have these visualizations in order to better conceptualize these functions.

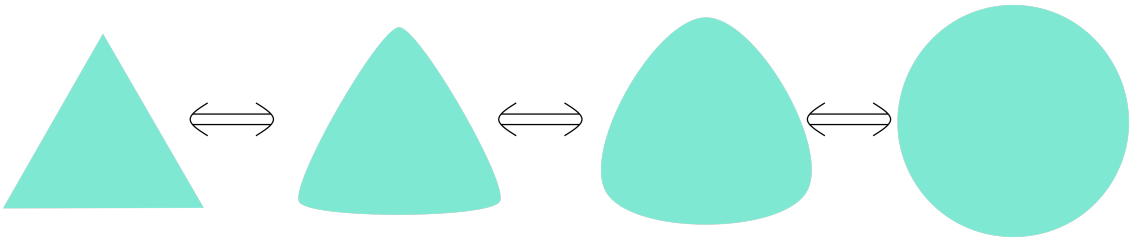


Figure 2.2.1: Visual example of a homeomorphism mapping a triangle to a circle.

2.3 Simplicial Complexes

An important idea from topology is that of simplicial complexes. Abstractly, a simplicial complex is a collection of connected elements where an element is either a point, a line, a triangle, a tetrahedron, or a higher dimensional simplex (which can be thought of as the analog for a triangle in greater than three dimensions). Simplicial complexes are important because they can be used to define a topological space. To rigorously define simplices, we will state a few definitions that will be helpful.

Definition 2.3.1. Let x be a point in a topological space, S. A *neighborhood* N_x is a subset of S, where for all points $p \in S$ the euclidean distance between x and p is less than some real value $d > 0$.

Definition 2.3.2. Let A be a subset of a topological space S. Any point $x \in S$ is a *boundary point* of A if for every neighborhood N_x of x, has a non empty intersection with A and a

non empty intersection with the complement of A. The set of all points in A that meet these conditions is the *boundary* of A.

Now we can formally define a simplicial complex as a finite collection of cells, also called complexes, where cells are defined by the following.

Definition 2.3.3. An $n-cell$ is a set whose interior region is homeomorphic to an n-dimensional disk, and whose boundary is a finite set of cells of lower dimension.

With this definition of cells, we can formally define a simplicial complex:

Definition 2.3.4. Let $K = \{K_1, K_2, ..., K_n\}$ be a set of simplices. We call K a simplicial complex if for every $k \in K$, every face of k is also in K, and the non-empty intersection of any two cells, k_1 and $k_2 \in K$, is a face of both k_1 and k_2.

Let us now consider an example of a simplicial complex. The left diagram in Figure 2.3.1 is a simplicial complex comprised of some 2-cells (yellow triangles), each of which is bounded by three lines, these are 1-cells. Each of the 1-cells is bounded by two points, these are 0-cells. The 0-cells are not bounded as they are single points and so have no boundary. We also have five 1-cells that are not faces of any 2-cells, this is not an issue, as we can have cells that are not faces of higher dimensional cells; so long as a cell is bounded by lower dimensional cells, and the intersection of any two cells is itself a cell, we have a valid complex. A non-example of a simplicial complex is shown in the right diagram of Figure 2.3.1. While each of the 2-cells is bounded by 1-cells, and each 1-cell is bounded by 0-cells, the intersection of the two 2-cells is not a cell in and of itself, thus this example is not a simplicial complex.

We consider simplicial complexes in part because they can represent other topological spaces, are often easier to conceptualize, and contain quite a bit of information about an object. It is important to note that a simplicial complex is itself a topological space. As such all the tools of topology apply to the simplicial complex, yet they are relatively easy to understand. By adding a notion of direction or orientation to each cell in a k-complex, we can define a notion of equivalent faces that can be thought of as "gluing" instructions, where faces are identified. In

Figure 2.3.1: (Left) An example of a simplicial complex comprised on 0,1, and 2-cells. (Right) A non example of a simplicial complex that is a set of 0, 1, and 2-cells.

such a way we are able to reconstruct a topological space from only a simplicial complex, and given a topological space, represent it as a simplicial complex.

We will see simplicial complexes applied in Chapter 3.1. There we will see how we can construct a special type of simplicial complexes, called the Vietoris-Rips complex, from a set of points sitting in some n-dimensional space. We can then use this particular complex to approximate the topological space that these points are sitting on. This will allow us to use topological ideas to analyze a set of discrete points.

2.4 Homology

The critical idea from topology that allows for the persistent homology algorithm to work is unsurprisingly something called homology. Homology is a complex concept that associates algebraic objects (usually groups) to topological spaces, but it can also tell us about features of a space, for example the connectedness of the space or how many "holes" an object has in some number of dimensions.

Homology is a concept that comes from algebraic topology. As is explained in [13], it serves as a reasonably powerful topological invariant and is thus useful for differentiating between spaces. We will not go in depth into invariants, but in this context it will suffice to say that an invariant is something that can be used to differentiate topological spaces. It is important to note that if two spaces have different invariants, they are guaranteed to be distinct, however having the

same invariant does not imply that the spaces are the same. In this case the invariant is a set of groups (where the cardinality of the set is the number of dimensions in which the topological space lives) called the homology groups, that are derived from imposing something called a simplicial complex chain on a simplicial complex representation of the space. This gives us a set of groups that depends only on the topological space, and not on the simplicial complex [9]. In [9] it is shown that it can be proven that these groups are in fact invariants, although such a proof is beyond the scope of this paper.

2.4.1 Chain Complexes

We will now defined all the components we need to define homology groups. This section is not intended to prove these ideas, or even give an in-depth description. It is simply an introduction of the basic terminology needed to state the definition of homology groups. For a more in depth explanation on the topic see [13].

It is not obvious how one would generate a group from a topological space. If you are just given a space, say a torus, how might we find a group that captures meaningful information about it? The answer lies in examining a simplicial complex representation of the space. The homology groups that we will define in section 2.4.2 are generated from chains of simplicial complexes. Again, this is an involved topic that we will not go into in too much depth, however it is important to have an understanding of how homology emerges.

As we saw in section 2.3, we can represent a topological space by a simplicial complex. Given this complex we can obtain an orientation from a direction on each k-cell that is inherited from the manifold that originally produced the simplicial complex. This lets us define a p-chain for some dimension p on this complex to be the weighted sum of a subset of the p-simplices, or formally:

Definition 2.4.1. A *p-chain* of a directed simplicial complex K, is the sum $c = \sum a_i \sigma_i$, where $a_i \in \mathbb{Z}$ are coefficients, and $\sigma_i \in K$ are the p-simplices in K. The p-chains form the p-chain group, denoted by $\boldsymbol{C}_p$.

We can think of p-chains as analogous to polynomials, in that they can be added together component-wise, and are values over which we can perform a notion of algebra [9]. This additivity will serve as the binary operation for the p-chain group. It is shown in [9] that such a group is abelian.

Note that there can be many chain groups formed from a single directed complex. We would like to be able to relate these groups in order to build the homology groups. We do this by introducing a notion of boundary. In this context we define the boundary as follows.

Definition 2.4.2. Given any p-chain c, the *boundary* of c is the sum of the $p-1$ dimensional faces of c, this is itself a $(p-1)$-chain, denoted $\partial_p c$.

Taking the boundary of a p-chain results in a $(p-1)$-chain. This should be intuitive as the faces of the p-chain, are themselves $(p-1)$ dimensional complexes, hence the boundary maps p-chains to $(p-1)$-chains. It can be shown that addition is commutative over the boundary mapping, meaning that taking the boundary is a homomorphism, we denote these boundary maps as ∂_p [9].

We can now define the chain complex.

Definition 2.4.3. A *chain complex* is a sequence of chain groups $\cdots \to C_{n+1} \to C_n \to C_{n-1} \to \cdots$ related by functions $b_1, b_2, b_3, \cdots$ where each function, $b_i : C_{n+1} \to C_n$, is a boundary homomorphism.

We will use the concept of boundary to define another important component we will need to define homology groups; p-cycles.

Definition 2.4.4. A *p-cycle* c on a directed chain complex K is a p-chain with empty boundary, $\partial c = 0$. The set of all p-cycles for a dimension p is denoted $\boldsymbol{Z}_p$

It can be shown that $\boldsymbol{Z}_p$ is actually a subgroup of the p-chain group. The other special type of p-chain we need to consider is the p-boundary. A p-boundary is a p-chain that itself is the boundary of a $(p+1)$-chain.

Definition 2.4.5. A p-boundary $b \in \boldsymbol{C}_p$ on a directed complex K is a p-chain such that there exists a $(p+1)$-chain, $c \in \boldsymbol{C}_{p+1}$ where $\partial c = b$. The p-boundaries form a subgroup of the chain group, known as the *boundary group*, denoted $\boldsymbol{B}_p$.

Now we have all the necessary definitions, we are prepared to approach homology groups.

2.4.2 Homology and Homology Groups

Given that we have a directed simplicial complex, a chain group defined over that complex, and the boundary and cycle subgroups, we can define the homology groups on that complex.

We note that the p-th boundary group $\boldsymbol{B}_p$ is a normal subgroup of the p-th cycle group $\boldsymbol{Z}_p$

Definition 2.4.6 (Homology group)**.** For a given directed complex K and some dimension p, the p-th homology group of K is the p-th cycle group modulo the p-th boundary group, $\boldsymbol{H}_p = \boldsymbol{Z}_p / \boldsymbol{B}_p$.

Another important aspect of homology groups are their Betti numbers. Informally this is the number that corresponds with the n-dimensional holes in a space for each dimension the space lives in. For example the 0-th Betti number of a topological space is the number of connected components in the space, while the 1-st Betti number is the number of loops present. The 3-rd Betti number is the number of 3-dimensional voids enclosed by the topological space, we can think of these as any pockets in the space that could be filled up with water without leaking it out.

Definition 2.4.7. Given a topological space S, and a natural number k, the *k-th Betti number* $\beta_k(S)$ is the rank of the k-th homology group.

Remark 2.4.8. Here the rank of a homology group is the number of linearly independent generators.

Now that we have developed an understanding of some basic topological principles, and a working intuition on homology groups, we can explore the topic at the core of this paper; persistent homology.

3
Persistent Homology, Bottleneck Matching, and Tools

3.1 Persistent Homology

We will now turn to the field of computational topology. One of the main tools from this area of study is persistent homology; an algorithm for computing how topological features persist as we examine a point cloud at different resolutions and scales. The topological features that are captured by persistent homology are, of course, homology groups.

Topological data analysis (TDA) is a relatively new field that began gaining traction in the early 1990's. It leverages ideas from algebraic topology, along with modern computational power to glean insights from data that cannot be easily seen with standard data analysis, and to give a new way to summarize and compare entire datasets without concern for individual datum. Topological data analysis has been used for a variety of tasks, from computational biology, to shape recognition, and has been suggested as a possible tool for studying bias in the context of detecting gerrymandering [6] and in examining financial bias [10]. The underlying assumption behind topological data analysis is that the topological features or the "shape" of a space constructed from a dataset will contain relevant information about the dataset as a whole, and the phenomena it represents. It seeks to give algorithmic tools to analyze that underlying topological structures and its features [11].

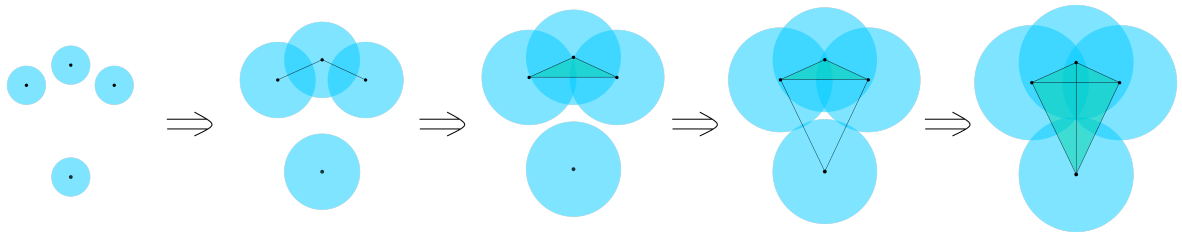


Figure 3.1.1: An example of how we build a simplicial complex by increasing the radius of 2-balls (discs) around the points as we go to the right.

Before we explore the details of persistent homology, we will take a brief look into some of its applications and the field of topological data analysis in general. One area in particular where TDA and persistent homology have seen some success is in the study of electoral redistricting. The 2020 paper [6], looks to classify redistricting plans as "partisan gerrymandering" (where lines of voting districts are drawn to favor victory in one political party) or not, by using persistent homology. Towards this end, they look at a set of many possible redistricting plans, called an ensemble, to form a backdrop on which to compare the redistricting plan in question.

3.1.1 Persistent Homology Intuition

As noted above, the primary tool of topological data analysis is a technique called persistent homology. This is an algorithm that keeps track of the various topological features of a dataset as we vary the notion of what it means for data points to be "close" to each other. Given a dataset in n-dimensional space, we can surround each point with a n-dimensional ball. For a given radius of these spheres, we can construct a n-simplex for every non-empty intersection of $n-1$ sphere. This will give us a cell complex, a construct that we can use to extract topological features. As we vary the radii of these spheres, we can see how long each topological feature persists. Using the tool of persistent homology, we can compare features of datasets as a whole, without concern for individual points. Figure 3.1.1 is a simple example of this process.

The topological features captured by persistent homology are the homology groups; informally one can think of the n-th homology group as capturing information about the n-dimensional voids in the topological space. The 0-dimensional homology group tells us the number of con-

nected components in the space. While 1-dimensional homology gives information on the circles present in the space. Similarly the 2-dimensional homology captures information on the voids encapsulated by the space. It is important to note here that we consider all these features up to homeomorphism.

The persistence, or "lifespan" of a homology group is considered to be all the radius values, α, of the expanding n-balls that result in a simplicial complex that will produce that homology group. Hence there is some α_0 where a homology group will first appear, and an α_d where it will disappear. This persistence is recorded in a persistence diagram. Figure 3.1.3 is an example of such a diagram. The persistence diagram is a two dimensional graph, where both axes represent the range of α values of the growing n-balls that define the complexes. The horizontal axis, labeled birth, is the α value for which a given feature appears (eg. a value of α that first makes a connected component appear); while the vertical axis, labeled death, is the α value for which a given feature disappears. Hence each point on the diagram shows the beginning and end of some feature (this is also why points will only ever appear above the $birth = death$ line. This may not seem like a lot of information, however when we consider the points of the persistence diagram in relation to each other, we realize there is quite a depth of information present in these figures.

3.1.2 Definition of Persistent Homology

At this point we have built an intuitive understanding for persistent homology. However our notion of expanding n dimensional balls and visualizing the induced topological features is far from a formal definition.

The first concept we will be defining in this section is the Vietoris-Rips Complex. The idea behind this type of complex, is to be able to bridge the gap between a set of discrete data points and a complete topological space.

Definition 3.1.1. Let X be a set of points in a metric space, and let $\alpha \geq 0$. The *Vietoris-Rips* complex $Rips_\alpha(X)$ is the set of simplices $\{x_0, ..., x_k\}$ where every $x \in X$ is a 0-simplex, and the

the distance, d_X, between every pair of vertices on every simplex x, $d_X(x_i, x_j) \leq \alpha$ for all pairs (i, j) [11].

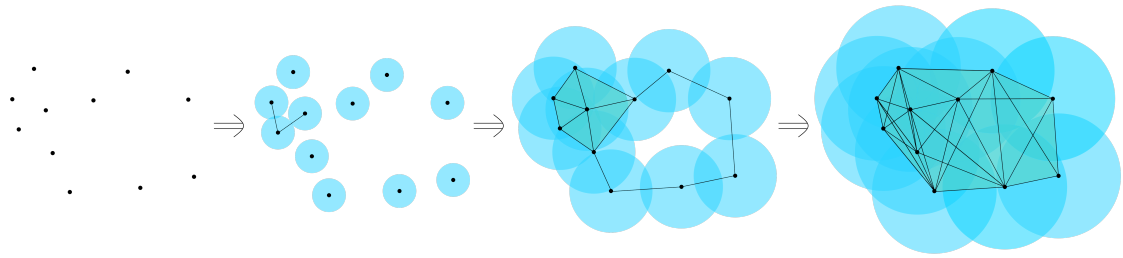


Figure 3.1.2: Example of the filtration generated by increasing the α value in a Vietoris-Rips complex.

The notion of expanding n-balls we use to build intuition, corresponds directly to varying the α value used to define the Vietoris-Rips complex. The resulting Vietoris-Rips complex for a given α can be thought of as a subcomplex of some other complex K, the subcomplexes we obtain by varying α will form a filtration, a concept that we will now define.

Given that we have a complex, K, we need to define a notion of filtration of a complex in order to give a definition for persistent homology. A filtration can be thought of as constructing K by sequentially adding small collections of simplices (this addition of simplices comes directly from the simplices that appear as we vary α in the Vietoris-Rips complex). We first define a monotonic function $f : K \to \mathbb{R}$, where in this context monotonic means that $f(\rho) \leq f(\sigma)$ whenever ρ is a face of σ. Because f is monotonic, it follows that $K(a) = f^{-1}(-\infty, a]$ is a subcomplex of K. If there are m simplices in K, then we get $n+1$ subcomplexes, where $n \leq m$. These subcomplexes are then arranged in ascending order, giving us a filtration, denoted

$\emptyset = K_0 \subseteq K_1 \subseteq ... \subseteq K_N = K$.

We now wish to relate through homomorphisms the homology groups on this sequence of subcomplexes in order to see the topological evolution as more simplices are added to the complex. For every i, j where $i \leq j$, there exists a homomorphism from the homology groups of K_i to those of K_j. As is explored in [9], These homomorphisms are called $f_p^{i,j} : H_p(K_i) \to H_p(K_j)$. Thus the filtration can be represented as a sequence of homology groups related through homo-

morphisms. As we go through this sequence, we record when new homology groups appear, and when they disappear and become trivial. Figure 3.1.2 is a visualization of such a filtration over a small set of 2-dimensional points.

Definition 3.1.2. The *p-th persistent homology groups* are the images of the homeomorphisms induced by inclusion, denoted $\boldsymbol{H}_p^{i,j} = \text{im}(f_p^{i,j}) = \boldsymbol{Z}_p(K_i)/\boldsymbol{B}_p(K_j) \cap \boldsymbol{Z}_p(K_i)$ for $0 \leq i \leq j \leq n$. The associated p-th persistent Betti numbers, β_p, are the ranks of these groups.

While the persistent homology groups are themselves homology groups, they are defined in this way to be more flexible, and to allow us to capture more information from the original data. This is necessary because of the limited topological information that is innate in a collection of points.

3.1.3 Algorithmic Implementation and Run-time of Persistent Homology

It is perhaps surprising, but persistent homology is polynomial-time reducible to Gaussian elimination (matrix reduction). This reduction is done by first ordering the simplices of the complex $\sigma_1, \sigma_2, \sigma_3 ... \sigma_m$ such that $i < j$ is implied by $f(\sigma_i) < f(\sigma_j)$. This ordering is valid due to the fact that f is monotone. We then use this ordering to set up an $m \times m$ matrix, ∂ of boundaries of simplices, where $\partial[i, j] = 1$ if σ_i is a co-dimension one face of σ_j, and $\partial[i, j] = 0$ otherwise [9]. This is sufficient to populate the matrix. Gaussian elimination is a well known algorithm, and is computable in $O(n^3)$ time, meaning that after a linear time reduction, the persistent homology groups are computable in at least cubic time. We can then read off the rank of the homology groups by counting the zero columns corresponding to p-simplices, these ranks correspond the Betti numbers of the persistent homology groups [9].

Remark 3.1.3. The rank of a matrix A is the dimension of the vector space generated by its columns.

3.1.4 Results of Persistent Homology

The persistent homology algorithm is a powerful tool for summarizing large amounts of information pertaining to the underlying topological features of a dataset. In order to understand this summary, we need to be able to interpret the resulting output. As mentioned before, this is known as a persistence diagram. Figure 3.1.3 is an example of such a diagram, a scatter plot with multiple colors of points, all above the main diagonal. As stated above, both axes of these diagrams represent the radii of the n-balls growing around each data point. The objects that are spawning and then die are the homology groups, or more concretely the topological features. Each point on the plot is colored according to the type of feature it represents (the dimension of the topological feature), with its horizontal position representing the n-balls radius for which that feature first appeared, and the vertical position representing the last radius for which it existed. We can also see how long a feature persists by its distance from the diagonal; if a point is close to the diagonal it vanished quickly after it first appeared. If a point is far off the diagonal, then its death was long after its birth, meaning it persisted and is likely a feature of the space the original data was sampled from. Hence the persistence diagram serves as a compact summary of all the topological information captured by persistent homology.

Let us now put our theoretical understanding to work with a simple example. Figure 3.1.4 is an example of a filtration for which we will compute persistent homology. Each simplex is labeled 1 through 7, we will now define the simplicial complex shown in part "d" of Figure 3.1.4 to be K. The simplicial complex K is comprised of simplices 1 through 7. The 0-simplices; $1, 2$ and 3 are our initial data points, with simplices 4 through 7 being introduced as we increase the radius of the discs. We can define K_i to be the subcomplex of K where K_i is the subcomplex obtained after adding simplex i ($i \in \{1, 2, 3, 4, 5, 6, 7\}$) to K_{i-1}. This gives us the filtration $\emptyset = K_0 \subseteq K_1 \subseteq K_2 \subseteq K_3 \subseteq K_4 \subseteq K_5 \subseteq K_6 \subseteq K_7 \subseteq= K$, where $K_i = \{1, ..., i\}$. In this example we will only look at the 0-th and 1-st persistent Betti numbers, they can be thought of as the number of connected components and loops respectively. We can simply read the persistent Betti numbers from each simplicial complex in the filtration, giving us the rank of the persistent

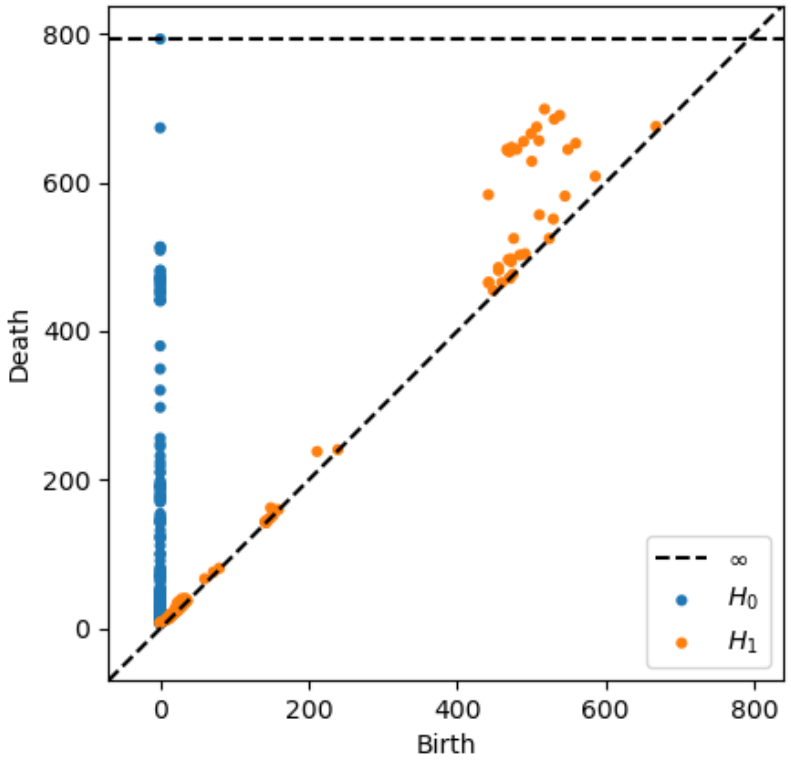


Figure 3.1.3: Example output of persistent homology. This persistence diagram was generated from a random subset of the data used in this research.

homology groups at each point in the filtration. We can refer to the radius of the discs when each new simplex i was added to the filtration as α_i, thus we can see what the persistent Betti numbers are at "times" α_i for each $i \in \{1, 2, 3, 4, 5, 6, 7\}$. From table 3.1.1, we can see that the number of connected components starts at 1 when the first simplex is added, goes up to 3 as simplices 2 and 3 are added, goes back down to 2 as simplices 2 and 3 are connected by simplex 4, then goes back down to 1 as simplices 1 and 2 are connected by simplex 5. Similarly we can see that the number of rings is 0 until simplex 6 is added, then drops back to 0 as soon as simplex 7 is added.

While this approach to persistent homology does not explicitly calculate the homology groups, it yields the persistent Betti numbers that contain the relevant information about the topological features of the space.

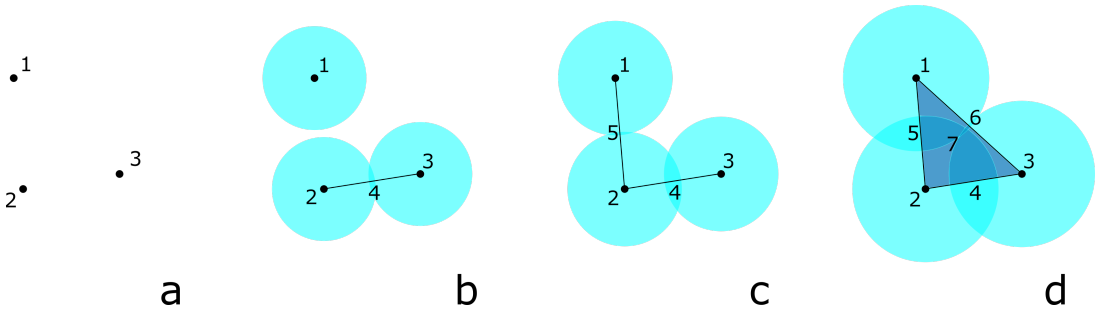


Figure 3.1.4: Example of a simplicial complex filtration.

α_i	β_0	β_1
α_1	1	0
α_2	2	0
α_3	3	0
α_4	2	0
α_5	1	0
α_6	1	1
α_7	1	0

Table 3.1.1: Table of persistent Betti numbers for Figure 3.1.4

3.2 Bottleneck Matching

In section 3.1.4 we built an understanding for interpreting the persistence diagrams that are produced by persistent homology. While understanding any one of these diagrams can provide much insight, the question then becomes; how can we compare these diagrams? How can we tell if they are summarizing a similar or distinct underlying topological space? There are several algorithms that have been designed to address just this problem, including; the modified Gromov–Hausdorff distance, sliced Wasserstein kernel, heat kernel, and bottleneck distance. For the purposes of this paper, we will be using bottleneck matching to compare persistence diagrams, as it is an intuitive approach to comparing any two diagrams, and is computable in $O(n^{1.5}log(n))$ time [7], something that is necessary for this work due to the large quantity of diagrams that must be compared. Simply put, the bottleneck distance between two diagrams

is the largest distance between any associated pairs of points, one from each diagram, in an optimal matching of points.

3.2.1 Bottleneck Matching Intuition

In the most basic case, we can think of bottleneck matching as pairing up points from some diagram A, to the closest corresponding point from a second diagram, B. Once we have these pairs, we can find all the distances between them, record the largest distance and define it to be the "bottleneck". We then consider this to be the distance between A and B. While this may seem like a simple idea, finding an optimal matching of the points from the two diagrams is anything but. We have to consider all possible matchings and see what the greatest distance is for a given matching, then minimize across all possible matchings. We also need to consider the case where the two diagrams that we are comparing have different numbers of points. To address the case where the cardinality of the two diagrams is different, we add all points on the diagonal to our consideration, if we see a point from one diagram that has no good corresponding point in the other, we match that outlying point to the nearest point on the diagonal.

We may also consider this problem through the lens of graph theory. If we consider the points in a diagram A and points in a diagram B we can consider their disjoint union $M = A \sqcup B$ to be the nodes of a bipartite graph. We then define the weighted edges of such a graph to be the distance between associations of pairs of points. We can now think of this problem as minimizing the maximal weights on all edges of the graph by considering all possible bipartite graphs, where the disjoint sets are defined to be A and B. While we will formally define bottleneck matching in terms of bijections from one diagram to another, it is this graph theoretic view of the problem that allows for a speedy algorithmic implementation, namely in $O(n^{1.5}log^2n)$ time [7].

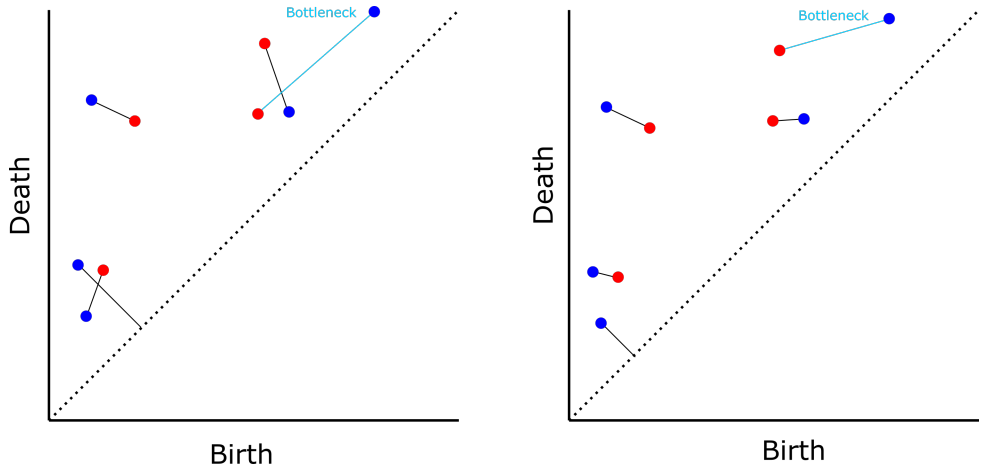


Figure 3.2.1: Here we see two possible matchings of points on a Red and Blue persistence diagram; one with high cost (left), and one with low cost (right).

3.2.2 Definition of Bottleneck Matching

The immediate goal of bottleneck matching is to take two diagrams and record their similarity or dissimilarity in a single scalar value, which is referred to as the bottleneck distance. To this end, we will formalize this concept by first defining the p-th persistence diagram as follows:

Definition 3.2.1. A *persistence diagram* of some dimension p, denoted H_p, is a finite multi-set of points union all points on the diagonal with infinite multiplicity;

$H_p = \{(x, y) | y \in [0, \infty) \text{ and } y > x\} \cup \{\cup_{i=1}^{\infty} \{(x, y) | x = y\}\}$.

Remark 3.2.2. Recall that the multiplicity of an element x in a multi-set is the number of times x appears in the multi-set.

Given two persistence diagrams A and B we wish to define a bijection between them, we will denote such a bijection $\eta : A \to B$. By defining a bijection we have a well defined association between pairs of points, as every point in A must be mapped to a unique point in B, and every point in B must have a unique point in A as its inverse image. We then record the supremum of all distances between pairs of points $x = (x_1, x_2) \in A$ and $\eta(x) = (y_1, y_2) \in B$ where the distance between x and $\eta(x)$ is defined to be $||x - \eta(x)||_\infty = max\{|x_1 - y_1|, |x_2 - y_2|\}$. This gives us a single cost value for a particular η.

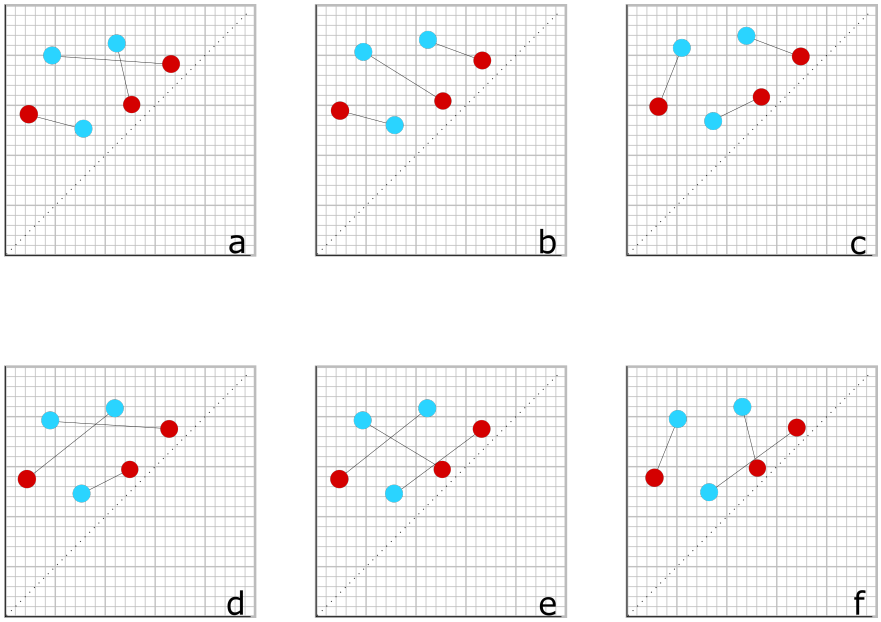


Figure 3.2.2: All possible mappings $\eta : A \to B$ where A is the set of red points, and B is the set of blue points.

We now wish to consider all possible bijections between our two diagrams A and B, and minimize the cost value of η. To do this we simply take the infimum of all η.

Hence the definition of the bottleneck distance follows directly from the values described above. We define the bottleneck distance as follows:

Definition 3.2.3. Let A and B be persistence diagrams. The *bottleneck distance* between A and B, denoted W_∞ is

$$W_\infty = \inf_{\eta: A \to B} \left(\sup_{x \in A} ||x - \eta(x)||_\infty \right)$$

.

Let us now look at a small example of bottleneck matching. Figure 3.2.2 shows us all the possible mappings from one persistence diagram (red) to another (blue). We will refer to the mappings described in each sub-figure as η_i where $i \in \{a, b, c, d, e, f\}$. First we must find the $\sup_{x \in A} ||x - \eta_i(x)||_\infty$ for each i. Let us start with η_a. We then see that $||x - \eta_a(x)||_\infty$ is $12, 5.5$ and 6 for each paring of $(x, \eta_a(x))$ respectively. Hence $\sup_{x \in A} ||x - \eta_a(x)||_\infty$ must be 12. We repeat this computation to obtain the values in table 3.2.1 .

$\eta_i(x)$	$\sup_{x\in A} \|\|x - \eta_i(x)\|\|_\infty$
$\eta_a(x)$	12
$\eta_b(x)$	8
$\eta_c(x)$	5.5
$\eta_d(x)$	12
$\eta_e(x)$	9
$\eta_f(x)$	8.5

Table 3.2.1: Table of supremum distance values for all bijections between the sets in Figure 3.2.2 .

Now we wish to find the $\inf_{\eta:A\to B}(\sup_{x\in A} ||x - \eta(x)||_\infty)$ which is simply $\inf(\{12, 8, 5.5, 12, 9.8.5\})$. Hence our bottleneck distance is $W = 5.5$.

Remark 3.2.4. We use inf and sup instead of maximum and minimum when defining bottleneck distances because homology groups can persist infinitely, so it is common to have points in a persistence diagram of the form (x, ∞).

3.3 Software and Tools

While the algorithms and techniques described in Chapters 3.1 and 3.2 are theoretically complicated, so too are their practical implementations. There have been several implementations of both persistent homology and bottleneck matching created over the past several years. As the goal of this work is not to modify or improve upon either of these algorithms, simply to assess their sensitivity to unfair bias in data, we do not write new implementations of these algorithms, but rather leverage powerful versions that are available as open source libraries.

3.3.1 Data

The data used in this paper is a dataset of criminal sentencing practices from the state of Minnesota, collected by the Minnesota Sentencing Guidelines Commission. The data was pulled from state records and made publicly available according to the statutory roll of the commission. The dataset was obtained through the Institute for the Quantitative Study of Inclusion, Diversity, and Equity (QSIDE).

In their 2021 paper, [12], Smith et. al. find significant racial disparities in the sentencing practices of federal judges. Given this, and given that our dataset is criminal sentencing data, we conclude that there is good reason to assume that unfair bias exists in this dataset.

The entire dataset consists of approximately 300,000 data points, with 81 attributes. Each data point represents an individual that has gone through the sentencing process. The attributes of the data cover a wide range of information, from the defendants' demographic information, to the statute violated, details about the offense, and sentencing length.

In order to use this data in a meaningful way, we used standard preprocessing techniques. First, we dropped all non-numerical attributes. While we could have used dummy variables to capture this information in a way that could be processed, this was unnecessary as there were only a handful of non-numerical attributes, and we did not think the additional dimensions would add meaningful insights. We then normalized the remaining attributes using z-score normalization. This was an imperative step as some attributes ranged in the thousands, while other attributes were binary. In an early attempt at running the persistent homology on the un-normalized data, we saw very messy results, as attributes like "sentencing year", which ranged from 2001 to 2019, completely drowned out attributes like race which only contained nominal values one through six. Hence normalizing the data allowed for each attribute to contribute equivalently to the computation.

3.3.2 RIPSER

The software package we use to compute persistent homology is called RIPSER, named for the Vietoris-Rips complexes that are crucial in computing the persistent homology. For this work we used the Python wrapper of the C++ library of the same name. RIPSER is a standard implementation of persistent homology, originally developed in C++ by Ulrich Bauer, and wrapped for Python by Tralie et. al. [3]. RIPSER stands out among other persistent homology libraries as it has taken into account all known computational speedups and combined them into one library. This allowed us to compute higher dimensional persistent homology on relatively large

datasets in a reasonable amount of time. Much research using persistent homology is limited by computational resources, so only computes 0-dimensional homology, however thanks to the speedups offered by RIPSER, we were able to compute $0, 1, 2$-dimensional homology easily.

3.3.3 Persim

For comparing the persistence diagrams that are output by RIPSER, we use a package called Persim. It is a library explicitly built for the comparison of persistence diagrams, and has a handful of useful algorithms. However for our purposes, we used it specifically for bottleneck matching. The implementation of bottleneck matching in Persim is based on the Hopcroft-Karp algorithm for finding maximal matchings on a bipartite graph. This algorithm uses the graph theory framing of the bottleneck matching problem discussed in section 3.2.1

3.3.4 Framework

For this work, all algorithms and data processing were run using the Python programming language. Due to its ease of development, vast library system, and status as one of the primary languages used in data science, it was the clear choice to use. Both RIPSER and Persim are available as Python packages allowing for simple integration into our data processing pipeline. Data preprocessing was done with the standard data processing tools from the Pandas library.

4
Applying Persistent Homology to Sentencing Data

4.1 Experimental Pipeline

The central question this paper seeks to answer is as follows: is persistent homology sensitive to unfair bias in data? For this to be the case, it would mean that partitioning the data along a particular protected attribute would yield subsets that had different underlying topologies. To approach this question we start with a dataset that we have good reason to believe contains unfair bias, and partition it along attributes that have historically been cause for discrimination. The hope being that in doing so we will either see distinct underlying spaces, providing evidence in support of our claim, or similar underlying spaces, providing evidence against.

In this chapter we will outline the experimental pipeline that we use in this work. It consists primarily of three steps; pre-processing data, applying persistent homology and, subsequently, bottleneck matching to the data, and finally analyzing and interpreting the results. The methodology for pre-processing the data is outlined in section 3.3.1 We apply this pipeline to three distinct partitions of the data. Firstly we partition based on the county where the sentencing was issued, we then partitioned the data by race and then by sex, as these are two of the biggest factors that lead to discrimination.

4.1.1 Use of Persistent Homology

Given a particular attribute that we wish to examine for bias, we partition the dataset along this attribute. This gives us n disjoint subsets, where n is the number of distinct values for this attribute. For each of the subsets in the partition, we apply the persistent homology algorithm supplied by the RIPSER package, and save the resulting persistence diagram for later processing. For each subset, we computed the 0th, 1st, and 2nd persistent homology groups. Due to the slower run-time of higher dimensional persistent homology, these were the only dimensions we computed. Our dataset contains over $300,000$ data points and as a result every subset in every partition had several thousand data points. However due to limitations in computational resources, and to the fact that some partitions contained just above 1000 data points, we ran all persistent homology computations with 1000 data points. This served to ensure that all computations were run with the same number of data points for consistency.

Despite the relative speed of the RIPSER implementation of persistent homology, it took about 12 hours to run due to the size of our data partitions, and the number of dimensions of homology groups we wanted to compute.

4.1.2 Use of Bottleneck Matching & Analysis of Bottleneck Distances

The next step is to compare our persistence diagrams. As discussed previously, we use the bottleneck matching algorithm provided in the Persim library. For any pair of persistence diagrams, we are well equipped to compare them. A direct comparison is simple for cases when we partition data along a binary value, such as sex, but what about when there are many distinct values in the partition? This would give us many distance values, the question then becomes what information can we see from a collection of distance values? To address this we take two approaches to analyzing bottleneck distances, depending on how many persistence diagrams are being considered. If there are six or fewer diagrams, we analyze them directly, by visually comparing the diagrams, and determine if the handful of distance values we get are large or

small as compared to the scale of the diagram. When we have more diagrams to compare, we calculate all possible distances, and examine their distribution.

To see how all the diagrams relate to each other we compute all $\binom{n}{2}$ distances, where n is the number of persistence diagrams we get from a given data partition. From an algorithmic design point of view, this approach is incredibly slow. One might think a work-around would be to select a single diagram to be a base line, and calculate the bottleneck distances of the remaining diagrams to the baseline. However due to the fact that bottleneck distances are not transitive knowing two diagrams distances to a third diagram does not give much information about the relative distance of the first two diagrams. In order to gain as much information about the relationships between the diagrams, we take this naive approach and calculate the distance between all distinct pairs of diagrams. Despite how quick the bottleneck matching algorithm is for comparing a pair of diagrams, computing all combinations took quite some time for experiments that partitioned on an attribute with more than a couple dozen distinct values. To speed up this process we employed a multiprocessing approach to gain a linear time speed up, where up to 12 distances were computed simultaneously. We were able to do this as each bottleneck distance computation is independent of any others.

We want to see how all the diagrams are related to each other. If all of the persistence diagrams are describing similar underlying topologies, then we expect the persistence diagrams to all be close to each other as measured by the bottleneck distance. If however they are describing two different underlying spaces, representative of biased and unbiased practices, then we expect there to be two distinct clusters of diagrams. In the first case, where they are all clustered near each other, we would expect to see a normal distribution of distance values. In the second case, we would expect a bimodal distribution, where we see lots of small distance values representing the distances between diagrams in a cluster, and lots of large distances between diagrams in one cluster versus diagrams in the other.

Our analysis thus becomes a search for bimodal versus unimodal distributions of distance values for the data partitioned by an attribute we suspect to be a determining factor of an unfair bias.

4.1.3 Experiments

Using the pipeline outlined above, we performed three experiments by partitioning the data along three attributes; county, race, and sex.

Given that our dataset relates to criminal sentencing practices in the state of Minnesota, we look to see if the data reveals discrepancies between different counties sentencing practices. If we expect that a subset of judges in the state will deliver biased sentencing, then we expect the diagrams representing the counties with those judges to have a high distance from the remainder of the diagrams. We also use this attribute to validate our approach to this sort of analysis. Because there are 87 distinct counties, we get $\binom{87}{2}$ distances, with this large number of distances we will be able to clearly see a unimodal or bimodal distribution if present.

The second experiment we run is partitioning the data by the 'race' attribute. There have been many reports like that done by Smith et. al. [12] that have empirically shown that criminal sentences are much harsher on people of color as opposed to their white counterparts. Given that this bias is so well documented, we expect to see vastly different persistence diagrams if it is the case that the bias results in different underlying topologies.

The third experiment we run is partitioning the data by the 'sex' attribute. We run this experiment for similar reasons as experiment two, a sex is historically a determinant of unfair bias. In addition, we wish to see how the persistence diagrams differ when we partition only on a binary value, as the dataset only provides 'male' and 'female' values.

4.2 Results

For this paper, we performed three experiments by partitioning the data by the attributes labeled County, Race, and Sex, and performing the analysis pipeline outlined in the previous section over each partition respectively.

4.2.1 Experiment 1: Partition by County

We first divided the data into collections by the county where each sentencing took place. Using this partition, there 87 subsets of data on which to perform persistent homology. By performing bottleneck matching on every pair of the resulting persistence diagrams, we get the distributions of bottleneck distances shown in Figure 4.2.1.

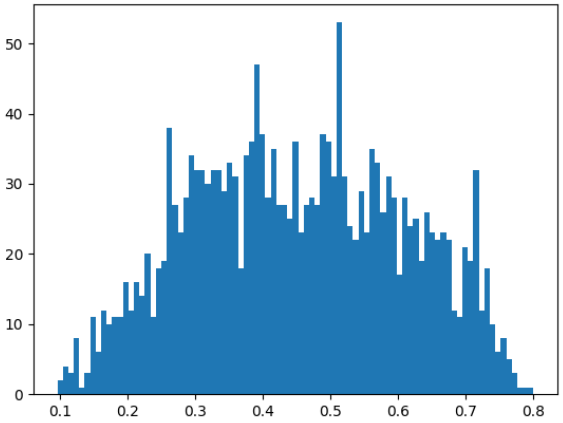
In Figure 4.2.1 we can clearly see unimodal distributions of bottleneck distances in each histogram plot. Based on the reasoning we outlined in section 4.1.2, the unimodality of these plots implies that we are unable to support the claim that an unfair bias results in significantly different underlying topologies.

While this is evidence the there is no real significant difference between the underlying topological spaces of this data partition, this is evidence that supports the idea that by looking at the modality of the bottleneck distance histograms, we can glean insight into how similar the subsets of a particular partition are. The fact that we see clear unimodal distributions, implies that all persistence diagrams are describing similar underlying topological spaces.

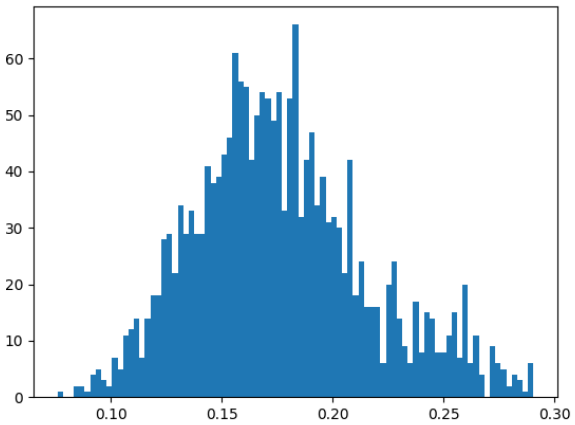
4.2.2 Experiment 2: Partition by Race

For the second experiment we partitioned the data by an attribute that we expect to show significant bias, and thus significantly different underlying topologies.

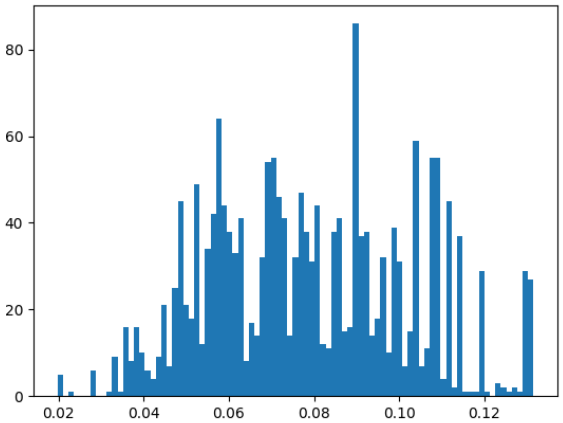
When performing the experiment with this partition, we get just 6 subsets, as the dataset only contains 6 distinct race labels. The analysis of bottleneck distances proceeds in the same manner as Experiment 1, we get the distributions shown in Figure 4.2.2.



(a) 0-dimensional homology persistence diagram comparisons

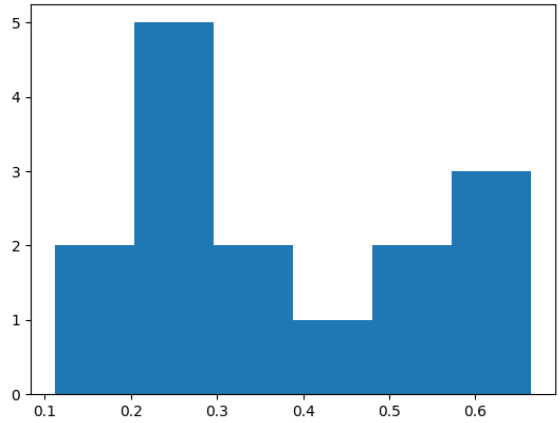


(b) 1-dimensional homology persistence diagram comparisons

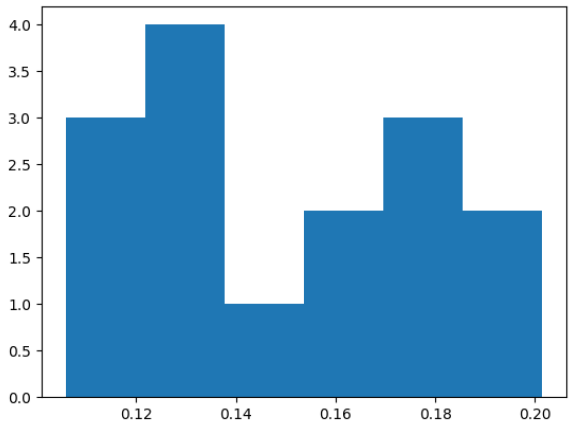


(c) 2-dimensional homology persistence diagram comparisons

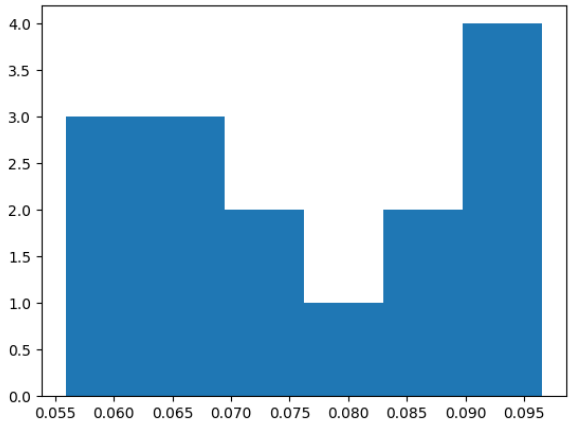
Figure 4.2.1: Histogram distribution of bottleneck distances between 0, 1, and 2-dimensional persistence diagrams of data partitioned by county. The horizontal axis is the bottleneck distance, and the vertical axis is the count.



(a) 0-dimensional homology persistence diagram comparisons



(b) 1-dimensional homology persistence diagram comparisons



(c) 2-dimensional homology persistence diagram comparisons

Figure 4.2.2: Histogram distribution of bottleneck distances between 0, 1, and 2-dimensional persistence diagrams of data partitioned by race. The horizontal axis is the bottleneck distance, and the vertical axis is the count.

The distributions of these bottleneck distances shown in 4.2.2 appear to be bimodal, however as there are only 15 distinct distance values, it is possible this is simply noise, so we examine the persistence diagrams in Figure 4.2.3 directly. Upon this examination we note that 5 of the 6 diagrams are quite similar, with the 6th being much more sparse. The fact that this single set in the partition is much more sparse is due to the fact that this value had much fewer data points than the others. This leads to a high distance between the first 5 distances and the last one, and low distance between the first 5. This accounts for the bimodality we see in the histogram. Because the bimodality we see is explained by noise, and the majority of the diagrams are quite close together, we must conclude that we do not see a difference in the underlying topologies when partitioning the data by the race attribute.

4.2.3 Experiment 3: Partition by Sex

For our third experiment, we partition our data by the sex attribute. Because this is a binary attribute we only get two persistence diagrams. To compare them we both visually examine the persistence diagrams, and look at the single distance value. If we examine the diagrams in Figure 4.2.4 we can see that they look quite similar. So we look at the absolute bottleneck distance values for each dimension, those values being: 0.1772, 0.1189, and 0.0818 for dimensions 0, 1, and 2 respectively. As these values are quite small, each being less than 5% of the 3.5 unit axis, we conclude that these diagrams are not significantly different from each other, meaning that they do not describe different underlying topologies.

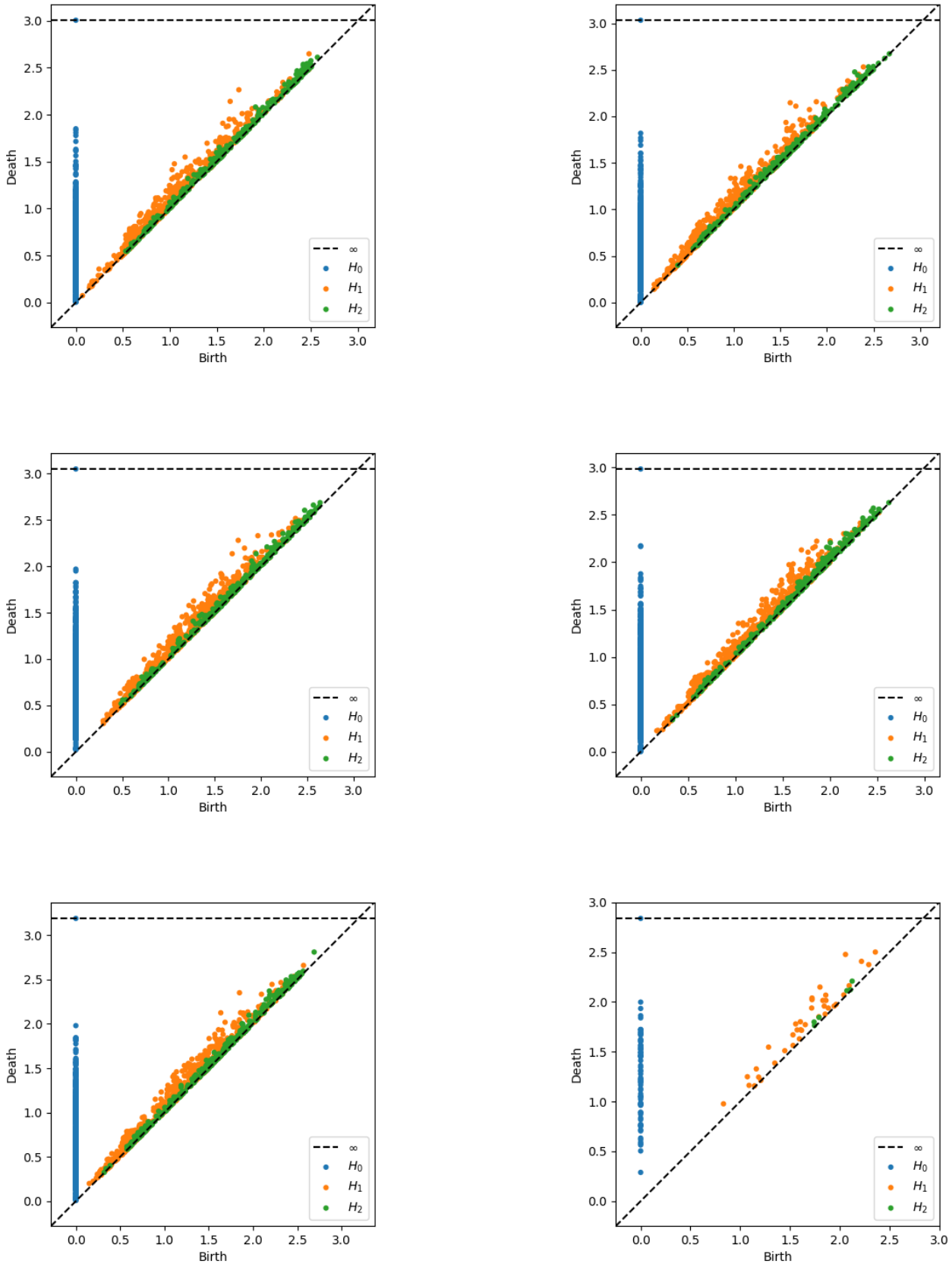


Figure 4.2.3: Persistence diagrams for each subsets partitioned by race.

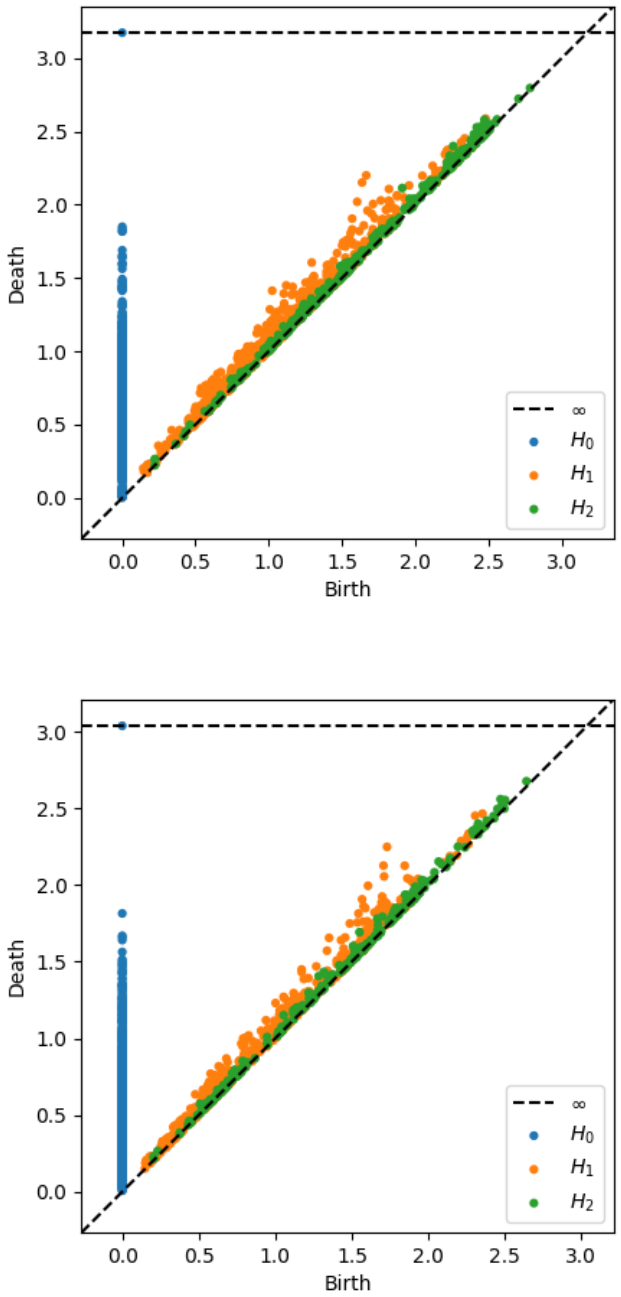


Figure 4.2.4: Persistence diagrams for each subset of the data partitioned by sex.

5
Future Directions

To more rigorously interrogate the question posed in this work there are a few steps that could be taken. Firstly before topological data analysis is used, analyzing a dataset with classical statistical tools to show a bias is present may be advantageous. While in this work we have good reason to suspect bias was present in our dataset, we did not see it directly through classical statistics.

Given that real world data is often messy, and statistical models might struggle to see an unfair bias even if one is present, another possible way to address the problem is to generate artificial data to perform topological data analysis techniques on. However there may be a potential problem with this approach. Real world data may have significant correlations between many if not all attributes, something that may not be simulated by randomly generated datasets.

One limitation of the approach we outlined in this work, is that we do not get a lot of information when partitioning data along an attribute with few distinct values. One way to address this is to partition along multiple attributes simultaneously. For example in our dataset we might partition along both race and county attributes.

Due to constraints on computational resources, we only performed persistent homology for 0, 1, and 2 dimensions. Future work could explore the effectiveness of using multiprocessing to compute the persistent homology, and using that speed up to apply persistent homology to all

dimensions up to the dimensionality of the data, as it is conceivable that discrepancies in the topological spaces only appear in higher dimensional homology groups.

Another avenue to explore is using alternate coefficient fields when running the persistent homology, the coefficient field corresponds to the scalar values used in defining the chains. By default the field $\mathbb{Z}/2\mathbb{Z}$ is used. The Persim documentation suggest that some features of a space are only seen when using alternate coefficient fields. They show that a notion of twisting can be captured by using the field $\mathbb{Z}/3\mathbb{Z}$. It may be illuminating to repeat the experimental pipeline outlined in this work any vary the coefficient field.

The final idea we will present as a possible area to explore in future work is using alternate metrics for persistence diagram comparison. While bottleneck matching is robust against noise, other metrics like q-Wasserstein distance are more sensitive to slight variations in the persistence diagrams. It may also be worthwile to look at metrics that map a persistence diagram to a continuous metric space so the entire collection can be compared together, rather than pairwise.

Over the course of our experimentation, we failed to find any evidence to support the claim that unfair bias in datasets is represented by different underlying topological spaces, and could be captured by persistent homology. However there are many avenues that could be taken in future works that would further examine this question and provide more definitive evidence.

Appendix A
Appendix

Artificial Intelligence is a sub-field of computer science that looks to develop computation systems that emulate the decision making abilities of sentient organisms. While there are several advanced approaches to AI that take inspiration directly from biology, many forms of AI are little more than statistical models. The ultimate goal of an AI system is for it to be able to make a decision when presented with a novel situation that is similar to situations that the model has seen before. One thing that the vast majority of AI techniques have in common, is "learning" how to make decisions based on patterns in large collections of data. The idea of "learning" in AI corresponds to detection of complicated patterns in a set of data called "training data" that a programmer will give to a model. Once an AI model has learned from a significant quantity of training data, the hope is that when given novel data that is in the same format as the training data, the AI model will be able to correctly make some decision based on it. One of the most powerful and widely used forms of AI is Machine Learning, an approach to AI that vaguely approximates the structure of a brain as a *neural network*. In Machine Learning, a neural network simulates neurons as nodes in a directed graph, where each node activates with some intensity based on the input from another node. Each node contains a function that modifies the input, and dictates the strength of the signal that the node will pass on to the next node. By systematically varying the parameters of each function in each node, the neural network can encode decision making abilities.

Bibliography

[1] Rachel K. E. Bellamy and Kuntal Dey and Michael Hind and Samuel C. Hoffman and Stephanie Houde and Kalapriya Kannan and Pranay Lohia and Jacquelyn Martino and Sameep Mehta and Aleksandra Mojsilovic and Seema Nagar and Karthikeyan Natesan Ramamurthy and John T. Richards and Diptikalyan Saha and Prasanna Sattigeri and Moninder Singh and Kush R. Varshney and Yunfeng Zhang, *AI Fairness 360: An Extensible Toolkit for Detecting, Understanding, and Mitigating Unwanted Algorithmic Bias*, arXiv preprint arXiv:1810.01943 **abs/1810.01943** (2018), available at `http://arxiv.org/abs/1810.01943`.

[2] Cathy O'Neil, *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*, Crown, New York, 2016.

[3] Christopher Tralie and Nathaniel Saul and Rann Bar-On, *Ripser.py: A Lean Persistent Homology Library for Python*, The Journal of Open Source Software **3** (Sep 2018), no. 29, 925, DOI 10.21105/joss.00925, available at `https://doi.org/10.21105/joss.00925`.

[4] Faisal Kamiran and Toon Calders, *Data preprocessing techniques for classification without discrimination*, Knowledge and information systems (3 December 2011), available at `https://link.springer.com/article/10.1007/s10115-011-0463-8`.

[5] Indre Zliobaite, *A survey on measuring indirect discrimination in machine learning*, arXiv preprint arXiv:1511.00148 (2015), available at `https://arxiv.org/abs/1511.00148`.

[6] Moon Duchin and Tom Needham and Thomas Weighill, *The (homological) persistence of gerrymandering*, posted on 2020, DOI 10.48550/ARXIV.2007.02390, available at `https://arxiv.org/abs/2007.02390`.

[7] Michael Kerber and Dmitriy Morozov and Arnur Nigmetov, *Geometry Helps to Compare Persistence Diagrams*, posted on 2016, DOI 10.48550/ARXIV.1606.03357, available at `https://arxiv.org/abs/1606.03357`.

[8] Sarit Agami, *Comparison of Persistence Diagrams*, Communications in Statistics-Simulation and Computation (2020), available at `https://arxiv.org/abs/2003.01352`.

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

[10] Ramya and Chander Srinivasan Ajay, *Understanding Bias in Datasets using Topological Data Analysis* (2019), available at `http://ceur-ws.org/Vol-2419/paper_9.pdf`.

[11] Frédéric and Michel Chazal Bertrand, *An introduction to Topological Data Analysis: fundamental and practical aspects for data scientists*, arXiv preprint arXiv:1710.04019 (2017), available at `https://arxiv.org/abs/1710.04019`.

[12] Christian M and Goldrosen Smith Nicholas and Ciocanel, *Racial Disparities in Criminal Sentencing Vary Considerably across Federal Judges* (2021), available at `osf.io/preprints/socarxiv/j2gbn`.

[13] Christine Kinsey, *Topology of Surfaces*, Springer-Verlag, 1993.