Adaptive Covers for Mapper Graphs Using Information Criteria

Nithin Chalapathi
University of California, Berkeley
Berkeley, CA, USA
nithinc@berkeley.edu

Youjia Zhou
University of Utah
Salt Lake City, UT, USA
zhou325@sci.utah.edu

Bei Wang
University of Utah
Salt Lake City, UT, USA
beiwang@sci.utah.edu

***Abstract*—The mapper construction is a widely used tool from topological data analysis in obtaining topological summaries of large, high-dimensional point cloud data. It has enjoyed great success in data science, including cancer research, sports analytics, and visualization. However, developing practical and automatic parameter selection for the mapper construction remains a challenging open problem for both the topological analysis and visualization communities. In this paper, we focus on parameter selection for the 1-dimensional skeleton of the mapper construction, called the mapper graph. Specifically, we explore how information criteria used in the X-means clustering algorithm can inform and generate adaptive covers for mapper graphs. Our approach thus makes novel progress towards automatic parameter selection for the mapper construction using information theory.**

***Index Terms*—Topological data analysis, topology in visualization, mapper, information theory**

I. Introduction

The amount of high-dimensional data has increased at an unprecedented rate, as has the demand for data science tools to explore such data. Topological data analysis (TDA) offers a rich set of tools for the exploratory analysis of high-dimensional data (see [1], [2] for surveys). One successful tool is the *mapper construction*, a topological summary of data in the form of a simplicial complex.

First introduced by Singh *et al.* [3], the mapper construction visualizes the topological structures of high-dimensional point clouds. At its core, it produces a function-induced soft-clustering that captures topological information about the data, such as branches or loops in high dimensions. The mapper construction has seen success across a variety of fields, including network visualization [4], cancer research [5], [6], neuroscience [7], machine learning interpretability [8], and more [9]. In recent years, the ecosystem for the mapper framework has rapidly developed in the form of open source libraries and interactive visualization tools, *e.g.*, *KeplerMapper* [10], *giotto-tda* [11], *Mapper Interactive* [12], *TDAview* [13], and *Guhdi* [14].

One major obstacle encountered by applications of the mapper construction and an active research topic in TDA is parameter selection and tuning, specifically the creation of a *cover*. We focus on parameter selection for the 1-dimensional skeleton of the mapper construction, referred to as the *mapper graph*. Carrière *et al.* [15] made some recent progress in this direction by introducing a statistical procedure for selecting the most stable parameters for the mapper graph, under certain assumptions (see Sect. III for details). Different from previous approaches, **we adapt information-theoretic measures used in X-means clustering to inform and generate adaptive covers for mapper graphs**.

Our contributions are as follows:

- First, inspired by X-means clustering, we propose a new strategy to generate adaptive covers for mapper graphs, referred to as the *multi-pass AIC/BIC* algorithm, based on the Akaike information criterion (AIC) and Bayesian information criterion (BIC).
- Second, we demonstrate our algorithm by generating adaptive covers for several synthetic and real-world datasets under the mapper framework.
- Lastly, we provide an open source implementation of our framework that is distributed both as a stand-alone Python package (https://github.com/tdavislab/mapper-xmean-cover), and as an extension to the *Mapper Interactive* toolbox [12] (https://github.com/MapperInteractive/MapperInteractive/tree/xmeans).

II. Technical Background

We first review the mapper construction as introduced by Singh *et al.* [3]. We focus on the 1-dimensional mapper construction, referred to as the mapper graph. We then discuss the X-means clustering that utilizes information criteria to determine the number of clusters in a dataset. X-means clustering inspires our proposed approach.

A. Mapper Graphs

Covers and nerves. To describe the mapper construction, we begin with the notions of a *cover* and the *nerve* of a cover. Given a high-dimensional point cloud $\mathbb{X} \subset \mathbb{R}^d$, a *cover* $\mathcal{V} = \{V_i\}_{i \in I}$ of $\mathbb{X}$ is an indexed set of open sets of $\mathbb{R}^d$ such that $\mathbb{X} \subset \bigcup_{i \in I} V_i$. The *nerve* $\mathcal{N}$ of $\mathcal{V}$ contains all finite subsets $J \subseteq I$ such that the intersection of the V_i whose indices are in J is non-empty.

In general, $\mathcal{N}$ is an abstract simplicial complex called the *nerve complex* of $\mathcal{V}$. The 1-dimensional skeleton of $\mathcal{N}$, denoted as $\mathcal{N}_1$, is a graph referred to as the *mapper graph*, which captures the intersections between elements of $\mathcal{V}$. Each node in $\mathcal{N}_1$ represents a cover element, while there is an edge between vertices i and j, if and only if $V_i \cap V_j$ is nonempty.

As illustrated in Fig. 1A, $\mathbb{X}$ is a 2-dimensional point cloud covered by a set $\mathcal{V}$ of rectangles as the cover elements. The 1-dimensional nerve $\mathcal{N}_1$ of $\mathcal{V}$ is shown in Fig. 1C, where there is an edge between nodes 1 and 3 since $V_1 \cap V_3 \neq \emptyset$.

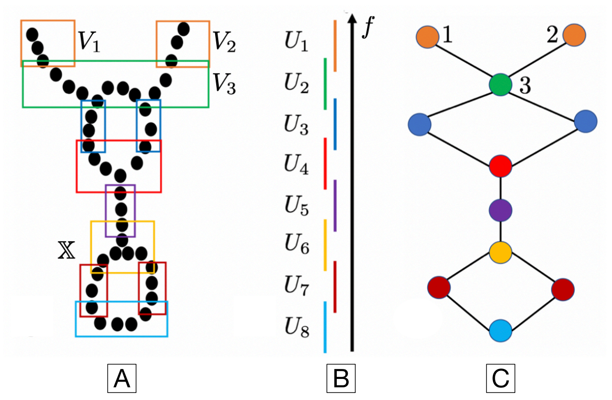

Fig. 1. An example mapper graph. (A) A 2-dimensional point cloud $\mathbb{X}$. (B) A height function $f : \mathbb{X} \to \mathbb{R}$ together with a cover of $f(\mathbb{X})$ consisting of intervals. (C) The resulting mapper graph.

Mapper graphs. Finding an appropriate cover $\mathcal{V}$ of $\mathbb{X}$ that captures its underlying structure is nontrivial. In the simplest case, given a point cloud $\mathbb{X} \subset \mathbb{R}^d$ equipped with a continuous function $f : \mathbb{X} \to \mathbb{R}$, a mapper construction induces a cover of $\mathbb{X}$ using f. Specifically, starting with a cover $\mathcal{U} = \{U_j\}_{j \in J}$ of $f(\mathbb{X})$, we can create a cover $\mathcal{V}$ of $\mathbb{X}$ by considering the clusters induced by $f^{-1}(U_j)$. In practice, a density based clustering algorithm, DBSCAN [16] is typically employed to cluster points in $f^{-1}(U_j)$. That is, for each $U_j \in \mathcal{U}$, we treat the clusters of $f^{-1}(U_j)$ (obtained by DBSCAN) as cover elements of $\mathbb{X}$ that constitute $\mathcal{V}$. The *mapper graph* is defined as the 1-dimensional nerve of $\mathcal{V}$, $\mathcal{M}(\mathbb{X}, f) := \mathcal{N}_1(\mathcal{V})$.

Take Fig. 1 as an example. A 2D point cloud $\mathbb{X}$ is equipped with a height function $f : \mathbb{X} \to \mathbb{R}$. We start with a cover of $f(\mathbb{X})$ with 8 open intervals of uniform lengths, $f(\mathbb{X}) \subseteq \bigcup_{j=1}^{8} U_j \subset \mathbb{R}$ in Fig. 1B. For each U_j, $f^{-1}(U_j)$ induces some clusters that are subsets of $\mathbb{X}$. These clusters form cover elements of $\mathbb{X}$. For instance, $f^{-1}(U_1)$ induces two clusters of points enclosed by the orange rectangles V_1 and V_2 respectively; while $f^{-1}(U_2)$ induces a single cluster enclosed by the green rectangle V_3, see Fig. 1A. The mapper graph in Fig. 1C is the 1-dimensional nerve of this cover and it captures the overall shape of the point cloud, including the two upward branches and two loops.

Parameters for mapper graphs. In practice, a practitioner needs to specify a number of parameters to construct a mapper graph: the function f, the number of cover elements l and their percentage of overlap p, the metric $d_{\mathbb{X}}$ on $\mathbb{X}$, and the clustering method. The function f is typically referred to as the *filter function* that captures either a known data attribute or the properties of the underlying point cloud, *e.g.*, the L_2-norm, geodesic distances, and eccentricity [3], [17]. For example, in Fig. 1, f is the height function, $l = 8$ and $p = 25\%$, $d_{\mathbb{X}}$ is Euclidean, and the clustering method is DBSCAN. The DBSCAN algorithm comes with two additional parameters: ε is the neighborhood size of a given point, and *minPts* is the minimum number of points needed to consider a set of points as a cluster.

In this paper, we study the parameter selection of the number of intervals l that specifies a cover for a mapper graph, assuming all other parameters (f, p, $d_{\mathbb{X}}$, ε, *minPts*, etc.) are fixed. The standard way to construct the cover $\mathcal{U}$ of $f(\mathbb{X})$ is by splitting the range of $f(\mathbb{X})$ into l equally sized intervals, each with p percentage overlap between adjacent intervals. We require the intervals in $\mathcal{U}$ to overlap in order to capture relations among the cover elements. This strategy of creating $\mathcal{U}$ is called the *uniform cover*. A second but less widely used strategy is the *balanced cover*, where the inverse image of each interval contains an equal number of points; such a strategy is offered by libraries such as the *giotto-tda* [11].

B. *X-means Clustering*

***k*-means clustering** is one of the most popular clustering algorithms [18], [19]. Determining the number of clusters k for k-means clustering is a widely discussed topic with established approaches such as cross validation [20], the silhouette score [21], and various indices [22].

Given a set of points $\mathbb{X} = \{x_1, x_2, \cdots, x_n\}$, the standard k-means clustering initializes a group of k randomly selected centroids and then performs an interactive process to optimize the centroids until they have stabilized or a pre-defined number of iterations have been reached. During each iteration, the algorithm proceeds by:

1) Assigning each point x_i to the centroid that is closest to x_i.
2) Updating centroids by computing the center of mass of points assigned to it.

However, there are several issues surrounding the k-means algorithm: the problem is NP-hard [23] (although it typically works fast in practice); it does not guarantee convergence to the global optimum; and determining the number of clusters k is often ambiguous.

***X*-means clustering** is a variant of k-means clustering, which attempts to address the parameter selection of k. It refines clusters by repeatedly attempting subdivision, and keeping the best resulting splits according to an information-theoretic criterion such as the Akaike information criterion (AIC) or Bayesian information criterion (BIC) [24].

To apply X-means clustering to a dataset $\mathbb{X}$, the user specifies as input a range $[k_{min}, k_{max}]$ in which the *true* number of clusters k reasonably lies. And the output is a value for $k \in [k_{min}, k_{max}]$ which scores best with a model selection criterion such as AIC or BIC [24].

At its core, X-means consists of two alternating operations: a k-means algorithm (`Improve-Params`) to determine the clusters for a chosen k, and a clustering splitting process (`Improve-Structure`) that optimizes the value of k according to the AIC/BIC. Starting with an initial choice of $k = k_{min}$, X-means runs the following steps until completion [24]:

1) `Improve-Params`: running standard k-means clustering until convergence.

2) `Improve-Structure`: (a) splitting the centroid of each cluster $\mathbb{X}_j$ into two children; (b) running a local 2-means algorithm for each pair of children in the region defined by the cluster $\mathbb{X}_j$; (c) performing model selection on all pairs of children: for each pair, if the AIC/BIC does not increase locally, the parent centroid is kept; otherwise, the children are kept and $\mathbb{X}_j$ is split into two, and k is updated accordingly (*i.e.*, $k \leftarrow k+1$).
3) If $k > k_{max}$, stop and report the best scoring model; otherwise go to step 1.

Our method is inspired by X-means' utility in parameter selection. In particular, we utilize and modify the `Improve-Structure` process for X-means to formulate adaptive covers for mapper graphs.

To clarify, there are two different clustering algorithms relevant to our pipeline. To construct a mapper graph with a given cover $\mathcal{U}$ of $f(\mathbb{X})$, DBSCAN is used to cluster points in $f^{-1}(U_j)$ (for each $U_j \in \mathcal{U}$) such that these clusters form a cover $\mathcal{V}$ of $\mathbb{X}$. The `Improve-Structure` process of X-means is used to determine whether a particular cover element U_j of $f(\mathbb{X})$ should be split or not during the cover refinement process (see Sect. IV for details).

AIC and BIC. Kullback-Leibler (KL) divergence (or relative entropy) [25] is at the center of information theoretic measures such as AIC and BIC. It provides a way to compare two probability distributions. Both AIC and BIC are commonly used for model selection to estimate the relative amount of information lost by a given model. In the X-means clustering method, assume we are given a dataset $\mathbb{X}$ and a family of models, where each model M corresponds to a clustering of $\mathbb{X}$ with different values of k. For a fixed model M, BIC is formulated as [24], [26], [27],

$$\mathrm{BIC}(M) = \hat{l}(M;\mathbb{X}) - \frac{\theta}{2}\log|\mathbb{X}|.$$

where $\hat{l}$ is the maximum log likelihood estimation of M given dataset $\mathbb{X}$ and θ is the number of parameters estimated in M. Similarly, AIC [28], [29] is defined as,

$$\mathrm{AIC}(M) = 2\hat{l}(M;\mathbb{X}) - 2\theta.$$

The log likelihood term $\hat{l}$ measures how likely a model is given the observed data, whereas the θ term penalizes models with more parameters, and hence discourages overfitting. For X-means, the model with the highest AIC or BIC score is selected[1].

In our setting, let $\mathbb{X} = \{x_1, \cdots, x_n\}$ be a point cloud of n points in d dimensions, we derive BIC and AIC based the work of Pelleg and Moore [24] with a minor correction based on multivariate Gaussian distribution [30], [31] (see Sect. A). In a nutshell, given a mapper graph with a fixed cover, we convert such a mapper graph into a hard clustering of points and evaluate its AIC/BIC scores. Following the derivation details in Sect. A, we have $\theta = (d+1)k$, $\mathbb{X}_{(i)}$ is the cluster assigned to x_i, $\hat{\sigma}$ is the maximum likelihood estimator for the variance under a spherical Gaussian assumption, and

$$\hat{l}(M;\mathbb{X}) = \sum_{i=1}^{n}\left[\log\frac{|\mathbb{X}_{(i)}|}{|\mathbb{X}|} - d\cdot\log(\sqrt{2\pi}\hat{\sigma})\right] - \frac{|\mathbb{X}|\cdot d}{2}.$$

III. Related Work

The mapper construction has been a popular and effective tool from TDA for capturing topological summaries of complex data [6], [32]–[34]. There are several open source implementations for computing and visualizing mapper graphs for general purposes [10]–[14], and for domain-specific applications [35], [36]. A number of theoretical questions surrounding the mapper construction are being studied by the TDA communities regarding its information content (*e.g.*, [37]–[39]), stability (*e.g.*, [38]), convergence (*e.g.*, [3], [37], [38], [40], [41]), and comparative measures (*e.g.*, [42]–[46], see [47], [48] for a survey).

However, parameter selection for the mapper algorithm, in particular, the selection of covers, remains an active topic of research. One common strategy is to employ a *best practice*, which is to find a range of parameters where the mapper graphs remain structurally stable. In the classic mapper construction [3], the *uniform cover* consists of a number of equal-sized intervals with a fixed overlap rate between successive intervals. The *giotto-tda* library [11] enables a *balanced cover*, where the length of intervals is adjusted so that approximately the same number of points fall into each interval.

Carrière *et al.* [15] introduced a statistical procedure for selecting the most stable parameters for the mapper graph. They assumed that the data is sampled under a well-behaved probability distribution, *i.e.*, an (a,b)-standard probability distribution, where a, b are some constant parameters. Their algorithm repeatedly computes the mapper graph over a sample of the data while comparing the extended persistence [15] between the mapper graph and its continuous analogue, the Reeb graph [49]. The mapper graph is induced by a δ-neighborhood graph of the input point cloud, where the threshold δ is determined by a and b, and the length of intervals is determined by the maximum value of filter function differences between any two points that have a distance less than or equal to δ. When a and b are unknown, a subsampling approach [50] is adapted to determine the threshold δ. They proved that the mapper graph is an optimal estimator for the Reeb graph [49]. However, their theoretical results are not guaranteed to hold when the Reeb graph is unknown or the independent sampling condition of the input point cloud is not satisfied.

Our framework, on the other hand, is a completely new approach based on information criteria and it does not make any assumptions on the input point cloud. To the best of our knowledge, this is the first time the information theory is used for automatically tuning the parameters of the mapper construction.

[1]In some literature, there is a factor of -1 for the definition of AIC/BIC; in this case, the computational process is the same but the model with the smallest score is selected instead.

IV. Method

To generate an adaptive cover for a mapper graph, we begin by articulating how to compute the AIC and BIC for a mapper graph and its subgraph. Then we describe our *multi-pass AIC/BIC* algorithm inspired by X-means.

A. Computing AIC or BIC of a Mapper Graph

Let $\mathbb{X} \subset \mathbb{R}^d$ be a point cloud with a filter function $f: \mathbb{X} \to \mathbb{R}$. Each point $x \in \mathbb{X}$ is a d-dimensional point with a vector of coordinates. We assume that $\mathbb{X}$ is equipped with the standard Euclidean metric. We convert a mapper graph $\mathcal{M}$ constructed from $(\mathbb{X}, f)$ to a hard clustering in order to compute its AIC or BIC.

Let V denote the set of nodes of $\mathcal{M}$ and $k = |V|$. We generate a centroid for points belong to each node of $\mathcal{M}$. That is, let $\mathbb{X}_j$ denote the set of points belong to a node in $\mathcal{M}$, its centroid is computed as $\frac{1}{|\mathbb{X}_j|}\sum_{x \in \mathbb{X}_j} x$. Then we assign each point in $\mathbb{X}$ to its nearest centroid. This is illustrated in Fig. 2. The resulting clustering in Fig. 2C contains 11 clusters; points are colored by their cluster ID with cluster centroids highlighted in red.

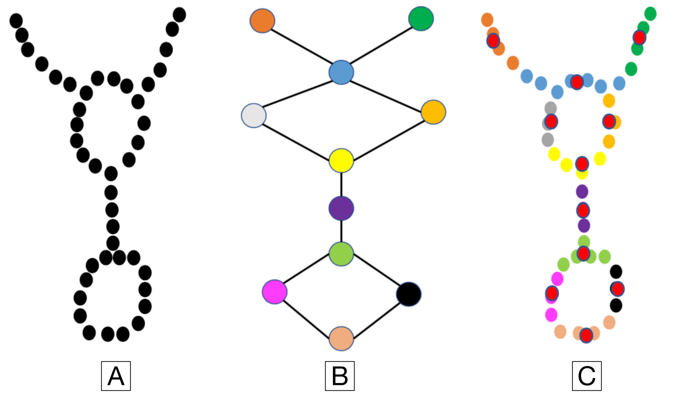

Fig. 2. Converting a mapper graph into a hard clustering. (A) The original point cloud of Fig. 1A. (B) The mapper graph in Fig. 1C, where nodes are colored by cluster ID. (C) The resulting hard clustering with cluster centroids highlights in red.

B. Generating Adaptive Covers

Our strategy is inspired by the `Improve-Structure` step of X-means, referred to as the *multi-pass AIC/BIC*. In the `Improve-Structure` phase, given a point cloud $\mathbb{X}$ with k clusters, a cluster $\mathbb{X}_j$ is treated as an independent point cloud, and a BIC (or AIC) score is computed before and after it is split into two clusters. The split is kept if it increases the information criterion.

Our algorithm starts with an initial mapper graph $\mathcal{M}$ on point cloud $\mathbb{X}$, with a filter function f, a cover of $f(\mathbb{X})$ with l intervals of uniform lengths, and p percent overlap. The *multi-pass BIC* iterates through the following steps until convergence using BIC. Assume intervals in $\mathcal{U}$ are sorted by their left endpoints. At each iteration, for a chosen unprocessed interval $U \in \mathcal{U}$, the algorithm proceeds by,

1) Considering the subgraph $\mathcal{M}_U$ of $\mathcal{M}$ induced by U (*i.e.*, $\mathcal{M}_U$ consists of nodes in $\mathcal{M}$ that correspond to clusters of $f^{-1}(U)$, and edges in $\mathcal{M}$ induced by these nodes) and computing the BIC of $\mathcal{M}_U$ as outlined in Sect. IV-A.
2) Splitting U into two overlapping intervals U' and U'' (with a p percent overlap) and computing a new mapper graph $\mathcal{M}'$ using the new cover $(\mathcal{U} \setminus U) \cup U' \cup U''$; see Fig. 3A-B.
3) Considering the subgraph $\mathcal{M}'_U$ of $\mathcal{M}'$ induced by $U' \cup U''$ and computing its BIC. The split is kept if it increases the information criterion, and $\mathcal{U}$ is updated and sorted; otherwise, the split is ignored.

The convergence criteria are reached when the number of iterations t has reached a pre-defined threshold (*e.g.*, $t \geq 100$), or when all intervals have been processed, and the BIC score has converged, that is, $BIC_t - BIC_{t-1} \leq \delta * BIC_{t-1}$ for some small percentage δ.

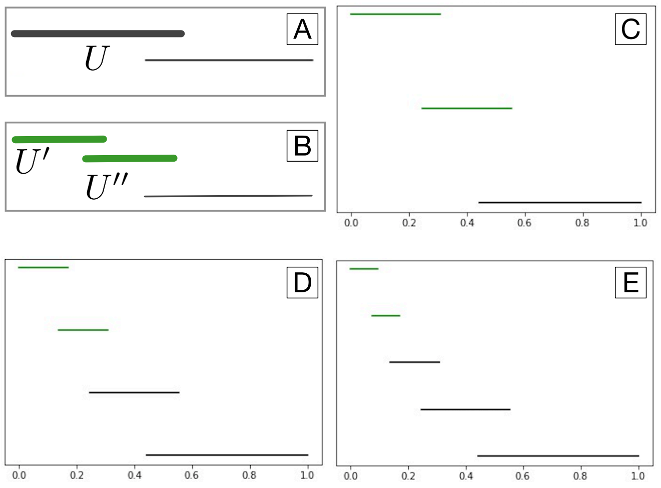

Fig. 3. (A)-(B) Splitting U into U' and U''. (C)-(E) First three iterations of a *DFS multi-pass BIC* starting with two initial intervals in (A).

The above *multi-pass AIC/BIC* algorithm has three variations, depending on how to choose an unprocessed interval $U \in \mathcal{U}$ at the beginning of each iteration. The *Random* strategy chooses an interval proportionally to its length, that is, a longer interval has a higher probability to be chosen. The *DFS* (depth-first search) strategy starts at the first (and left-most) interval for each iteration, and explores as deep as possible before backtracking and moving to the second interval and so on (see Fig. 3C-E for an example). The *BFS* (breadth-first search) strategy starts at the first interval and explores all intervals at the present depth. The interval with the largest change in AIC or BIC is split permanently. This process is then repeated until convergence.

V. Results

We apply our multi-pass AIC/BIC algorithm to several synthetic and real-world datasets. Using multi-pass AIC/BIC, we ask the following question: can we use information criteria to adaptively refine the cover of a mapper graph to better capture its underlying structure? The answer is yes.

A key takeaway is that our framework offers a practical and automatic way to adaptively adjust the cover for an initial mapper graph, thus obtaining a refined mapper graph that better captures the structure of its underlying dataset.

A. 2D Concentric Circles

Our first dataset consists of 2000 points sampled from two concentric circles with a filter function, the sum of x and y coordinates (Fig. 4A). Therefore the hand-tuned mapper graph (regarded as the “ground truth”) is expected to also contain two circles (Fig. 4B).

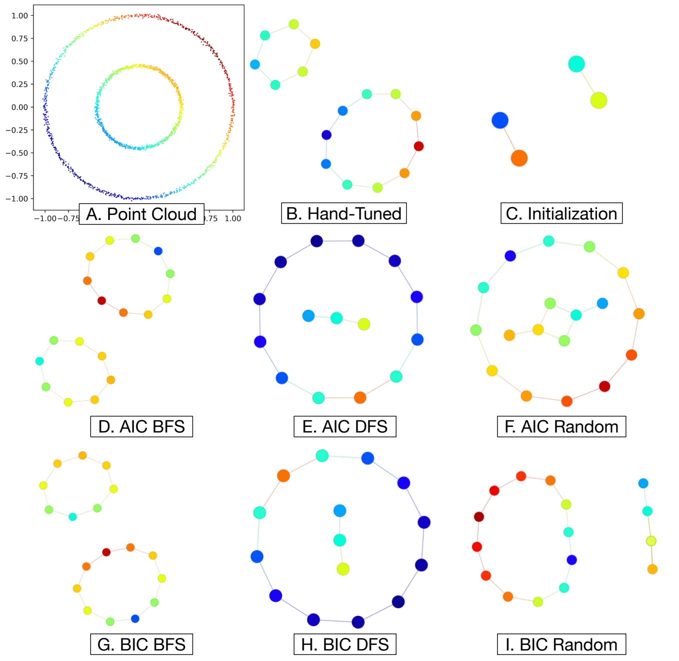

Fig. 4. *Circles* dataset. For mapper graphs, $p = 20\%$. For DBSCAN, $\varepsilon = 0.1$, and *minPts* $= 5$. (A) point cloud visualized as a scatter plot. (B) Hand-tuned mapper graph ($l = 6$). (C) Initial mapper graph ($l = 2$). (D-F) AIC results using BFS, DFS and Random strategies respectively. (G-I) BIC results using BFS, DFS and random strategies respectively. Nodes are colored by the (0-1 normalized) average value of the filter function (sum of x and y coordinates).

As illustrated in Fig. 4C, the initial mapper graph with $l = 2$ uniform intervals is too coarse to capture the circular structures of the underlying data. We then apply multi-pass AIC/BIC to the initial mapper graph. Fig. 4D-F are multi-pass AIC results using BFS, DFS and Random strategies respectively. We set $\delta = 0\%$ and $t = 50$. It can be seen that both BFS and Random strategies recover the two circles, and the DFS strategy recovers the outside circle only. Fig. 4G-I are multi-pass BIC results using BFS, DFS, and Random strategies, respectively. Using the BFS strategy, both circles are recovered, and the DFS and Random strategies recover the outside circle. This example indicates that there are fine differences among the three strategies of multi-pass AIC/BIC, and the effectiveness of the results are likely initialization-dependent and data-dependent.

B. 3D Human Dataset

We explore a second point cloud dataset from [51], which is a 3D human shape, see Fig. 5A. The dataset consists of 4706 points. The hand-tuned mapper graph (Fig. 5B) with a height (filter) function is expected to capture the head (the red branch), as well as the arms (the two green branches) and legs (the two blue branches) of the human.

We apply multi-pass AIC/BIC ($\delta = 0\%$, $t = 50$) to a very simple initial mapper graph. The initial mapper graph Fig. 5C with $l = 2$ uniform intervals does not capture any expected bodily features. The multi-pass AIC Random result in Fig. 5D is able to capture the head feature; however, the arms and legs are not clearly separated. On the other hand, the multi-pass BIC Random result in Fig. 5E is able to recover the head, arms, and legs of the human as expected.

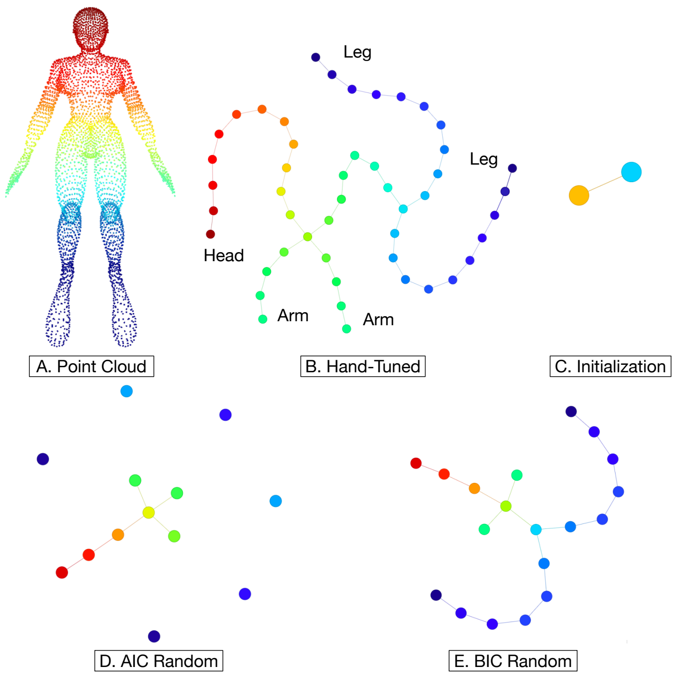

Fig. 5. *Human* dataset. For mapper graphs, $p = 40\%$. For DBSCAN, $\varepsilon = 0.1$, and *minPts* $= 5$. (A) Point cloud visualized as a scatter plot. (B) Hand-tuned mapper graph ($l = 30$). (C) Initial mapper graph ($l = 2$). (D) AIC Random (resulting in $l = 5$). (E) BIC Random (resulting in $l = 11$). Nodes are colored by the (0-1 normalized) average value of the height function with a rainbow colormap.

We show the distributions of cover elements in Fig. 6 before and after multi-pass AIC Random (left) and multi-pass BIC Random (right) strategies. The refined mapper graphs contain 5 (AIC) and 11 (BIC) uneven intervals, respectively.

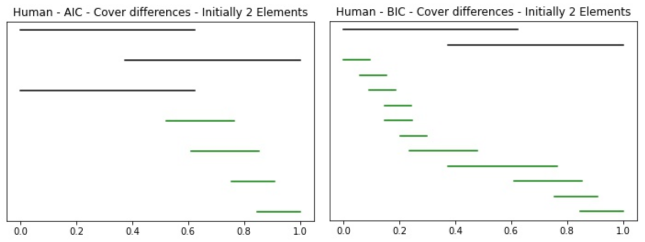

Fig. 6. *Human* dataset. Distributions of cover elements before and after multi-pass AIC Random (left) and multi-pass BIC Random (right), starting with $l = 2$ initial intervals.

C. 3D Horse Dataset

Our third dataset is a point cloud sampled from a 3D horse shape with 8430 points (Fig. 7). The filter function is the distance to the tail. The hand-tuned mapper graph captures different parts of the animal and their connectivities: the head, four legs, and a tail.

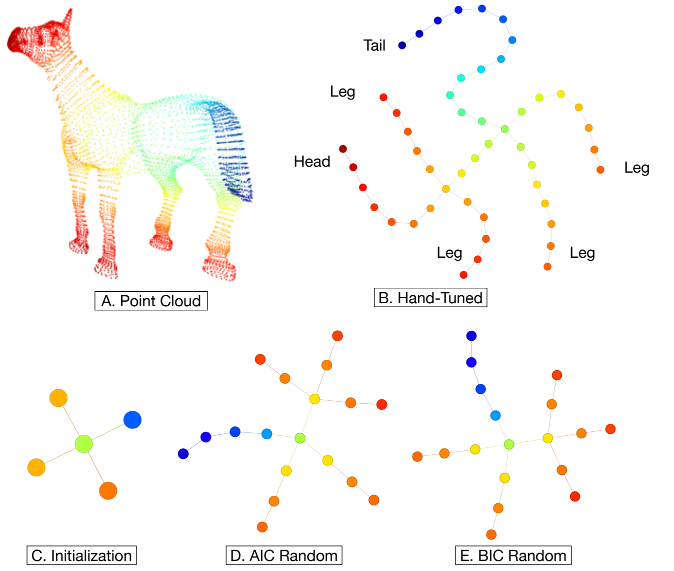

Fig. 7. *Horse* dataset. For mapper graphs, $p = 20\%$. For DBSCAN, $\varepsilon = 0.1$, and $minPts = 5$. (A) Point cloud visualized as a scatter plot. (B) Hand-tuned mapper graph ($l = 25$). (C) Initial mapper graph ($l = 3$). (D) AIC Random (resulting in $l = 8$) (E) BIC Random (resulting in $l = 8$). Nodes are colored by the (0-1 normalized) average value of the filter function (distance to the tail) with a rainbow colormap.

We show the distributions of cover elements in Fig. 8 before and after multi-pass AIC Random (left) and BIC Random (right) strategies. The refined mapper graphs both contain (identical) 8 intervals, respectively.

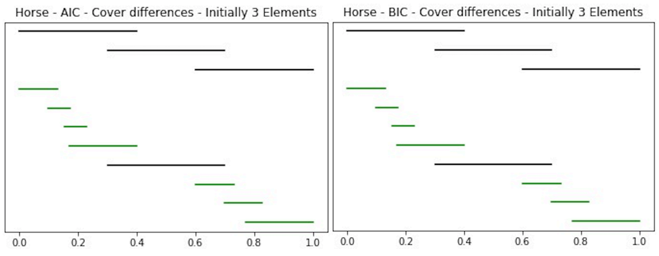

Fig. 8. *Horse* dataset. Distributions of cover elements before and after multi-pass AIC Random (left) and multi-pass BIC Random (right), starting with $l = 3$ initial intervals.

D. 3D Ant Dataset

Our next dataset is the point cloud of a 3D ant shape from [51]. The dataset consists of 6370 points. The filter function is the distance to the tip of its abdomen (Fig. 9A).

In the hand-tuned mapper graph, Fig. 9B, there are notable branches for the two antennas on the head, six legs, and one abdomen of the ant. The abdomen of the ant is represented by the blue branch. The refined mapper graphs using multi-pass AIC/BIC Random strategies are shown in Fig. 9D-E, respectively. The BIC Random strategy produces a mapper graph that resembles that of the hand-tuned version; while the AIC Random strategy over-segments the data (*i.e.*, some parts of the antenna appear to break off from the main body).

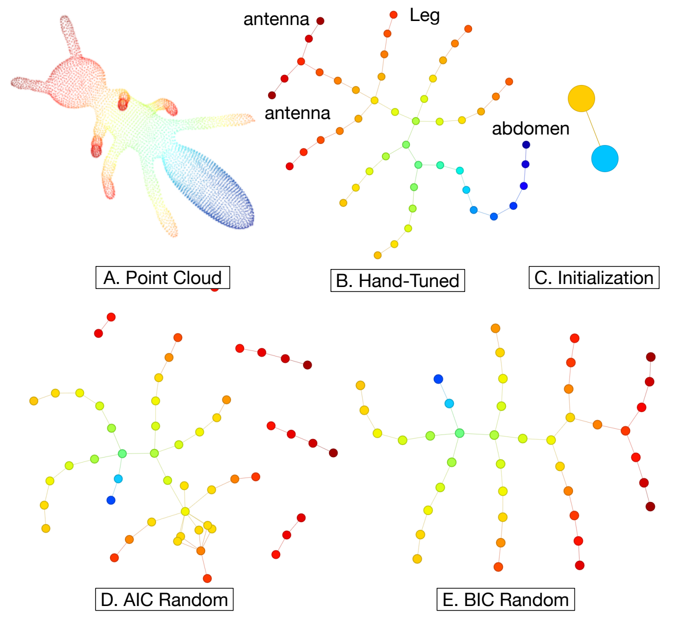

Fig. 9. *Ant* dataset. For mapper graphs, $p = 40\%$. For DBSCAN, $\varepsilon = 0.025$, and $minPts = 5$. (A) Point cloud visualized as a scatter plot. (B) Hand-tuned mapper graph ($l = 20$). (C) Initial mapper graph ($l = 2$). (D) AIC Random (resulting in $l = 14$) (E) BIC Random (resulting in $l = 12$). Nodes are colored by the (0-1 normalized) average value of the filter function with a rainbow colormap.

We show the distributions of cover elements in Fig. 8 before and after multi-pass AIC Random (left) and BIC Random (right) strategies, resulting in 14 and 12 intervals, respectively.

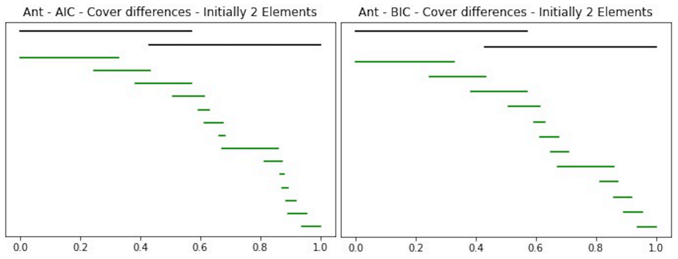

Fig. 10. *Ant* dataset. Distributions of cover elements before and after multi-pass AIC Random (left) and multi-pass BIC Random (right), starting with $l = 2$ initial intervals.

E. 5D Klein Bottle Dataset

Our fifth dataset is a point cloud sample of the Klein bottle embedded in $\mathbb{R}^5$ with 15876 points. The dataset is obtained from the *Gudhi* library [14] [2], see Fig. 11A. The filter function is the first coordinate of the point cloud. The hand-tuned mapper graph (Fig. 11B) is expected to capture the loop shown in Fig. 11A. We apply multi-pass AIC/BIC BFS ($\delta = 0\%$, $t = 50$) strategy to an initial mapper graph with $l = 4$ uniform

[2]https://github.com/GUDHI/gudhi-devel

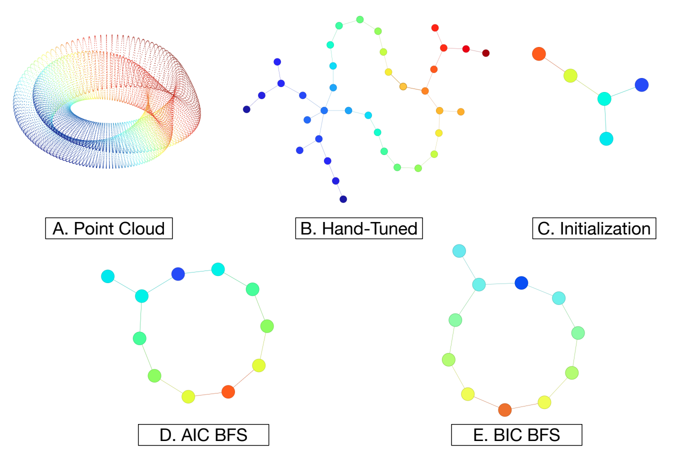

Fig. 11. *Klein bottle* dataset. For mapper graphs, $p = 40\%$. For DBSCAN, $\varepsilon = 0.21$, and $minPts = 5$. (A) Point cloud visualized with a scatter plot using dimension 1, 2, and 3. (B) Hand-tuned mapper graph ($l = 19$). (C) Initial mapper graph ($l = 4$). (D) Multi-pass AIC BFS resulting in $l = 6$. (E) Multi-pass BIC BFS resulting in $l = 6$. Nodes are colored by the (0-1 normalized) average value of the filter function (the first coordinate) with a rainbow colormap.

intervals. As shown in Fig. 11C, using 4 uniform intervals does not capture the loop, whereas the multi-pass AIC BFS and BIC BFS both capture the loop.

We show the distributions of cover elements in Fig. 12 before and after multi-pass AIC BFS (left) and multi-pass BIC BFS (right) strategies. Both of the refined mapper graphs contain (the same) 6 uneven intervals.

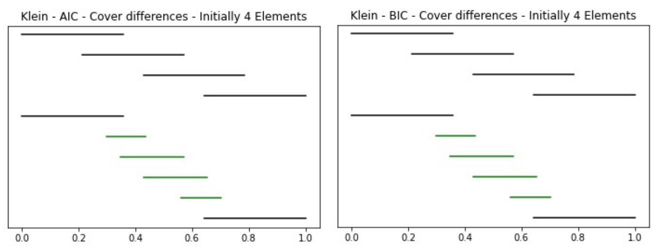

Fig. 12. *Klein bottle* dataset. Distributions of cover elements before and after multi-pass AIC BFS (left) and multi-pass BIC BFS (right), starting with $l = 4$ initial intervals.

F. 7D COVID Dataset

The first real-world dataset [12] contains 1431 daily records of COVID-19 cases in nine states from April 12, 2020 to September 18, 2020[3]. These nine states (AZ, CA, FL, GA, IL, NC, NJ, NY, TX) are chosen to be the ones with the largest number of confirmed cases. For each record, the dataset contains seven statistical measures: number of confirmed cases, death cases, active cases, people tested, testing rate, mortality rate, and incidence rate (i.e., the number of cases per 100K persons). The hand-tuned mapper graph of this dataset has been studied previously [12] (Fig. 13A), which helps to distinguish states with different epidemic trends. Specifically, two main branches stand out: Arizona and Georgia, Florida and Texas, which are two pairs of states sharing similar epidemic trends. Each pair emerges from the main branch and bifurcates, where the branching points (two black arrows in Fig. 13A) indicate when their trends start to diverge.

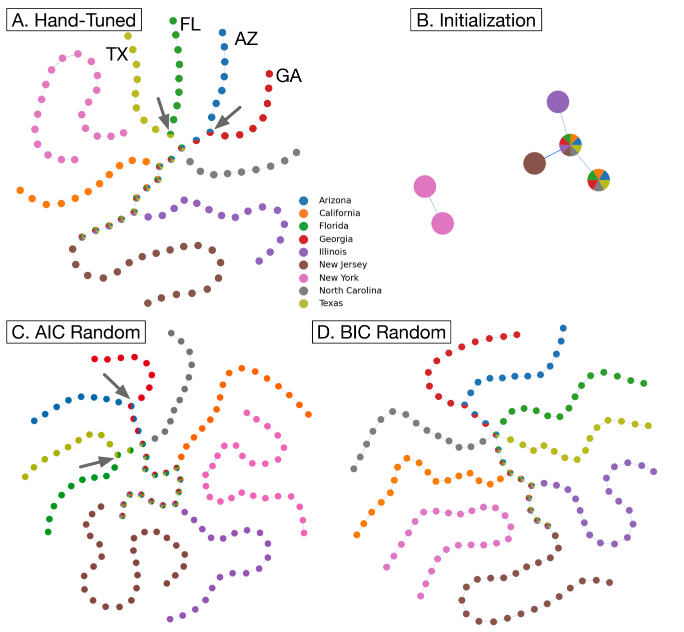

Fig. 13. *COVID* dataset. For mapper graphs, $p = 50\%$, filter function: the number of recorded days. For DBSCAN, $\varepsilon = 0.15$, and $minPts = 5$. (A) Hand-tuned mapper graph ($l = 20$). (B) Initial mapper graph ($l = 2$). (C) Multi-pass AIC Random ($l = 26$). (D) Multi-pass BIC Random ($l = 24$). Nodes are colored by pie chart of states composition.

While the initial mapper graph with $l = 2$ uniform intervals shows some branching structure (Fig. 13B), it is far from the hand-tuned mapper graph as most states remain mixed. Applying multi-pass AIC/BIC ($\delta = 1\%$, $t = 100$) to the initial mapper graph results in adaptive covers whose corresponding refined mapper graphs look very similar to the hand-tuned result (Fig. 13C-D). In particular, the refined mapper graph using the AIC Random strategy recovers the full branching and bifurcation of Arizona and Georgia, as well as Florida and Texas (see black arrows in Fig. 13C). Results from BFS and DFS strategies are fairly similar thus omitted here.

We also show the distributions of cover elements in Fig. 14 before and after multi-pass AIC Random (left) and multi-pass BIC Random (right) strategies. The refined mapper graphs contain 26 (AIC) and 24 (BIC) uneven intervals, respectively.

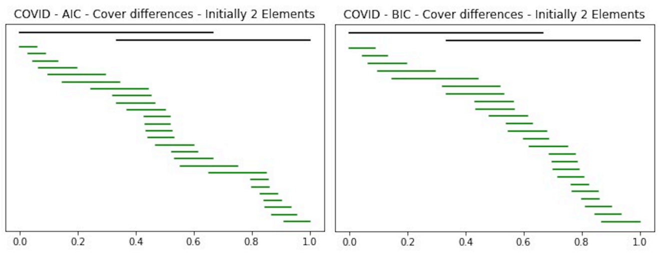

Fig. 14. *COVID* dataset. Distributions of cover elements before and after multi-pass AIC Random (left) and multi-pass BIC Random (right), starting with $l = 2$ initial intervals.

[3]https://github.com/CSSEGISandData/COVID-19/

G. 512D CIFAR-10 Dataset

Our second real-world dataset is created by passing 50K input images from CIFAR-10 [52] to a *ResNet-18* neural network, and collecting spatial activation vectors (*i.e.*, combinations of neuron firings) from the last layer of the network. The images are from 10 image classes (ship, truck, automobile, horse, deer, bird, dog, cat, fog, and airplane), and they give rise to activation vectors in 512 dimensions. This dataset has been studied previously [8], [12] using mapper graphs. The filter function is the L_2 norm of each activation vector. We use this dataset to show that our adaptive cover strategy recovers prior findings; for a dedicated study on applying the mapper construction to ResNet-18, we refer the reader to Rathore *et al.* [8].

As illustrated in Fig. 15A, a hand-tuned mapper graph [12] is shown with $l = 40$ uniform intervals and $p = 20\%$ overlap. Nodes are colored by the class ID, and a pie chart represents a set of activation vectors with mixed classes. This mapper graph was shown to not only cluster images from each class into a separate branch, but also highlight relationships among closely related classes [12]. For instance, the "horse" and "deer" images first jointly emerge from the main branch and then bifurcate into their own branches, indicating similarities between these two classes of images. The "automobile" and "truck" images behave in a similar fashion. The hand-tuned mapper graph of Fig. 15A, regarded as the "ground truth", captures these class bifurcations quite well (see the two black arrows).

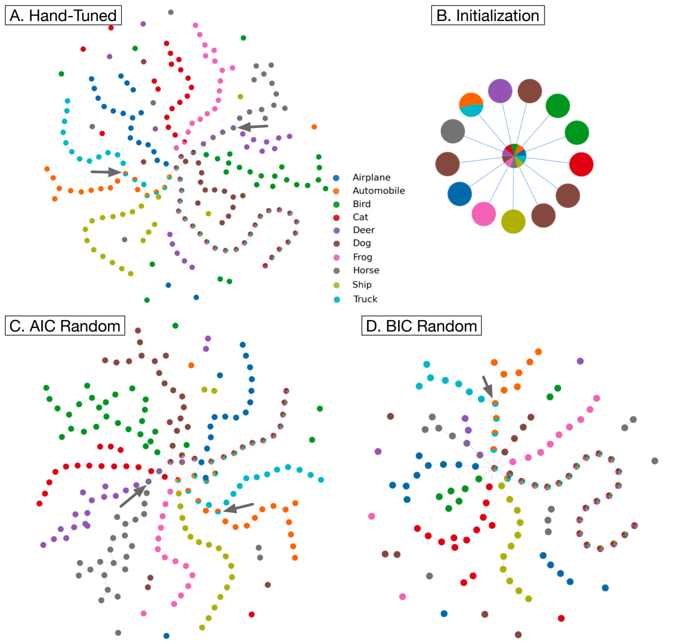

Fig. 15. *CIFAR-10* dataset. For mapper graphs, $p = 20\%$, filter function: L_2-norm. For DBSCAN, $\varepsilon = 8.71$, and $minPts = 5$. (A) Hand-tuned mapper graph ($l = 70$). (B) Initial mapper graph ($l = 2$). (C) AIC Random ($l = 42$). (D) BIC Random ($l = 40$). Nodes are colored by pie chart of categories composition.

As illustrated in Fig. 15B, the initial mapper graph with $l = 2$ uniform intervals is a star-shaped graph, which is too coarse to show any detailed branching structures. Applying multi-pass AIC/BIC to this initial mapper graph ($\delta = 1\%$, $t = 100$) is able to uncover the main branching structures from the hand-tuned mapper graph. In particular, multi-pass AIC Random is able to recover both horse-deer and truck-automobile bifurcations (see black arrows in Fig. 15C).

Fig. 16 shows the cover elements before and after running multi-pass AIC Random and BIC Random respectively, *w.r.t.* the initial mapper graph (at $l = 2$). AIC Random produces $l = 42$ intervals and BIC Random produces $l = 40$ intervals. It is clear that multi-pass AIC/BIC adaptively refines the cover for the initial mapper graph to capture meaningful structure of the data.

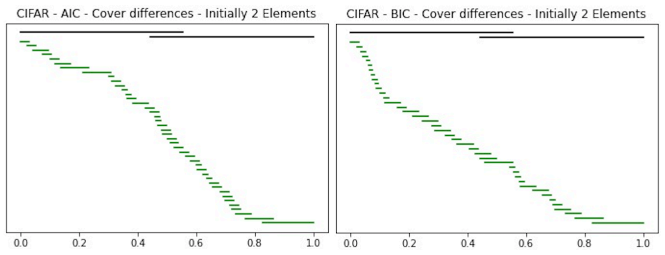

Fig. 16. *CIFAR-10* dataset. Distributions of cover elements before and after multi-pass AIC Random (left) and multi-pass BIC Random (right), starting with $l = 2$ initial intervals.

VI. Comparisons with State-of-the-Art

In this section, we compare our results with those obtained by two state-of-the-art tools that implement mapper graphs with different cover selection strategies.

A. Comparison with Gudhi

First, we compare our results with the *Gudhi* library [14], which includes the automatic parameter tuning approach proposed by Carrière *et al.* [15] (referred to as *statistical cover* here).

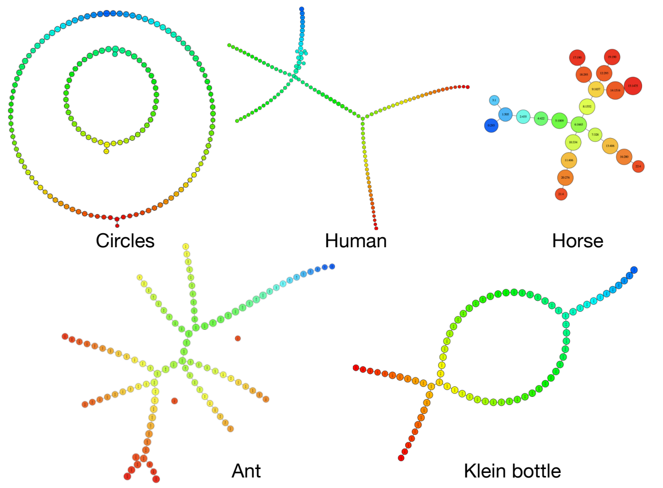

Fig. 17. Applying *statistical cover* strategy to five synthetic datasets: *circles* ($l = 50$), *human* ($l = 58$), *horse* ($l = 10$), *ant* ($l = 40$), and *Klein bottle* ($l = 46$) datasets.

We begin with the synthetic datasets. As shown in Fig. 17, our *adaptive cover* strategy and the *statistical cover* strategy are both able to recover the hand-tuned mapper graphs. The main difference between these two approaches is that our method is able to recover the hand-tuned mapper graph by refining some very coarse initial covers. In addition, the statistical cover strategy typically produces a larger number of (even-length) intervals in comparison to our approach.

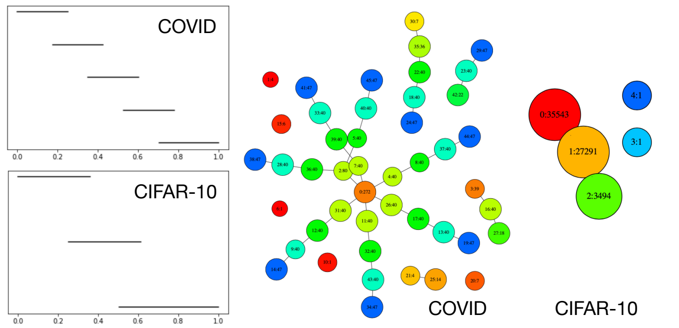

Fig. 18. Applying *statistical cover* strategy to two real-world datasets: *COVID* ($l = 5$) and *CIFAR-10* ($l = 3$) datasets.

We show the results associated with the two real-world datasets in Fig. 18. In both cases, *statistical cover* strategy is not able to recover the hand-tuned mapper graphs in comparison with our strategy, after some extensive testing.

We hypothesize that this is because the independent sampling condition of the input point cloud is not satisfied following the framework in [15]. In the *Covid* dataset, points are not independently distributed. Points sampled from later recording dates tend to have more number of confirmed cases, death cases, people tested, etc. For the *CIFAR-10* dataset, although the points (activations) are supposed to be independently distributed (since each activation is randomly sampled from a single image), they are not likely to come from a uniform probability distribution. Therefore, *statistical cover* strategy performs poorly on these two datasets.

B. Comparison with giotto-tda

Second, we compare our results with the *balanced cover* strategy implemented within the *giotto-tda* library. We run the balanced cover strategy using the final number of intervals after a multi-pass BIC strategy. In particular, we set $l = 6, 11, 8, 12, 6$ for the synthetic datasets, *circles*, *human*, *horse*, *ant*, and *Klein bottle* datasets, respectively, see Fig. 19.

We set l to be 24 and 40 for the real-world *COVID* and *CIFAR-10* datasets, respectively, see Fig. 20. The balanced cover strategy recovers the branching structures in the *COVID* dataset reasonably well; while it fails to recover the branching structures in the *CIFAR-10* dataset.

In summary, the main difference between our *adaptive cover* strategy and the *balanced cover* strategy is that the latter does not perform any automatic parameter tuning. In other words, choosing the appropriate number of intervals relies entirely on the end user.

C. Runtime Comparison

Finally, we report the runtime of our *adaptive cover* strategy in comparison with the *statistical cover* strategy in *Gudhi* and the *balanced cover* strategy of *giotto-tda*.

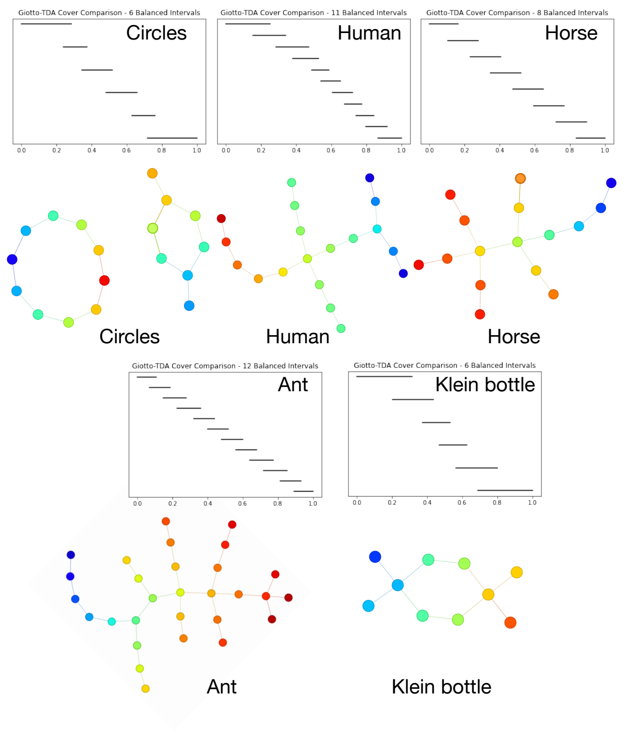

Fig. 19. Applying *balanced cover* strategy to synthetic datasets.

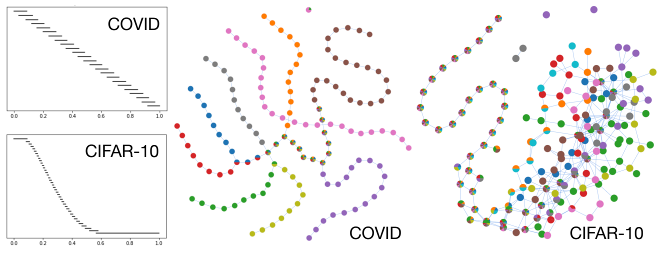

Fig. 20. Applying *balanced cover* strategy to *COVID* and *CIFAR-10* datasets.

For the *CIFAR-10* dataset, all three cover strategies are applied with a $5K$ subsample of the dataset. The rest of the datasets are run as is with the same parameters. We use a 2.3 GHz quad-core i7 laptop with 32 GB of RAM. The reported times are the average over 5 trials each. All times are reported in seconds.

In comparison with the statistical cover strategy, for the

TABLE I
RUNTIME (IN SECONDS) AND SIZE OF EACH DATASET.

data	**size**	**dim**	**adaptive cover**	**statistical cover**	**balanced cover**
circles	2000	2	4.02	2.38	0.05
Human	4706	3	4.99	2.74	0.25
horse	8430	3	3.66	10.35	0.88
ant	6370	3	5.92	5.51	0.50
Klein bottle	15876	5	6.33	25.48	2.75
COVID	1431	7	4.63	0.43	0.17
CIFAR-10	5K sample	512	22.21	46.97	0.53

circles and *human* datasets, our approach is slower. However, for the *Klein bottle* and *CIFAR-10* (5K sample), our approach is significantly faster. This is likely because that we do not have to search a large parameter space in comparison with the statical cover strategy. On the *COVID* dataset, it is interesting to note that the statistical cover finishes significantly faster than our approach but at the cost of poorer result quality, as explained in prior sections. Moreover, it is unsurprising that balanced cover is significantly faster than either adaptive or statistical cover strategies since it does not perform any parameter tuning. Our experiments and prior work by Zhou *et al.* [12] indicate that the most time consuming step in computing the mapper graph is the clustering step with DBSCAN, which is performed many times in our adaptive cover strategy. This to its slower runtime in comparison with the balanced cover strategy.

VII. Conclusion

Inspired by X-means clustering, we generate adaptive covers for mapper graphs using information criteria such as AIC and BIC. To the best of our knowledge, our approach, *for the first time*, investigates practical and automatic parameter selection for mapper graphs using information theory. We demonstrate the utility of our framework with several datasets in obtaining refined mapper graphs that better capture the structures of the underlying data. There are a few points for discussion.

Evaluation. Our current evaluation is based on visual inspection. Comparing mapper graphs via distance measures such as bottleneck and Wasserstein distances between persistence diagrams is an immediate direction to explore. Carrière *et al.* [15] used the bottleneck distance between extended persistence diagram to assess if a mapper graph is a good approximation of the Reeb graph. Additionally, we could also study class consistencies in the case where labels are known for real-world datasets following [24], this is left for future work.

From 1D to 2D covers. Our experiments focus on 1D cover. It would be interesting to explore the extension of our framework for 2D or higher-dimensional covers. We believe this is feasible, albeit certain technical details in converting a mapper graph into hard clustering in higher dimensions.

Computational complexity. There has been a number of recent progress in scalable computation of mapper graphs. Both *giotto-tda* and *Mapper Interactive* are equipped with on-the-fly computation of mapper graphs. In particular, the command line API of *Mapper Interactive* computes mapper graphs for 1 million points of 256 dimensions in 3 minutes, and its GPU implementation provides an additional 2 times acceleration [12]. Our implementation is included as an extension to the *Mapper Interactive*.

X-means algorithm was proposed to address both the scalability and the parameter selection issues of the k-means algorithm for large datasets [53]. As shown in Sect. V-G, our framework is able to refine the mapper graph of a high-dimensional point cloud dataset (50K points in 512D). Further experiments are needed to explore the scalability of our approach for datasets of increased size and dimension. Similar to k-means [54] and X-means clusterings, analyzing the computational complexity and the quality of approximation of our multi-pass AIC/BIC algorithm is challenging and remains an open problem.

Acknowledgement

This work was partially supported by NSF IIS-1910733, DBI-1661375, and DOE DE-SC0021015.

Appendix

We provide detailed derivations of BIC and AIC from the hard clustering of a point cloud; our derivations are based on the original derivation of Pelleg and Moore [24] with minor correction based on multivariate Gaussian distribution [30].

Let $\mathbb{X} = \{x_1, \cdots, x_n\}$ be a point cloud of n points in d dimensions, $|\mathbb{X}| = n$. For a fixed hard clustering model M, suppose $\mathbb{X}$ is partitioned into k clusters assumed to be spherical Gaussians. Let $\mathbb{X}_{(i)}$ denote the cluster whose centroid $\mu_{(i)}$ is the closest to the point $x_i \in \mathbb{X}$. The maximum likelihood estimator for the variance under the identical spherical Gaussian assumption is [31] (a correction from [24]),

$$\hat{\sigma}^2 = \frac{1}{|\mathbb{X}| \cdot d} \sum_{i=1}^{n} \|x_i - \mu_{(i)}\|^2.$$

The point probability of x_i is [30],

$$\hat{P}(x_i) = \frac{|\mathbb{X}_{(i)}|}{|\mathbb{X}|} \cdot \frac{1}{(\sqrt{2\pi}\hat{\sigma})^d} \cdot \exp\left(-\frac{1}{2\hat{\sigma}^2}\|x_i - \mu_{(i)}\|^2\right).$$

The joint probability on the data is modeled as $\prod_{i=1}^{n} \hat{P}(x_i)$, taking its log-likelihood, we obtain

$$\begin{aligned}\hat{l}(M;\mathbb{X}) &= \log\left(\prod_{i=1}^{n} \hat{P}(x_i)\right) = \sum_{i=1}^{n} \log(\hat{P}(x_i)) \\ &= \sum_{i=1}^{n}\left[\log\frac{|\mathbb{X}_{(i)}|}{|\mathbb{X}|} - d\cdot\log(\sqrt{2\pi}\hat{\sigma}) - \frac{1}{2\hat{\sigma}^2}\|x_i - \mu_{(i)}\|^2\right] \\ &= \sum_{i=1}^{n}\left[\log\frac{|\mathbb{X}_{(i)}|}{|\mathbb{X}|} - d\cdot\log(\sqrt{2\pi}\hat{\sigma})\right] - \frac{1}{2\hat{\sigma}^2}\sum_{i=1}^{n}\|x_i - \mu_{(i)}\|^2 \\ &= \sum_{i=1}^{n}\left[\log\frac{|\mathbb{X}_{(i)}|}{|\mathbb{X}|} - d\cdot\log(\sqrt{2\pi}\hat{\sigma})\right] - \frac{|\mathbb{X}|\cdot d}{2}.\end{aligned}$$

Recall that BIC is defined as

$$\mathrm{BIC}(M;\mathbb{X}) = \hat{l}(M;\mathbb{X}) - \frac{\theta}{2}\log|\mathbb{X}|,$$

where θ is the number of free parameters estimated by the underlying statistical model M. It is simply the sum of $k-1$ cluster probabilities, $d\cdot k$ centroid coordinates, and one variance estimation [24]. That is, $\theta = k - 1 + d\cdot k + 1 = (d+1)k$.

References

[1] N. Otter, M. A. Porter, U. Tillmann, P. Grindrod, and H. A. Harrington, "A roadmap for the computation of persistent homology," *EPJ Data Science*, vol. 6, no. 1, 2017.

[2] L. Wasserman, "Topological data analysis," *Annual Review of Statistics and Its Application*, vol. 5, no. 1, pp. 501–532, 2018.

[3] G. Singh, F. Memoli, and G. Carlsson, "Topological methods for the analysis of high dimensional data sets and 3D object recognition," *Eurographics symposium on point-based graphics*, 2007.

[4] M. Hajij, P. Rosen, and B. Wang, "Mapper on graphs for network visualization," arXiv preprint arXiv:1804.11242, 2019.

[5] J. C. Mathews, S. Nadeem, A. J. Levine, M. Pouryahya, J. O. Deasy, and A. Tannenbaum, "Robust and interpretable PAM50 reclassification exhibits survival advantage for myoepithelial and immune phenotypes," *npj Breast Cancer*, vol. 5, no. 1, p. 30, 2019.

[6] M. Nicolau, A. J. Levine, and G. Carlsson, "Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival," *Proceedings of the National Academy of Sciences*, vol. 108, no. 17, pp. 7265–7270, 2011.

[7] C. Geniesse, O. Sporns, G. Petri, and M. Saggar, "Generating dynamical neuroimaging spatiotemporal representations (DyNeuSR) using topological data analysis," *Network neuroscience*, vol. 3, no. 3, pp. 763–778, 2019.

[8] A. Rathore, N. Chalapathi, S. Palande, and B. Wang, "TopoAct: Visually exploring the shape of activations in deep learning," *Computer Graphics Forum*, vol. 40, no. 1, pp. 382–397, 2021.

[9] A. Patania, F. Vaccarino, and G. Petri, "Topological analysis of data," *EPJ Data Science*, vol. 6, no. 1, p. 7, 2017.

[10] H. J. van Veen, N. Saul, D. Eargle, and S. W. Mangham, "Kepler mapper: A flexible python implementation of the mapper algorithm." *Journal of Open Source Software*, vol. 4, no. 42, p. 1315, 2019.

[11] G. Tauzin, U. Lupo, L. Tunstall, J. B. Pérez, M. Caorsi, A. Medina-Mardones, A. Dassatti, and K. Hess, "giotto-tda: A topological data analysis toolkit for machine learning and data exploration," *Journal of Machine Learning Research*, vol. 22, pp. 1–6, 2020.

[12] Y. Zhou, N. Chalapathi, A. Rathore, Y. Zhao, and B. Wang, "Mapper interactive: A scalable, extendable, and interactive toolbox for the visual exploration of high-dimensional data," *IEEE Pacific Visualization Symposium*, 2021.

[13] K. Walsh, M. A. Voineagu, F. Vafaee, and I. Voineagu, "TDAview: an online visualization tool for topological data analysis," *Bioinformatics*, vol. 36, no. 18, pp. 4805–4809, 2020.

[14] C. Maria, J.-D. Boissonnat, M. Glisse, and M. Yvinec, "The GUDHI library: simplicial complexes and persistent homology," in *International congress on mathematical software*. Springer, 2014, pp. 167–174.

[15] M. Carrière, B. Michel, and S. Oudot, "Statistical analysis and parameter selection for mapper," *Journal of Machine Learning Research*, vol. 19, no. 12, pp. 1–39, 2018.

[16] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the second international conference on knowledge discovery and data mining*. AAAI Press, 1996, pp. 226–231.

[17] S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno, "Reeb graphs for shape analysis and applications," *Theoretical Computer Science*, vol. 392, pp. 5–22, 2008.

[18] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[19] D. Arthur and S. Vassilvitskii, "K-Means++: The advantages of careful seeding," *Proceedings of the 18th annual ACM-SIAM symposium on discrete algorithms*, pp. 1027–1035, 2007.

[20] J. M. Phillips, *Mathematical Foundations for Data Analysis*, ser. Springer Series in the Data Sciences. Springer-Verlag, 2021.

[21] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[22] D. T. Pham, S. S. Dimov, and C. D. Nguyen, "Selection of K in K-means clustering," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 219, no. 1, pp. 103–119, 2005.

[23] M. Mahajan, P. Nimbhorkar, and K. Varadarajan, "The planar k-means problem is NP-hard," *Lecture Notes in Computer Science*, vol. 5431, pp. 274–285, 2009.

[24] D. Pelleg and A. Moore, "X-means: Extending k-means with efficient estimation of the number of clusters," *Proceedings of the 17th international conference on machine learning*, pp. 727–734, 2000.

[25] S. Kullback and R. A. Leibler, "On information and sufficiency," *Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[26] R. E. Kass and L. Wasserman, "A reference bayesian test for nested hypotheses and its relationship to the schwarz criterion," *Journal of the American Statistical Association*, vol. 90, no. 431, pp. 928–934, 1995.

[27] G. Schwarz, "Estimating the dimension of a model," *The Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[28] H. Akaike, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 716–723, 1974.

[29] K. P. Burnham and D. R. Anderson, *Model Selection and Multimodel Inference*, 2nd ed. Springer-Verlag, 2002.

[30] S. Prince, *Computer Vision: Models, Learning, and Inference*. Cambridge University Press, 2012.

[31] B. Hancock, D. Frank, A. Foglia, and R. Yozzo, "Notes on Bayesian information criterion calculation for *x*-means clustering," https://github.com/bobhancock/goxmeans/blob/master/doc/BIC_notes.pdf, 2014.

[32] P. Y. Lum, G. Singh, A. Lehman, T. Ishkanov, M. Vejdemo-Johansson, M. Alagappan, J. Carlsson, and G. Carlsson, "Extracting insights from the shape of complex data using topology," *Scientific reports*, vol. 3, no. 1, pp. 1–8, 2013.

[33] A. Robles, M. Hajij, and P. Rosen, "The shape of an image: A study of mapper on images," *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2018.

[34] M. Saggar, O. Sporns, J. Gonzalez-Castillo, P. A. Bandettini, G. Carlsson, G. Glover, and A. L. Reiss, "Towards a new approach to reveal dynamical organization of the brain using topological data analysis," *Nature communications*, vol. 9, no. 1, pp. 1–14, 2018.

[35] M. Kamruzzaman, A. Kalyanaraman, B. Krishnamoorthy, S. Hey, and P. Schnable, "Hyppo-X: A scalable exploratory framework for analyzing complex phenomics data," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2019.

[36] Y. Zhou, M. Kamruzzaman, P. Schnable, B. Krishnamoorthy, A. Kalyanaraman, and B. Wang, "Pheno-Mapper: an interactive toolbox for the visual exploration of phenomics data," *Proceedings of the 12th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*, pp. 1–10, 2021.

[37] T. K. Dey, F. Mémoli, and Y. Wang, "Topological analysis of nerves, Reeb spaces, mappers, and multiscale mappers," *Proceedings of 33rd International Symposium on Computational Geometry*, vol. 77, no. 36, pp. 1–16, 2017.

[38] M. Carriére and S. Oudot, "Structure and stability of the one-dimensional mapper," *Foundations of Computational Mathematics*, vol. 18, no. 6, pp. 1333–1396, 2018.

[39] Y. Zhou, N. Saul, I. Safarli, B. Krishnamoorthy, and B. Wang, "Stitch fix for mapper and topological gains," in *Research in Computational Topology 2*, E. Gasparovic, V. Robins, and K. Turner, Eds. Springer International Publishing, 2021.

[40] A. Babu, "Zigzag coarsenings, mapper stability and gene-network analyses," Ph.D. dissertation, Stanford University, 2013.

[41] E. Munch and B. Wang, "Convergence between categorical representations of Reeb space and mapper," *Proceedings of 32nd International Symposium on Computational Geometry*, vol. 51, pp. 53:1–53:16, 2016.

[42] U. Bauer, X. Ge, and Y. Wang, "Measuring distance between Reeb graphs," *Proceedings of the 30th Annual Symposium on Computational Geometry*, pp. 464–474, 2014.

[43] U. Bauer, E. Munch, and Y. Wang, "Strong equivalence of the interleaving and functional distortion metrics for Reeb graphs," *Proceedings of 31st International Symposium on Computational Geometry*, vol. 34, pp. 461–475, 2015.

[44] U. Bauer, B. D. Fabio, and C. Landi, "An edit distance for Reeb graphs," in *Eurographics Workshop on 3D Object Retrieval*, A. Ferreira, A. Giachetti, and D. Giorgi, Eds. The Eurographics Association, 2016.

[45] U. Bauer, C. Landi, and F. Memoli, "The Reeb graph edit distance is universal," *Foundations of Computational Mathematics*, 2020.

[46] V. De Silva, E. Munch, and A. Patel, "Categorified Reeb graphs," *Discrete & Computational Geometry*, vol. 55, no. 4, pp. 854–906, 2016.

[47] L. Yan, T. B. Masood, R. Sridharamurthy, F. Rasheed, V. Natarajan, I. Hotz, and B. Wang, "Scalar field comparison with topological descriptors: Properties and applications for scientific visualization," *Computer Graphics Forum*, vol. 40, no. 3, pp. 599–633, 2021.

[48] B. Bollen, E. Chambers, J. A. Levine, and E. Munch, "Reeb graph metrics from the ground up," *arXiv preprint arXiv:2110.05631*, 2021.

[49] G. Reeb, "Sur les points singuliers d'une forme de pfaff completement intergrable ou d'une fonction numerique (on the singular points of a complete integral pfaff form or of a numerical function)," *Comptes Rendus Acad. Science Paris*, vol. 222, pp. 847–849, 1946.

[50] B. T. Fasy, F. Lecci, A. Rinaldo, L. Wasserman, S. Balakrishnan, and A. Singh, "Confidence sets for persistence diagrams," *The Annals of Statistics*, vol. 42, no. 6, pp. 2301–2339, 2014.

[51] X. Chen, A. Golovinskiy, and T. Funkhouser, “A benchmark for 3D mesh segmentation,” *ACM Transactions on Graphics*, vol. 28, no. 3, pp. 1–12, 2009.

[52] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” University of Toronto, Tech. Rep. TR-2009, 2009.

[53] M. Ahmed, R. Seraj, and S. M. S. Islam, “The k-means algorithm: A comprehensive survey and performance evaluation,” *Electronics*, vol. 9, no. 8, p. 1295, 2020.

[54] J. Blömer, C. Lammersen, M. Schmidt, and C. Sohler, “Theoretical analysis of the k-means algorithm – a survey,” *Algorithm Engineering. Lecture Notes in Computer Science*, vol. 9220, 2016.