Enabling Multi-Scale Simplicial Complex Generation for `Mapper`

Matt Piekenbrock* and Derek Doran†

Abstract. `Mapper` is perhaps the most widely used algorithm for practical topological data analysis. The primary result of the algorithm is a graph depicting topological relationships and structures usable for comparing, visualizing, and analyzing high dimensional data. As an unsupervised, exploratory tool that may be used for multimodal and heterogeneous forms data, `Mapper` naturally relies on a number of parameters which explicitly control the quality of the resulting construction; one such parameter controls the entire relational component of the output graph, and offers little guidance or intuition on what values may provide 'better' results. In this effort, we provide a closed-form solution that allows the efficient computation of all possible realizations of this crucial parameter, and we discuss further extensions and theoretical advancements that may be made as a result of this development.

Key words. Mapper, Topology Theory, Unsupervised Learning

AMS subject classifications. 54H99, 54C05, 54J05

Introduction. The notion of *nearness* plays an important role in studying and understanding relationships in data. Within the realm of unsupervised machine learning, there is now a well-developed theory of how measures of nearness may be exploited for the purpose of modeling, understanding, or visualizing data, even if the data is sufficiently complex or higher dimensional. For example, in manifold learning, distance interpolation methods along manifold surfaces enable realistic, perceptual changes between images to be more semantically clear [14]. In clustering, distance measures are used to express the (dis)similarity between objects towards creating a meaningful representation suitable for classification, based on characteristics of the data itself [8]. Self-organizing maps (SOMs), often referred to in the context of artificial neural networks for unsupervised learning, have been used extensively for mapping data to lower-dimensional representations which preserve some degree of topological properties of the input space, allowing for robust feature representation and visualization [9]. Indeed, it is not difficult to find unsupervised models which exploit notions of nearness to describe something useful or interesting about the space the data, or functions of the data, were observed in.

Although there are many approaches in unsupervised learning, perhaps the most foundational field that unifies these various concepts of "nearness" is the field of *topology theory*. Topology theory explores deformations of maps and spaces, and has been referred to as the natural evolution of the notions of proximity and continuity [6]. The development of topological methods and algorithms that bring this theory into practice are emerging, and specific applications of topology in data mining tasks have been referred to as *topological data analyses* (TDA). Perhaps the leading tool for applying TDA in practice is `Mapper`, developed by Singh

*Department of Computer Science, Wright State University, Dayton, OH (piekenbrock.5@wright.edu, http://www.wright.com/~piekenbrock.5/).

†Department of Computer Science, Wright State University, Dayton, OH (derek.doran@wright.edu, http://knoesis.org/people/derek/).

et al. [13]. `Mapper` is a coordinatization framework capable of summarizing high-dimensional data sets, or functions of them, and has proven its worth as a useful tool for data visualization [7], genetic recombination characterization [4], and dimensionality reduction [13]. `Mapper` has also had much success in commercial settings.[1] In its simplest interpretation, `Mapper` is a topologically-motivated framework for reducing high dimensional data into an interpretable graph $G(V, E)$, where vertices correspond to clusters of data, and edges correspond to interactions (non-empty intersections) between such clusters. The goal of `Mapper` is to provide a simplified representation of high-dimensional data that not only *preserves* certain topological structures of interest, but is amenable to qualitative analysis and visualization. The goal of `Mapper` is *not* to achieve a complete representation of the data, but to capture significant areas of interest in an explainable way, enabling fruitful exploratory data analysis.

As an unsupervised framework capable of generalizing to heterogeneous formats of data, `Mapper` naturally requires a number of parameters. Provided a reference map f, a covering over a metric space $\{\mathcal{U}\}$, a clustering algorithm $\mathcal{C}$ using some distance metric d, and their corresponding parameters, `Mapper` deterministically generates G. `Mapper` admits a large parameter space, requiring several choices to be made by the user based on the goal of the analysis at hand. It's well known in both unsupervised and supervised learning that more often than not, a large parameter space implies a large solution space: changing the value of any particular parameter in may lead to a different set of solutions, which may in turn lead to inconsistent interpretations and analyses. This is true for the `Mapper` construction as well. Solutions which reduce this complexity would not only enable more robust analyses of the topological solutions produced by `Mapper`, but may indeed accelerate the adoption of TDA as a useful tool for data analysis compared to other forms of learning.

The focus of this effort is to analyze the *overlap* parameter of `Mapper`, which we describe further in Section 1, as it solely determines the connectivity, and thus the relational aspect, of the output graph G. That is, when the overlap parameter is set to 0, the graph is completely disconnected, and as the parameter increases, the graph generally becomes more connected. Unfortunately, there is little guidance on how to determine parameter values relevant to the data set being analyzed. If the overlap is too high, spurious connections may be shown in the output graph, conveying a relational structure that is misrepresentative of the topology. Conversely, if it is too low, subtle yet important connections may be missed.

Is there an optimal scale that best captures the topology of the data with respect to `Mapper`'s overlap parameter? To answer this, it's worth exploring how such scaling choices are performed for other types of TDA. Consider a well-known topological construction, such as the Vietoris-Rips Complex at scale ϵ, which is a simplicial complex obtained by adding n-dimensional simplices spanning the points of X if the pairwise intersection of $B(X_1, \epsilon) \cap B(X_2, \epsilon) \cap \cdots \cap B(X_n, \epsilon)$ is non-empty, where $B(X_i, \epsilon) = \{x \in X \mid d(X_i, x) \leq \epsilon\}$. As the sole parameter which determines connectivity, the ϵ parameter is analogous to the overlap parameter for `Mapper`, and as Ghrist argues: *"Despite being both computable and insightful, the homology of a complex associated to a point cloud at a particular ϵ is insufficient: it is a mistake to ask which value of ϵ is optimal"* [5]. We extend this assertion to `Mapper`: a fixed

[1]The Mapper framework actually lies at the core of the Ayasdi platform (https://www.ayasdi.com/solutions/)

scale parameter is insufficient if the task is study the structure in the context of *homology*. Rather, the solution is to look at the *continuum* of possible values towards understanding how the complex changes between ranges of parameter settings. Intuitively, substructures which seem to "persist" across parameter ranges forms a foundation for many approaches in Persistent Homology [12]. This solution extends to exploratory use-cases of `Mapper` as well—rather than choose an optimal scale to extract the "best" output of `Mapper`, a better solution may be to allow an analyst to explore the space of possible parameter values. By viewing multiple topological representations of the data at once, an analyst could compare and contrast solutions, derive insights, and extract richer knowledge on the topology of the data. The analyst could also further understand how the topological representation of the data (i.e. the output graph of `Mapper`) changes across other settings of interest.

This notion of studying topological constructions on a continuum of parameter ranges is not new. Methods which measure some notion of 'stability' or 'persistence' of a given real-valued function with respect to its domain have proven useful in both theoretical and practical instances of TDA [5]. In the context of the `Mapper` construction explicitly, theoretical extensions which analyze `Mapper` at multiple resolutions are emerging [2, 3]. Nonetheless, using such extensions to *performing TDA on empirical data* requires knowledge of how the `Mapper` construction changes with respect to ranges of parameter inputs, and what those parameter ranges are. Such types of analysis may only be enabled tractably, however, if the set of valid and useful values for the overlap parameter can be computed efficiently.

In this work, we derive a closed-form solution to compute the set of minimal parameter values that, at a given resolution, produce 'distinct' topological renderings under the `Mapper` algorithm, with the exact definition of 'distinct' to be defined later in Section 2. We further show that the solution can be computed very efficiently on a data set in $O(nd)$ time, where n is the size of the data set (i.e. the number of 'points' in the point cloud data), and d is the dimensionality of the space. We outline the benefits in tractability our approach enables in the context of 1-skeleton generation and in other application areas, and we note that our effort may also allow similar computational gains in generating the full simplicial complex.

In the next section, we discuss some prerequisite background material including preliminary notation that will be used throughout the paper. We then give a step-by-step overview of `Mapper` with examples. In Section 2, we derive a closed form solution for finding the minimal set of values for the overlap setting. In Section 3, examples are given and discussed in detail to further convey the motivation, significance, and utility of our solution. Finally, we discuss future applications and work related to `Mapper` in Section 5.

1. Background.

This section provides a succinct summary of the `Mapper` algorithm, leaving the reader to refer to the original paper by Singh *et. al* [13] for deeper background material. An even greater treatment of the topological ideas underlying `Mapper` can be found in Carlsson's exposition [1]. Still further, Merkulov et. al [10] provides much information on simplicial complexes, and Ghrist [6] may be consulted for a high-level overview of using topological constructs for applied topology.

We start by providing notation and some well-known topological constructs needed to describe `Mapper` and our corresponding derivation. Let S be a discrete set. An *abstract simplicial complex* is a collection X of finite subsets of S, closed under restriction. That is,

for each $\sigma \in X$, all subsets of σ are also in X. Each element $\sigma \in X$ is called a k-simplex whose faces are the simplices corresponding to all subsets of $\sigma \subset S$. The standard k-simplex is defined:

$$\Delta^k := \left\{ x \in [0,\infty)^{k+1} : \sum_{i=0}^{k} x_i = 1 \right\}$$

For simplicity, we may occasionally use the terms 'vertex', 'edge', 'triangle', and so on to describe the 0, 1, 2, and higher dimensional simplexes, respectively. The resulting topological construction returned by `Mapper` is a *geometric realization* of X, inductively defined by the k-skeletons of X, $k \in \mathbb{N}$, to be the quotient space:

$$X^{(k)} := \left(X^{(k-1)} \bigcup \prod_{\sigma:\dim\sigma=k} \Delta^k \right) \Big/ \sim$$

where $\sim$ is the equivalence relation that identifies faces of Δ^k with the corresponding combinatorial faces of σ in $X^{(j)}$ for $j < k$. Thus, the 0-skeleton $X^{(0)} = S$ is a discrete set. The 1-skeleton $X^{(1)}$ is a space homeomorphic to a collection of vertices (S) and edges connecting vertices, i.e. a topological graph. Define a *cover* of a set X as a collection of sets U_α whose union contains X as a subset, i.e. $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ is a cover of X if $X \subseteq \bigcup_{\alpha \in A} U_\alpha$ for some index set A. In this paper, we will always assume that each U_α is path-connected and a cover means a finite open cover. Define the *nerve* of a covering $\mathcal{U}$ as the simplicial complex $N(\mathcal{U})$ whose k-simplices correspond to non-empty intersections of $k+1$ distinct elements of $\mathcal{U}$, $N(\mathcal{U}) = \{\sigma \subseteq A \mid \cap_{\alpha \in \sigma} U_\alpha \neq \emptyset\}$. Given a continuous map $f : X \to Z$ where Z is equipped with a cover $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$, we will write $f^*(\mathcal{U})$ as the cover of X obtained by considering the path connected components of $f^{-1}(U_\alpha)$ for each α. Given this map f, `Mapper` produces a *topological construction* M defined as the nerve of the $f^*(\mathcal{U})$, $M(\mathcal{U}, f) := N(f^*(\mathcal{U}))$.

Unless otherwise stated, we will use lowercase symbols to represent scalars, capital symbols to represent sets, bold lowercase symbols to represent vectors ($1 \times m$), and bold capital symbols to represent matrices ($n \times m$). With a slight abuse of notation, we will occasionally express vectors with a scalar in `typewriter` font when the vector contains identical elements along all of its dimensions, e.g. $\boldsymbol{k} = \mathtt{k}$ implies $\boldsymbol{k} = [k_1, k_2, \ldots, k_m]$ where each $k_i = k \; \forall \; i \in [1, m]$. This is used to simplify the mathematical expressions that follow, and their corresponding examples, although we note the derivation still holds when the components are not identical. This simplification also aligns our analysis more closely with several practical applications of `Mapper`—many of the algorithmic implementations of `Mapper` allow the user to input scalar values out of simplicity, which then convert to their vector-valued equivalents internally. This is elaborated on in the next section.

1.1. The Mapper Algorithm. Mapper constructs a simplicial complex from a given set of 'point cloud data' (PCD) $X \subset \mathbb{R}^d$ by the following (informally presented) process:

1. Define a reference map $f : X \to Z$ where Z is some reference metric space. Note that because Z is a metric space, it's assumed that it is possible to compute inter-point distances between the points in the data of Z.
2. Select a finite covering $\{U_\alpha\}_{\alpha \in A}$ of Z.
3. Construct the subsets $X_\alpha = f^{-1}U_\alpha$. Since f is continuous, these sets also form an open covering of X.

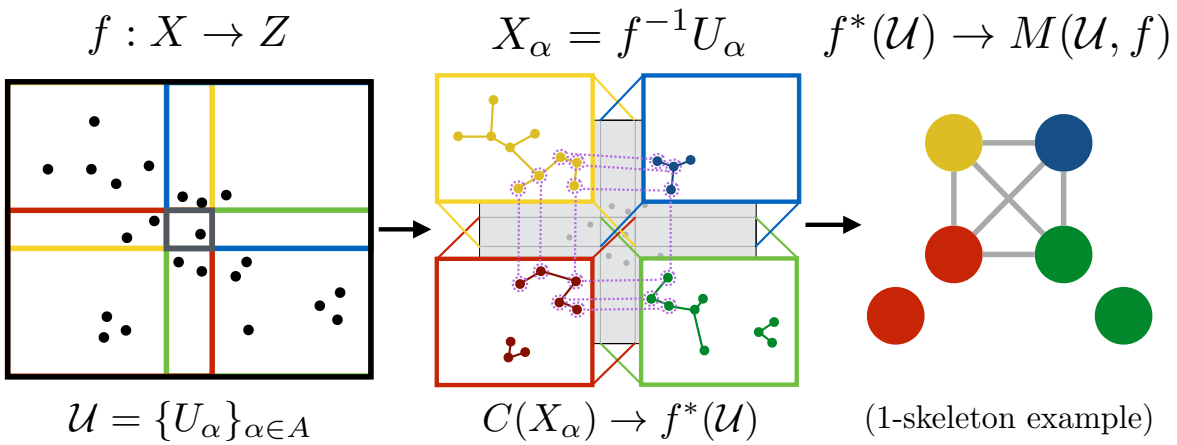

Figure 1. *Mapper step by step example. The left panel shows the mapping of a given data set $X \to Z$ via f (where $Z \in \mathbb{R}^2$) and four overlapping sets that construct the covering $\mathcal{U}$ of Z. The center figure shows the isolation the points in each of these sets that is used for partial clustering and the connected components represent the results of this clustering. Points which appear in multiple subsets are represented by the dashed lines. The final figure on the right represents the 1-skeleton realization of the simplicial complex constructed from the nerve of the covering.*

4. Given a clustering algorithm $\mathcal{C}$, construct the set of clusters by applying $\mathcal{C}$ to the sets X_α. This induces a covering of X parametrized by pairs (α, c) where $\alpha \in A$ and c is one of the clusters of X_α.
5. Construct the simplicial complex $N(\mathcal{U})$ whose vertex set is the set of all possible such pairs (α, c) and where a family $\{(\alpha_0, c_0), (\alpha_1, c_1), \ldots, (\alpha_k, c_k)\}$ spans a k-simplex if and only if the corresponding clusters have a point in common.

Proceeding item-wise through these steps, one begins a TDA with `Mapper` by defining a reference map, sometimes referred to as a *filter* function, which takes as input a set of 'point cloud data' defined in the *data space* and maps to an appropriate metric space, which we'll refer to as the *filter space* or Z-space. The outcome produced by `Mapper` is highly dependent on this function. The choice of this function is inevitably application-specific, however details on common functions that may of interest are discussed in [13]. In step 2, a parameterized covering is constructed on the Z-space, partitioning subsets of the data into subsets in the filter space based on which set of the covering they lie within. We will occasionally refer to these sets which compose the cover as *level sets*. Because level sets generally overlap, any given data point may be within multiple level sets. Since f is continuous, the constructed cover of Z also forms a covering of X. `Mapper` relies on the idea of *partial clustering*, wherein subsets of X are clustered independently of other subsets as a means of simplifying the data into topological prototypes. These subsets are determined by the parameterization of the covering in the Z-space. The prototypes formed within these level sets are represented as 0-simplexes, the interaction of these prototypes with other 0-simplexes form the higher dimensional simplexes in the resulting simplicial complex. Intuitively, a simplicial complex is an expression of a given space as a union of points, intervals, triangles, and higher dimensional

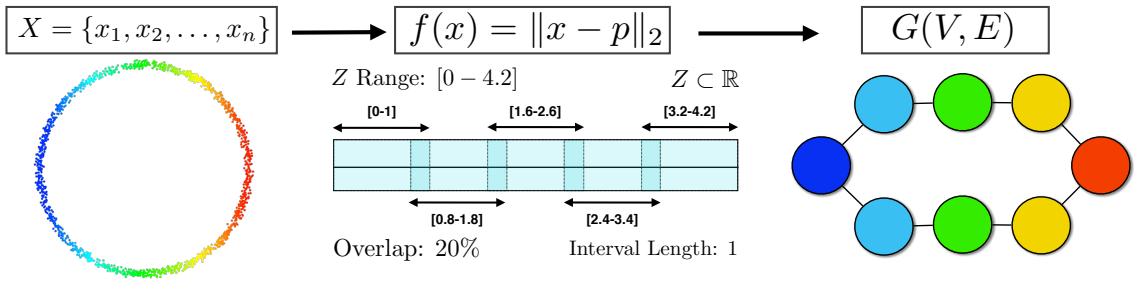

Figure 2. *Mapper ring data example discussed in Example 1.1. In this example, $Z \in \mathbb{R}$ represents the distance from each point $x_i \in X$ to the point with the smallest coordinate in $X^{(1)}$, i.e. the "left-most" point. Note how the three inner intervals resulted in two clusters each, which connect at each end point to a single cluster in the first and last intervals, forming a ring.*

analogues [1]. Figure 1 illustrates, at a high level of abstraction, how the `Mapper` algorithm create a simplicial complex from a given set of data.

Example 1.1: Mapper Ring Data Example

To further illustrate `Mapper`, consider the example given by Singh *et. al* in [13], shown in Figure 2. The left figure displays the ring data X in $\mathbb{R}^2$. The filter function f used computes the distance of every point $x \in X$ to the left-most point p, resulting in a filter space in $\mathbb{R}$. This filter space is shown in the center figure, from which a covering is constructed of 5 sets, each with 20% overlap (the points lie in the filter space are not shown). The right figure shows the resulting 1-skeleton of the constructed simplicial complex, represented as a graph G.

1.2. Cover Parameterization. Many of the parameters of `Mapper` are user supplied. This includes the reference map (step 1), all parameters related to forming the cover (steps 2-3), and the clustering algorithm (as well as its corresponding hyper-parameters) (step 4). Because steps 1 and 4 are generally data set specific, we will focus on steps 2-3: constructing a cover within the Z-space.

There are numerous ways to construct a suitable covering. Although it's assumed each level set U_α is path-connected, and that the cover $\mathcal{U}$ is finite, `Mapper` makes no assertions restricting the shape of the level sets used to construct the covering. For example, Figure 2 of [13] demonstrates two potential coverings of the parameter space in $\mathbb{R}^2$, one with rectangles and another with hexagons. In practice, perhaps due to its simplicity or its computational tractability, a rectangular covering is often used. As a 2-parameter family of coverings, the rectangular cover generally requires the user to supply 1) a tuple of positive integers representing the number of intervals to construct (per dimension) and 2) a tuple of positive real values representing the percent overlap to allow between adjacent intervals (per dimension). Examples of this cover are shown in Figures 1 and 2.

Consider steps 2 and 3 of the `Mapper` framework: constructing a finite covering $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ of Z, given a user-supplied continuous map $f : X \to Z$. The hyper-rectangular covering method requires as input the number of rectangles $\boldsymbol{k} = [k_1, k_2, \ldots, k_m]$ to distribute

along each dimension (where $\boldsymbol{k} \in \mathbb{Z}^m$), and an overlap percentage $\boldsymbol{g} = [g_1, g_2, \ldots, g_m]$ neighboring rectangles should intersect (where $\boldsymbol{g} \in [0,1)^m$). One proceeds constructing the cover by first finding the range of the function f restricted to the set of given points Z. That is, denote the i^{th} point of Z as $\boldsymbol{z}_i$, and its j^{th} component of this point as $z_i^{(j)}$. Similarly, denote the j^{th} component Z as $\boldsymbol{z}^{(j)}$. The extrema vectors of Z are defined as follows:

$$\begin{aligned} \boldsymbol{z_{max}} &= [\ \max(\boldsymbol{z}^{(1)}), \max(\boldsymbol{z}^{(2)}),\ \ldots, \max(\boldsymbol{z}^{(m)})\ ] \\ \boldsymbol{z_{min}} &= [\ \min(\boldsymbol{z}^{(1)}), \min(\boldsymbol{z}^{(2)}),\ \ldots, \min(\boldsymbol{z}^{(m)})\ ] \end{aligned}$$

And the range vector is the component-wise difference between them:

$$\overline{\boldsymbol{z}} = \boldsymbol{z_{max}} - \boldsymbol{z_{min}}$$

Given a user-supplied input for $\boldsymbol{g}$ and $\boldsymbol{k}$, a cover $\mathcal{U}[\boldsymbol{r}, \boldsymbol{e}]$ is constructed where:

$$\boldsymbol{r} = \frac{\overline{\boldsymbol{z}}}{(\boldsymbol{k} - \boldsymbol{g}(\boldsymbol{k}-1))}, \quad \boldsymbol{e} = \boldsymbol{r} \circ (1 - \boldsymbol{g}) \tag{1}$$

We'll refer to $\boldsymbol{r}$ as the *interval length* of a given covering, and $\boldsymbol{e}$ as the *step size* (which is inversely proportional to the overlap $\boldsymbol{g}$ with respect to $\boldsymbol{r}$). When $\boldsymbol{k} > 1$, this cover implies Z is effectively divided up into a number of overlapping, rectangular level sets. Each point in Z is lies within at least one level set of $\mathcal{U}[\boldsymbol{r}, \boldsymbol{e}]$. To formalize operations on these sets, it's necessary to express this cover as an *indexed family* of sets. That is, the cover $\mathcal{U}[\boldsymbol{r}, \boldsymbol{e}] = \{U_\alpha\}_{\alpha \in A}$ of the space Z is an indexed family of level sets U_α, each of length $\boldsymbol{r}$, indexed by a given index set A represented by the mapping function $A_\alpha : A \to U_\alpha$.

From a theoretical perspective, there is no restriction on how the index set is represented. But consider the case where $\boldsymbol{k} = \mathtt{k}$ and A is given by the m-fold Cartesian product of $\boldsymbol{k}$, i.e.

$$A = \kappa^m = \underbrace{\kappa \times \cdots \times \kappa}_{m} = \{(l_1, l_2, \ldots, l_m) \mid l_i \in \kappa \text{ for all } i = 1, \ldots, m\}$$

where $\kappa = [0, 1, \ldots, \boldsymbol{k} - 1]$. This indexing scheme offers the practical advantage of not only being easy to compute, but it also explicitly records locational information with regards to the position of each level set in the cover. For example, when this indexing scheme is used, the level set indexed by the tuple $(0, 0)$ contains any point in Z between $[\boldsymbol{z_{min}}, \boldsymbol{r})$, and similarly the level set indexed by $(k-1, k-1)$ will contain any point in Z between $[\boldsymbol{r} \circ (\mathtt{k} - 1), \boldsymbol{z_{max}}]$. This approach has additional advantages, which we elaborate on in the next section.

2. Cover Analysis. We now derive the main result of the paper: a closed-form solution to find the minimal set of overlap values which may be used to compute distinct simplicial complexes. As discussed above, we specifically consider the case where a (possibly multivariate) hyper-rectangular covering method is used, and we focus on the case where the number of intervals $\boldsymbol{k}$ is fixed.[2] Additionally, we only concern ourselves with the 1-skeleton, which is often sufficient for most types of analysis [].

[2]This is the only covering method supported by the *Python Mapper* library for $\mathbb{R}^d$ when $d > 1$ and the R package *TDAmapper*.

The goal of the following analysis is to create an method for computing the minimal set of input values $\boldsymbol{g}$ which realize all possible *distinct* simplicial complexes in a closed-form solution. To formalize this, it's necessary to first define what is meant by a 'distinct' solution. Recall that the `Mapper` construction provides a discrete, combinatorial summary of the data used to construct it, $M(\mathcal{U}_{\boldsymbol{r}}, f)$, given by the nerve of $f^*(\mathcal{U}_{\boldsymbol{r}})$. Although the overlap parameter $\boldsymbol{g}$ lies on a continuum, there may be ranges of values which produce identical simplicial complexes. For example, consider a covering constructed from $\boldsymbol{g} = \mathtt{g}$ and another from $\boldsymbol{g}' = \mathtt{g} + \epsilon$ for a sufficiently small ϵ. Denote the corresponding interval lengths as $\boldsymbol{r}$ and $\boldsymbol{r}'$. Since $\boldsymbol{k}$ is fixed, $\boldsymbol{r}'$ will be slightly larger than $\boldsymbol{r}$ in the corresponding covers, $\mathcal{U}_{\boldsymbol{r}}$ and $\mathcal{U}_{\boldsymbol{r}'}$. However, if the point membership of each level set does not change between the two covers, then the corresponding dissimilarities between all points within each level set will also not change; there will be no new vertices or edges introduced between the two `Mapper` constructions. This motivates a definition of *distinction* between possible realized values of $\boldsymbol{g}$.

Definition 2.1 (Distinct Simplicial Complex). *Given a finite set of point cloud data $\boldsymbol{Z}_n \in Z$ where $Z \subset \mathbb{R}^m$ and two covers $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ and $\mathcal{V} = \{V_\alpha\}_{\alpha \in A}$ of Z which share the same index set (but may index different families), denote the corresponding membership sets of $\boldsymbol{Z}_n \cap U_\alpha$ and $\boldsymbol{Z}_n \cap V_\alpha$ as follows:*

$$\begin{aligned} \mathcal{U}_\alpha(\boldsymbol{Z}) &= \{\boldsymbol{z} : \boldsymbol{z} \in \boldsymbol{Z}_n \text{ and } \boldsymbol{z} \in U_\alpha\} \\ \mathcal{V}_\alpha(\boldsymbol{Z}) &= \{\boldsymbol{z} : \boldsymbol{z} \in \boldsymbol{Z}_n \text{ and } \boldsymbol{z} \in V_\alpha\} \end{aligned} \tag{2}$$

We define the simplicial complex $N(\mathcal{U})$ as being distinct from the simplicial complex $N(\mathcal{V})$ if:

$$\text{Any } \mathcal{U}_\alpha(\boldsymbol{Z}) \neq \mathcal{V}_\alpha(\boldsymbol{Z}) \; \forall \; \alpha \in A \tag{3}$$

Thus, given two coverings with the same index set, if there is at least one membership set that is not equal for some index, then the resulting simplicial complexes are considered distinct.

It's worth noting that under this definition of distinctness, it is possible to observe two `Mapper` constructions whose 1-skeletons share the same simplex representation. That is, it is *not* guaranteed that . However, under this definition, the *mapping* of simplices to points in $\boldsymbol{Z}_n$ will be different.

With an established notion of a distinct simplicial complex, we begin our derivation by assuming we are given a finite point cloud $\boldsymbol{Z}_n \in Z$, and consider the special case where the topological space Z is zero-dimensional with respect to its Lebesgue covering dimension. This implies that each open set of the cover of Z is disjoint, or equivalently $\boldsymbol{g} = 0$. For brevity, we'll occasionally refer to this case as the *base cover*. Under this case, every point $z_i \in \boldsymbol{Z}_n$ is contained in exactly one set of the cover, and additionally, is mapped by exactly one index $\boldsymbol{\alpha} \in A$. Let this initial mapping be expressed as $\psi : \boldsymbol{Z}_n \to \boldsymbol{\alpha}$, where $\boldsymbol{\alpha} \in A$, and denote $\boldsymbol{A}$ as the matrix of mappings by ψ over a set of n data points (i.e. where row $\boldsymbol{A}_i$ corresponds to mapping $\psi(\boldsymbol{z}_i)$).

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{\alpha_0} & \boldsymbol{\alpha_1} & \dots & \boldsymbol{\alpha_n} \end{bmatrix}^\intercal = \begin{bmatrix} \psi(\boldsymbol{z}_0) \\ \psi(\boldsymbol{z}_1) \\ \vdots \\ \psi(\boldsymbol{z}_n) \end{bmatrix} \quad (\text{where } \boldsymbol{A} \in \mathbb{Z}^{n \times m}) \tag{4}$$

Given a set of *fixed* parameters, the parameters of the cover $\boldsymbol{r}$ and $\boldsymbol{e}$ are traditionally defined as in (1). We first need to establish the parameterization of the cover with the defined mapping $\boldsymbol{A}$, i.e. when $\boldsymbol{g} = 0$. In doing so, we define the *base interval length* as follows:

$$\bar{\boldsymbol{r}} = \overline{\boldsymbol{z}} \circ \boldsymbol{k}^{-1} \tag{5}$$

Since $\boldsymbol{e}$ is inversely proportional to $\boldsymbol{g}$, the corresponding step size $\boldsymbol{e} = 0$. This establishes a lower bound on the interval length (any interval length below this value is not a cover of Z), and provides a starting point for our solution. Given a base interval length, each point $\boldsymbol{z}_i$ is contained in exactly one level set $\boldsymbol{\alpha}_i$, given by the map $\psi(\boldsymbol{z}_i)$, which we'll refer to the points' *originating set*. A simple illustrations of the realizations of these definitions for a small set of points is given in Example 1.1.

Example 1.1

Consider a set of points $\boldsymbol{Z}_n \subset \mathbb{R}^2$ with the defined range vector $\overline{\boldsymbol{z}} = [2, 4]$. Suppose $\boldsymbol{k} = 2$, i.e. $\boldsymbol{k} = [2, 2]$. Assume a base cover is be constructed $\mathcal{U}[\bar{\boldsymbol{r}}, 0]$ where $\bar{\boldsymbol{r}} = [1, 2]$. The index set A is given by the cartesian product $\kappa \times \kappa = \{(0,0), (0,1), (1,0), (1,1)\}$, and it's corresponding family of level sets is given by $\{\{[0,1],[0,2]\}, \{(1,2],[0,2]\}, \{[0,1],(2,4]\}, \{(1,2],(2,4]\}\}$. Observe that when $\boldsymbol{k}$ is fixed, the family of coverings of $\mathcal{U}[\bar{\boldsymbol{r}}, \boldsymbol{e}]$ for all values of $\boldsymbol{e}$ have the same indexing set.

When $\boldsymbol{g} = 0$, there are no simplices of dimension ≥ 1 in the resulting `Mapper` construction, i.e. M is a fully disconnected graph. Consider the case when $\boldsymbol{g}' = 0 + \epsilon$ for a sufficiently small ϵ. What is the smallest ϵ that will produce a distinct simplicial complex? According to our definition of a distinct simplicial complex 2.1, this question is equivalent to asking: what is the smallest overlap percentage $\boldsymbol{g}' > 0$ that induces a set membership change, where $\mathcal{U}_\alpha(\boldsymbol{Z})$ corresponds to the cover $\boldsymbol{g} = 0$, and $\mathcal{V}_\alpha(\boldsymbol{Z})$ corresponds to the cover $\boldsymbol{g}' > 0$? Clearly, we must consider overlap values $\boldsymbol{g}' > \boldsymbol{g}$, and thus interval lengths larger than $\hat{\boldsymbol{r}} > \bar{r}$. For the membership sets between two parameterizations to be different, a point $\boldsymbol{z}_i \in \boldsymbol{Z}_n$ must intersect another level set, outside of its originating set. To compute this efficiently, it's necessary know the *relative* position of each point within it's originating set, such that the metric distance between any pair of points from the same originating level set is unchanged. Consider a projection of a point z_i onto a single index set, such that $0 \leq z_i^{(d)} \leq \bar{r}^{(d)} \; \forall \; d \in \{1, 2, \ldots, m\}$, with projected positions $\widetilde{\boldsymbol{Z}}$ computed as follows:

$$\widetilde{\boldsymbol{Z}} = (\boldsymbol{Z}_n - \boldsymbol{z_{min}}) - \boldsymbol{A} \circ \bar{\boldsymbol{r}} \quad (\text{where} \;\; \widetilde{\boldsymbol{Z}} \in \mathbb{R}^{n \times m}) \tag{6}$$

Equation 6 describes $\widetilde{\boldsymbol{Z}}$ as a translation of Z to the origin, followed by a projection of every point to the level set $[0, \bar{r}]$. Note that this projection is only possible with the original surjective mapping of $\psi(\boldsymbol{z}_i) \to \boldsymbol{\alpha}_i$, where $\boldsymbol{\alpha}_i \in \mathbb{Z}^m$ is a vector of integers representing the indexed location of each point in the base cover. A visual illustration of the projection is shown in Figure 3. Given $\widetilde{\boldsymbol{Z}}$, we can collectively compute the distance of every point in $\boldsymbol{Z}_n$ to its *nearest* adjacent set, per dimension, by considering which halfspace each individual point lies within. If a given point $\tilde{\boldsymbol{z}}_i$ lies in the lower halfspace along dimension d, then the nearest level set along dimension d that $\boldsymbol{z}_i$ is simply the d^{th} component of $\tilde{\boldsymbol{z}}_i$. If $\tilde{\boldsymbol{z}}_i$ lies within the upper halfspace

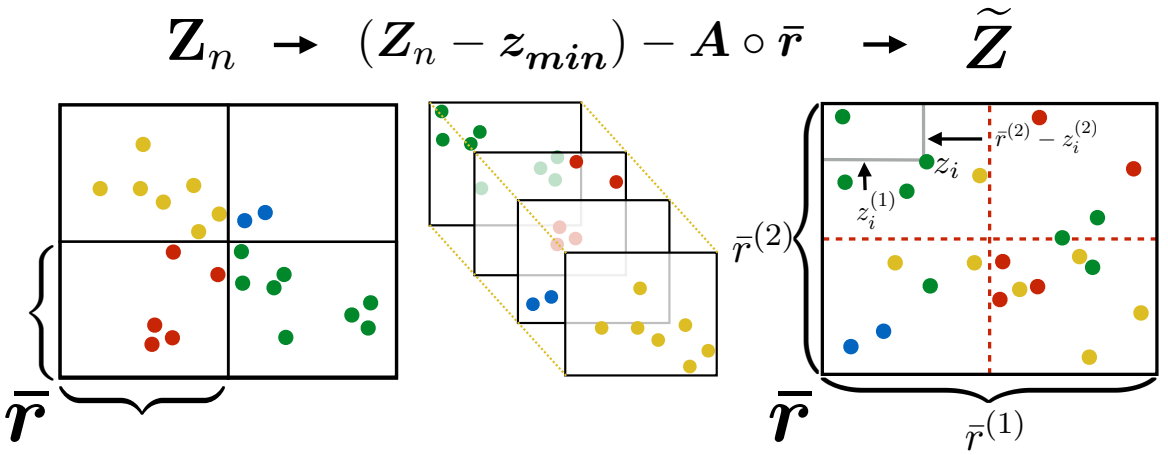

Figure 3. *The left panel shows the given $\boldsymbol{Z}_n$ in the Z-space (where $Z \in \mathbb{R}^2$) and four overlapping sets that construct the covering $\mathcal{U}$ of Z. The center figure shows conceptually how the projection is aligning each level set into one level set. The final figure on the right illustrates how each points corresponding $\tilde{\boldsymbol{z}}_i^{(d)}$ value depends on which halfspace (delimited with red dashed lines) it lies in. Note that the left panel is showing four level sets, and the right panel is showing only one defined on the interval $[0, \boldsymbol{r}]$.*

along dimension d, then the distance to the nearest adjacent level set is given by $\bar{r}^{(d)} - \tilde{\boldsymbol{z}}_i^{(d)}$. These simple observations express all the information needed to compute the entire finite set of $\boldsymbol{g}$ that produce distinct simplicial complexes, and in the next section, we consider two use-cases of where these observations become useful for 1-skeleton generation.

2.1. Restricted Cover. In practice, it's often sufficient to assume reasonable restrictions on parameter ranges of the cover. For example, if we consider using a rectangular cover over $\mathbb{R}^2$ under the case where $e < \frac{r}{2}$, there will exist regions where at most four rectangles will intersect, and thus the dimension of the simplices used to construct the complex will be 3 or less. We first analyze this situation in the context of our end-goal.

Consider the projection previously discussed in equation 6 and the corresponding component wise differences, $\tilde{\boldsymbol{z}}_i^{(d)}$ if $\tilde{\boldsymbol{z}}_i$ lies in the lower halfspace of $\widetilde{\boldsymbol{Z}}$ along dimension d, and $\bar{r}^{(d)} - \tilde{\boldsymbol{z}}_i^{(d)}$ if it lies in the upper halfspace. Since these distances are complements to each other with respect to $\boldsymbol{r}$, regardless of which halfspace $\tilde{z}_i^{(d)}$ lies within, the distance to the closest adjacent level set may be simplified by minimum of the two values: $\delta(\boldsymbol{z}_i) = \min(\tilde{\boldsymbol{z}}_i^{(d)}, \bar{\boldsymbol{r}}^{(d)} - \tilde{\boldsymbol{z}}_i^{(d)})$. Denote this mapping as follows:

$$\boldsymbol{Z}_\Delta = \begin{bmatrix} \delta(\boldsymbol{z}_1^{(1)}) & \delta(\boldsymbol{z}_1^{(2)}) & \dots & \delta(\boldsymbol{z}_1^{(m)}) \\ \delta(\boldsymbol{z}_2^{(1)}) & \delta(\boldsymbol{z}_2^{(2)}) & \dots & \delta(\boldsymbol{z}_2^{(m)}) \\ \vdots & \vdots & \ddots & \vdots \\ \delta(\boldsymbol{z}_n^{(1)}) & \delta(\boldsymbol{z}_n^{(2)}) & \dots & \delta(\boldsymbol{z}_n^{(m)}) \end{bmatrix} \quad (\text{where } \boldsymbol{Z}_\Delta \in \mathbb{R}^{n \times 1}) \tag{7}$$

These minimal distances, when added to the base interval length, define the *smallest* interval length (for a given point $\boldsymbol{z}_i$) $\hat{\boldsymbol{r}}$ which induces a change in the membership set of the

corresponding cover $\mathcal{U}$. Furthermore, they can be computed simply:

$$\hat{\boldsymbol{R}} = \boldsymbol{Z}_{\Delta} + \bar{\boldsymbol{r}} \quad (\text{where} \quad \hat{\boldsymbol{R}} \in \mathbb{R}^{n \times m}) \tag{8}$$

Note that because these distances are all orthogonal, we have all of the information needed to compute when *any* set of the covering will intersect the originating set of a given point $\boldsymbol{z}_i$. For example, consider any subset in the cover which is not along one of the orthogonal directions in $\hat{\boldsymbol{r}}$. The smallest interval length wherein a given point $\boldsymbol{z}_i$ outside of this set will intersect the subset is achieved when $\boldsymbol{r} = \mathtt{r}_{max}$ where $\mathtt{r}$ is an m-length vector of values all set to $\max(\hat{\boldsymbol{r}})$. Since the step size $\boldsymbol{e}$ is a function of $\boldsymbol{g}$ and $\boldsymbol{r}$, and because $\boldsymbol{k}$ is fixed, knowledge of the smallest interval length(s) which cause a level set membership change enables $\boldsymbol{g}$ to be computed in closed-form. Recall that fixed `Mapper` traditionally takes as input the number of intervals to divide the filter space along each dimension, $\boldsymbol{k}$, and the overlap (percentage) between these sets $\boldsymbol{g}$, as shown in equation 1. We take the inverse of this function, supplying the set of $\hat{\boldsymbol{R}}$ values computed from the $\boldsymbol{Z}_n$ point cloud as input with the fixed $\boldsymbol{k}$ to recover $\boldsymbol{g}$. The derivation, given in the appendix 5, yields:

$$\boldsymbol{G} = \frac{(\bar{\boldsymbol{z}} - \hat{\boldsymbol{R}}\boldsymbol{k})}{\hat{\boldsymbol{R}}(-\boldsymbol{k} + 1)} \quad (\text{where} \quad \boldsymbol{G} \in \mathbb{R}^{n \times m}) \tag{9}$$

Given $\boldsymbol{G}$, the step size can be computed analogously:

$$\hat{\boldsymbol{E}} = \hat{\boldsymbol{R}}(1 - \boldsymbol{G}) \tag{10}$$

Each entry $\boldsymbol{g}_i^{(j)} = \boldsymbol{G}_{ij}$ expresses the smallest overlap percentage necessary for the corresponding point $\boldsymbol{z}_i$ to intersect it's nearest adjacent level set along the j^{th} dimension. Since we only consider the restricted cover parameterization in this section, these values encompass all of the possible values to produce distinct simplicial complexes. More specific uses cases for when $\boldsymbol{G}$ are available is discussed more in Section 3. We next discuss the case where $e \geq \frac{r}{2}$, or equivalently, where the overlap is less than 50%.

2.2. Unrestricted Cover. In this situation, $\boldsymbol{g}$ is not bounded above by 0.50. As a result, the value of $\boldsymbol{z}_i^{(d)}$ may intersect more than just its nearest and originating level sets. For example, consider a point $z = 2.1$ in 2, but instead consider the case where the cover was constructed with an overlap of 80%. When $k = 5$, the corresponding interval length comes out to ≈ 2.41. In this situation, z intersects every level set of the cover. This implies that a more general formulation of equation 9 is needed. Note that this is only needed when $\boldsymbol{g} \geq 0.50$.

We use the same strategy as before, but now replace equation 7 with two variables, $\boldsymbol{Z}_{near}$ and $\boldsymbol{Z}_{far}$. Set $\boldsymbol{Z}_{near} = \boldsymbol{Z}_{\Delta}$, and compute $\boldsymbol{Z}_{far}$ analogously but instead by looking at the distance to the *farther* level set, i.e. $\delta(\boldsymbol{z}_i) = \max(\tilde{\boldsymbol{z}}_i^{(d)}, \bar{\boldsymbol{r}}^{(d)} - \tilde{\boldsymbol{z}}_i^{(d)})$. Additionally, we add natural multiples of $\bar{R}$ to account for level sets which are not immediately adjacent to the originating level sets. The $\boldsymbol{p}^{\text{th}}$ nearest level sets correspond to:

$$\begin{aligned} \hat{\boldsymbol{R}}_{near} &= \boldsymbol{Z}_{near} + (\boldsymbol{p} - 1)\bar{\boldsymbol{R}} \quad (\text{where} \quad \hat{\boldsymbol{R}}_{near} \in \mathbb{R}^{1 \times m}) \\ \hat{\boldsymbol{R}}_{far} &= \boldsymbol{Z}_{far} + (\boldsymbol{p} - 1)\bar{\boldsymbol{R}} \quad (\text{where} \quad \hat{\boldsymbol{R}}_{far} \in \mathbb{R}^{1 \times m}) \end{aligned} \tag{11}$$

where $\boldsymbol{p}$ is an m-dimensional vector where each component $\boldsymbol{p}^{(d)} \in [1, 2, \ldots, \boldsymbol{k}^{(d)}]$.

Note that the magnitude of each component of $\boldsymbol{p}$ dramatically increases the total number of overlap values. Consider the case where $\boldsymbol{p} = \mathtt{k}$.

2.3. Practical considerations. In the previous sections, we outlined methods for compute the full finite set of overlap values under both the cases where $0 \leq g < 0.50$ and $0.50 \leq g < 1$. The analysis performed so far has generally assumed that all level sets within distance $\hat{\boldsymbol{R}}$ were *valid* level sets in the cover. However, it may be the case that for a given point z_i, the level set(s) that were considered in computing $\hat{\boldsymbol{R}}$ lie outside of the cover. For example, consider a point $z_i = 0.2$ positioned in the filter space of the the ring data set from example 2. When $0 \leq g < 0.50$, z_i never intersects a level set beyond its originating set. Because z_i lies at the extrema of the filter space, the halfspace distance computed for z_i does not correspond to a level set of the cover, and therefore its corresponding value in G must be discarded. When $0.50 \leq g < 1$, the corresponding overlap values for z_i may be computed using multiples of $\hat{\boldsymbol{R}}_{far}$, as given in equation 11, for all $p = \{1, 2, \ldots, 4\}$.

We note that this situation is simply an exception case, does not detract from the validity of the solutions given above, and can be readily handled by simply ensuring the index of the level set whose distance is being computed is within the index set of the cover. Furthermore, these indexes may be readily computed. Consider the use of the m-fold Cartesian product index set described in section 1. If $e < \frac{r}{2}$, these are simply the sets adjacent to $\boldsymbol{A}$ in the direction of the halfspace each point in $\boldsymbol{Z}_n$ lies. Conceptually, this index set $\hat{\boldsymbol{A}}$ represents the mapping as follows:

$$\hat{\boldsymbol{A}} = \begin{bmatrix} \psi(\boldsymbol{z}_0 + \hat{\boldsymbol{R}}_0) \\ \psi(\boldsymbol{z}_1 + \hat{\boldsymbol{R}}_1) \\ \vdots \\ \psi(\boldsymbol{z}_n + \hat{\boldsymbol{R}}_n) \end{bmatrix} \quad (\text{where } \boldsymbol{A} \in \mathbb{Z}^{n \times m}) \tag{12}$$

Since we directly account for this offset in equation 11, this mapping can be simplified for both the restricted and unrestricted covers as follows:

$$\hat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{\gamma} \circ \boldsymbol{p} \tag{13}$$

$$\text{where } \boldsymbol{\gamma} = \begin{cases} 1 & \text{if } \bar{\boldsymbol{r}} - \widetilde{\boldsymbol{Z}} < \frac{\hat{\boldsymbol{R}}}{2} \\ -1 & \text{if } \bar{\boldsymbol{r}} - \widetilde{\boldsymbol{Z}} \geq \frac{\hat{\boldsymbol{R}}}{2} \end{cases}$$

Any index $\hat{\boldsymbol{A}} \notin A$ corresponds to invalid overlap parameter, and can be excluded through a simple post-processing step.

3. Implementation. In 2013, Müllner et. al introduced the *Python Mapper* software tool [11] as a limited yet extensible realization of the `Mapper` framework for data analysis. The *Python Mapper* library supports three types of covers for $\mathbb{R}$, and one type of cover for $\mathbb{R}^d$ (multidimensional 'patch' covering, the one studied in this effort). Similarly, Pearson [] has an R package *TDAmapper* which relies on the hyper-rectangular covering method as well. All of the derived equations as part of this effort are available in both Python and R scripts, available

open-source online, as unofficial extensions to these libraries.[3] In the following subsections, we demonstrate the utility of this result with a few example data sets previously analyzed with `Mapper`.

4. Applications and Discussion. There are several applications of `Mapper` enabled from this work. We outline only a few here.

4.1. Improvement in Efficiency. TODO

4.2. Multiscale Mapper: Expanding n-orthotopes. TODO

4.3. Sensitivity and Stability. TODO

5. Conclusion and Future Work. In this paper, we've derived a closed-form solution to computing all of the possible parameter values to the overlap parameter in `Mapper`, enabling efficient computation of not only the 1-skeleton of the simplicial complex traditionally produced by `Mapper`, but also of the full simplicial complex. Theoretical evidence was shown indicating that this solution may also be used to compare output constructions from `Mapper` very quickly across a discrete set of parameter settings, which in turn may provide a useful foundation for both comparative and exploratory analysis of how the topological structure of G changes as a function of such parameter changes. Finally, we included a small discussion of how the solution provided enables further development in the application of topological data analysis to various theoretical and application areas.

Acknowledgement. This work was supported by an appointment to the Internship/Research Participation Program at the U.S. Air Force Research Laboratory (AFRL), administered by the Oak Ridge Institute for Science and Education through an interagency agreement between the U.S. Department of Energy and Wright-Patterson Air Force Base.

Appendix: Overlap Inverse. Traditionally, given the number of intervals $\boldsymbol{k}$ and the percentage overlap between consecutive intervals $\boldsymbol{g}$, Z may equipped with a hyper-rectangular covering $\mathcal{U}[\boldsymbol{r}, \boldsymbol{e}]$ given one has a finite point cloud $\boldsymbol{Z}_n \subset Z$, where $\boldsymbol{r}$ and $\boldsymbol{r}$ are computed as follows:

$$\boldsymbol{r} = \frac{\boldsymbol{z}}{(\boldsymbol{k} - \boldsymbol{g}(\boldsymbol{k}-1))}, \ \boldsymbol{e} = \boldsymbol{r}(1-\boldsymbol{g}) \tag{14}$$

Since $\boldsymbol{e}$ is dependent on $\boldsymbol{g}$, and assuming $\boldsymbol{k}$ is fixed, it's clear that all one needs to compute $\boldsymbol{g}$ is a supplied interval length $\boldsymbol{r}$. This is shown below:

$$\begin{aligned} \boldsymbol{r}(\boldsymbol{k} - \boldsymbol{g}(\boldsymbol{k}-1)) &= \bar{\boldsymbol{z}} \\ \boldsymbol{r}\boldsymbol{k} - \boldsymbol{r}\boldsymbol{g}\boldsymbol{k} + \boldsymbol{r}\boldsymbol{g} &= \bar{\boldsymbol{z}} \\ \boldsymbol{g}(-\boldsymbol{r}\boldsymbol{k} + \boldsymbol{r}) &= \bar{\boldsymbol{z}} - \boldsymbol{r}\boldsymbol{k} \\ \boldsymbol{g} &= \frac{\bar{\boldsymbol{z}} - \boldsymbol{r}\boldsymbol{k}}{\boldsymbol{r}(-\boldsymbol{k}+1)} \end{aligned} \tag{15}$$

It's possible that a user of `Mapper` could parameterize the cover using $\boldsymbol{r}$ and k directly instead of g, however determining what values of $\boldsymbol{r}$ are relevant is difficult since the range of Z will

[3] TODO:link github

inevitably differ in value and interpretation with respect to the data set and reference map used. This is perhaps the reason its preferred by the open-source implementations of `Mapper` to parameterize the cover by g, which always carries the same interpretation as a percentage. We consider an alternative approach which explicitly computes $\boldsymbol{r}$ from a finite set of data, thus reducing the possible values of $\boldsymbol{g}$ (for a given set of data) to a finite number of parameter settings.

REFERENCES

[1] G. Carlsson, *Topology and data*, Bulletin of the American Mathematical Society, 46 (2009), pp. 255–308.

[2] T. K. Dey, F. Mémoli, and Y. Wang, *Multiscale mapper: Topological summarization via codomain covers*, in Proceedings of the twenty-seventh annual acm-siam symposium on discrete algorithms, SIAM, 2016, pp. 997–1013.

[3] T. K. Dey, F. Mémoli, and Y. Wang, *Topological analysis of nerves, reeb spaces, mappers, and multiscale mappers*, arXiv preprint arXiv:1703.07387, (2017).

[4] K. J. Emmett and R. Rabadan, *Characterizing scales of genetic recombination and antibiotic resistance in pathogenic bacteria using topological data analysis*, in International Conference on Brain Informatics and Health, Springer, 2014, pp. 540–551.

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

[6] R. W. Ghrist, *Elementary applied topology*, Createspace, 2014.

[7] C. Heine, H. Leitte, M. Hlawitschka, F. Iuricich, L. De Floriani, G. Scheuermann, H. Hagen, and C. Garth, *A survey of topology-based methods in visualization*, in Computer Graphics Forum, vol. 35, Wiley Online Library, 2016, pp. 643–667.

[8] A. K. Jain, *Data clustering: 50 years beyond k-means*, Pattern recognition letters, 31 (2010), pp. 651–666.

[9] T. Kohonen, *Self-organized formation of topologically correct feature maps*, Biological cybernetics, 43 (1982), pp. 59–69.

[10] S. MERKULOV, *Hatcher, a. algebraic topology (cambridge university press, 2002)*, Proceedings of the Edinburgh Mathematical Society, 46 (2003), pp. 511–512.

[11] D. Müllner and A. Babu, *Python mapper: An open-source toolchain for data exploration, analysis, and visualization*, URL http://math. stanford. edu/muellner/mapper, (2013).

[12] E. Munch, *A users guide to topological data analysis*, Journal of Learning Analytics, 4 (2017), pp. 47–61.

[13] G. Singh, F. Mémoli, and G. E. Carlsson, *Topological methods for the analysis of high dimensional data sets and 3d object recognition.*, in SPBG, 2007, pp. 91–100.

[14] J. B. Tenenbaum, *Mapping a manifold of perceptual observations*, in Advances in neural information processing systems, 1998, pp. 682–688.