Adaptive Covers of the Mapper Graph using Information Theory

Nithin Chalapathi
University of Utah

UUCS-21-009

School of Computing
University of Utah
Salt Lake City, UT 84112 USA

27 April 2021

Abstract

Topological data analysis (TDA) has recently grown in popularity for analyzing and visualizing high-dimensional data. One of the most popular tools from TDA is the mapper construction, known as the mapper graph in the 1-dimensional setting. An obstacle in the applications of the mapper graph is parameter selection, specically the choice of a cover. In this thesis, we explore strategies for computing an adaptive cover for the mapper graph using information theoretic measures, namly the Akaike information criterion and Bayesian information criterion. We develop a new strategy inspired by X-means, called multi-pass X-means, for adaptively splitting cover elements. We demonstrate that multi-pass X-means produces mapper graphs that approximate the topology of hand-tuned mapper graphs via experimental results on synthetic and real-world datasets. Our preliminary results show that the usage of information theoretic measures is a promising direction for parameter selection. Finally, we study a variant of the mapper graph, called the enhanced mapper graph, introduced by Brown et al. [3]. We provide an open-source library with both our adaptive cover strategies and one of the rst implementations of the enhanced mapper graph.

ADAPTIVE COVERS OF THE MAPPER GRAPH USING INFORMATION THEORY

by

Nithin Chalapathi

A Senior Honors Thesis Submitted to the Faculty of
The University of Utah
In Partial Fulfillment of the Requirements for the

Honors Degree in Bachelor of Science

In

Computer Science

Approved:

Bei Wang Phillips

Bei Wang Phillips, PhD
Thesis Faculty Supervisor

Mary Hall, PhD
Director, School of Computing

Thomas C. Henderson, PhD
Honors Faculty Advisor

Sylvia D. Torti, PhD
Dean, Honors College

April 2021

ABSTRACT

Topological data analysis (TDA) has recently grown in popularity for analyzing and visualizing high-dimensional data. One of the most popular tools from TDA is the mapper construction, known as the mapper graph in the 1-dimensional setting. An obstacle in the applications of the mapper graph is parameter selection, specifically the choice of a cover. In this thesis, we explore strategies for computing an adaptive cover for the mapper graph using information theoretic measures, namly the Akaike information criterion and Bayesian information criterion. We develop a new strategy inspired by X-means, called multi-pass X-means, for adaptively splitting cover elements. We demonstrate that multi-pass X-means produces mapper graphs that approximate the topology of hand-tuned mapper graphs via experimental results on synthetic and real-world datasets. Our preliminary results show that the usage of information theoretic measures is a promising direction for parameter selection. Finally, we study a variant of the mapper graph, called the enhanced mapper graph, introduced by Brown *et al.* [3]. We provide an open-source library with both our adaptive cover strategies and one of the first implementations of the enhanced mapper graph.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Over the last two decades, the amount of unlabeled high-dimensional data has increased at an unprecedented rate, as has the demand for data science tools to explore such data, particularly by domain scientists. Topological data analysis (TDA) offers a rich set of tools for the exploratory data analysis of high-dimensional data. TDA aims to find structure within the data, particularly for datasets that are not easy to visualize and understand. As a field, TDA encompasses a wide variety of approaches, including persistent homology [23] and manifold learning [37]. One successful tool has been the *mapper construction*, a summary of the data in the form of a simplicial complex.

First introduced by Singh *et al.* [33], the mapper construction is a way to visualize high-dimensional point clouds. At its core, the mapper construction is a function-induced soft-clustering method that captures topological information about the data (e.g., branches or loops in high dimensions). The mapper construction has seen success in a variety of fields, including network visualization [12], cancer research [20, 22], neuroscience [10], and more [25]. In recent years, the ecosystem for developers and users alike has rapidly developed in the form of Python libraries (e.g., KeplerMapper [35] and giotto-tda [34]) and robust visualization tools for interactive data analysis (e.g., Mapper Interactive [39] and TDAview [36]).

One major obstacle encountered by applications of the mapper construction is parameter selection, specifically the creation of a *cover*. We focus solely on the 1-dimensional skeleton of the mapper construction, referred to as the *mapper graph*. In this thesis, we explore how information theoretic measures used in hard clustering can inform and generate adaptive covers for mapper graphs. Our main contributions are as follows:

- We explore three strategies to apply the Akaike information criterion (AIC) and Bayesian information criterion (BIC) to the problem of cover selection. We then demonstrate a specific strategy, referred to as the multi-pass X-means, on several synthetic and real-world datasets; see Chapter 3.

- We provide an open-source library with all three adaptive cover strategies.

- We provide, to the best of our knowledge, the first implementation of the enhanced mapper graph [3]. Our library includes GPU computation support, first introduced by Zhou *et al.* [39], and the ability to export graphs to Mapper Interactive, a state-of-the-art visualization tool for mapper graphs; see Chapter 4.

The rest of this thesis has the following structure. Chapter 2 details the necessary technical background and related works. Chapter 3 outlines our three strategies for an adaptive cover and provides results. Chapter 4 articulates the enhanced mapper graph and discusses our implementation. Finally, we conclude in Chapter 5 with some closing remarks.

Manuscript Relevant to the Thesis:

- *Generating Adaptive Covers for Mapper Graphs using Information Theory.* Nithin Chalapathi, Bei Wang. In preparation. 2021.

Other Publications:

- *Mapper interactive: A scalable, extendable, and interactive toolbox for the visual exploration of high-dimensional data.* Youjia Zhou, Nithin Chalapathi, Archit Rathore, Yaodong Zhao, Bei Wang. IEEE Pacific Visualization Symposium, 2021.

- *TopoAct: Visually Exploring the Shape of Activations in Deep Learning.* Archit Rathore, Nithin Chalapathi, Sourabh Palande, Bei Wang. Computer Graphics Forum, 40(1), pages 382-397, 2021.

- *Correctness-Preserving Compression of Datasets and Neural Network Models.* Vinu Joseph, Nithin Chalapathi, Aditya Bhaskara, Ganesh Gopalakrishnan, Pavel Panchekha, Mu Zhang. 4th International Workshop on Software Correctness for HPC Applications Archive Listing, 2021.

- *Interactive Visualization of Interdependent Power and Water Infrastructure Operation.* Han Han, Konstantinos Oikonomou, Nithin Chalapathi, Masood Parvania, Bei Wang. IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT), 2020.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, we describe the technical background of the mapper graph and information theoretic clustering measures. We end this chapter by describing the datasets used in our experiments.

2.1 Mapper Graphs

2.1.1 Construction of a Mapper Graph

We lay out the formal definition of the mapper construction as introduced by [33]. We focus on the 1-dimensional mapper construction. For the rest of this thesis, the mapper construction refers to the process of generating a mapper graph.

Suppose we have some high-dimensional point cloud $\mathbb{X} \subset \mathbb{R}^n$. A cover $\mathcal{V} = \{V_i\}_{i \in I}$ of $\mathbb{X}$ is an indexed set of open sets of $\mathbb{R}^n$ endowed with the standard topology such that $\mathbb{X} \subset \bigcup_{i \in I} V_i$. Given $\mathcal{V}$, we can construct a 1-dimensional nerve $\mathcal{N}$ of $\mathcal{V}$. In general, $\mathcal{N}$ is a simplicial complex. However, $\mathcal{N}_1$, the 1-dimensional skeleton of $\mathcal{N}$, is a graph capturing the intersections between elements of $\mathcal{V}$. There are $|I|$ vertices in $\mathcal{N}_1(\mathcal{V})$, each representing one of the cover elements. $\mathcal{N}_1(\mathcal{V})$ contains an edge between vertices i and j if and only if $V_i \cap V_j$ is nonempty (for $V_i, V_j \in \mathcal{V}$).

Although the construction of $\mathcal{N}_1$ given $\mathcal{V}$ is straightforward, finding $\mathcal{V}$ for $\mathbb{X}$ is nontrivial and that is where the mapper construction comes in. The mapper construction begins with a high-dimensional point cloud $\mathbb{X} \subset \mathbb{R}^n$ as well as a continuous function $f : \mathbb{X} \to \mathbb{R}^m$ where typically $n >> m$ and $m = 1$ or $m = 2$. Suppose we can create a cover $\mathcal{U} = \{U_j\}_{j \in J}$ (J is an index set) of $f(\mathbb{X})$. Then we can create $\mathcal{V}$ by considering the connected components $f^{-1}(\mathcal{U})$. That is, for each $U_j \in \mathcal{U}$, we treat the connected components of $f^{-1}(U_j)$ as cover elements that make up $\mathcal{V}$. The *mapper graph* is then defined as the 1-dimensional nerve of $\mathcal{V}$, $\mathcal{M}(\mathbb{X}, f) := \mathcal{N}_1(\mathcal{V})$.

At first glance, it is not apparent that using a cover of $f(\mathbb{X})$ alleviates the issue of constructing $\mathcal{V}$ since we still need to define $\mathcal{U}$, the cover of $f(\mathbb{X})$. However, in practice, f is a scalar valued function with known meaning, and the cover $\mathcal{U}$ is a set of open intervals in $\mathbb{R}$. The most common choices for f fall into two categories. First, f may be a known data attribute. For example, if $\mathbb{X}$ is a set of high-dimensional points representing basketball players in the National Basketball League, f could be a function mapping players to their average number of points scored per game. Second, f may be derived from the data, for example, the L_2-norm and eccentricity.

Assuming that f is a scalar function, the de facto method of constructing $\mathcal{U}$ 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 to overlap in order to capture relations among the cover elements. This method of creating $\mathcal{U}$ is called the *uniform cover*. A second, but less widely used, method is the *balanced cover*. Again, the range of $f(\mathbb{X})$ is split into l intervals, but each interval contains an equal number of points, which ensures that denser regions of $f(\mathbb{X})$ are covered by more intervals and are a finer representation. Once again, each adjacent interval has p percent overlap.

To illustrate the process of computing a mapper graph, we provide an example in Fig. 2.1. In (a), we have the original point cloud $\mathbb{X}$ with the points in black. In (b) we have an example

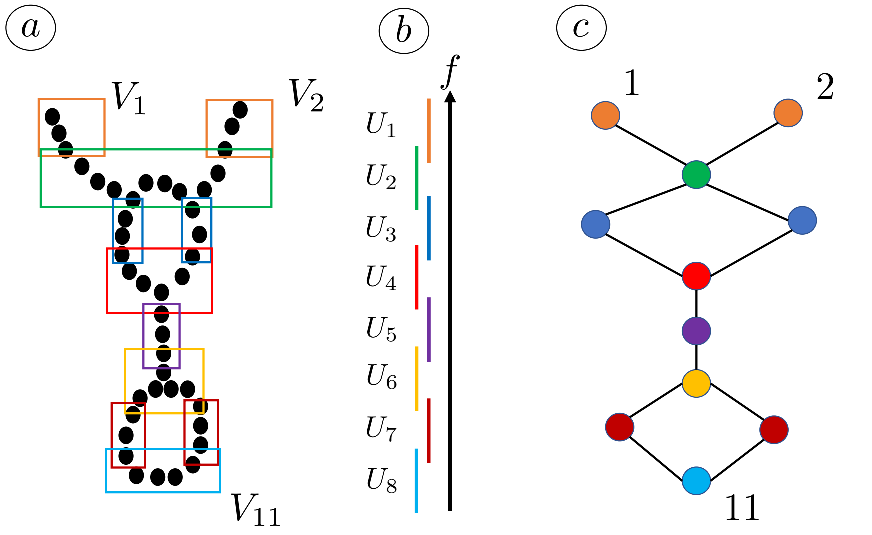


Figure 2.1. Example mapper graph computation. (a) The point cloud $\mathbb{X}$. (b) The filter function along with a cover. (c) The resulting mappar graph.

filter function. For each point in $\mathbb{X}$, f provides a corresponding value. Fig. 2.1(b) has a uniform cover $\mathcal{U}$ with eight elements. In (a), we overlay the connected components of each $f^{-1}(U_i)$ in (b). V_1, V_2, and V_{11} are the first, second, and eleventh elements of $\mathcal{V}$, an open cover on $\mathbb{X}$. Each V_i becomes a vertex in the mapper graph shown in (c). Edges in (c) are added when there are overlapping elements between two V_i.

2.1.2 Mapper Construction Parameters

In practice, the above construction has a number of parameters a practitioner must specify. First, a filter function f, sometimes called a "lens"; l, the number of intervals to cover $f(\mathbb{X})$; and p, the percent overlap. Finally, a clustering algorithm must be used to find the connected components of $f^{-1}(U_i)$. Although any clustering algorithm works, most common applications use an algorithm without a fixed number of clusters. DBSCAN (Density-based spatial clustering of applications with noise [9]) is a common choice. However, DBSCAN introduces two new parameters: ϵ and $minPts$. DBSCAN clusters points by density; clusters are formed if there are $minPts$ within ϵ distance of each other. During the clustering step, $\mathbb{X}$ must also be endowed with a metric, such as the Euclidean distance.

2.1.3 Related Works

Parameter selection has long been a challenge faced by users of the mapper framework. Carrière *et al.* [5] proved that under certain assumptions about the input point cloud, the mapper graph is an optimal estimator for the Reeb graph. They also introduced a procedure for selecting the most stable parameters. Stability in this setting means that under parameter perturbations, the resulting mapper graph does not change. Their procedure involves repeatedly computing the mapper graph over a sample of the data while comparing the extended persistence [24] between the mapper graph and a ground truth.

Brown *et al.* [3] studied the convergence between the Reeb graph and the mapper graph, extending the work of Munch and Wang [21]. In [3], the authors introduced a theoretical object called the *enhanced mapper graph*. This variant captures more geometric information about the point cloud than the classic mapper graph. The construction process of the enhanced mapper graph can be found in Chapter 4.

2.2 Information Theoretic Clustering Measures

Within the field of clustering, there are many measures for assessing the quality of a clustering. Broadly speaking, the question of clustering quality is a question of model selection. Given some set of models $M_1, \ldots, M_n$, how does one select the best model? In the supervised setting where the data have labels and our goal is to group points with similar labels together, we can use external validation (e.g., cross validation). However, the mapper graph is primarily used as a data exploration tool and well-defined labels are rare. In situations like this, we use internal evaluation, the process of deriving some score based on the internal properties of the clustering. Information theoretic measures are one class of internal evaluation methods.

AIC and BIC. At the core of information theoretic measures is the idea of minimizing the Kullback-Leibler (KL) divergence, also known as relative entropy [6]. The KL divergence is a way of informally describing the distance between two probability distributions. Note that the KL divergence is not a metric since it is not symmetric nor does it satisfy the triangle inequality. Nonetheless, it still provides a way to compare two probability distributions. If P and Q are discrete probability functions defined on the same probability space $\mathcal{X}$, then the KL divergence is:

$$D_{KL}(P||Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{Q(x)} \right) \tag{2.1}$$

Suppose we have a dataset $\mathbb{X}$. We assume that there is an underlying generative process from which $\mathbb{X}$ is sampled. Let the associated probability function of the generative process be P. $M_1, \ldots M_n$ is a set of parameterized probability models with parameters $\theta_1 \ldots \theta_n$, and denote the probability functions as $P(x; M_i)$ for model i. The best model will be the one that minimizes the KL divergence.

One measure derived from this idea is the Akaike information criterion (AIC) [1, 4]. For a given model M_i, the AIC is:

$$\text{AIC}(M_i) = 2 \cdot \hat{l}(M_i; \mathbb{X}) - 2 \cdot p_i \tag{2.2}$$

where $\hat{l}$ is the maximum log likelihood estimation of M_i given dataset $\mathbb{X}$ and p_i is the number of parameters estimated in M_i (i.e., $|\theta_i|$).

Another commonly used information theoretic measure is the Bayesian information criterion (BIC) [15, 26, 32]. If we assume that $P(x; M_i)$ has a prior distribution, the KL divergences gives rise to the following formulation of the BIC:

$$\text{BIC}(M_i) = \hat{l}(M_i; \mathbb{X}) - \frac{p_i}{2} \cdot \log |\mathbb{X}| \tag{2.3}$$

The model with the highest AIC or BIC score is the selected model. In some of the literature, there is a factor of -1. In this case, the process for the computation is the same but the model with the smallest score is instead selected. In this thesis, we use the maximization variant. Neither formulation directly gives a quality assessment for a particular model. Their main usage is in comparing a set of models and selecting the best one. In other words, the magnitude of a single AIC or BIC score is not relevant. Similarly, all the models must operate on the same probability space.

Both methods balance the trade-off between fit and complexity. The log likelihood term measures how likely a model is given the observed data whereas the p_i term penalizes models with more parameters, and hence discourages overfitting.

2.2.1 X-Means

One of the most popular clustering algorithms is k-means [2]. In k-means, the selection of the parameter k is a widely discussed topic with multiple established approaches such as cross validation [29], the silhouette score [31], and various indices [28]. One clustering algorithm that heavily uses the BIC is X-means [26], a variant of k-means. X-means attempts to solve three large issues that plague k-means: poor computational scaling, parameter selection, and getting stuck in local minima. For our purposes, parameter selection is most relevant.

At its core, X-means consists of two alternating steps. First, given a k, the method finds the best k-means clustering. In most implementations, this is done using Lloyd's algorithm [14, 18] or its variants. Given the best k clustering, X-means then runs an Improve Structure step. Improve Structure determines if and where new clusters should be added by splitting a particular cluster. For the i-th cluster, there is a subset of the data D_i assigned to cluster i. Treating D_i as the whole dataset, X-means computes its BIC score. It then runs a 2-means clustering and again computes the BIC score. If the 2-means BIC score is larger than the 1-means BIC score, the original centroid is replaced with the two new centroids computed. The 2-means clustering is initialized by splitting the i-th centroid in two. The two new centroids are moved along a random direction with the distance proportional to the size of the region D_i. Note that when running the 2-means, only points in D_i contribute to the location of the new centroids.

2.3 Datasets

Here we outline the three datasets used in Chapter 3, one synthetic dataset and two real-world datasets: Circles, COVID-19, and CIFAR-10. For each, we also define the filter function. All three datasets have either a known mapper graph or have been studied by prior work. Thus, they provide a collection of known datasets to compare our new cover selection schemes.

Circles The circles dataset is comprised of two concentric 2D circles with a total of 2000 points. The filter function is the second dimension of each point (i.e., the y-axis). See Fig. 2.2 for an image of the two circles.

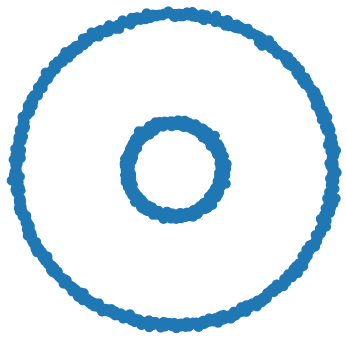


Figure 2.2. Visualization of the Circles dataset.

COVID-19 Within the United States, various states are affected by the COVID-19 pandemic differently. The second dataset consists of statistics for different states from April 12, 2020 to September 18, 2020[1]. We specifically look at Arizona, California, Florida, Georgia,

[1]COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University: https://github.com/CSSEGISandData/COVID-19

Illinois, New Jersey, New York, North Carolina, and Texas. There are 1431 data points, each corresponding to a daily snapshot of a state with respect to seven measures: confirmed cases, deaths, active cases, people tested, testing rate, mortality rate, and incidence rate. All rate values are per 100k individuals. The filter function we use is the number of days since April 12, 2020. Zhou *et al.* [39] applied the Mapper framework and found that particular states form branching structures, suggesting that they have different epidemic trends.

CIFAR-10. The final dataset comes from deep learning, following the procedure outlined by Rathore *et al.* [30]. One of the main challenges in deep learning is interpretability. That is, given a neural network, how do we explain its behavior? In our case, we use ResNet-18 [13], a popular image classification neural network with 18 layers and 8 residual blocks, trained on CIFAR-10 [16]. CIFAR-10 is an image classification dataset with 60,000 28x28 images spread across 10 classes (e.g., dog, cat, airplane). To collect the activations, we run a forward pass of a trained neural network using all images and extract the intermediate representations between layers. We use a sample of the activation vectors from the third residual block. Each activation vector has 256 dimensions, and we sample 50,000 of the possible 1 million activations. We ensure each class has 5,000 activation vectors so that each class is represented. Similar to [30], we use the L_2 norm as the filter function. In [30], the authors note that the classes start to separate in the form of bifurcations within the mapper graphs.

CHAPTER 3

COVER SELECTION

3.1 Methodology

In order to explain the three strategies for generating an adaptive cover, we first articulate how to compute the AIC and BIC for mapper graphs.

3.1.1 AIC and BIC on Mapper Graphs

Let $\mathbb{X}$ be a point cloud with a lens function $f : \mathbb{X} \to \mathbb{R}$ and x_i denote the i-th point of $\mathbb{X}$. We assume that $\mathbb{X}$ is equipped with the standard Euclidean metric. Given a mapper graph $\mathcal{M}$, we can create a hard clustering. Let V be the vertices of $\mathcal{M}$ and $k = |V|$. We can generate k centroids, each representing a vertex of the mapper graph by taking the mean of the corresponding points. If $\mathbb{X}_j$ represents the points belonging to vertex v_j, then the location of the j-th centroid is:

$$\mu_{v_j} = \frac{1}{|\mathbb{X}_j|} \sum_{x_i \in \mathbb{X}_j} x_i \tag{3.1}$$

For points that belong to only one vertex, we assign their membership to the corresponding cluster. However, for each point that exists in two or more vertices, we assign it to the nearest center. Suppose point x_i belongs to a set of vertices $V_{(i)}$ in $\mathcal{M}$. Then the assignment of x_i is $\text{argmin}_{v_j \in V_{(i)}} ||x_i - \mu_{v_j}||^2$. Points that do not belong to any vertex are assigned to the vertex with the nearest centroid. In Fig. 3.1, we use the point cloud and mapper graph from Fig. 2.1 to compute a hard clustering. The colors of the mapper graph vertices correspond to the clusters, and the center of each cluster is a red dot.

We now have k centroids and a unique assignment for each point x_i. We treat the set of centroids $\mu_{v_1} \dots \mu_{v_k}$ as the means for k identically spherical Gaussian distributions. Under this assumption, the set of centroids acts as the centroids for k-means clustering. To derive the likelihood function, we first start with the probability of point x_i:

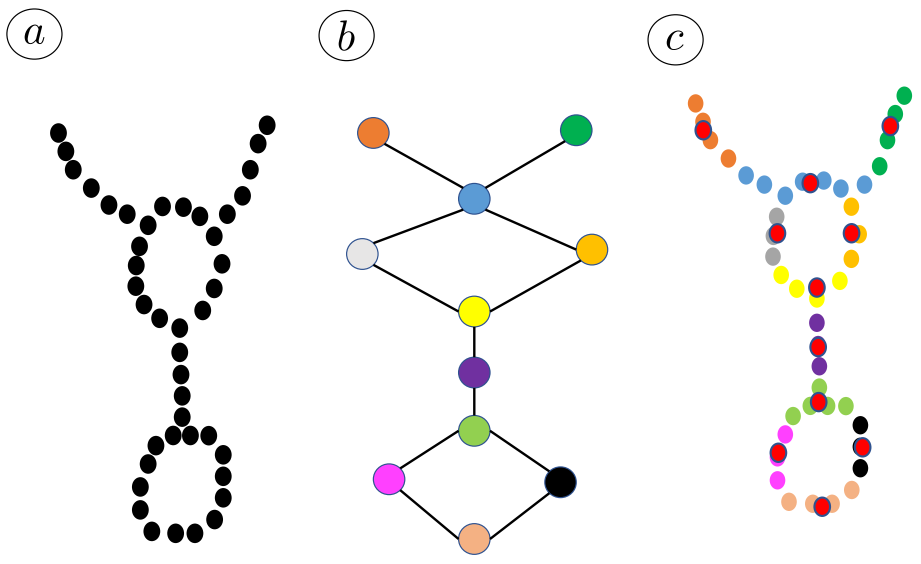


Figure 3.1. Hard clustering of the point cloud from Fig. 2.1. (a) The original point cloud. (b) The mapper graph. (c) The resulting hard clustering with cluster centroids marked by red dots.

$$P(x_i) = \sum_{j=1}^{k} P(x_i \in \mathbb{X}_j) \cdot P(x_i | x_i \in \mathbb{X}_j) \tag{3.2}$$

In the maximum likelihood case, the first term $P(x_i \in \mathbb{X}_j)$ becomes $\frac{|\mathbb{X}_j|}{|\mathbb{X}|}$ [15, 26]. The second term is the standard definition of a Gaussian distribution:

$$P(x_i | x_i \in \mathbb{X}_j) = \frac{1}{(2 \cdot \pi \cdot \sigma^2)^{d/2}} \cdot \exp \frac{-||x_i - \mu_{v_j}||^2}{2\sigma^2} \tag{3.3}$$

where d is the dimension of the points. We can then take the log likelihood of equation (3.2).

$$\begin{aligned} l(\mathcal{M}; \mathbb{X}) &= \log(\prod_i P(x_i)) \\ &= \sum_{j=1}^{k} \left(|\mathbb{X}_j| \cdot \left(\log\left(\frac{|\mathbb{X}_j|}{|\mathbb{X}|}\right) - \frac{d}{2} \cdot \log(2\pi \cdot \sigma^2) \right) - \frac{1}{2 \cdot \sigma^2} \sum_{x_i \in \mathbb{X}_j} ||x_i - \mu_{v_j}||^2 \right) \end{aligned} \tag{3.4}$$

We have yet to compute the variance. Since cluster j is a Gaussian distribution at μ_{v_j}, the unbiased estimator of the variance is:

$$\hat{\sigma}_j^2 = \frac{1}{d \cdot (|\mathbb{X}_j| - 1)} \sum_{x_i \in \mathbb{X}_j} ||x_i - \mu_{v_j}||^2 \tag{3.5}$$

The identically spherical Gaussian distributions assumption presumes that each cluster has the same variance. Thus, for any cluster j,

$$\hat{\sigma}^2 = \frac{1}{d \cdot (|\mathbb{X}_j| - 1)} \sum_{x_i \in \mathbb{X}_j} ||x_i - \mu_{v_j}||^2 \tag{3.6}$$

Now consider the sum of all distances:

$$\begin{aligned}\sum_{x_i \in \mathbb{X}} ||x_i - \mu_{(i)}||^2 &= \sum_{j=0}^{k} \sum_{x_i \in \mathbb{X}_j} ||x_i - \mu_{v_j}||^2 \\ &= d \cdot (|\mathbb{X}| - k) \cdot \hat{\sigma}^2 \\ \implies \hat{\sigma}^2 &= \frac{1}{d \cdot (|\mathbb{X}| - k)} \sum_{x_i \in \mathbb{X}} ||x_i - \mu_{(i)}||^2 \end{aligned} \tag{3.7}$$

where $\mu_{(i)}$ is the centroid x_i is assigned to. Putting everything together, we now have the following likelihood estimate:

$$\hat{l}(\mathcal{M}; D) = \sum_{j=1}^{k} (|\mathbb{X}_j| \log |\mathbb{X}_j|) - |\mathbb{X}| \log |\mathbb{X}| - \frac{|\mathbb{X}| \cdot d}{2} \log(2 \cdot \pi \cdot \hat{\sigma}^2) - \frac{d \cdot (|\mathbb{X}| - k)}{2} \tag{3.8}$$

Recall that the AIC and BIC are defined as:

$$\text{AIC}(\mathcal{M}) = 2 \cdot \hat{l}(\mathcal{M}; \mathbb{X}) - 2 \cdot p_{\mathcal{M}} \tag{3.9}$$

$$\text{BIC}(\mathcal{M}) = \hat{l}(\mathcal{M}; \mathbb{X}) - \frac{p_{\mathcal{M}}}{2} \cdot \log |\mathbb{X}| \tag{3.10}$$

We have shown how to find $\hat{l}(\mathcal{M}; \mathbb{X})$, but we have yet to compute $p_{\mathcal{M}}$, the number of free parameters in the probability distribution generated by k Guassian centroids. The hard clustering of $\mathcal{M}$ contains k centroids, each of dimension d. Hence, there are $k \cdot d$ parameters representing the centroids. Since all clusters have the same variance, one variance estimate, $\hat{\sigma}^2$, needs to be computed. Finally, during the derivation of $P(x_i)$, $P(x_i \in \mathbb{X}_j)$ is an empirical estimate of the number of points in cluster j. Note that if we are given $P(x_i \in \mathbb{X}_j)$ for all j

except $j = k$, we automatically know $P(x_i \in \mathbb{X}_k)$ since $P(x_i \in \mathbb{X}_k) = 1 - \sum_{j=1}^{k-1} P(x_i \in \mathbb{X}_j)$. Computing each $P(x_i \in \mathbb{X}_j)$ for k clusters yields an additional $k-1$ parameters. Therefore, $p_{\mathcal{M}} = (k \cdot d) + 1 + (k-1) = k \cdot (d+1)$. Putting this all together, we have the following equations for the AIC and BIC:

$$\begin{aligned}
\mathrm{AIC}(\mathcal{M}) = &\sum_{j=1}^{k} (2 \cdot |\mathbb{X}_j| \log |\mathbb{X}_j|) - 2 \cdot |\mathbb{X}| \log |\mathbb{X}| - |\mathbb{X}| \cdot d \log(2 \cdot \pi \cdot \hat{\sigma}^2) - d \cdot (|\mathbb{X}| - k) \\
&- 2 \cdot (k \cdot (d+1)) \qquad (3.11) \\
\mathrm{BIC}(\mathcal{M}) = &\sum_{j=1}^{k} (|\mathbb{X}_j| \log |\mathbb{X}_j|) - |\mathbb{X}| \log |\mathbb{X}| - \frac{|\mathbb{X}| \cdot d}{2} \log(2 \cdot \pi \cdot \hat{\sigma}^2) - \frac{d \cdot (|\mathbb{X}| - k)}{2} \\
&- \frac{k \cdot (d+1)}{2} \cdot \log |\mathbb{X}| \qquad (3.12)
\end{aligned}$$

3.1.2 Strategy 1: Grid Search

In this strategy, we aim to maximize the overall value of the information criterion by selecting the mapper graph whose AIC or BIC value is maximal. Suppose we have a fixed percent overlap and a set of the number of intervals to select from $\{l_1 \dots l_n\}$. For each l_i, we construct a uniform cover with p overlap and l_i intervals. The mapper graph $\mathcal{M}_{l_i}$ is computed along with its AIC and BIC values using the formulas from section 3.1.1. We then select the cover with the largest AIC or BIC score.

3.1.3 Strategy 2: Single-Pass X-Means

Our second strategy is to treat the mapper graph as an initialization for X-means [26]. After running X-means until completion, we convert the resulting clustering to a mapper graph by constructing a new cover based on the centroids returned by X-means.

First we create a hard clustering from a given mapper graph $\mathcal{M}$ using the process outlined in section 3.1.1. After the hard clustering step, we have k centroids and each point has a unique assignment. We then run X-means till convergence or reaching a maximum number of clusters, and we receive k' centroids where $k' \geq k$.

To reconstruct a mapper graph, we first create a suitable cover of the image of the lens function f. We start with a naive cover:

$$\mathcal{U} = \{(\min f(\mathbb{X}_i), \max f(\mathbb{X}_i)) | i = 1 \ldots k'\} \tag{3.13}$$

We note two problems with the naive cover of $f(\mathbb{X})$, $\mathcal{U}$, based on equation (3.13). First, two cover elements $U_i, U_j \in \mathcal{U}$ may exist such that $U_i \subseteq U_j$. Second, an overlap between cover elements is not guaranteed. These two issues lead us to algorithm 1 for computing a new cover $\mathcal{U}'$ based on $\mathcal{U}$. Given the new cover $\mathcal{U}'$ of f, we can generate a new mapper graph $\mathcal{M}'$, the final result. We evaluate the final graph using the AIC and BIC formulas from section 3.1.1

Algorithm 1 Algorithm used in strategy 2 to generate a new cover $\mathcal{U}'$

```
1: procedure GENERATE COVER($\mathcal{U}$, min_overlap)        ▷ Start with $\mathcal{U}$ and returns $\mathcal{U}'$.
2:     $\mathcal{U}' \leftarrow$ sort($\mathcal{U}$)        ▷ Sort intervals based on the start of each interval
3:                                                    Break ties by prioritizing longer intervals
4:     for $i = 0 \ldots |\mathcal{U}'| - 1$ do
5:         for $j = i + 1 \ldots |\mathcal{U}'| - 1$ do
6:             if $U_j \subseteq U_i$ then
7:                 Remove $U_j$ from $\mathcal{U}'$.
8:     for $i = 0 \ldots |\mathcal{U}'| - 2$ do        ▷ Ensure a minimum overlap
9:         if $U_i \cap U_{i+1} <$ min_overlap $\cdot \min(|U_i|, |U_{i+1}|)$ then
10:            Extend $U_i$ and $U_j$ equally till minimum overlap is achieved.
11:    Return $\mathcal{U}'$
```

3.1.4 Strategy 3: Multi-Pass X-Means

Strategy 3 is based on X-means's Improve Structure step. In X-means's Improve Structure phase, each cluster is treated independently, and the associated points are treated as the dataset. The BIC score is then computed for $k = 1$ and $k = 2$ representing the option to keep the current clustering or splitting it into two clusters. In our strategy, we compute an AIC or BIC value for each interval. We then split the interval and compare the AIC or BIC values, opting to keep a split if it increases the information criterion.

First, we define what it means to "split an interval" with overlap p. Let (a, b) be an open interval. If we want two equal length intervals $(a, a+\delta)$ and $(b-\delta, b)$ as a result of the splitting operation, we need to compute δ that respects the overlap. Since $b - a = 2 \cdot \delta - \delta \cdot p$, $\delta = (b-a)/(2-p)$, we now have two new intervals $(a, a+\delta)$ and $(b-\delta, b)$. Analogously, merging two overlapping intervals (a, b) and (c, d) is done by replacing both with (a, d). We illustrate this in Fig. 3.2

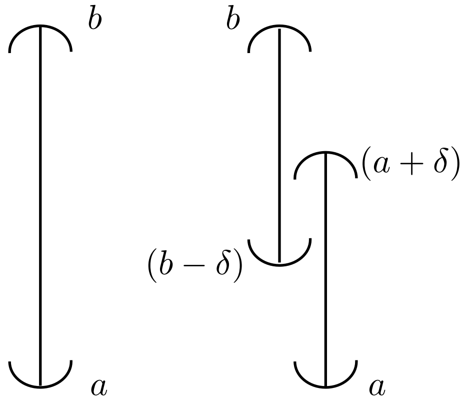


Figure 3.2. An illustration of splitting an interval that belongs to the cover of $f(\mathbb{X})$.

Now we can describe the entire process. Consider a mapper graph $\mathcal{M}$ on point cloud $\mathbb{X}$, with filter function f, m cover elements, and p percent overlap. For cover element U_j, we consider the set of vertices V_j of $\mathcal{M}$ that correspond to connected components of $f^{-1}(U_j)$. We compute the AIC or BIC score as outlined in section 3.1.1 on the subgraph induced by connected components V_j of $f^{-1}(U_i)$. We then split U_j into two overlapping intervals: U'_j, U'_{j+1}. We compute a mapper graph $\mathcal{M}'$ using the new cover that contains $m+1$ elements. The subgraph from U'_j and U'_{j+1} is used to compute another AIC or BIC score. If the new score is larger than before splitting the cover element, we retain the split. We provide pseudocode for one iteration of the procedure in algorithm 2.

We detail the steps of algorithm 2. First, we iterate through each interval of the cover of $f(\mathbb{X})$ and compute the AIC or BIC of the induced subgraph. Then, we split the interval and compute a new mapper graph $\mathcal{M}'$. If the AIC or BIC value of the induced subgraph of $\mathcal{M}'$ from the two new intervals is larger than the subgraph prior to splitting, we mark the interval as a candidate for splitting. After iterating through all of the cover elements, we split the cover element with the largest increase in the AIC or BIC value. This concludes the one iteration detailed in algorithm 2. Algorithm 2 shows one iteration when using the BIC. In the AIC version, the process is the same except that on lines 6 and 13, we compute the AIC value of the subgraphs instead of the BIC.

Algorithm 2 One iteration of multi-pass X-means (BIC Version)

```
1: procedure MULTI-PASS $X$-MEANS($\mathbb{X}$, $f$, $\mathcal{U}$)                ▷ $\mathcal{U}$ is the initial cover
2:     mark_for_splitting ← {}
3:     $\mathcal{M}$ ← GenerateMapper($\mathbb{X}$, $f$, $\mathcal{U}$)
4:     for $i = 0 \ldots m-1$ do
5:         subgraph ← $\mathcal{M}$.subgraph($U_i$)
6:         original-BIC-value ← BIC(subgraph)
7:         $\mathcal{U}' \leftarrow \mathcal{U}$
8:         $\mathcal{U}'$.split_element($U'_i$)                ▷ New cover by splitting element $i$
9:         $\mathcal{M}'$ ← GenerateMapper($\mathbb{X}$, $f$, $\mathcal{U}'$)
10:
11:        // Since $U_i$ was split, the adjacent cover element $U_{i+1}$ is the new cover element
12:        subgraph← $\mathcal{M}'$.subgraph($U_i$, $U_{i+1}$)
13:        new-BIC-value ← BIC(subgraph)
14:        if original-BIC-value < new-BIC-value then
15:            mark_for_splitting.add($U_i$)
16:    // Construct a new cover with the best split
17:    Return $\mathcal{U}$.split_element(mark_for_splitting)
```

Since algorithm 2 is only one iteration, we can repeat the process until convergence. One consistent obstacle with this approach is selecting the initial cover. Because intervals are only split in two, the initialization plays a significant role. For our results, we use parameters that are near the hand-tuned parameters for each dataset.

3.2 Results

In this section, we include the relevant results of our strategies and focus on the results of multi-pass X-means. We also provide classic mapper graphs generated from the hand-tuned parameters of prior works [30, 39]. The number of intervals passed to the classic mapper graph and to initialize multi-pass X-means is denoted by l. The nodes in the mapper graphs for the Circles dataset are colored by function value. For COVID-19 and CIFAR-10, the nodes are represented by pie charts showing the composition of states and classes, respectively.

Circles. In the circles dataset, we use a DBSCAN ϵ value of 0.1, *minPts* of 5, and 20% overlap. In this case, the hand-tuned number of intervals for the classic mapper graph is 7. By visual inspection, we know the classic mapper graph at $l = 7$ captures the ground-truth, visualized in Fig. 3.3 with two concentric circles. At $l = 6$, the classic mapper graph in Fig. 3.4(a) disconnects the inner circle. Multi-pass X-means using the AIC, Fig. 3.4(b), is able to recover the inner circle and converges to the ground truth.

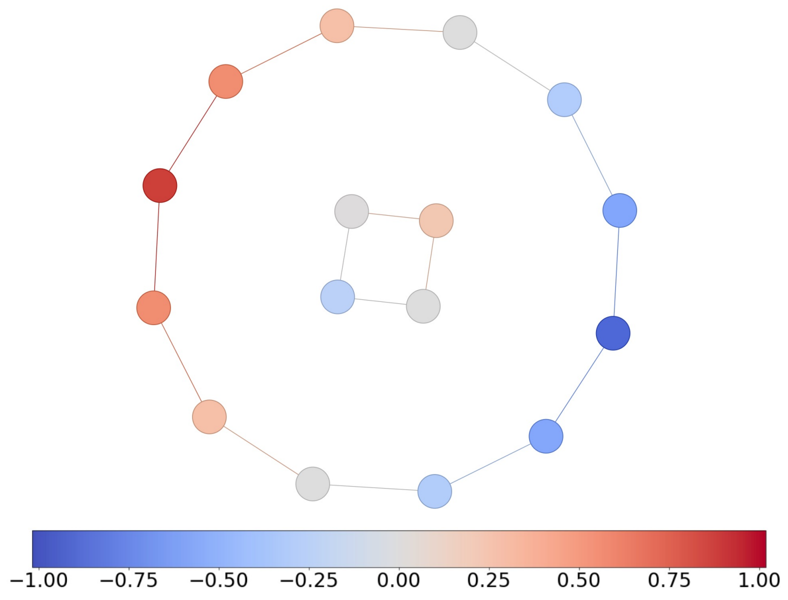


Figure 3.3. Circles: Ground truth mapper graph ($l = 7$)

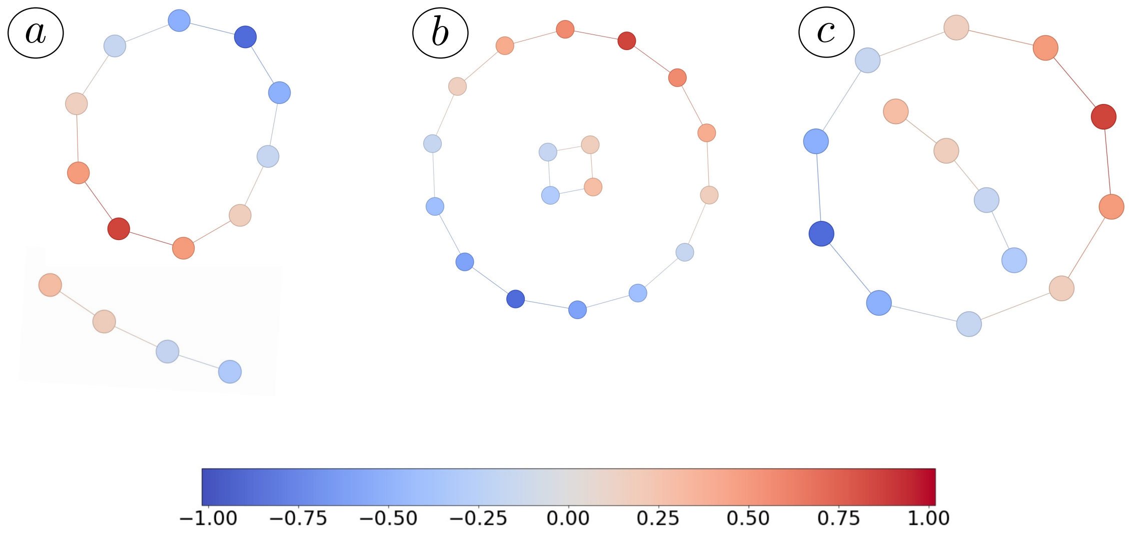


Figure 3.4. Circles: Generated mapper graphs when $l = 6$. (a) Classic mapper graph (b) Multi-pass AIC (c) Multi-pass BIC.

In Fig. 3.5, we use a parameter setting of $l = 2$. The classic mapper graph in (a) shows neither circle. However, both multi-pass X-means using the AIC (b) and the BIC (c) recover one of the two circles. Neither converges directly to the ground-truth but are significantly closer than the classic mapper graph in (a). In addition to demonstrating that multi-pass X-means progresses towards the hand-tuned topological configuration, it suggests that multi-pass X-means is able to discover some loop structures.

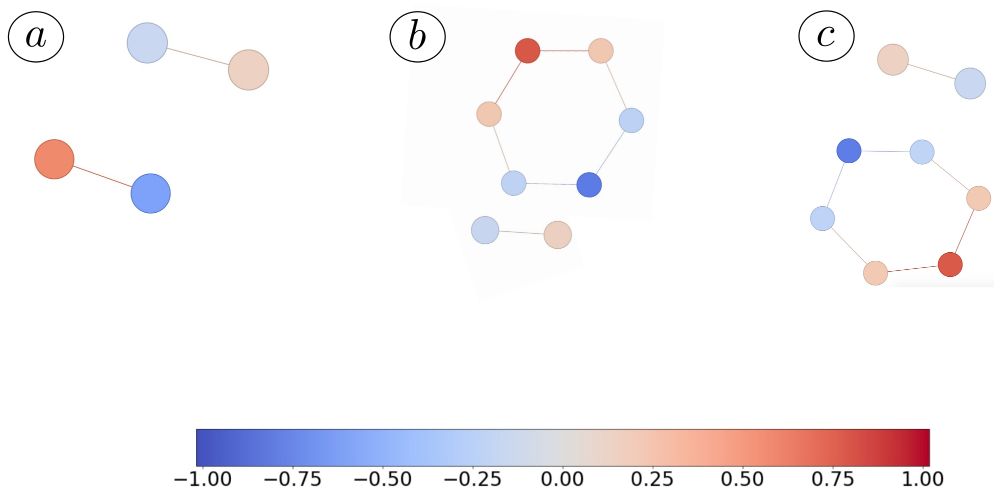


Figure 3.5. Circles: Generated mapper graphs when $l = 2$. (a) Classic mapper graph (b) Multi-pass AIC (c) Multi-pass BIC.

Finally, in Fig. 3.6, we include a plot of the AIC (a) and BIC (b) before and after running multi-pass X-means. For most choices of l, the local maxima found by multi-pass X-means is the same as the global AIC or BIC value of the classic mapper graph. However, the resulting mapper graphs are significantly improved.

COVID-19. The next dataset is the COVID-19 dataset. We use a DBSCAN ϵ of 0.15, $minPts$ of 5, and an overlap of 50%. In Fig. 3.7, we show the hand-tuned configuration of $l = 20$ discovered by Zhou *et al.* [39]. In the hand-tuned classic mapper graph, we can see clear branching structures for each state. In particular, Texas (light green) vs Florida (dark green) as well as Georgia (red) vs Arizona (blue) form two interesting branches due

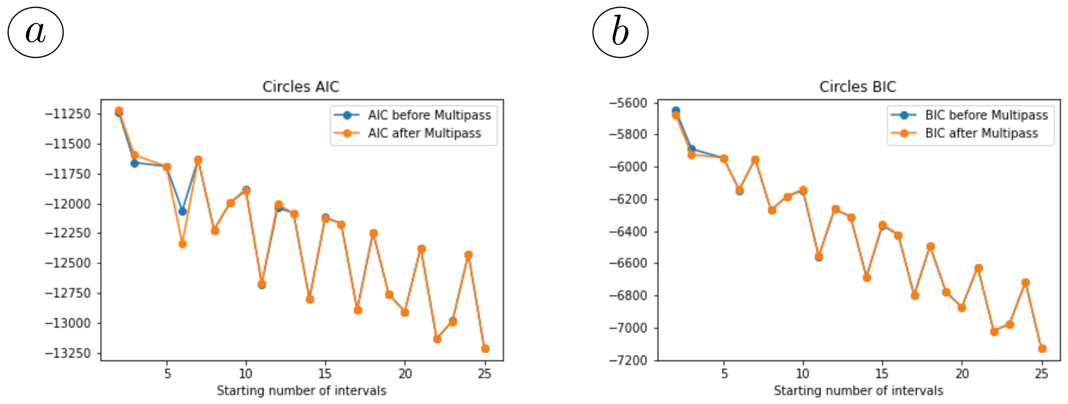


Figure 3.6. Circles: (a) AIC before and after multi-pass. (b) BIC before and after multi-pass.

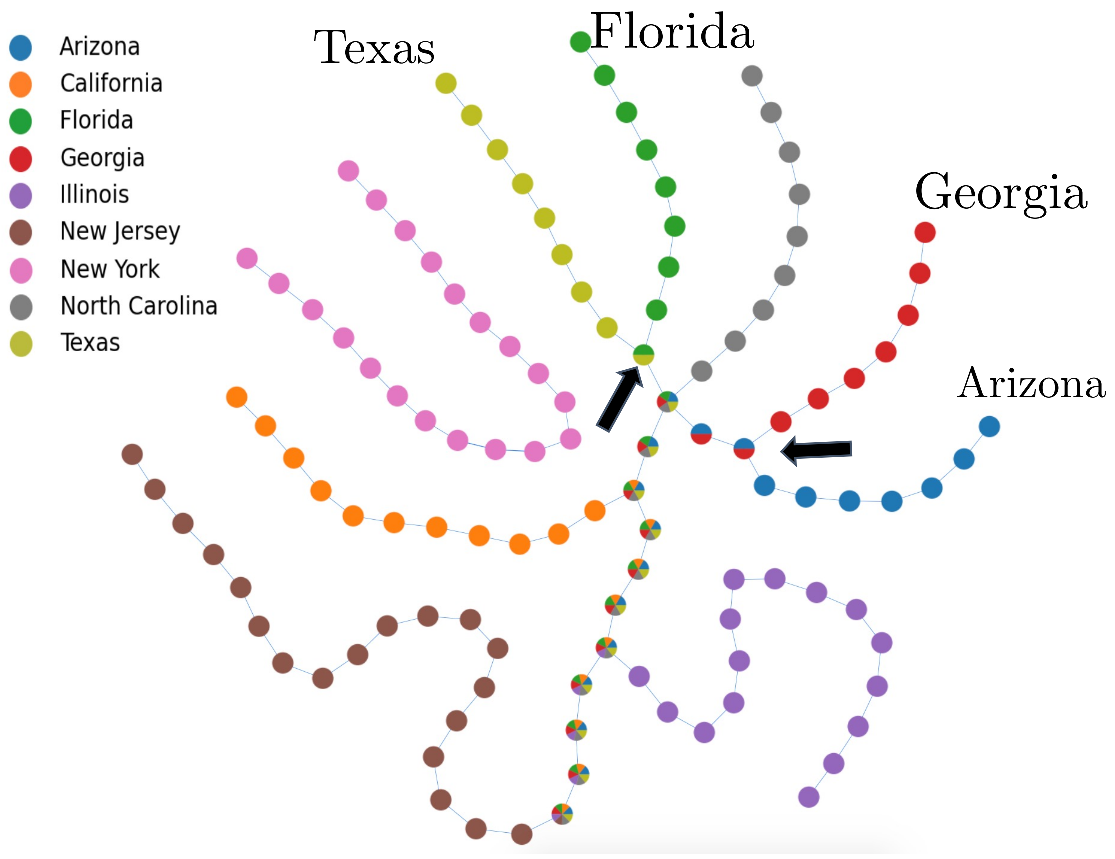


Figure 3.7. COVID: Hand-tuned mapper graph ($l = 20$)

to their epidemic trend; the branching points for both are pointed out by arrows.

For $l = 10$ and above, the classic mapper graph already shows clear branching structures. In order to properly test multi-pass X-means, we first start with $l = 2$ in Fig. 3.8. In the classic mapper graph (a), the connected component with 4 vertices is entirely mixed; there is no clear separation between the states. Both multi-pass X-means with the AIC (b) and BIC (c) generate mapper graphs that are significantly closer to the hand-tuned graph in Fig. 3.7. (b) and (c) show the Arizona vs Georgia bifurcation starting from a node that contains only Arizona and Georgia data points. The results of multi-pass X-means recover the Florida vs Texas branch but fail to include a root node containing only Florida and Texas points. The branching points are indicated by black arrows. Examining the distribution of intervals shown next to the graphs of (b) and (c), we can see that multi-pass X-means focuses on subdividing intervals with higher filter function values (i.e., after COVID-19 starts to spread in a state). This supports multi-pass X-means converging towards the hand-tuned mapper graph (treated as the "ground truth") since the branching of states occurs during higher filter function values. The results of multi-pass X-means reflects the fact that most states start with the same initial epidemic trends, but quickly branch based on the state's response and unique attributes (e.g., population density).

If we start with a more generous initialization of $l = 5$, shown in Fig. 3.9, the classic mapper graph (a) shows some branching structure. However, most states are still mixed. While better than $l = 2$, the classic mapper graph (a) is still far from the hand-tuned mapper graph in Fig. 3.7. Multi-pass X-means using the AIC (b) and the BIC (c) converge to mapper graphs that look similar to the hand-tuned graph. In particular, multi-pass X-means is able to recover the bifurcation of Arizona vs Georgia as well as Florida vs Texas (see Fig. 3.9 black arrows). Both structures branch from the main connected component and bifurcate from a node that contains only Arizona and Georgia or Florida and Texas data points. As with $l = 2$, multi-pass X-means focuses on subdividing intervals with higher filter function.

Finally, Fig. 3.10 shows the AIC (a) and BIC (b) values after performing multi-pass X-means. Quantitatively, both methods decrease the AIC and BIC values for choices of l less than 40. However, as discussed above, qualitatively, the local maxima that multi-pass X-means finds are significantly closer to the hand-tuned graph.

CIFAR-10. Our last set of results comes from neural network activations on CIFAR-10. The DBSCAN ϵ parameter is set to 8.71, $minPts$ as 5, and an overlap of 20%. Fig. 3.11 contains the hand-tuned parameter of $l = 40$ from Zhou *et al.* and Rathore *et al.* [30, 39].

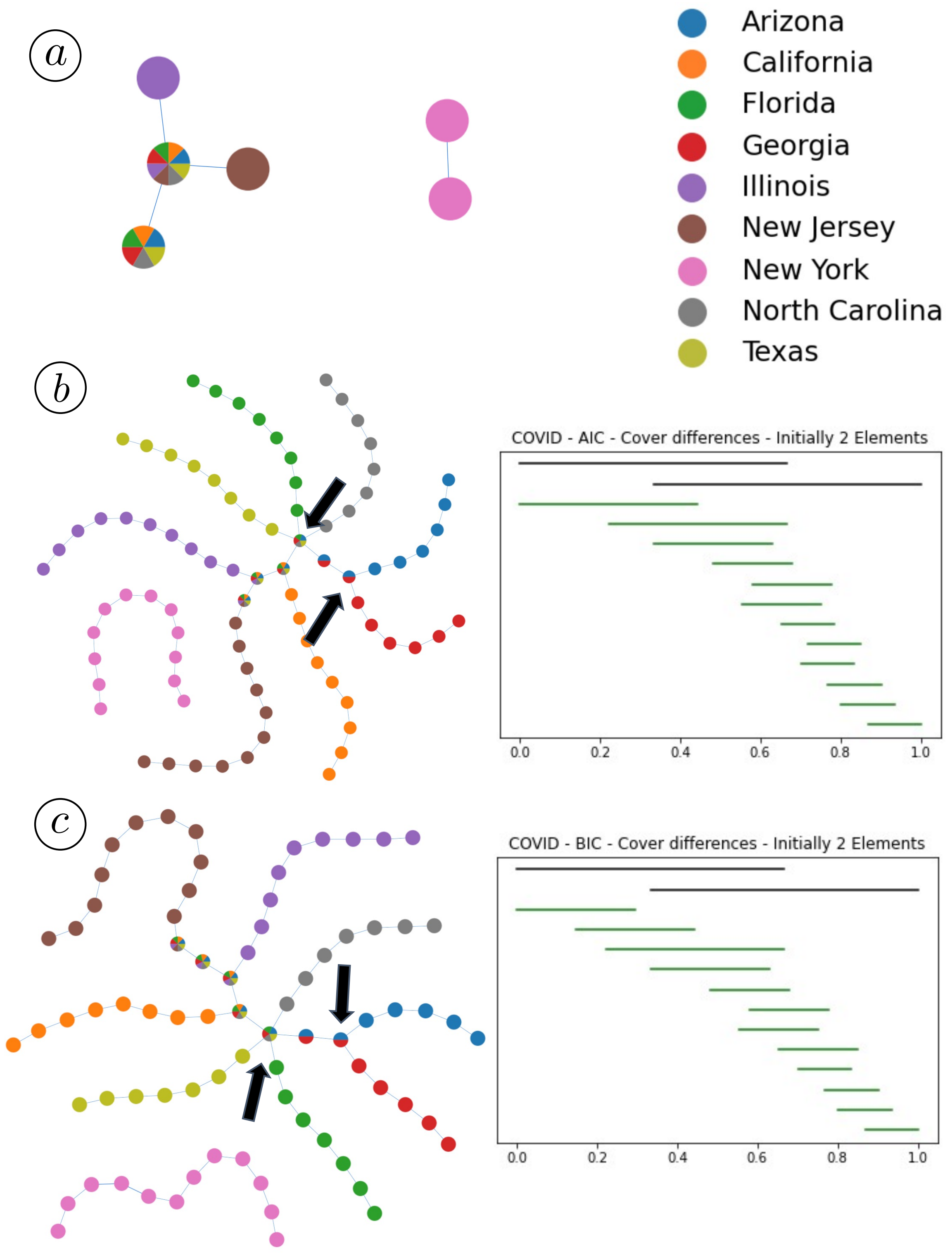


Figure 3.8. COVID: Generated mapper graphs when $l = 2$. (a) Classic mapper graph (b) Multi-pass AIC (c) Multi-pass BIC. The multi-pass figures also include the distribution of cover elements.

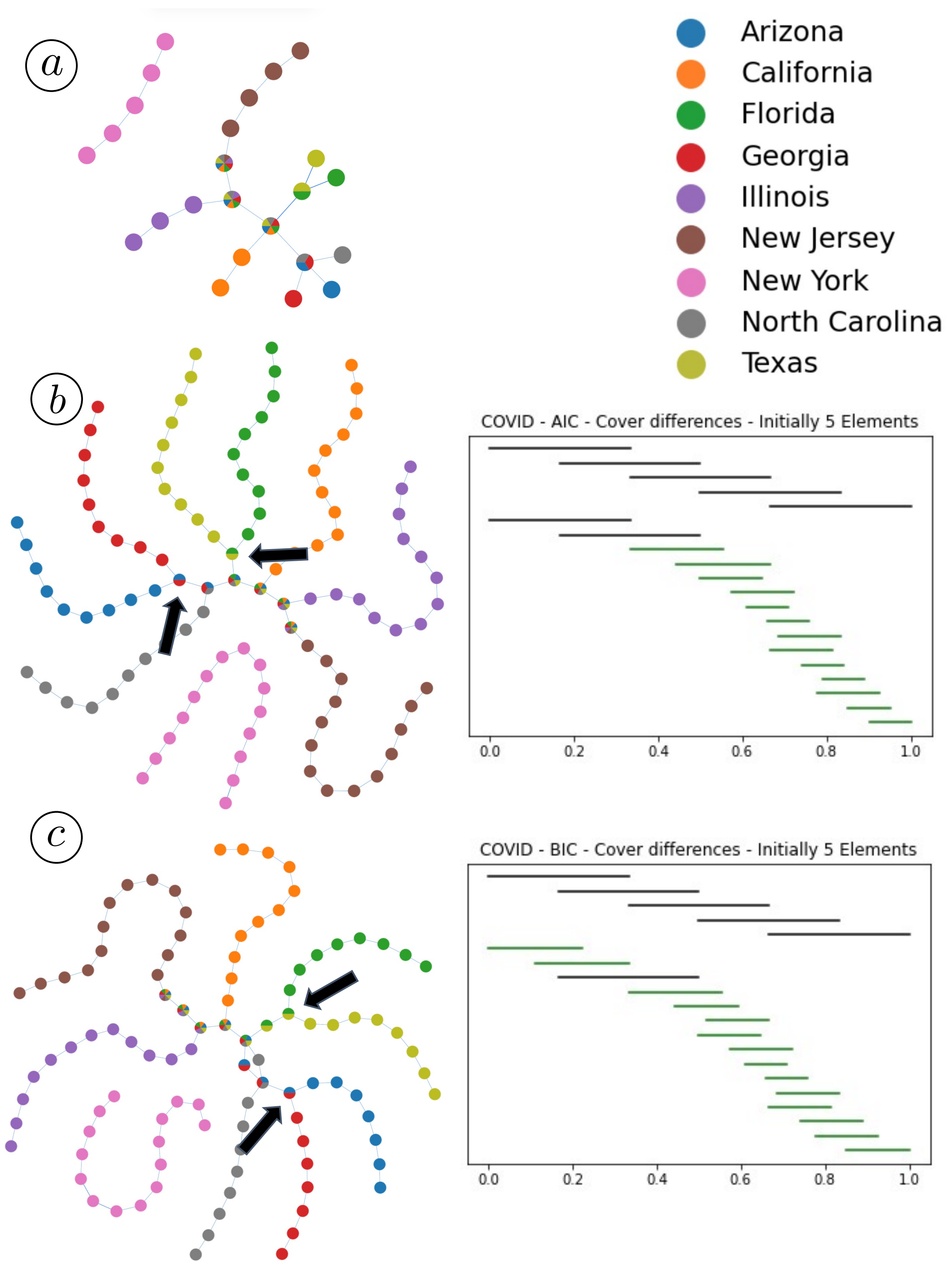
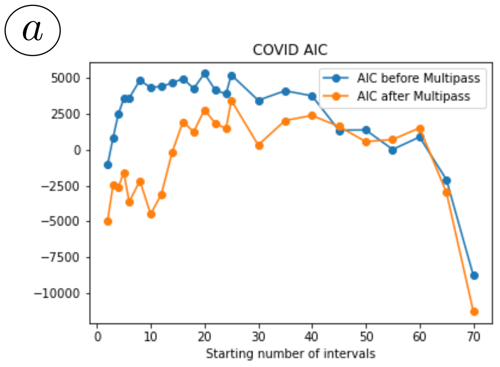


Figure 3.9. COVID: Generated mapper graphs when $l = 5$. (a) Classic mapper graph (b) Multi-pass AIC with the distribution of cover elements (c) Multi-pass BIC with the distribution of cover elements. For the distribution of cover elements, black represents unmodified cover elements and green represents new cover elements after splitting.



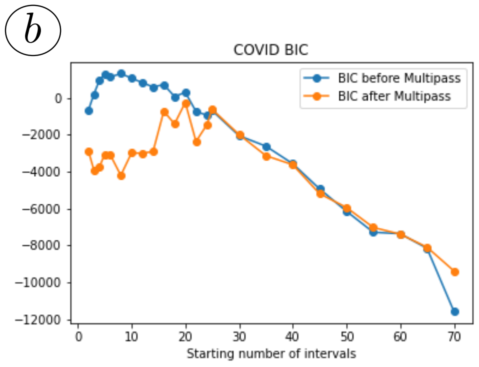


Figure 3.10. COVID: (a) AIC before and after multi-pass. (b) BIC before and after multi-pass.

The key branches we expect to see with multi-pass X-means are the Truck (light blue) vs Automobile (orange) branch and the Deer (purple) vs Horse (gray) branch. The branches split from the main connected component into a node that contains only the two classes. They then further bifurcate into branches that only contain one class. In Fig. 3.11, the branching points are indicated with black arrows.

We show the results of $l = 2$ in Fig. 3.12. The classic mapper graph (a) shows no clear branching structure. Multi-pass X-means using the AIC (b) and BIC (c) are able to uncover the main branching structure from the ground truth. Compared to (a), both (b) and (c) are closer to the hand-tuned mapper graph. Multi-pass X-means is also able to recover the bifurcation between Horse and Deer starting at a Horse and Deer mixed node, similar to the hand-tuned mapper graph. However, neither method is able to recreate the branch of Automobile and Truck, grouping both into one branch. It may be the case that $l = 2$ is too far from the hand-tuned parameter for multi-pass X-means to completely recover the hand-tuned graph. Examining the covers before and after multi-pass X-means, it is clear that both methods focus on activations with low to mid-range L_2-norms. This is indicative of multi-pass X-means focusing on the correct cover elements; activations with high L_2-norm are naturally separated in the high-dimensional space.

Using $l = 5$, Fig. 3.13, the classic mapper graph (a) begins to show small branching structures; Airplane (dark blue) and Bird (dark green) are both in their own branches. Similar to the $l = 2$ case, multi-pass X-means (b,c) separates most classes and forms a Horse vs Deer branch that is similar to the hand-tuned mapper graph. Both multi-pass

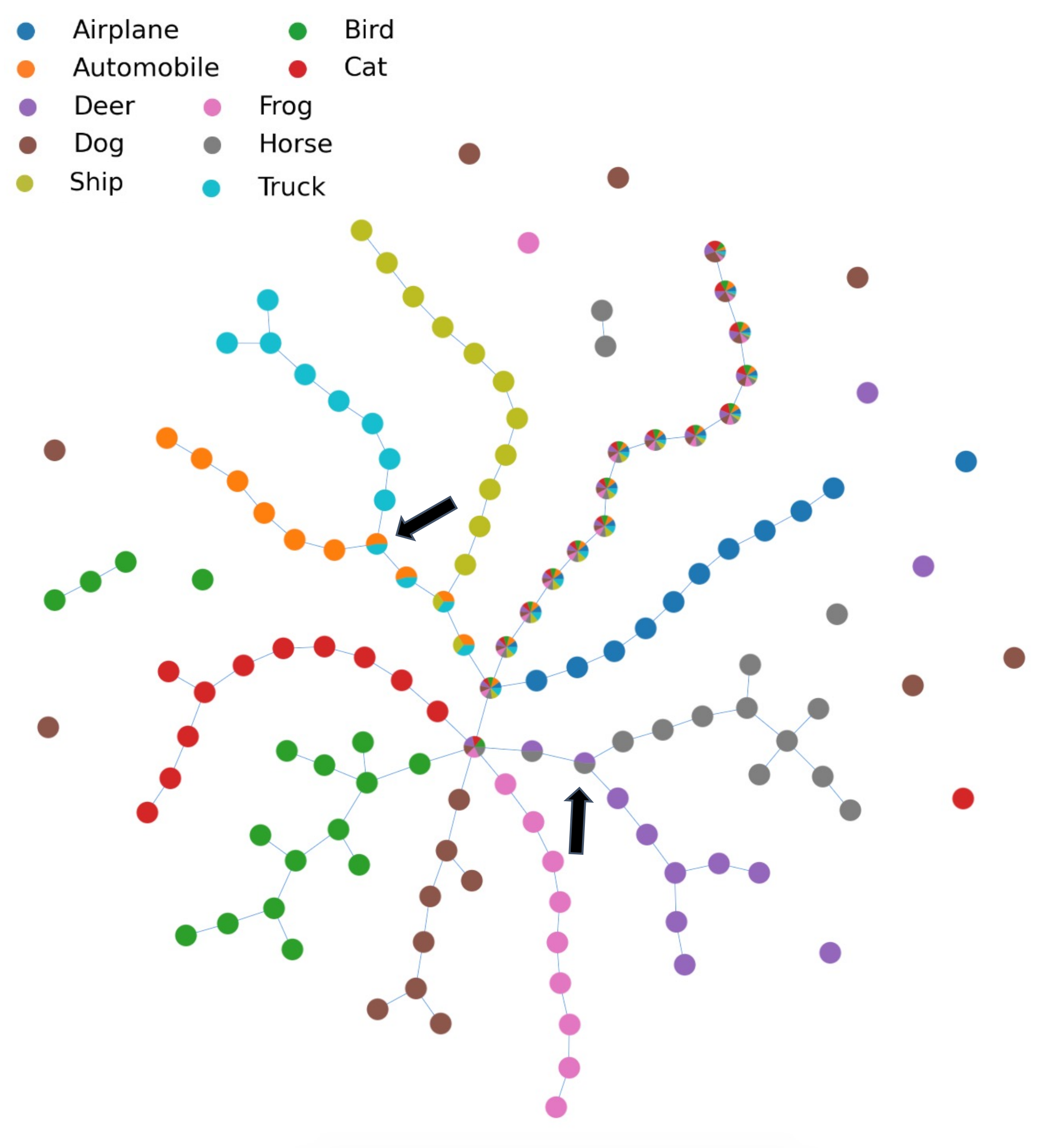


Figure 3.11. CIFAR: Hand-tuned mapper graph (l = 40)

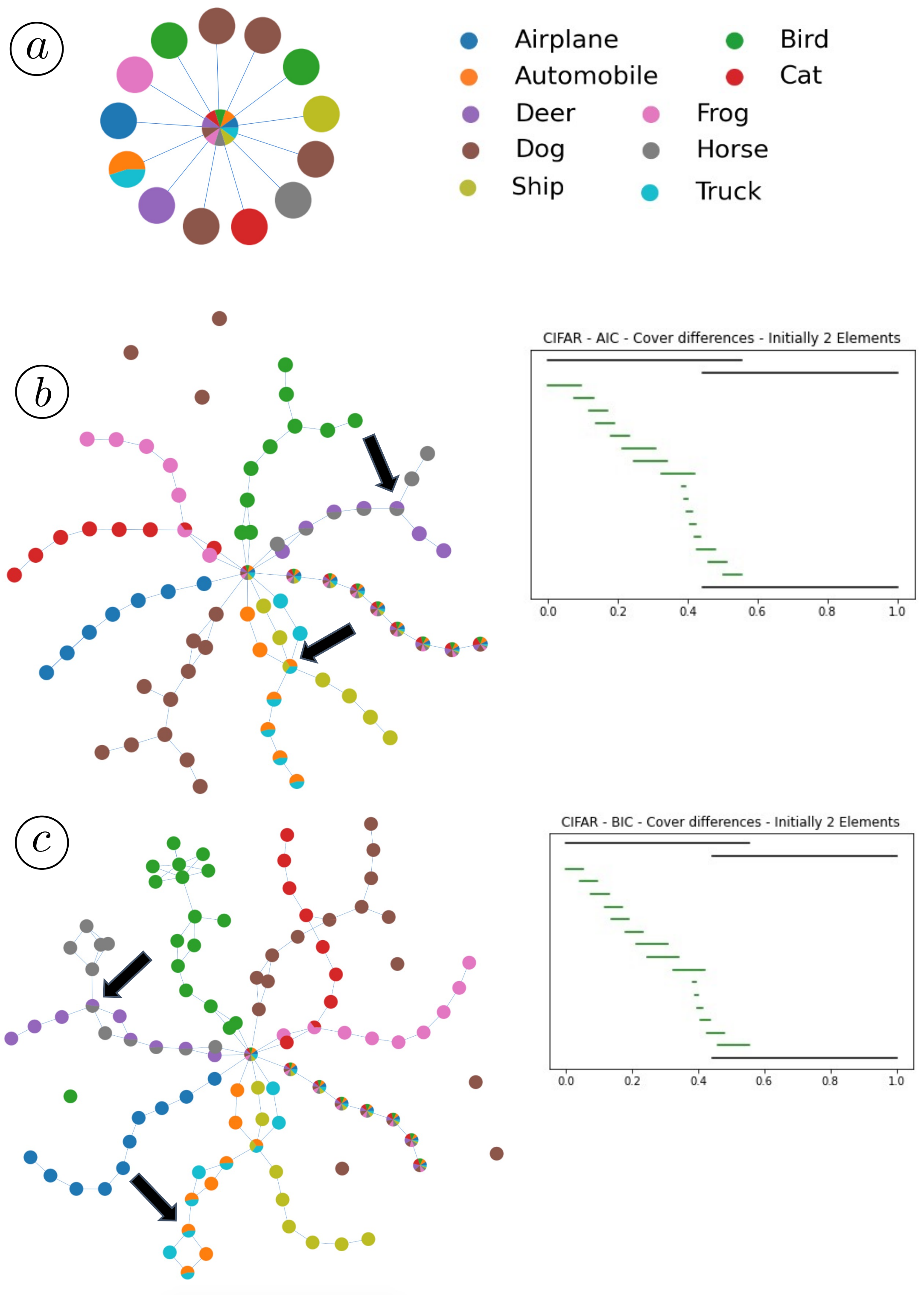


Figure 3.12. CIFAR: Generated mapper graphs when $l = 2$. (a) Classic mapper graph (b) Multi-pass AIC with the distribution of cover elements (c) Multi-pass BIC with the distribution of cover elements.

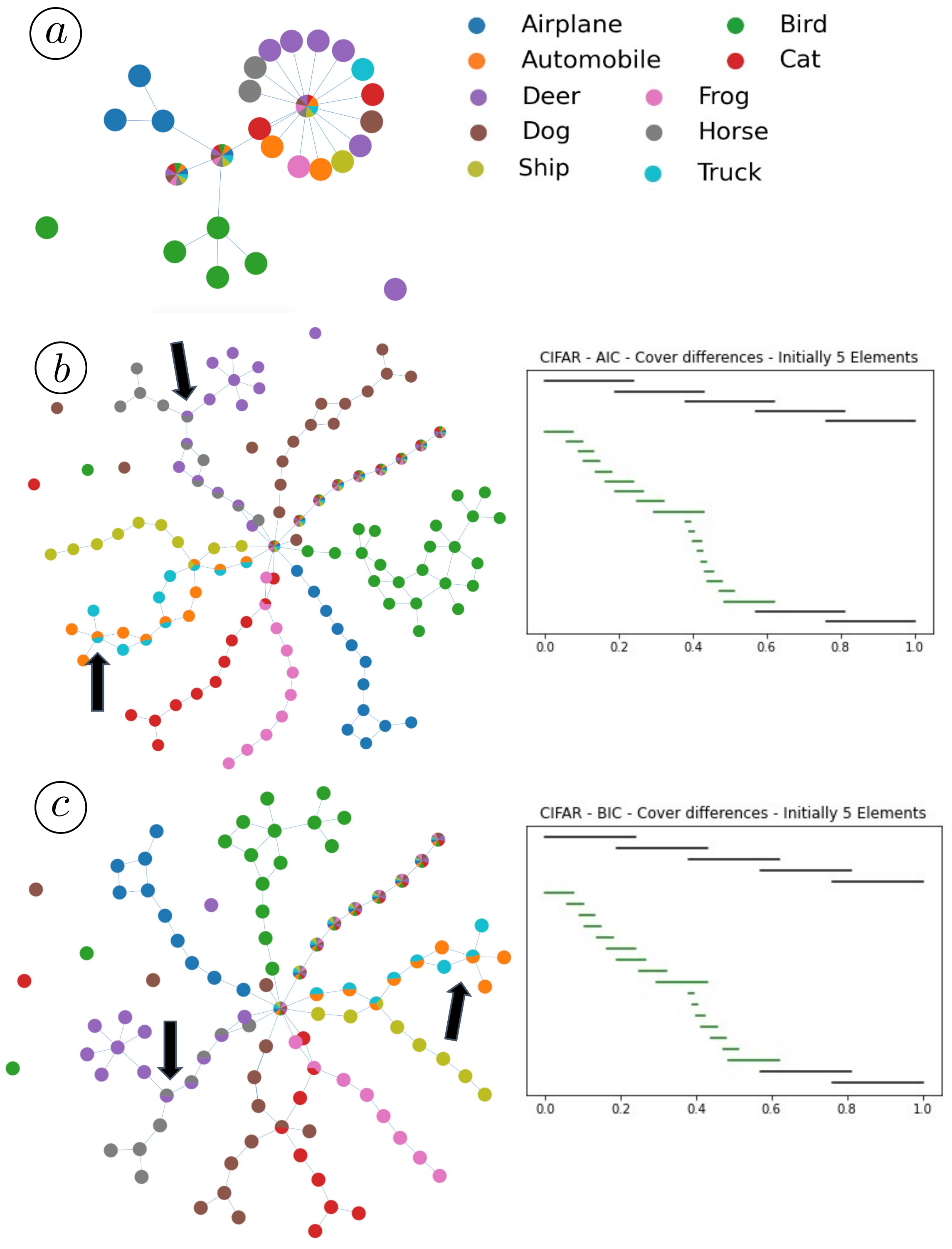


Figure 3.13. CIFAR: Generated mapper graphs when $l = 5$. (a) Classic mapper graph (b) Multi-pass AIC with the distribution of cover elements (c) Multi-pass BIC with the distribution of cover elements.

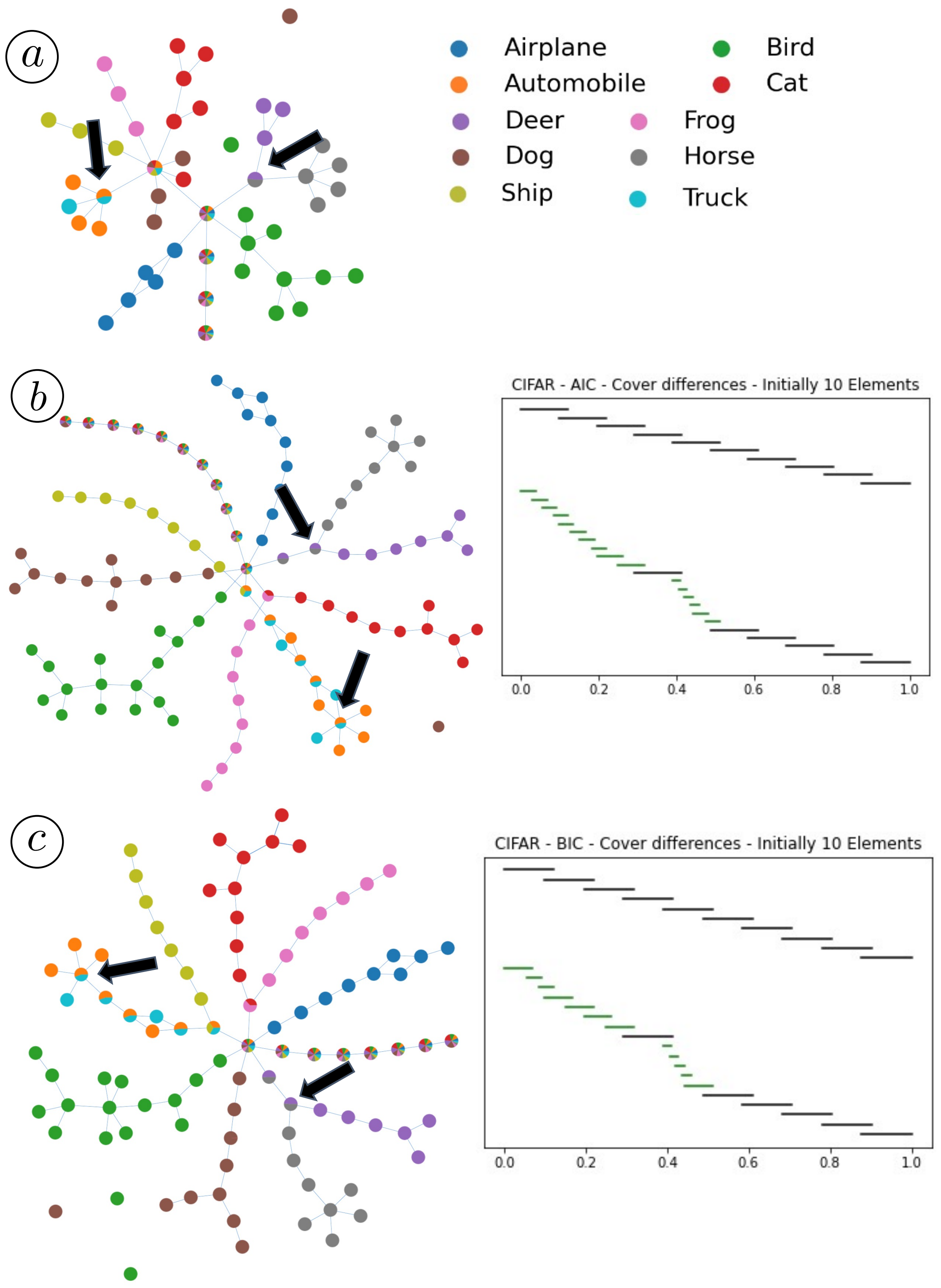

a
Airplane
Bird
Automobile
Cat
Deer
Frog
Dog
Horse
Ship
Truck
b
CIFAR - AIC - Cover differences - Initially 10 Elements
0.0
0.2
0.4
0.6
0.8
1.0
c
CIFAR - BIC - Cover differences - Initially 10 Elements
0.0
0.2
0.4
0.6
0.8
1.0

Figure 3.14. CIFAR: Generated mapper graphs when $l = 10$. (a) Classic mapper graph (b) Multi-pass AIC with the distribution of cover elements (c) Multi-pass BIC with the distribution of cover elements.

X-means graphs start the Automobile vs Truck branch but do not expand it further. We also include the results of using $l = 10$ in Fig. 3.14 to see how multi-pass X-means performs with a finer cover. Both versions of multi-pass X-means, Fig. 3.14(b,c), preserve the start of the Automobile vs Truck branch as well as consistently generating the Horse vs Deer bifurcation.

Fig. 3.15 and Fig. 3.16 include the classic mapper graph and the resulting mapper graph of multi-pass X-means for $l = 15$ and $l = 20$, respectively. When $l = 15$, the Automobile vs Truck branch start to form. Increasing the resolution of the initial uniform cover to $l = 20$ in Fig. 3.16, multi-pass X-means using the AIC (b) expands the branch found in the classic mapper graph (a); multi-pass X-means using the BIC (c) is similar. At this initialization, there is a clear branching node for both the Truck vs Automobile branch and the Deer vs Horse branch.

For the final set of mapper graphs, we examine behavior of multi-pass X-means when given the hand-tuned parameter of $l = 40$, shown in Fig. 3.17. Both multi-pass variants preserve the primary branching structures but are able to further refine the cover to create loops in the graph. The largest loops are pointed out with blue arrows. Similar to the Circles dataset, multi-pass X-means using the AIC (b) or BIC (c) is able to discover loops that are not detected in the classic mapper graph. As noted by Rathore *et al.* [30], the meaning of loops in the space of neural network activations is slightly harder to interpret. Nevertheless, our strategy demonstrates that there are non-trivial loops in the high-dimensional activation space. If we examine the differences between the uniform cover and adaptive cover in Fig. 3.17(b, c), it is clear that multi-pass X-means refines the cover in the middle of the filter function. This aligns with Rathore *et al.* where they point out that the majority of activations have an L_2-norm in the middle of the filter function [30]. Multi-pass X-means is able to recognize this and refines the cover to provide a more detailed mapper graph.

In addition to the four values of l we've shown, we also tested our method across a range of parameters from 3 to 200 and plotted the respective AIC and BIC values in Fig. 3.18. For most parameter settings, mutlipass X-means improves the AIC (a) or BIC (b) value. This is especially true for smaller values of l. The plots in Fig. 3.18 align with our qualitative findings since multi-pass X-means changes the structure of the classic mapper graph the most for small l.

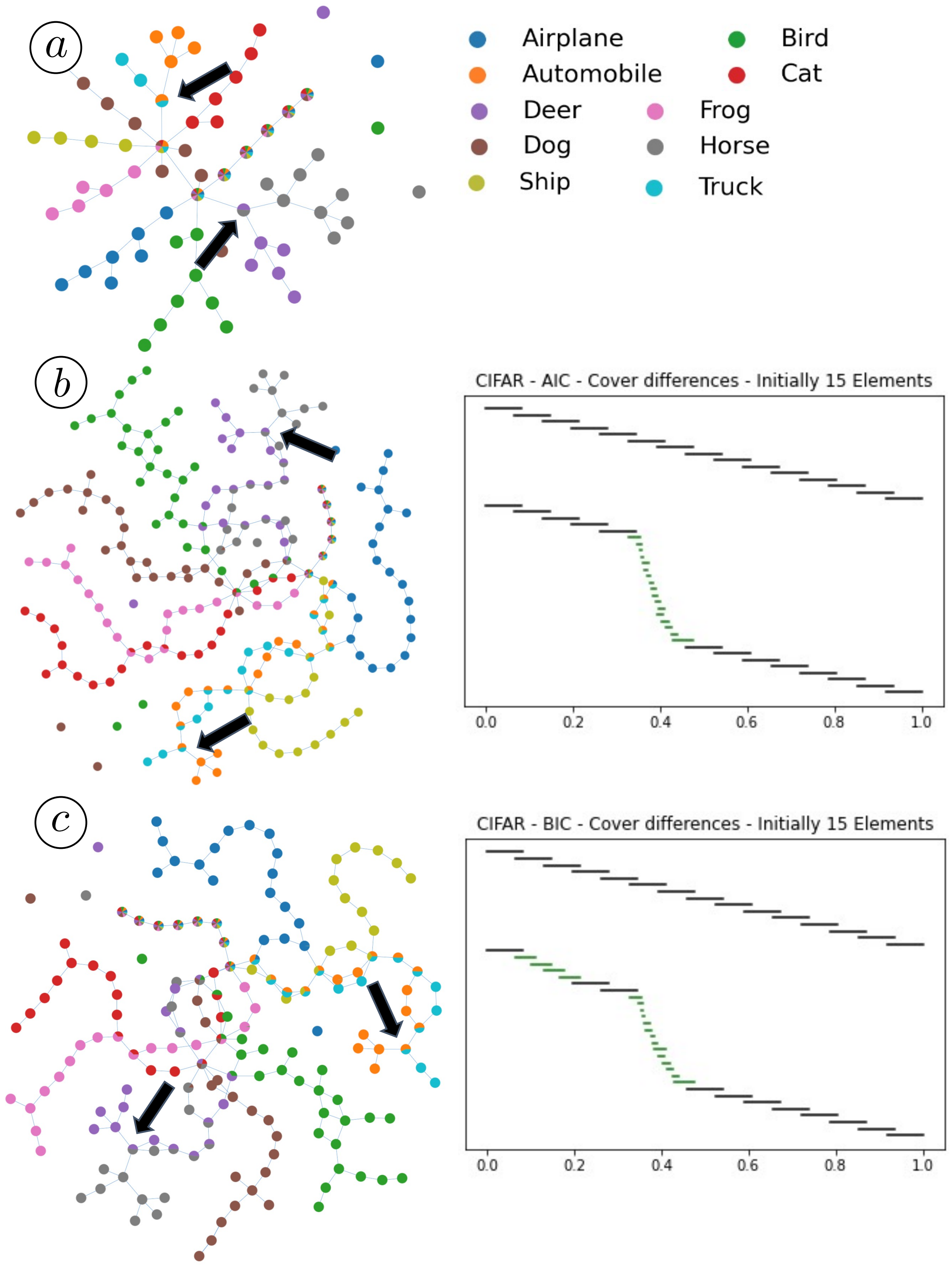


Figure 3.15. CIFAR: Generated mapper graphs when $l = 15$. (a) Classic mapper graph (b) Multi-pass AIC (c) Multi-pass BIC.

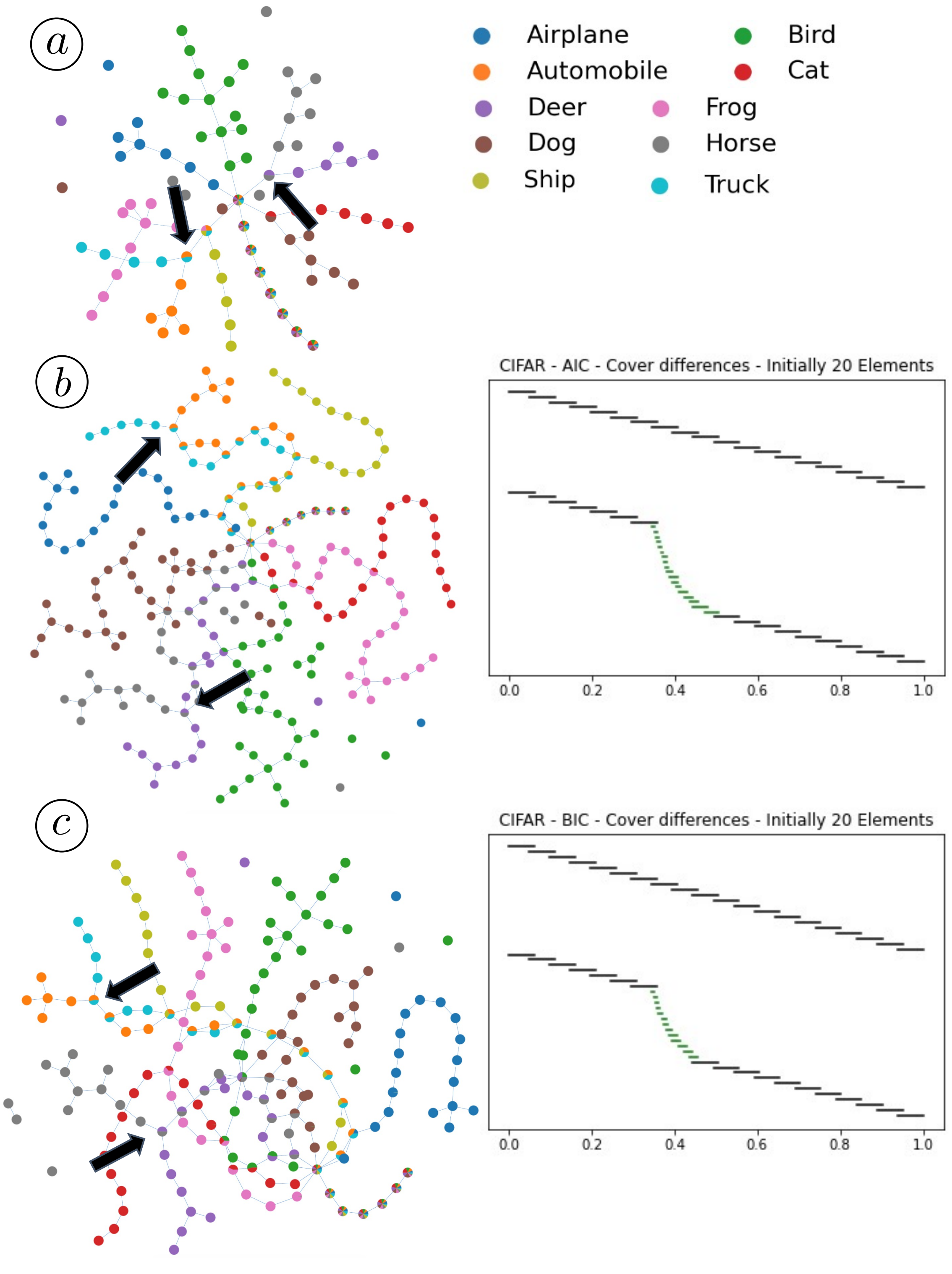


Figure 3.16. CIFAR: Generated mapper graphs when $l = 20$. (a) Classic mapper graph (b) Multi-pass AIC (c) Multi-pass BIC.

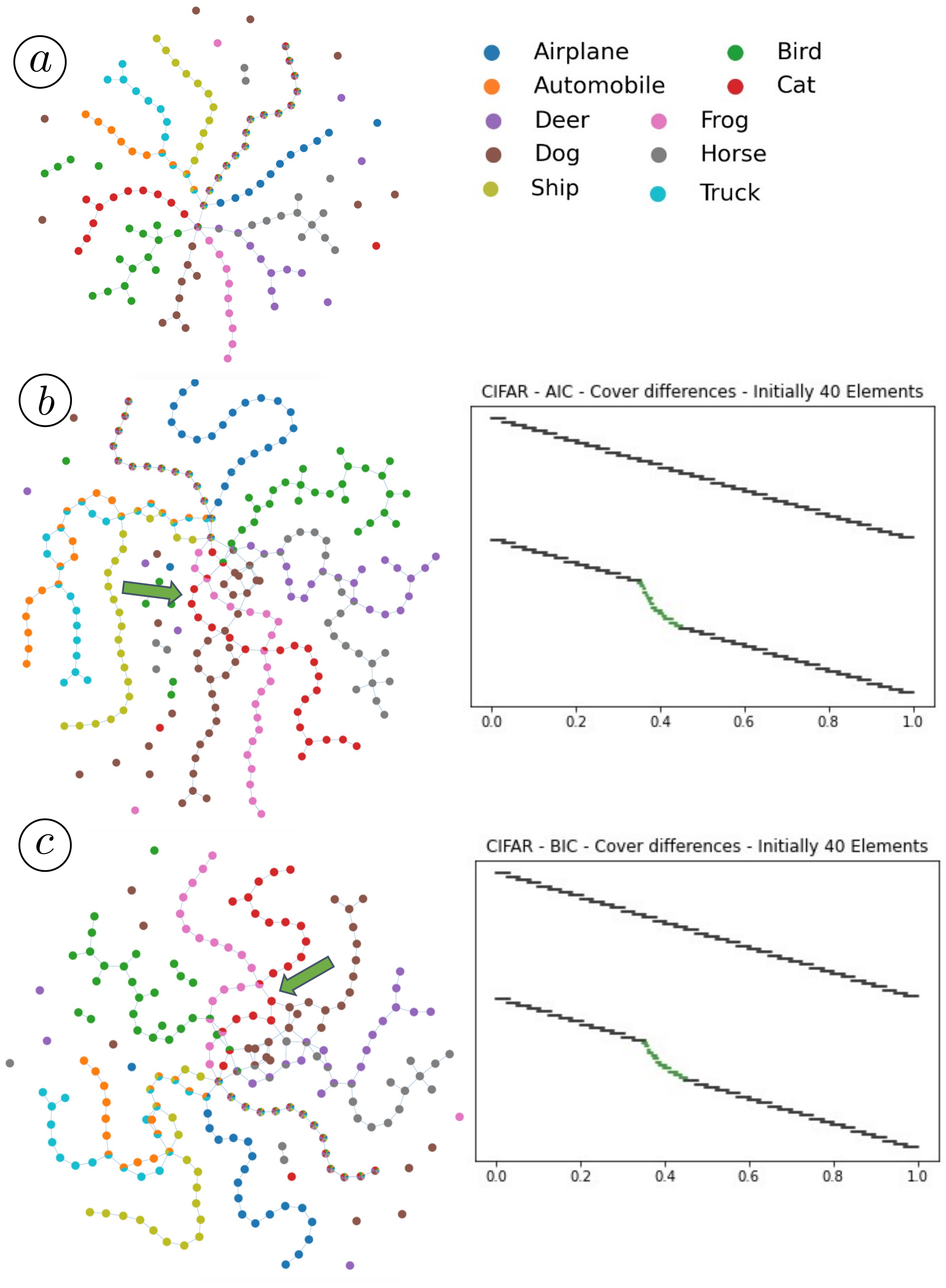
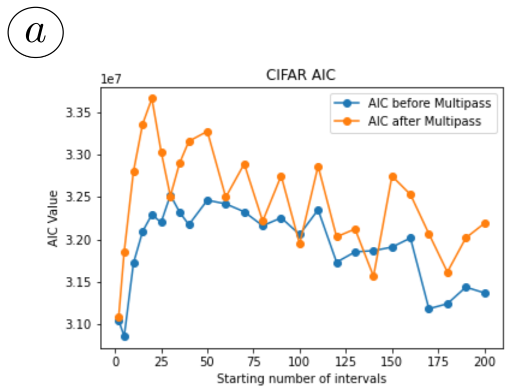


Figure 3.17. CIFAR: Generated mapper graphs when $l = 40$. (a) Classic mapper graph (b) Multi-pass AIC (c) Multi-pass BIC.



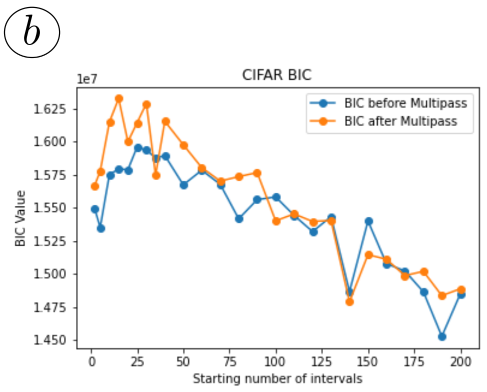


Figure 3.18. CIFAR: (a) AIC before and after multi-pass. (b) BIC before and after multi-pass.

3.3 Discussion

3.3.1 Overall Evaluation

While omitted in the main results section, we find that single-pass X-means converges to a set of isolated connected components with no significant branching or loops. Multi-pass X-means, as demonstrated on three datasets, converges to the hand-tuned mapper graph in a number of cases. On the real-world datasets, we tested aggressively small parameters to see how multi-pass X-means performs. We found that, even parameters far from the hand-tuned value, the resulting multi-pass X-means mapper graphs converge to the hand-tuned graphs. We also observe that multi-pass X-means is able to uncover some loop structures. This is first noticeable on the circles dataset and is further illustrated by CIFAR-10 by running multi-pass X-means on the hand-tuned mapper graph. For both Circles and CIFAR-10, the AIC and BIC local maxima found by multi-pass X-means is higher than the classic mapper graph's AIC or BIC value. On the COVID-19 dataset, the AIC and BIC local maxima is smaller than classic mapper graph's AIC or BIC. However, qualitatively, multi-pass X-means, provides a mapper graph that is a refinement of the classic mapper graph generated from the initial parameters.

3.3.2 Parameter Selection Recommendations

If a user is met with an unknown dataset and would like to employ the mapper framework, how should they proceed? With no knowledge of the ideal parameters, we recommend users to try multi-pass X-means across various overlap parameters with small values for the number of intervals. By doing this, the user will gain some intuition and knowledge on what the hand-tuned "ideal" mapper graph might be. With this knowledge, the user may opt to hand tune parameters, using the number of intervals in the multi-pass X-means adaptive cover as a starting point for a new uniform cover. If the user has a vague idea of the hand-tuned parameters, we recommend running multi-pass X-means on those parameters to study how the hand-tuned graph may be improved. With these guidelines, it is easier for a practitioner to refine a hand-tuned classic mapper graph.

3.3.3 Future Work

Some researchers have applied topological concepts to information theory [19]. Numerous concepts of entropy for graphs and simplicial complexes have also been proposed [7, 8]. Because of the strong connection between entropy and the KL divergence, it may be possible to derive a topologically informed score for mapper graphs. Our strategy tries to minimize the KL divergence between the underlying true probability distribution and a set of identically spherical Gaussians; there may be a way codify topological structures into probability functions.

In the future, we would also like to explore the possibility of merging cover elements. Suppose that there is an extremely refined cover. We would like to work backwards by merging rather than splitting cover elements. In terms of the existing theory, the mapper construction has been proven to recover the underlying Reeb graph. In some sense, this avenue of work would entail computing as close of an approximation to the Reeb graph as possible with refinement as necessary via an adaptive cover.

CHAPTER 4

ENHANCED MAPPER

The second contribution of this thesis is an open source Python library that computes the enhanced mapper graph. We first outline an algorithmic description of the enhanced mapper graph and then introduce our library's features. We end with some examples of how the enhanced mapper graph might be used to derive new insights about both the COVID-19 and CIFAR-10 datsets.

4.1 Enhanced Mapper Graph Description

We give an algorithmic description of the enhanced mapper graph and discuss the differences to the classic mapper graph. For a mathematical description of the enhanced mapper graph, we refer the reader to [3].

Let $\mathbb{X}$ be a point cloud with filter function f. Suppose that $\mathcal{U}$ is a cover of the image of f, constructed the same way as the classic mapper graph. Now construct a set of closed sets $\mathcal{O}$ by considering cover elements in $\mathcal{U}$ with a nonempty intersection. That is, $\mathcal{O} = \{U_i \cap U_j | U_i \cap U_j \not\subseteq \emptyset, U_i, U_j \in \mathcal{U}\}$. Because $\mathcal{U}$ is a list of adjacent intervals on $\mathbb{R}$, $\mathcal{O}$ is the region of overlap between two consecutive intervals. Note that $|\mathcal{O}| = |\mathcal{U}| - 1$. In Fig. 4.1, we show how to construct $\mathcal{O}$ (b) for the point cloud and cover in (a) coming from Fig. 2.1.

Similar to the classic mapper graph, we compute the connected components of $f^{-1}(U_i)$ for $U_i \in \mathcal{U}$. We also consider the connected components of $f^{-1}(O_j)$ for $O_j \in \mathcal{O}$. For convenience, define π to be a function that returns the connected components.

We can now construct the enhanced mapper graph $\mathcal{M}$. The enhanced mapper graph is defined by three objects: the vertex set V, the edge set E, and a function $g : V \to \mathbb{R}$. Consider one interval U_i in $\mathcal{U}$ and one connected component of $f^{-1}(U_i)$, c_j. Let O^- and O^+ be the intersections with U_{i-1} and U_{i+1} respectively. We define two vertices in the mapper graph, v_j^+ and v_j^-, with an edge between them. $g(v_j^+)$ is defined by comparing the maximum value of $f(c_j)$ with connected components of O^+. Formally:

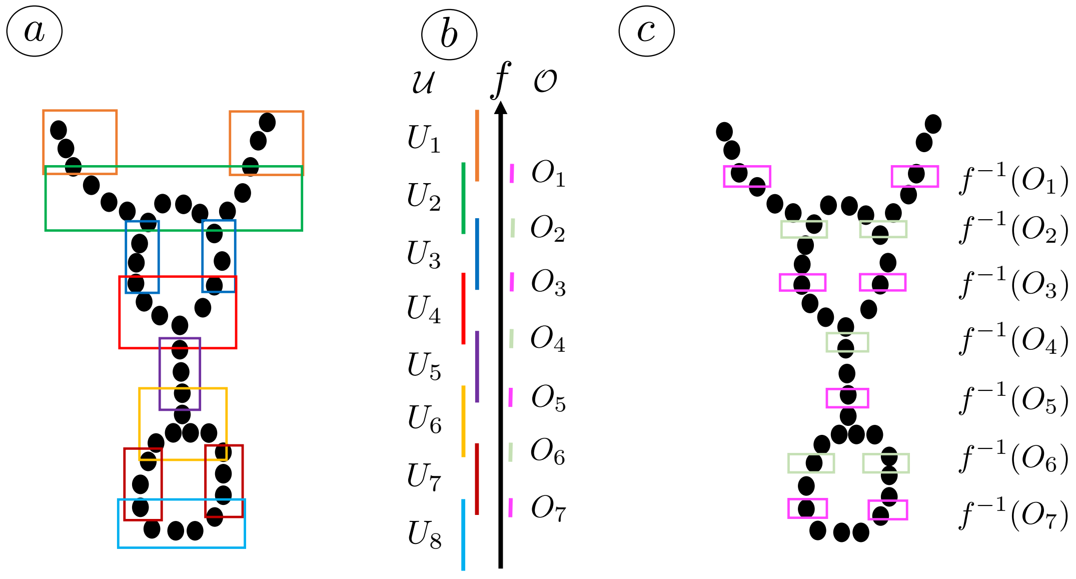


Figure 4.1. An example construction of $\mathcal{O}$ and $f^{-1}(\mathcal{O})$. (a) The point cloud from Fig. 2.1 with connected components of $f^{-1}(\mathcal{U})$ in the boxes. (b) The filter function f with $\mathcal{U}$ and overlaps $\mathcal{O}$. (c) The connected components of $f^{-1}(\mathcal{O})$.

$$g(v_j^+) = \begin{cases} \min\limits_{\substack{o_k \in \pi(O^+) \\ o_k \cap c_j \not\subseteq \emptyset}} (\min f(o_k)) & \text{if there exists } o_k \in \pi(O^+) \text{ such that } c_j \cap o_k \not\subseteq \emptyset \\ \max f(c_j) & \text{otherwise} \end{cases} \tag{4.1}$$

Similarly, we can define the value of $g(v_i^-)$ by examining O^-:

$$g(v_j^-) = \begin{cases} \max\limits_{\substack{o_k \in \pi(O^-) \\ o_k \cap c_j \not\subseteq \emptyset}} (\max f(o_k)) & \text{if there exists } o_k \in \pi(O^-) \text{ such that } c_j \cap o_k \not\subseteq \emptyset \\ \min f(c_j) & \text{otherwise} \end{cases} \tag{4.2}$$

At this stage, $\mathcal{M}$ consists only of vertices with one edge between them. Before proceeding further, we provide an example of where in the pipeline we are. In Fig. 4.2, (c) contains the connected components of $f^{-1}(\mathcal{U})$. For each connected component, we split it into a positive and negative node in (d). Notice that there are no edges between nodes coming from different connected components of $f^{-1}(\mathcal{U})$. We now explain how to compute the pink edges in (e), the final enhanced mapper graph.

Consider two connected components c_i and c_j coming from two overlapping cover elements in $\mathcal{U}$. Suppose that the intersection between c_i and c_j is nonempty. Let v_i^+, v_i^- be vertices from c_i and v_j^+ and v_j^- be vertices from c_j. If $g(v_i^+) < g(v_j^-)$, we add an edge between

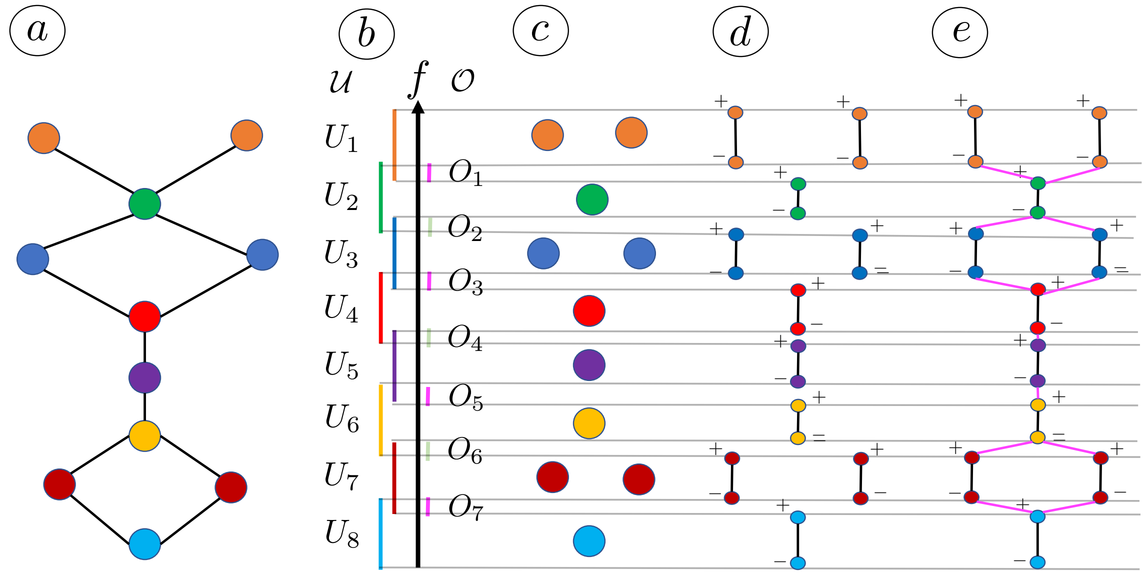


Figure 4.2. The enhanced mapper graph construction of the point cloud in Fig. 4.1. We include lines across to denote the filter function value. (a) The classic mapper graph. (b) The filter function with cover elements $\mathcal{U}$ and overlaps $\mathcal{O}$. (c) The connected components of $f^{-1}(\mathcal{U})$. (d) The generation of individual vertices in the enhanced mapper graph with "+" and "-" denoting the positive and negative vertices. (e) The final enhanced mapper graph. The added edges, in comparison to (d), are in pink.

v_i^+ and v_j^-. Otherwise, we add an edge between v_i^- and v_j^+. We have now constructed the enhanced mapper graph. Returning to Fig. 4.2, (e) contains the final enhanced mapper graph. Each of the pink edges are added by this process.

4.1.1 Comparison to the Classic Mapper Graph

The enhanced mapper graph provides more geometric information about the point cloud. In particular, the function g gives a more refined understanding of the topological structure. For each interesting feature in the enhanced mapper graph, there is an associated filter function value where the feature occurs. Moreover, each of the edges in the enhanced mapper graph has an associated length. We demonstrate how to use this information in section 4.3.

4.2 Implementation and Features

Our library is written in Python and supports Numba, a high-performance “just-in-time” Python compiler [17]. In addition to implementing the enhanced mapper graph, our library has three features that make it appealing to practitioners wanting to leverage the mapper framework: programmatic GPU acceleration, easy exportation, and quality of life features.

Programmatic GPU acceleration. Zhou *et al.* [39] showed that using a GPU provides a speed-up between 6x to 12x for 1 million data points of 256 dimensions. They also provide a command line interface (Bash) for accessing the GPU computation. However, we take it a step further and provide a programmatic way to access the GPU computation. By making the GPU acceleration available as a library, users with existing Python code can use our library to speed up their existing pipelines. For our GPU computations, we use PyTorch and include the ability to parallelize across multiple GPUs. As noted by Zhou *et al.* [39], the slowest component of the mapper pipeline is the DBSCAN subroutine, specifically the pairwise distance computation. We can instead run this on a GPU.

Easy exportation. Our library includes the ability to take computed mapper graphs and export them to other libraries and tools. We provide converters to NetworkX [11], a graph and network analysis library, PyVis [27], a Python based network visualization library, and Mapper Interactive [39], a state-of-the-art visualization tool for mapper graphs. This feature allows users to quickly prototype and try new ideas without worrying about implementing a new visualization.

Quality of life features. We have a variety of features to streamline the development experience of users. We provide built-in methods for computing the distribution of the k-th nearest neighbor to aid users when selecting ϵ when using DBSCAN. Moreover, our mapper graph object is treated as a Python dictionary with additional methods for indexing and analysis. Our analysis tools include computing connected components, shortest paths, and the Fowlkes Mallows Index with another clustering of the data. Because our mapper graph is also a Python dictionary, it can access all of Python’s existing libraries that operate on dictionaries.

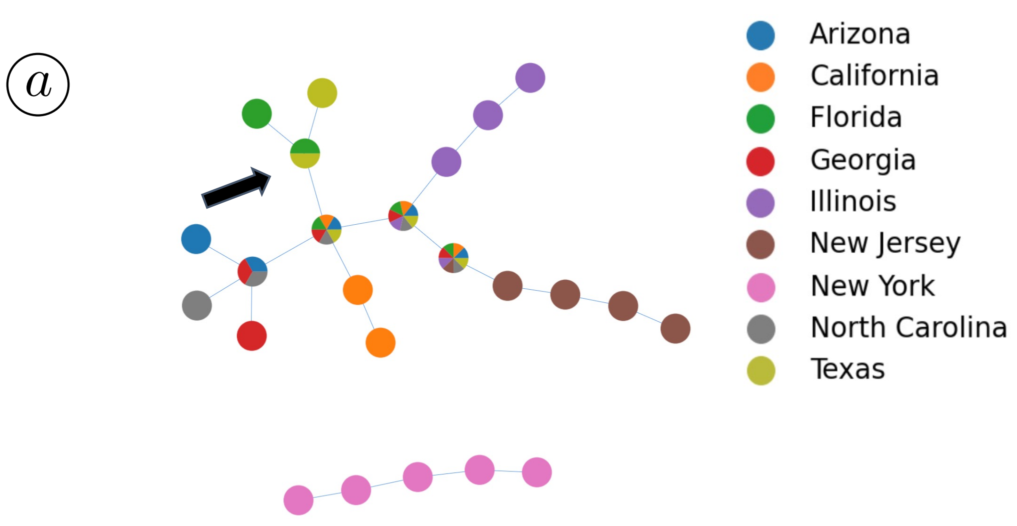
4.3 Results

To illustrate the power of the enhanced mapper graph, we show a few examples using the COVID-19 and CIFAR-10 real-world datasets. For the COVID-19 examples, we set DBSCAN's ϵ to 0.15 and $minPts$ to 5. For CIFAR-10, ϵ is 8.71 and $minPts$ is 5.

COVID-19. In Fig. 4.3, we have the classic mapper graph (a) and the enhanced mapper graph (b) when the number of cover elements is 5 with 50% overlap. This is one of the first parameter choices to show the Texas (light green) and Florida (dark green) branching structure. In (a), the branching node is indicated by a black arrow. If we examine the corresponding positive and negative vertices of the branching node in the enhanced mapper graph, again denoted by a black arrow in Fig. 4.3(b), we can find a corresponding function value for when the branch forms. The negative branching node has a function value of 106 days and the positive branching node has a function value of 132.5 days. This result implies that Texas and Florida branch from the other states at around July 27, 2020. On August 22, 2021, Texas and Florida bifurcate from each other.

We can perform similar analysis on the branch containing Arizona (blue) and Georgia (red). In Fig. 4.4, (a) contains the classic mapper graph computed with 20 intervals and 50% overlap, and the enhanced mapper graph is shown in (b). The classic mapper graph has two nodes before Arizona and Georgia completely bifurcate. The function value of the node closest to the mixed node is 90 and the function value of the node right before the bifurcation is 106. Analogously, Arizona and Georgia break away from other states around July 11, 2020 and closely resemble each other's epidemic trends. Arizona and Georgia continue on the same trend until July 26, 2020 when the two states follow different epidemic trajectories.

Fig. 4.5 contains the classic mapper graph (a) and the enhanced mapper graph (b) for 22 intervals and 50% overlap. 22 intervals is the largest number of cover elements where the classic mapper graph connects New Jersey to the rest of the states. This suggests that New Jersey quickly diverged from the epidemic trends of other states. Using the enhanced mapper graph, we can gain a better understanding of when New Jersey branches from the other states. Looking at the branching node, denoted by a black arrow, the enhanced mapper graph tells us the function value of the negative branching node is 6.9. That is, within one week of data collection (April 19, 2020), New Jersey starts along its own epidemic trend. The New York connected component in Fig. 4.5(b, pink) appears as an isolated island



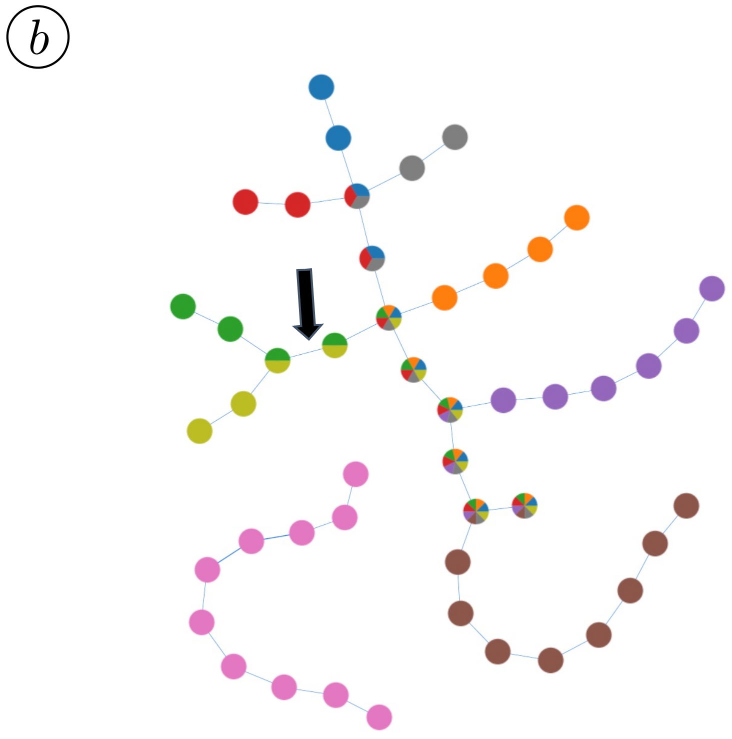


Figure 4.3. COVID-19: (a) Classic mapper graph with 5 intervals and 50% overlap. (b) The corresponding enhanced mapper graph.

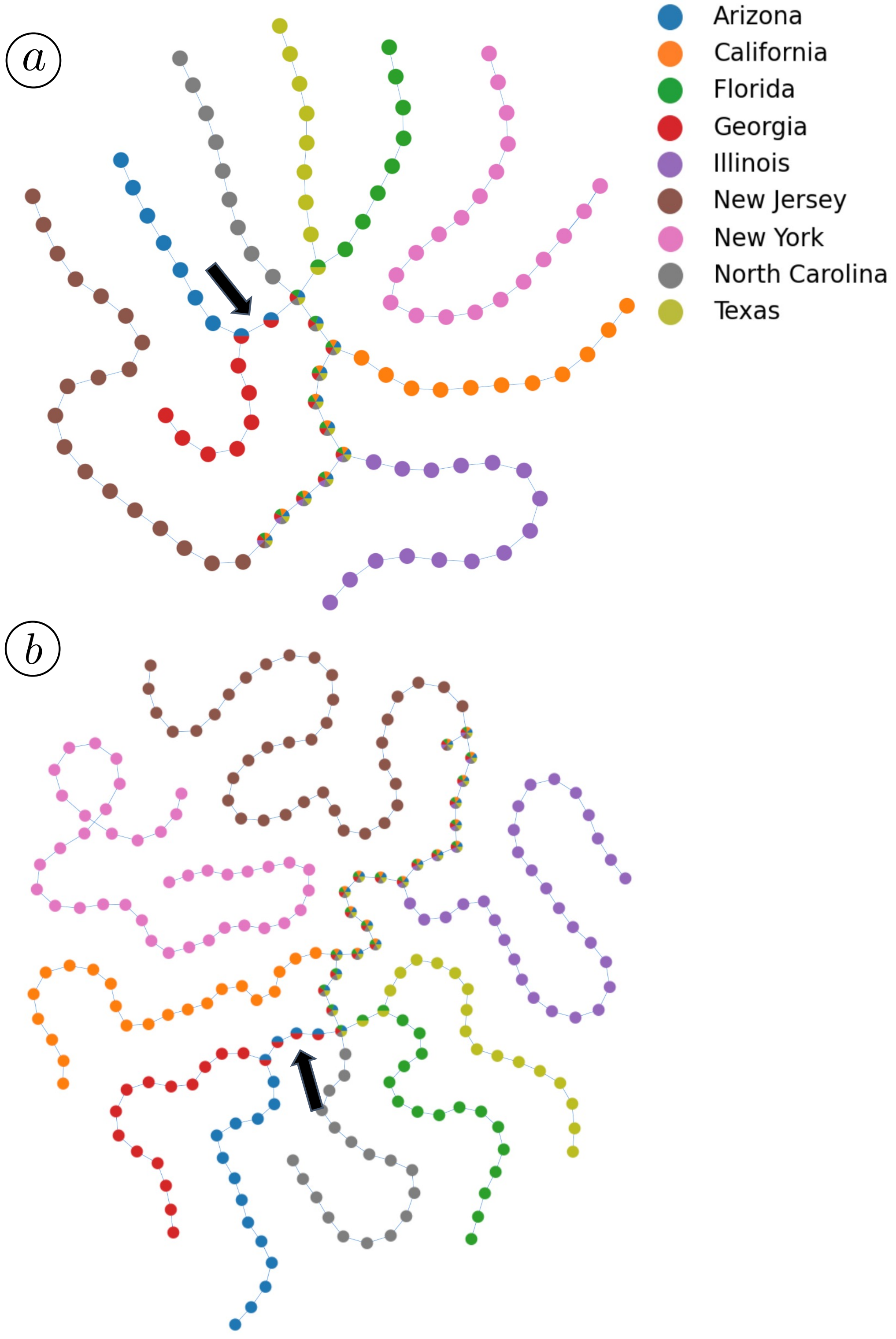


Figure 4.4. COVID-19: (a) Classic mapper graph with 20 intervals and 50% overlap. (b) The corresponding enhanced mapper graph.

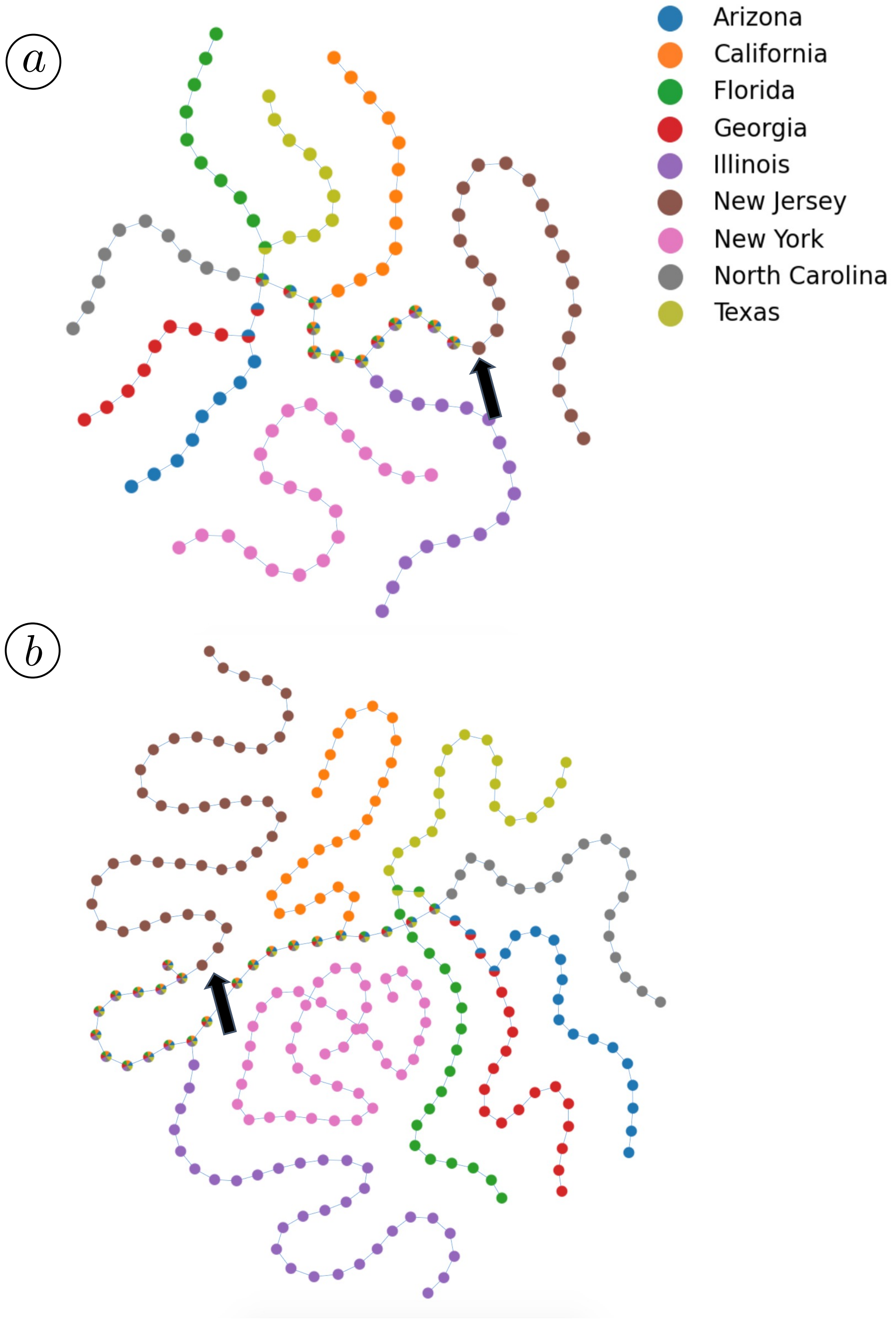


Figure 4.5. COVID-19: (a) Classic mapper graph with 22 intervals and 50% overlap. (b) The corresponding enhanced mapper graph.

under a number of parameters. To investigate further, we examine the filter function value of the two endpoint nodes. Each node has a value of 0 and 160. During the entire data collection window, New York had a different epidemic trend from all states.

CIFAR-10. Our final example in Fig. 4.6 comes from the CIFAR-10 dataset. The classic mapper graph (a) and the enhanced mapper graph (b) are generated using 70 intervals with 20% overlap. Suppose that we wish to investigate the loop in (a) at the end of the black arrow. We can examine the corresponding function values of the two vertices that link the loop to the rest of the graph in (b). In particular, one vertex has a function value (L_2-norm) of 15.4 and another with 16.0. That is, the loop is created at an L_2-norm of 15.4 and disappears at 16.0. While the meaning of loops is slightly harder to interpret than branches in the space of neural network activations, it is still informative to know the filter function values when the loops first appear.

4.4 Discussion

Our four examples show cases where the enhanced mapper graph uniquely informs our understanding of the underlying topology of the data. While the classic mapper graph allows us to study the general structure and interesting topological features of a point cloud, the enhanced mapper graph allow us to make pointed statements about the data (e.g., when New Jersey's epidemic trends deviate from other states). More broadly, the edge lengths given by the enhanced mapper graph gives us geometric information about interesting topological features.

Future work. Future work along this line includes first exploring existing datasets that have known mapper graph results [10, 20, 22, 25]. In all the referenced examples, the classic mapper graph has shown interpretable results. We can then use the enhanced mapper graph to further explore these known results.

We would also like to explore the usage of machine learning with the enhanced mapper graph. Graph neural networks have seen increasingly promising results [38]. Using the recent advances in graph neural networks and the geometric information the enhanced mapper graph provides, we can attempt to use the mapper graph as input to graph neural networks. This presents both implementation and research challenges. In terms of implementation, our library needs to be extended to work with existing deep learning

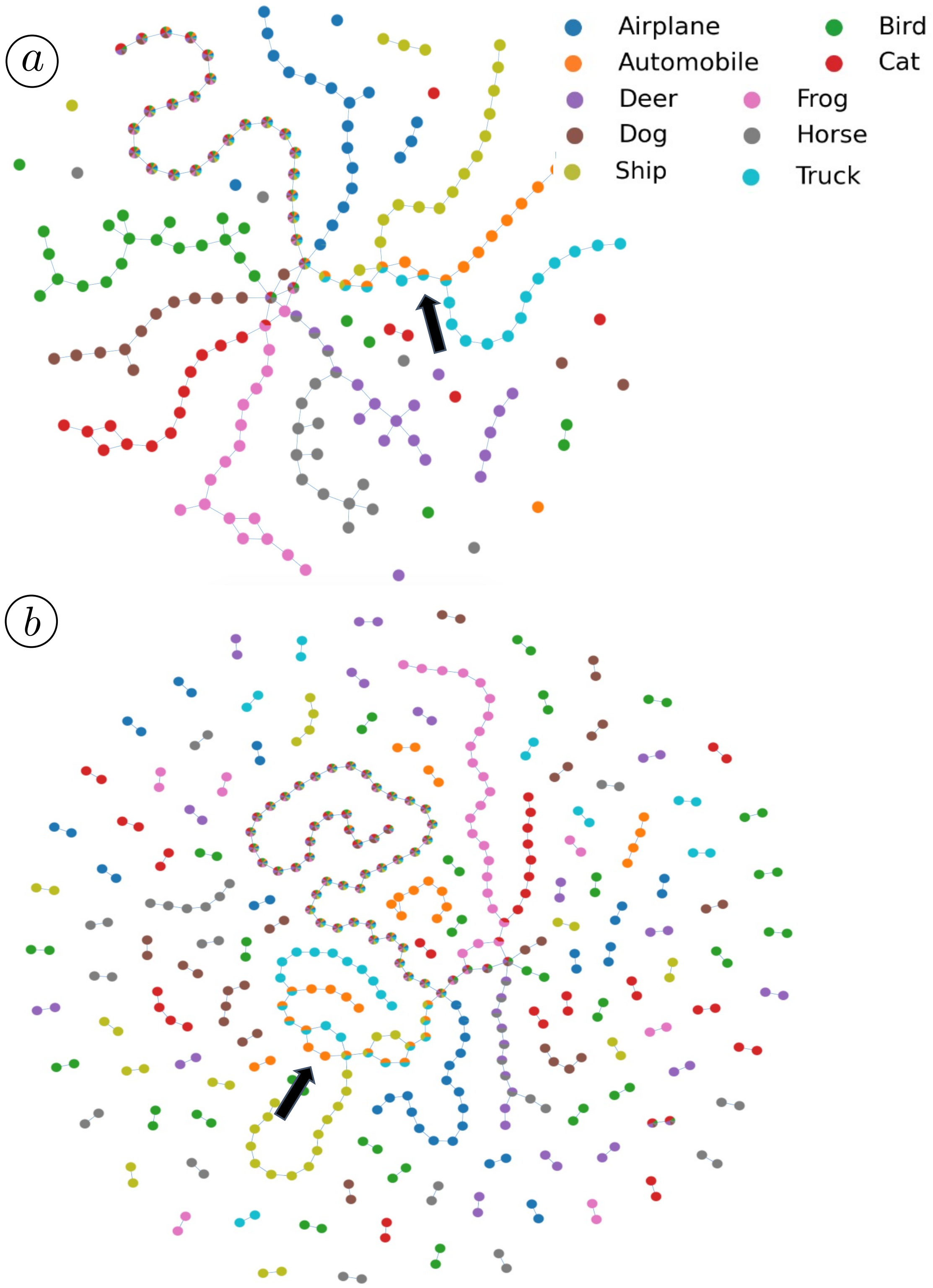


Figure 4.6. CIFAR-10: (a) Classic mapper graph with 70 intervals and 20% overlap. (b) The corresponding enhanced mapper graph.

libraries, like PyTorch. On the research side, it would be nontrivial to demonstrate that the enhanced mapper graph with graph neural networks leads to an improvement over existing methods.

CHAPTER 5

CONCLUSION

The mapper construction is a popular data exploration and visualization tool from topological data analysis. In this thesis, we explored the question of parameter selection and released an open source library for computing the enhanced mapper graph.

We explored three strategies for tuning the parameters based on the Akaike information criterion (AIC) and the Bayesian information criterion (BIC) in Chapter 3. We show that, experimentally, multi-pass X-means converges to the hand-tuned or "ground-truth" mapper graph. Our testing indicates that, under a range of parameters, multi-pass X-means is able to recover significant branching and some loop structures even when the classic mapper graph at the initialization parameters does not exhibit these features. These results point to information theoretic measure being a very promising direction for adaptive cover selection. For future work, we would like to explore the merging of cover elements and various notions of entropy on simplicial complexes to derive a new information criterion for the mapper graph.

In chapter 4, we articulate the algorithmic description of the enhanced mapper graph, detail features of our open source library, and provide four examples of the type of analysis the enhanced mapper graph supports. By using information from the enhanced mapper graph, we can relate interesting topological features with specific filter function values. In the COVID-19 dataset, for example, we can pinpoint the number of days since April 12, 2020 when states start experiencing different epidemic trends. We hope to use the enhanced mapper graph to provide interpretable results on new and known datasets in the future. Another potential direction for the enhanced mapper graph is to integrate with graph neural networks. We would like to explore how the extra information provided by the enhanced mapper graph can inform machine learning. We hope that users of the mapper graph will be able to use the results of this thesis to enhance their analysis and visualization pipeline.

REFERENCES

[1] Akaike, H. A new look at the statistical model identification. *IEEE Transactions on Automatic Control 19*, 6 (1974), 716–723.

[2] Arthur, D., and Vassilvitskii, S. K-Means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms* (USA, 2007), SODA '07, Society for Industrial and Applied Mathematics, pp. 1027–1035.

[3] Brown, A., Bobrowski, O., Munch, E., and Wang, B. Probabilistic convergence and stability of random mapper graphs. *Journal of Applied and Computational Topology 5*, 1 (Mar. 2021), 99–140.

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

[5] Carrire, M., Michel, B., and Oudot, S. Statistical analysis and parameter selection for mapper. *Journal of Machine Learning Research 19*, 12 (2018), 1–39.

[6] Cover, T. M., and Thomas, J. A. *Elements of information theory (wiley series in telecommunications and signal processing)*. Wiley-Interscience, USA, 2006.

[7] Dantchev, S., and Ivrissimtzis, I. Simplicial complex entropy. In *Mathematical methods for curves and surfaces. MMCS 2016. Lecture notes in computer science*, M. Floater, T. Lyche, M. L. Mazure, K. Mrken, and L. Schumaker, Eds., vol. 10521. Springer, 2017.

[8] Dehmer, M., and Mowshowitz, A. A history of graph entropy measures. *Information Sciences 181*, 1 (Jan. 2011), 57–78.

[9] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. 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* (1996), KDD'96, AAAI Press, pp. 226–231.

[10] Geniesse, C., Sporns, O., Petri, G., and Saggar, M. Generating dynamical neuroimaging spatiotemporal representations (DyNeuSR) using topological data analysis. *Network neuroscience (Cambridge, Mass.) 3*, 3 (July 2019), 763–778. Publisher: MIT Press.

[11] Hagberg, A. A., Schult, D. A., and Swart, P. J. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th python in science conference* (Pasadena, CA USA, 2008), G. Varoquaux, T. Vaught, and J. Millman, Eds., pp. 11 – 15.

[12] Hajij, M., Rosen, P., and Wang, B. Mapper on graphs for network visualization. *Unpublished Manuscript* (2019).

[13] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE conference on computer vision and pattern recognition (CVPR)* (2016), pp. 770–778.

[14] Kanungo, T., Mount, D., Netanyahu, N., Piatko, C., Silverman, R., and Wu, A. An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 7 (July 2002), 881–892.

[15] Kass, R. E., and Wasserman, L. A reference bayesian test for nested hypotheses and its relationship to the schwarz criterion. *Journal of the American Statistical Association 90*, 431 (1995), 928–934.

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

[17] Lam, S. K., Pitrou, A., and Seibert, S. Numba: A llvm-based python jit compiler. In *Proceedings of the second workshop on the LLVM compiler infrastructure in HPC* (2015), pp. 1–6.

[18] Lloyd, S. Least squares quantization in PCM. *IEEE Transactions on Information Theory 28*, 2 (Mar. 1982), 129–137.

[19] Martin, K. Topology in information theory in topology. *Computational Structures for Modelling Space, Time and Causality 405*, 1 (Oct. 2008), 75–87.

[20] Mathews, J. C., Nadeem, S., Levine, A. J., Pouryahya, M., Deasy, J. O., and Tannenbaum, A. Robust and interpretable PAM50 reclassification exhibits survival advantage for myoepithelial and immune phenotypes. *npj Breast Cancer 5*, 1 (Sept. 2019), 30.

[21] Munch, E., and Wang, B. Convergence between Categorical Representations of Reeb Space and Mapper. *International Symposium on Computational Geometry (SOCG)* (2016).

[22] Nicolau, M., Levine, A. J., and Carlsson, G. Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proceedings of the National Academy of Sciences 108*, 17 (2011), 7265–7270.

[23] Otter, N., Porter, M. A., Tillmann, U., Grindrod, P., and Harrington, H. A. A roadmap for the computation of persistent homology. *EPJ Data Science 6*, 1 (Aug. 2017).

[24] Oudot, S. *Persistence Theory: From Quiver Representations to Data Analysis*, vol. 209 of *Mathematical Surveys and Monographs.* American Mathematical Society, 2015.

[25] Patania, A., Vaccarino, F., and Petri, G. Topological analysis of data. *EPJ Data Science 6*, 1 (June 2017), 7.

[26] Pelleg, D., and Moore, A. X-means: Extending k-means with efficient estimation of the number of clusters. In *In proceedings of the 17th international conf. on machine learning* (2000), Morgan Kaufmann, pp. 727–734.

[27] Perrone, G., Unpingco, J., and Haw-minn Lu. Network visualizations with Pyvis and VisJS. In *Proceedings of the 19th Python in Science Conference* (2020), M. Agarwal, C. Calloway, D. Niederhut, and David Shupe, Eds., pp. 58 – 62.

[28] Pham, D. T., Dimov, S. S., and Nguyen, C. D. Selection of K in K-means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science 219*, 1 (Jan. 2005), 103–119. Publisher: IMECHE.

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

[30] Rathore, A., Chalapathi, N., Palande, S., and Wang, B. TopoAct: Visually exploring the shape of activations in deep learning. *Computer Graphics Forum 40*, 1 (2021), 382–397.

[31] Rousseeuw, P. J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics 20* (Nov. 1987), 53–65.

[32] Schwarz, G. Estimating the dimension of a model. *The Annals of Statistics 6*, 2 (1978), 461 – 464.

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

[34] Tauzin, G., Lupo, U., Tunstall, L., Prez, J. B., Caorsi, M., Medina-Mardones, A., Dassatti, A., and Hess, K. giotto-tda: A topological data analysis toolkit for machine learning and data exploration. *Journal of Machine Learning Research* (2020).

[35] van Veen, H. J., Saul, N., Eargle, D., and Mangham, S. W. Kepler mapper: A flexible python implementation of the mapper algorithm. *Journal of Open Source Software 4*, 42 (2019), 1315.

[36] Walsh, K., Voineagu, M. A., Vafaee, F., and Voineagu, I. TDAview: an online visualization tool for topological data analysis. *Bioinformatics 36*, 18 (Sept. 2020), 4805–4809.

[37] Wasserman, L. Topological data analysis. *Annual Review of Statistics and Its Application 5*, 1 (2018), 501–532.

[38] Z. Zhang, P. Cui, and W. Zhu. Deep Learning on Graphs: A Survey. *IEEE Transactions on Knowledge and Data Engineering* (2020), 1–1.

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