Efficient Cover Parameterization and Simplicial Complex Generation for `Mapper`

Matt Piekenbrock* and Derek Doran†

Abstract. The primary result of the `Mapper` framework 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, 54J05, 54C05

Introduction. The notion of *nearness* plays an important role in studying and understanding relationships in data. Within the realm of unsupervised 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, metric distance interpolation methods along the surface of the manifold enable realistic, perceptual changes between images to be more semantically clear [11]. 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 [6]. 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 [7]. Indeed, it is not difficult to find unsupervised models which exploit notions of nearness to describe something useful or interesting about the underlying space the data were observed.

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 may be thought as the natural evolution of the notions of proximity and continuity [4]. 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 *et al.* [10]. `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 visualiza-

*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/).

tion [5], genetic recombination characterization [2], and dimensionality reduction [10]. `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. `Mapper` prioritizes data *summarization* over *accuracy of representation*. That is, the goal of `Mapper` is *not* to achieve a fully accurate representation of the data, but significant areas of interest should be captured, and ideally in an understandable or explainable way, allowing for exploratory data analysis.

`Mapper` is an unsupervised form of learning. 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. We'll generically refer to the choices of function, covering, clustering algorithm, distance metric, and all other user-supplied inputs as *parameters*. `Mapper` admits a large parameter space, requiring several choices to be made by the user based on the goal of the analysis at hand. More often than not, a large parameter space implies a large solution space: changing the value of any particular parameter in `Mapper` may lead to different set of topological solutions, which may in turn lead to inconsistent interpretations and analyses. Solutions which reduce this complexity would indeed accelerate the adoption of TDA as a useful tool for data analysis.

The *overlap* parameter of `Mapper`, which we describe further in section 1, determines the connectivity, and thus the relational component, of the output graph G. Unfortunately, there is little guidance on how to determine parameter values relevant to the data set being analyzed. It is not possible to identify an ideal overlap parameter without extensive knowledge of the underlying topology of a dataset, which is seldom known. The best solution, then, may be to allow an analyst to explore the space of possible overlap parameter values, thus comparing, contrasting, and deriving insights by viewing multiple topological representations of the data at once. The analyst, then, could understand how the topological representation of the data (i.e. the output graph of `Mapper`) changes across other settings of interest. For example, methods which measure some notion of 'stability' of a given real-valued function with respect to its domain have proven useful in both theoretical and practical instances of TDA [3], yet require knowledge of how the image of the function changes with respect to ranges of parameter inputs, and what those parameter ranges are. We discuss this more in-depth with several examples in section 3. Such analysis can only be enabled, 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 efficiently on a data set of size n in $O(n)$ time. We outline the benefits in tractability our approach enables in the context of simplicial complex

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

generation and in terms of enabling future stability-type analyses.

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 usefulness of our solution. Finally, we discuss future applications and work related to `Mapper` in Section 4.

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* [10] 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 [8] provides much information on simplicial complexes, and Ghrist [4] 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. Given such a collection, define the *nerve* of the covering $\mathcal{U}$ as the simplicial complex $N(\mathcal{U})$ whose k-simplices correspond to non-empty intersections of $k+1$ distinct elements of $\mathcal{U}$.

We will use bold notation for all vectors, and capital symbols for all matrices. With a slight abuse of notation, we will occasionally express vectors with a scalar 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 $k_i = k$. This is used to simplify the mathematical expressions that follow, and their corresponding examples, although our derivation still holds when the components 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

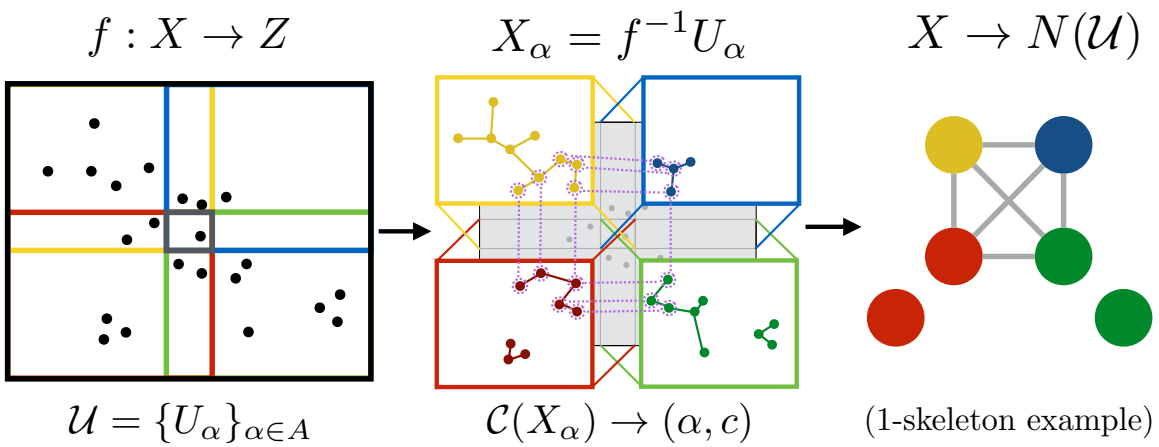

Figure 1. *Mapper step by step example. The left panel shows the mapping of a given data set $X \to Z$ via f (where $Z \in \mathbb{R}^2$) and four overlapping sets that construct the covering $\mathcal{U}$ of Z. The center figure shows the isolation the points in each of these sets that is used for partial 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.*

scalar values out of simplicity, which then convert to their identical vector-valued expressions internally.

1.1. The Mapper Algorithm.

Mapper constructs a simplicial complex from a given data set $X \subset \mathbb{R}^d$ 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. It is assumed that the indexing set A is finite.
3. Construct the subsets $X\alpha = f^{-1}U\alpha$. Since f is continuous, these sets also form an open covering of X.
4. Given a clustering algorithm $\mathcal{C}$, construct the set of clusters by applying $\mathcal{C}$ to the sets $X\alpha$. This induces a covering of X parametrized by pairs (α, c) where $\alpha \in A$ and c is one of the clusters of $X\alpha$.
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' and maps to an appropriate metric 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 [10]. In step 2, a parameterized covering is constructed on the Z space, partitioning subsets of the

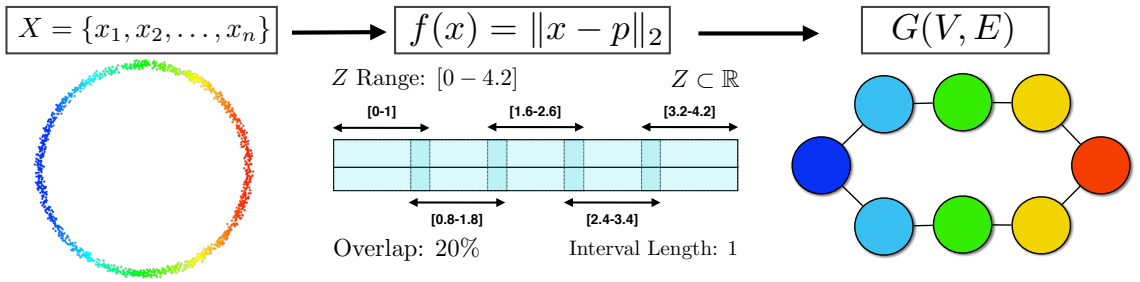

Figure 2. *Mapper ring data example discussed in Example 1.1.*

data into subsets based on which set of the covering they lie within. Note that, because the subsets of this cover generally overlap, any given data point may be within multiple subsets of the cover. Now, 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 subsets are represented as 0-simplexes, the interaction of these prototypes with other 0-simplexes form the higher dimensional simplexes in the resulting simplicial complex. 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 how `Mapper` works, consider the example given by Singh *et. al* in [10], 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. Parameter Selection. Many of the steps of the `Mapper` algorithm require choices from the user. This includes the reference map (step 1), and the clustering algorithm (step 4). Although these choices are important, we will not focus on how to select these for `Mapper`. Once they are defined, but before partial clustering can be applied, it's necessary to choose how to discretize the data into overlapping subsets appropriately. This manifests algorithmically in the choice of method to cover the embedded topological space (step 2-3).

There are numerous ways to construct a suitable covering, and `Mapper` makes no assertions restricting the covering chosen, so long as the covering is finite. The default approach, and the one used most often in practice [] is to consider a rectangular covering $\mathcal{U}[\boldsymbol{r}, \boldsymbol{e}]$ consisting of intervals of the form $[\boldsymbol{kr} - \boldsymbol{e}, (\boldsymbol{k}+1)\boldsymbol{R} + \boldsymbol{e}]$, where $\boldsymbol{R}$ and $\boldsymbol{e}$ are positive real numbers. Products of these intervals give a multidimensional covering of $\mathbb{R}^d$, whose corresponding index set A

of $\mathcal{U}$ is given by the d-fold Cartesian product of $\{0, 1, \ldots, k\}$. `Mapper` takes as input a *fixed* setting of (R, e) as a means of constructing a covering over Z, and thus, a fixed simplicial complex. Since these scaling parameters necessarily describe *how* the simplicial complex is constructed, and because the simplicial complex summarizes an aspect of the metric structure for a given data set, it is essential that the analyst understand how the choice of covering affects the simplicial complex generated. Because iterating over the set of all possible (R, e) is infeasible (there are an ∞ set of possible settings), a method to compute the smallest set of parameter settings providing all possible simplicial complexes becomes crucial.

2. Cover parameterization. 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. We specifically consider a multi-dimensional hyper-rectangular covering method[2], parameterized internally by an interval length R and overlap value e as above. Additionally, we exclusively restrict our attention to the case where $e < \frac{R}{2}$. In practice, it's often the case the 1-skeleton of the simplicial complex is sufficient for further analysis and visualization, however we note that our derivations easily extend when higher dimensional simplicial complexes are considered. The hyper-rectangular covering method requires as input the number of rectangles k to distribute along each dimension, and an overlap percentage $\boldsymbol{g} = [g_1, g_2, \ldots, g_m]$ neighboring rectangles should intersect. We seek to compute the minimal set of input values $\boldsymbol{g}$ which realize all possible *distinct* simplicial complexes.

We begin by analyzing step 2 of the `Mapper` framework, which requires as input a vector representing the number of sets to distribute along each dimension of the filter space $\boldsymbol{k}$, and a vector representing the overlap provided between these sets $\boldsymbol{g}$, so that $0 < g_i < 0.5 \; \forall g_i \in \boldsymbol{g}$ and $k_i > 0 \; \forall k_i \in \boldsymbol{k}$. If $\boldsymbol{k}$ is fixed, then the *indices* of the index set A remain constant, even though the collection of subsets that enumerate the cover $\mathcal{U}$ of Z will inevitably vary with the setting of $\boldsymbol{g}$. However, there may be values of $\boldsymbol{g}$ which produce identical simplicial complexes. For example, consider a covering constructed from $\boldsymbol{g} = [g]$ and another from $\boldsymbol{g}' = [g + \epsilon]$ for sufficiently small ϵ. It is possible that the set membership of each subset of $\mathcal{U}$ then may not change. This motivates a definition of *distinction* between possible realized values of $\boldsymbol{g}$.

Definition 2.1 (Distinct Simplicial Complex). *Given a set of values $Z \in \mathbb{R}^m$ and two covers $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ and $\mathcal{V} = \{V_\alpha\}_{\alpha \in A}$ of Z, denote the corresponding membership sets of $Z \cap \mathcal{U}_\alpha$ and $Z \cap \mathcal{V}_\alpha$ as $Z_{\mathcal{U}_\alpha}$ and $Z_{\mathcal{V}_\alpha}$, respectively. We define the simplicial complex $N(\mathcal{U})$ as being distinct from the simplicial complex $N(\mathcal{V})$ if:*

$$\text{Any } Z_{\mathcal{U}_\alpha} \neq Z_{\mathcal{V}_\alpha} \; \forall \; \alpha \in A \tag{1}$$

Thus, given two covering with the same index set, if there is at least one subset between them whose membership set is not equal, then the resulting simplicial complexes will be distinct.

Given a definition of what constitutes a 'distinct' parameterization of $\boldsymbol{g}$, we define some of the preliminary variables required by the `Mapper` algorithm. Assume, a reference map $f : X \to Z$ where $Z \subset \mathbb{R}^m$ is a set of n m-tuples representing the transformed coordinatization

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

of X. Denote the i^{th} m-tuple of Z as $\boldsymbol{z}_i$, and its j^{th} component as $z_i^{(j)}$. Similarly, denote the j^{th} component Z as $\boldsymbol{z}^{(j)}$. The extrema vectors of Z are defined as follows:

$$\boldsymbol{z_{max}} = [\ \max(\boldsymbol{z}_1^{(1)}), \max(\boldsymbol{z}_2^{(2)}),\ \ldots, \max(\boldsymbol{z}_n^{(m)})\ ]$$
$$\boldsymbol{z_{min}} = [\ \min(\boldsymbol{z}_1^{(1)}), \min(\boldsymbol{z}_2^{(2)}),\ \ldots, \min(\boldsymbol{z}_n^{(m)})\ ]$$

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

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

Assume a rectangular covering $\mathcal{U}[R, e]$ is constructed over Z whose index set 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 $\kappa = [0, 1, \ldots, m-1]$. Given an overlap percentage $\mathbf{g}$, a suitable covering of Z can be parameterized by $(R = \mathcal{Z} \circ (k - \mathbf{g}(k-1))^{-1}, e = R(1-\mathbf{g}))$. We'll refer to R as the *interval length* of a given covering, and e as the *step size* (which is proportional to the overlap $\mathbf{g}$). The goal is to determine *every* value of $\mathbf{g}$ which produces distinct topological solutions.

Our derivation begins by considering the special case where the topological space Z has is zero-dimensional with respect to its Lebesgue covering dimension. This implies that each open set of the cover is disjoint, or equivalently $\boldsymbol{g} = 0$. Under this case, every point $z_i \in Z$ is contained in exactly one set of the cover. This implies that each z_i is mapped by exactly one tuple $\boldsymbol{\alpha} \in A$. Let this initial mapping be expressed as $\psi : Z \to \boldsymbol{\alpha}$, where $\boldsymbol{\alpha} \in A$, and denote $A(Z)$ as the set of mappings by ψ over a set of n data points:

$$A(Z) = \{\boldsymbol{\alpha_0}, \boldsymbol{\alpha_1}, \ldots, \boldsymbol{\alpha_n}\} \quad (\text{where } \ \boldsymbol{\alpha} \in A,\ \boldsymbol{\alpha} \in \mathbb{Z}^{1\times m}) \tag{2}$$

Given $\boldsymbol{g} = 0$, the corresponding interval length is $R = \mathcal{Z} \circ k^{-1}$ (with step size $e = 0$). This establishes a 'base' interval length, which we'll denote as $\bar{r}$, and establishes a lower bound on the interval length (any interval length below this value is not a cover of Z). A simple example of one possible realization of these variable is given in Example 1.1.

Example 1.1: Illustrating the covering

Consider a set of points $Z_n \subset \mathbb{R}^2$ with the defined range vector $\mathcal{Z}_n = [2, 4]$. Suppose $\boldsymbol{k} = 2$, i.e. $\boldsymbol{k} = [2, 2]$. A valid covering can be constructed $\mathcal{U}[\bar{r}, 0]$ where $\bar{r} = [1, 2]$. The index set $A = \{(0,0), (0,1), (1,0), (1,1)\}$ is produced by the cartesian product $k \times k$. For all $z_i \in Z$, the ordered regions, or *level sets* A corresponds to the set of intervals $\{\{[0,1],[0,2]\}, \{(1,2],[0,2]\}, \{[0,1],(2,4]\}, \{(1,2],(2,4]\}\}$, respectively. Note that all of the coverings of $\mathcal{U}[\bar{r}, \epsilon]$ for different values of ϵ have the same indexing set [10].

For a given point z_i to intersect another set outside of its originating set, we must consider interval lengths larger than $\hat{\boldsymbol{r}} > \bar{r}$. Before considering what this larger interval length may be for a given point, it's necessary to know the *relative* position of each point in Z within

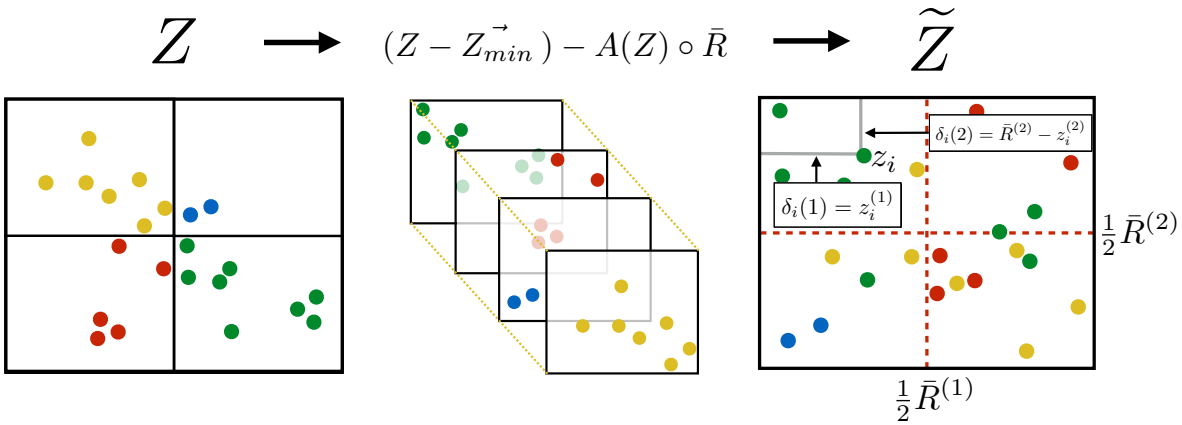

Figure 3. *The left panel shows the mapping of a given data set $X \to Z$ via f (where $Z \in \mathbb{R}^2$) and four overlapping sets that construct the covering $\mathcal{U}$ of Z. The center figure shows the isolation the points in each of these sets that is used for partial clustering and the connected components represent the results of this clustering. The final figure on the right represents the 1-skeleton realization of the simplicial complex constructed from the nerve of the covering.*

it's originating set. To determine this, we consider a projection $z_i^{(d)}$ of a point z_i onto a single index set, such that $0 \leq z_i^{(d)} \leq \bar{r}^{(d)} \; \forall \; d \in \{1, 2, \ldots, m\}$. The projected positions, $\widetilde{Z}$, is computed as follows:

$$\widetilde{Z} = (Z - \boldsymbol{z_{min}}) - A(Z) \circ \bar{\boldsymbol{r}} \quad (\text{where} \quad \widetilde{Z} \in \mathbb{R}^{n \times m}) \tag{3}$$

Equation 3 describes $\widetilde{Z}$ as a translation of Z to the origin, followed by a projection to the level set restricted to $\bar{r}$. This transformation preserves the *relative* position of all points, such that the metric distance between any pair of points from the same originating level set is unchanged. Given $\widetilde{Z}$, we can collectively compute the distance of every point in Z 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 $\boldsymbol{z}_i$. If $\tilde{z}_i$ lies within the upper halfspace along dimension d, then the distance to the nearest adjacent level set is given by $\bar{r}^{(d)} - \tilde{\boldsymbol{z}}_i^{(d)}$. Thus, since these distances are inversely proportional, whichever halfspace $\tilde{z}_i^{(d)}$ lies within, the distance to the closest adjacent level set is given by the minimum of the two values: $\delta(\tilde{\boldsymbol{z}}_i) = \min(\tilde{z}_i^{(d)}, \bar{r}^{(d)} - \tilde{\boldsymbol{z}}_i^{(d)})$. An visual illustration of this projection is shown in Figure 3.

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 $\mathcal{U}$. They can be computed simply:

$$\hat{\boldsymbol{r}} = \bar{\boldsymbol{r}} + \min(\widetilde{Z}, \bar{\boldsymbol{r}} - \widetilde{Z}) \quad (\text{where} \quad \hat{\boldsymbol{r}} \in \mathbb{R}^{1 \times m}) \tag{4}$$

Since the step size e is a function of $\boldsymbol{g}$ and R, and because $\boldsymbol{k}$ is fixed, knowledge of the smallest

R vector or 'rectangle' which causes a distinct set membership change in the cover enables $\boldsymbol{g}$ to be computed in closed-form. Recall that `Mapper` takes as input the number of intervals to divide the filter space along each dimension, $\boldsymbol{k}$, and the overlap (percentage) between these sets $\boldsymbol{g}$. Given these parameters, the properties of the cover r and e can be computed. We take the inverse, supply our computed values of $\hat{\boldsymbol{r}}$ to recover $\boldsymbol{g}$. The derivation is given in the appendix subsection 5.1, but the final solution is as follows:

$$\mathcal{G} = \frac{(\overline{\boldsymbol{z}} - \hat{\boldsymbol{r}}\boldsymbol{k})}{\hat{\boldsymbol{r}}(-\boldsymbol{k}+1)} \quad (\text{where } \mathcal{G} \in \mathbb{R}^{n\times m}) \tag{5}$$

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

$$\hat{e} = \hat{\boldsymbol{r}}(1-\mathcal{G}) \tag{6}$$

Note that $\mathcal{G}$ is an $n \times m$ matrix expressing the overlap values of each component. Specifically, entry $\mathcal{G}_{ij}$ corresponds to the scalar value $g_i^{(j)}$ expressing the smallest overlap percentage necessary for the corresponding point $\boldsymbol{z}_i$ to intersect it's adjacent level set, in the j^{th} direction. Respective maximums of selected components give the corresponding overlap percentages for level sets that lie in non-orthogonal directions.

$\mathcal{G}$ in and of itself is useful in many ways. For example, consider the situation where one wishes to compute higher order simplexes, or even the full simplicial complex of the covering. In this situation, although $\boldsymbol{g}$ is not bounded above by 0.50, one can still use this process to aid in computing realizations of higher order k-skeletons (e.g. when $k > 1$). As an example, consider the values of $\mathcal{G}$ derived when $\hat{\boldsymbol{r}}$ are computed as follows:

$$\hat{\boldsymbol{r}} = (\overline{\boldsymbol{r}} + \min(\widetilde{Z}, \overline{\boldsymbol{r}} - \widetilde{Z})) + m\overline{\boldsymbol{r}} \quad (\text{where } \hat{\boldsymbol{r}} \in \mathbb{R}^{1\times m}) \tag{7}$$

where $m \in \mathbb{Z}$. Let m be any integer value, say $m \geq 1$. This would derive the overlap values necessary to order higher order simplexes. The computational gains are discussed more in section 3.

3. Implementation and Discussion.

In 2013, Müllner et. al introduced the *Python Mapper* software tool [9] 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). All of the derived equations are available in a Python script as an extension to this library, available open-source online. In the following subsections, we discuss a potential use-case scenarios where a knowledge of the potential covering parameter values may be beneficial for a application-specific analysis.

3.1. Comparing Mapper Solutions.

The projection to base level set $\widetilde{Z}$ not only allows for a unified treatment of all of the coordinate values of a given reference set Z, but it also enables the a means of computing the resulting simplicial complex for all $\mathbf{g}$ values with a much lower runtime complexity.

Consider using the full `Mapper` algorithm, as given in the *Python Mapper* tool, to compute the full simplicial complex with two different overlap parameterizations, say, g_1 and g_2). As before, assume a fixed k. In the first overlap parameterization (g_1), a cover is constructed, the

points in each level set are independently clustered into 'nodes.' Each pairwise combination of nodes is checked to see if the intersection between them is non-empty. Since there are $\binom{n}{2}$ such combinations, where n = total number of nodes, this requires $O(n^2)$ tests.

Now suppose one wishes to compare the simplicial complex computed using g_1 against a simplicial complex computed using g_2. Because the partial clustering depends on the dissimilarity between points in each level set, and because this dissimilarity changes with the addition (if $g_2 > g_1$) or removal (if $g_2 < g_1$) of points to their respective level sets, whenever the membership set of *any* level set changes from one overlap parameterization to another, the clustering of these level sets must be recomputed to account for this change. In the current open source implementations, this implies the entire `Mapper` algorithm must be re-run completely.

This can be markedly improved, namely through an *iterative* processing step for `Mapper`. Before explaining how this may be done, a useful analogy to understanding the motivation where one starts with a covering of disjoint rectangles ($g = 0$), and imagine "expanding" the interval length of these rectangles until the appropriate maximum length is reached ($\hat{\boldsymbol{r}} = \frac{e}{2}$ is the 1-skeleton of the simplicial complex is sufficient). First, compute the full `Mapper` solution with a fixed k and under the parameterization where $g = 0$ and with a rectangular covering $\mathcal{U}[R = \mathcal{Z} \circ k^{-1}, e = 0][$, such that every point is in exactly one subset of the covering, and every subset in the cover is disjoint. Compute, using the solution given in Section 2, the possible values of g which produce distinct topological solutions. Each value g_i of $\mathbf{g}$ has a corresponding point in Z, where g_i expresses the smallest overlap values where, per dimension, the point will intersect an adjacent level set. Assume $g_1 < g_i <= g_2$. Then this implies that, minimally, the level set that the point intersects with at g_i must have it's simplexes updated in the simplicial complex generated from g_1, but *only* the simplexes originating from the newly updated level set. If $e < \frac{R}{2}$, these are simply the sets adjacent to $A(Z)$ in the direction of the halfspace each point in $\widetilde{Z}$ lies:

$$\hat{A} = A(Z) + \Gamma(Z) \quad \text{subject to } \hat{A} \in A(Z)$$
$$\text{where } \Gamma(Z) = \begin{cases} 1 & \text{if } \bar{r} - \widetilde{Z} < \frac{\hat{\boldsymbol{r}}}{2} \\ -1 & \text{if } \bar{r} - \widetilde{Z} \geq \frac{\hat{\boldsymbol{r}}}{2} \end{cases} \tag{8}$$

4. 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.

5. Appendix.

5.1. Overlap Inverse. The motivation is that, traditionally, given the number of intervals k and g, one can compute the subsequent values of (R, e) required to create the rectangular covering $\mathcal{U}[R, e]$ of Z. These parameter are traditionally as follows:

$$R = \frac{\mathcal{Z}}{(k - g(k-1))}, \; e = R(1-g) \tag{9}$$

Since e is dependent on g, and assuming k is fixed, it's clear that all one needs to compute g is a given interval length R. This is shown below:

$$\begin{aligned} R(k - g(k-1)) &= \mathcal{Z} \\ Rk - Rgk + Rg &= \mathcal{Z} \\ g(-Rk + R) &= \mathcal{Z} - Rk \\ g &= \frac{\mathcal{Z} - Rk}{R(-k+1)} \end{aligned} \tag{10}$$

It's of course possible that a user of `Mapper` could supply the interval length R and discretization parameter k instead of the overlap g, and then compute the covering. The problem is determining what values of R are relevant. Since the range of Z will inevitably differ with respect to the data set and reference map used, it may not be clear what values of R are relevant. On the contrary, g will always be between $(0, 0.5]$, which is perhaps why it's preferred by the open-source implementations of `Mapper`. We consider an alternative approach which deterministically computes R, and as a result, possible values of g.

REFERENCES

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

[2] 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.

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

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

[5] 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.

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

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

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

[9] 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).

[10] 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.

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