2023

Topological Data Analysis Using the Mapper Algorithm

Jessica Girard
University of Central Florida

TOPOLOGICAL DATA ANALYSIS USING THE MAPPER ALGORITHM

by

JESSICA GIRARD
B.S. University of Florida, 2021

A thesis submitted in partial fulfilment of the requirements
for the degree of Master of Science
in the Department of Mathematics
in the College of Sciences
at the University of Central Florida
Orlando, FL

Summer Term
2023

ABSTRACT

Topological data analysis is an expanding field that attempts to obtain qualitative information from a data set using topological ideas. There are two common methods of topological data analysis: persistent homology and the Mapper algorithm; the focus of this thesis is on the latter. In this thesis, we will be discussing the key ideas behind the Mapper algorithm, following the flow from Morse Theory to Reeb graphs to the topological version of the algorithm and finally to the statistical version. Lastly, we will present an application of Mapper to the USAIR97[11] data set using the R TDAmapper package.

I dedicate this thesis to my incredible parents, Michael and Tammie Girard.

ACKNOWLEDGMENTS

First and foremost, I want to acknowledge and thank my thesis chair, Dr. Junho Lee, for helping me every step of the way; his guidance and wisdom is very much appreciated. Secondly, I want to thank my thesis committee members, Dr. Joseph Brennan and Dr. Carlos Borges, for dedicating their time to be on my panel. To sum up the acknowledgements, I want to make sure to thank my family and friends for their overwhelming support throughout my time studying mathematics; I truly could not have done it without them.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Topological data analysis is a field of study named quite accurately, as it encompasses the analysis of data sets from a topological approach. As technology continues to rapidly grow over time, so does demand for strategies to analyze big data. Topological data analysis, or TDA for convenience, provides that sense of structure to a data set that can appear otherwise too complicated or noisy to analyze. There are many methods for tackling mathematical modeling for the purpose of data analysis in existence, but every method will have its drawbacks. Topological data analysis is a newer field created with the hopes to counteract these possible drawbacks by allowing for data analysis through the measuring of a data set with an associated metric, which requires methods from topology.

A specific tool of TDA lies in the Mapper algorithm. In this thesis, we will summarize the idea behind the Mapper algorithm and analyze the data set USAIR97[11] using the R TDAmapper package.

The concept of topological data analysis starts with some data set with a finite number of points carrying some notion of distance. We will use the concepts introduced in Chapter 1 to form a sense of shape from the given data, though methods on this will vary and we will be specifically focusing on the method utilized by the Mapper algorithm (more on this in Chapter 2). Like other methods in TDA, Mapper serves as an instrument for the simplification of data analysis by constructing combinatorial representations of data sets that focus on perhaps a specific aspect of the data or reveal a pattern that may otherwise be tricky to uncover. With the implementation of methods like Mapper and the creation of a corresponding shape for our data set, we can extract qualitative properties and analyze them to find relationships within the data.

A prime example of extracting qualitative properties from data shape already occurred in 1979,

before the notion of topological data analysis, when Reaven and Miller conducted a study on 145 patients of some diabetes relation to determine the relationship between insulin and glucose tolerance [12]. Reaven and Miller analyzed this six-dimensional data set using the projection pursuit method. This method produced an shape with a main concentration of data in the center and two wings flaring off from the center, separating the data into three subgroups: those without diabetes (center), those with overt diabetes (wing), and those with chemical diabetes (other wing). This interestingly shaped figure serves to answer a qualitative question on the identification of subtypes of diabetes by examining the connected components of the graph, i.e. clusters. This connection between topological ideas and a graphical output from a data analysis method describes the basic notion of topological data analysis.

This study was later replicated using Mapper within the same paper describing the invention of the algorithm, Singh et al. [1] applying specific parameters to produce the same three subgroups as a nerve complex, which will be defined in Chapter 1. A famous application of Mapper involving Type 2 diabetes subgroup analysis is briefly described in Chapter 3 [3].

In Chapter 1 of this thesis, we go over some important preliminary information necessary for understanding the Mapper algorithm. Chapter 2 then goes into both the topological and statistical version of Mapper, explaining the steps behind the algorithm and giving a detailed description of its parameters. Chapter 2 will follow the flow of progression from Morse Theory to Reeb Graphs, then to the topological version of the algorithm, and finally to the statistical version. Chapter 3 features an applications of the Mapper algorithm, analyzing USAIR97[11] with a focus on flight patterns. Our conclusion will close off this thesis with a brief summary of the Mapper algorithm and a discussion on possible improvements.

CHAPTER 2: PRELIMINARIES

To understand topological data analysis, we must first understand the underlying concept of topology. We will now cover the necessary definitions and theorems for the topics outlined in this thesis.

We introduce definitions of a simplex, both a geometric and an abstract simplicial complex, the corresponding underlying space, and a particular example of an abstract simplicial complex, a nerve complex. For more detailed information on topology, see [5] and [4]; for a detailed background on topological data analysis, see [2].

The foundation for the TDA process is that of creating simplicial complexes from the given data, which requires the defining of both simplices and simplicial complexes.

Definition 2.0.1. For a linearly independent set of points $\{v_0, v_1, ..., v_n\}$ in $\mathbb{R}^N$, a **simplex** can be defined as the span of those points such that

$$\sum_{i=0}^{n} a_i v_i \quad \text{for} \quad \sum_{i=0}^{n} a_i = 1.$$

A simplex is often referred to as the *convex hull* of $\{v_0, v_1, ..., v_n\}$. A *face* of an arbitrary simplex σ is any simplex spanned by a subset of $\{v_0, v_1, ..., v_n\}$. With the knowledge of simplices, we can now extend this definition to introduce simplicial complexes and their corresponding underlying spaces.

Definition 2.0.2. A (geometric) **simplicial complex** $\mathscr{K}$ in $\mathbb{R}^N$ is defined as a set of simplices with the following two properties:

1. For any simplex $\sigma \in \mathscr{K}$, every face of σ is also in $\mathscr{K}$.

2. For any two simplices $\sigma, \tau \in \mathscr{K}$, their intersection $\sigma \cap \tau$ is a face of both σ and τ.

Definition 2.0.3. The **underlying space** of a simplicial complex $\mathscr{K}$ in $\mathbb{R}^N$, denoted $|\mathscr{K}|$, is the union of all simplices in $\mathscr{K}$.

Most know of a simplicial complex through the geometric definition, but the second axiom of this definition is difficult to check and not entirely necessary for simplicial homology. We find we only really need the first axiom, which is brings forth the definition of an abstract simplicial complex, which removes the associated geometry from the above definition and provides a purely combinatorial description.

Definition 2.0.4. We define an **abstract simplicial complex** as some family of sets $\mathscr{F}$ such that every non-empty subset $Y \subseteq X$, where $X \in \mathscr{F}$, implies $Y \in \mathscr{F}$.

An abstract simplicial complex $\mathscr{F}$ isomorphic to the vertex scheme of a geometric simplicial complex $\mathscr{K}$ gives rise to the geometric realization $\mathscr{K}$ of $\mathscr{F}$. As geometric realizations of the same abstract simplicial complex are homeomorphic, there is a unique geometric realization for every abstract simplicial complex, up to isomorphism [5].

It is important to note that a one-dimensional simplicial complex is equivalent to a simple graph. The vertices correspond to the singleton sets within the simplicial complex, the edges to the two-element sets, and so on.

To illustrate the above definitions, let's pose a simple example. Let $X = \{1,2,3\}$ with the collection of subsets $S = \{\{1\},\{2\},\{3\},\{1,2\},\{2,3\},\{1,3\}\}$ represent an abstract simplicial complex. We can see that our set S contains the points $\{1\},\{2\}$ and $\{3\}$, which act as vertices, and the subsets $\{1,2\}$ and $\{2,3\}$ represent the edges between vertices. The triangle depicted in Figure 2.1 represents the geometric realization of the abstract simplicial complex (X,S). Notice that our triangle

in Figure 2.1 is not shaded in; adding the set $\{1,2,3\}$ to the collection of subsets S would grant us the interior. Through this triangle, we have constructed a geometric simplicial complex from an abstract simplicial complex. The converse can be performed in a similar manner.

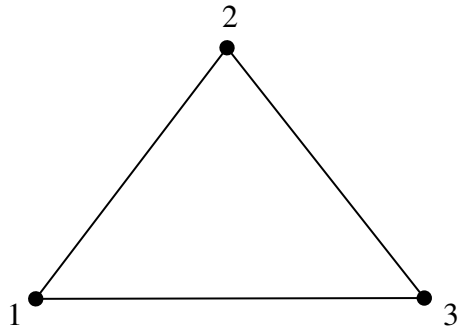

Figure 2.1: Geometric realization of (X,S)

A particular representation of an abstract simplicial complex is that of a nerve complex.

Definition 2.0.5. The **nerve** of some finite covering $\mathscr{U} = \{U_i\}_{i\in I}$ is the abstract simplicial complex $N(\mathscr{U})$ containing all finite subsets J (with subindices j) of the indexing set I such that the intersection of all U_j's is non-empty, i.e.

$$N(\mathscr{U}) = \{J \subseteq I \mid U_{i_1} \cap U_{i_2} \cap ... \cap U_{i_j} \neq \emptyset \text{ for } i \in I, j \in J\}.$$

The nerve complex can provide a combinatorial description of a relationship between data, found through intersection analysis.

An example of this can be seen in Figure 2.2 below. This figure displays a random point cloud with $\mathscr{U} = \{U_1, U_2, U_3\}$. The nerve of $\mathscr{U}$ is then $N(\mathscr{U}) = \{\{1,2\},\{1\},\{2\}\}$, which can also be described as an abstract one-simplex.

We will see the above preliminary concepts combine to form the topological version of the Mapper

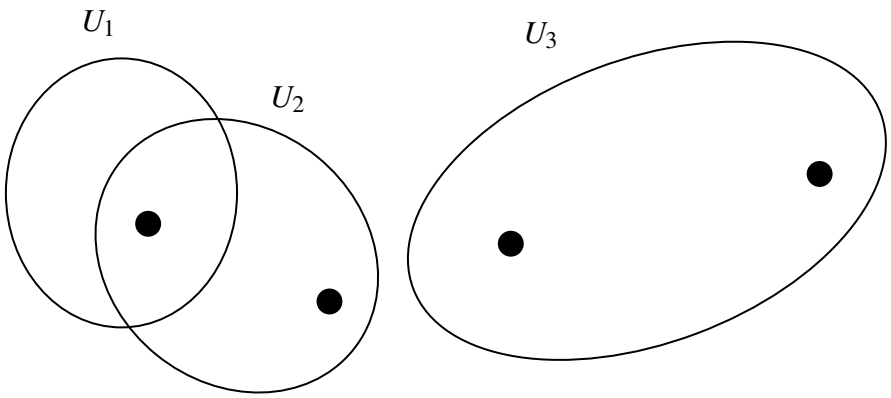

Figure 2.2: Simple example for nerve complex

algorithm, which will be described in the next chapter.

CHAPTER 3: MAPPER ALGORITHM

Within this chapter, we will dive into all things Mapper, explaining the concept behind its implementation. Its origin can be tied back to Gurjeet Singh, Facundo Mémoli, and Gunnar Carlsson in 2007 [1].

We will follow the flow of progression from Morse theory to Reeb graphs, then finishing the chapter with describing both the topological and statistical versions of Mapper. With this flow, an important observation is that as we advance through the topics, we are gaining more flexibility and freedom of input, but also losing the topological strength of our results. This becomes more clear with the choice of function in each section.

Morse Theory

One of the foundations of the Mapper algorithm lies in selecting a filter function f. We do have the freedom to select any function, but in selecting a *Morse function*, we are able to preserve topological features that might have otherwise been lost through the algorithm process. For more details on Morse theory, see [6].

Let's start with introducing this definition of *Morse function* and describe the relevant topological results associated.

Definition 3.0.1. A function $f : M \to \mathbb{R}$ is defined as a **Morse function** if at every critical point of f, the determinant of the Hessian of f is nonzero.

An example of a Morse function can be found in Figure 3.1, which illustrates a torus with the height function of $h : S^1 \times S^1 \to \mathbb{R}$, mapping the four critical points to their corresponding heights.

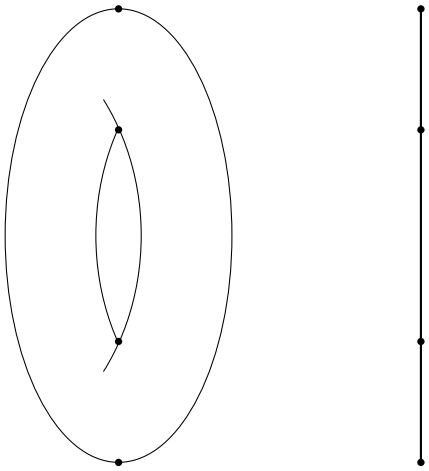

Figure 3.1: A torus with its height function

The height function is a common choice for a Morse function.

The aim of Mapper is to provide a sense of coordinatization; for Morse theory specifically, this concept can be described in the following **Morse Lemma**.

Lemma 3.0.1. Letting c_0 be a critical point of $f : M \to \mathbb{R}$ such that the determinant of the Hessian of f at c_0 is nonzero, we can choose a local coordinate system $(X_1, X_2, ..., X_m)$ about c_0 where f is of the form:

$$f = -X_1^2 - X_2^2 - ... - X_\lambda^2 + X_{\lambda+1}^2 + ... + X_m^2 + c,$$

where c_0 corresponds to the origin and c is a constant.

Referring to Figure 3.1, we can regard each critical point of the torus with a function f.

Morse theory is the strongest in terms of topology among the topics we will discuss today, and an important topological result in Morse theory is that of the **Morse Inequality**.

Theorem 3.0.1. For a closed n-manifold M and a Morse function $f : M \to \mathbb{R}$, we have the following inequality:

$$\kappa_\lambda \leq b_\lambda(M),$$

where κ_λ represents the number of critical points of index λ and $b_\lambda(M)$ the λ-dimensional Betti number of M. The **index** λ is defined as the number of negative eigenvalues of the Hessian.

In our transition from Morse theory to Reeb graphs, we again note that using a Morse function will provide us with stronger topological results.

Reeb Graph

We now introduce Reeb graphs, which drop the condition of our function being a Morse function and can now take any continuous function f as an input.

Definition 3.0.2. For some continuous map $f : X \to R$, where X is a topological space and R a specified target space, we define an equivalence relation $x \sim x'$ if $f(x) = f(x')$ and x, x' belong to the same connected component $f^{-1}(f(x))$. The **Reeb graph** is then defined as the quotient space $X/\sim$.

The Reeb graph consists of the points created by collapsing the connected components of each **level set**, which can be defined as $f^{-1}(a)$ for some real value a, given a continuous function f. Level sets are always evolving, which consequently evolves the connected components within them. From the Reeb graph, we can retrieve topological information related to the level sets formed by a given function.

The use of Morse functions in Reeb graph constructions provides a clearer analysis as it has distinct critical values, which is why using Morse functions is often considered a "good" choice for Reeb graph constructions. We see why in the following lemma [9].

Lemma 3.0.2. Given a connected, orientable 2-manifold, the Reeb graph of a Morse function preserves the number of holes.

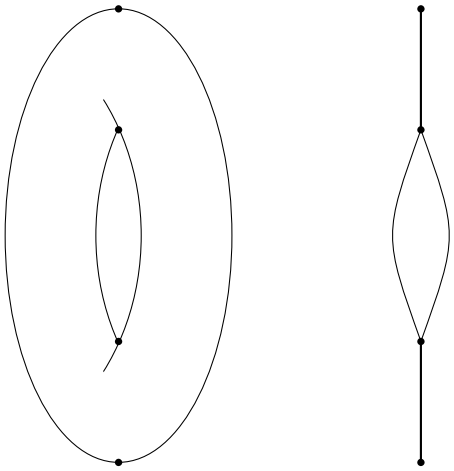

Figure 3.2: Reeb graph corresponding to a torus with the height function h

We continue to provide the same torus example in Figure 3.2, seeing the Reeb graph that corresponds to the torus with the height function. As we chose to use a Morse function, we see the above lemma reflected.

Reeb graphs are commonly seen as outputs of the topological version of the Mapper algorithm, which we will now detail.

Topological Version

The basis for the topological version of Mapper lies in the Reeb graph construction, which we have just outlined above. The connection lies in the fact that if the target space of our function is $\mathbb{R}$, which is a common choice, then we can recover the Reeb graph under a suitable cover.

The algorithm starts with a given continuous function $f : X \to Z$, where Z is equipped with a covering $\{U_i\}_{i \in I}$ for some indexing set I. As f is continuous, the sets $f^{-1}(U_i)$ form an open covering of X. The set of connected components of open sets $f^{-1}(U_i)$ is an open cover of X, which gives the nerve (abstract simplicial complex).

A map can be constructed from some space X to $N(\mathscr{U})$ with the addition of a partition of unity.

Definition 3.0.3. A **partition of unity** can be defined as a family of continuous functions such that every function is valued between 0 and 1, the sum of all functions is equal to 1, and the closure of all functions greater than 0 is within U_i, following the same notation as above.

We can then coordinatize the points within our simplex to k-tuples of real numbers through the concept of barycentric coordinatization.

Definition 3.0.4. Barycentric coordinatization refers to the bijective correspondence satisfying the conditions of $0 \leq n_i \leq 0$ and $\sum_{i=0}^{k} n_i = 1$, producing the arbitrary **barycentric coordinates** $(n_0, n_1, ..., n_k)$.

With these notions combined, we can produce a sort of coordinatization of our space X, taking values from the nerve complex. The nerve complex, as a specific notion of an abstract simplicial complex, is then the output of the Mapper algorithm.

This topological version of Mapper can be extended to point cloud data sets by replacing the connected components construction with the above concept of partitioning in the form of a clustering algorithm, therefore allowing the statistical implementation that is Mapper.

Statistical Version

The statistical version of the Mapper algorithm is used to convert a point cloud of data points into a two-dimensional graph, showing connections and attributes depending on the set parameters. The process the algorithm follows can be described as:

1. Choose some notion of distance between data points

2. Choose a filter function f (also commonly known as a lens)

3. Choose a cover for the co-domain, specifying the number of intervals and the percentage overlap for those intervals

4. Choose a clustering algorithm, specifying the number of bins when clustering

5. Produce a visual representation, where each vertex represents a cluster and two clusters connect if their intersection is non-empty

There are many possibilities with selecting a distance metric D. A commonly used metric is Euclidean distance; Hausdorff distance is another. Any notion of distance can be used.

As Mapper has many applications, seen in major fields such as medicine, biology, and finance, the user has much flexibility on setting random parameters; with that flexibility comes the necessary manipulation in order to achieve the desired results.

There are many commonly used filter functions, such as eccentricity, graph Laplacian, isometric mapping (isomap), principal component analysis (PCA), uniform manifold approximation and projection (UMAP), etc. To grant more understanding of a what a filter function is and its purpose in the Mapper algorithm, let's look at eccentricity as an example. The eccentricity filter function can be defined as, for $x \in X$,

$$f(x,p) = (\frac{1}{n}\sum_{y\in X} d(x,y)^p)^{\frac{1}{p}}.$$

For $p = \infty$, we can express the eccentricity filter function as

$$f(x,\infty) = \max_{y\in X}\ d(x,y).$$

The purpose of the eccentricity filter function is to find points with the largest distances away from

the center of the data, without actually specifying a center point. This filter can be useful in the case of outliers in the data, as eccentricity will be higher the farther away from the notion of a center. Different filter functions provide different functionalities, and that is why Mapper is so useful in the freedom a user has to choose a filter based on the circumstances.

Though many programming languages have a wide variety filter functions readily available for use within a TDA library, a filter function f can be constructed to cater to the need of the user. Many choose this path if they have a very specific goal in mind before implementing the algorithm to achieve the nerve graph for optimal analysis.

The choice of cover is just as important when running this algorithm, in the same way that altering the number of intervals and overlap will drastically effect the visualization. It is standard practice to choose to cover the data points already put through the desired filter function f that is very regular in distribution, such as taking a particular number of intervals n of equal length spaced evenly over the filter function values $f(X)$ given some data set X. As mentioned above concerning filter functions, a specific cover can also be constructed if desired.

The algorithm itself is not dependent on any specific clustering algorithm, though one of its main purposes does lie in cluster analysis. The default clustering method for the programming language R is that of single-linkage hierarchical clustering, and it is the method we will be utilizing in a later example, but Mapper will run with whatever clustering algorithm the user specifies, or with the default clustering algorithm of the program used to implement it. Other common examples of clustering algorithms are k-means clustering, density-based spatial clustering of applications with noise (or DBSCAN for short), and mean-shift clustering, just to name a few.

To demonstrate the process of the single-linkage hierarchical clustering algorithm, we pose the following example:

Let $X = \{a, b, c, d\}$ be plotted as in Figure 3.3, noting the distances between points.

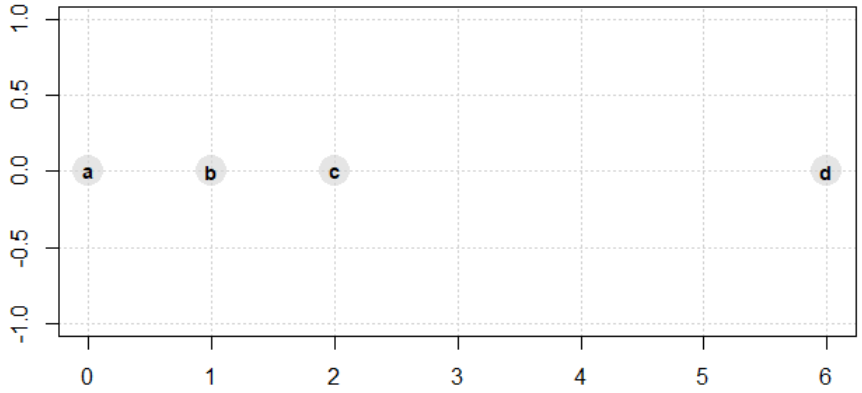

Figure 3.3: Visual representation of X

After X goes through the clustering algorithm, we have the following histogram and dendrogram representations (Figures 3.4 and 3.5). A **dendrogram** is a diagram typically associated with hierarchical clustering, demonstrating the relationships between clusters using tree-like structure. The number of bins when clustering is selected to be five, as demonstrated in the histogram, where 6 represents the maximum distance between points.

The single-linkage hierarchical clustering algorithm will separate d into its own cluster, which is shown by the red cut-off. The red cut-off line height of 2.5 comes from the halfway point of the first empty bin in the histogram representation. Depending on the number of bins chosen, this cut-off line will change, and consequently change the clusters themselves.

We will use this same clustering algorithm later in this thesis with an implementation of a data set in R.

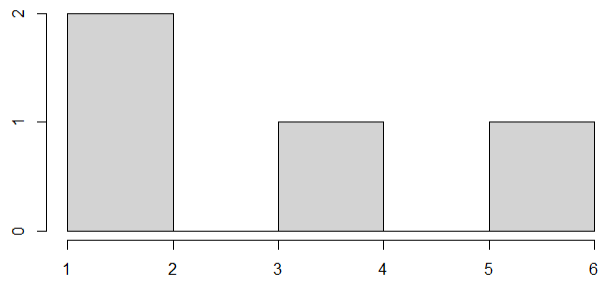

Figure 3.4: Histogram representation of X

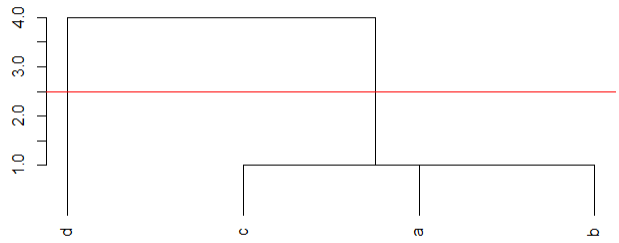

Figure 3.5: Dendrogram produced from clustering algorithm, with cut-off line

Note that the entire data set itself is not put through the clustering algorithm, only the preimage of the function values for each interval of the applied cover. The first few steps of Mapper must be followed before applying the clustering algorithm.

The last step in the Mapper process is to output the graph based on the given data and parameters chosen. This visualization can be separated into two types: nodes and edges. The **nodes** are represented by the vertices of the graph, i.e. the clusters created, and the **edges** are represented by the intersections between nodes. If there are no intersections between nodes, with every component

of each vertex being independent of each other, the graph will reflect this independence with all separated nodes. The makings of the vertices are determined by the earlier steps, but each vertex can also be separated into its own level set. Mapper will separate the nodes into different level sets depending on the parameters.

If it is not clear enough by now, the Mapper algorithm is merely a canvas to be painted with specifications depending on any situation. That is the key to applying this algorithm in so many different fields, as it allows such flexibility with input to grant a desired output for analysis. We see some examples of Mapper in action in the next chapter.

CHAPTER 4: APPLICATIONS OF MAPPER ALGORITHM

Now that we have a solid idea of what the Mapper algorithm consists of, we can briefly discuss a common example and fully detail an implementation using the programming language R. Let us start with the type 2 diabetes study.

Type 2 Diabetes Study

A well-known application of the Mapper algorithm can be seen in a type 2 diabetes study conducted by Li Li et al. in 2015 [3], which characterized three particular subgroups of individuals with type 2 diabetes based on the topology of a patient-patient network visualization. The study used the cosine distance metric, along with two filter functions: L-infinity centrality and principal metric singular value decomposition. This specific study, in its results, explored the physical attributes of the three different subgroups classified by the diseases each patient possessed through cluster analysis.

USAIR97

Topological data analysis is heavily used within the medical field, as demonstrated with the above diabetes application, but it can be consequently more difficult to interpret without a sufficient background in medicinal terminology and practices. To enhance understanding of the Mapper algorithm, we choose the simple data set of USAIR97 to implement and analyze, as it only contains one dimension and allows for easy understanding of statistical results. We will be using the programming language R to perform the Mapper algorithm and our subsequent analysis. The code used is linked here.

The data set chosen for this analysis is that of USAIR97 [11], consisting of 332 airports within the United States and the flights to and from each airport. In order to use the Mapper algorithm on this data set, we must first convert each of the 332 airports into a node and each flight to an edge, 2126 of them to be exact, specifically represented by a (weighted) network graph.

Weighted graphs are representations of network graphs with a specific weight between connections, **network graphs** simply being graphs with nodes and edges connecting them.

Each edge will represent a flight between airports with a corresponding weight, the weight being the normalized distances between airports per personal communication with A. Mrvar [11].

In [7], Mapper was applied to USAIR97 to analyze a specific grouping of airports. In their implementation of Mapper, they chose the filter function l_3, which uses eigenvectors of the third smallest eigenvalues of the unnormalized Laplacian of the weighted graph determined using the weights given in the data set, and a specified cover that they manipulate to achieve different visual results. With this specific choice of filter and cover, they analyze flight patterns of said specific grouping; we select the same grouping for replication purposes, though the choice of filter function and cover differ. Their reasoning behind their choice of filter and a specifically created cover with different interval lengths is not clearly justified, which is why we are following a more logical approach: our choice of filter is focused on an airport of interest (Bethel Airport in our case) and we choose a regularly spaced cover.

As every airport cannot have a direct flight to every other airport, we then compute the distance as the sum of the shortest direct flights. For example, if there was only a direct flight from airport A to airport B, but not airport C, we could add the the distances from airport A to airport B and then airport B to airport C to get a flight distance for the trip from airport A to airport C, assuming that combination gives us the shortest possible flight distance from airport A to airport C. This describes a practical implementation of *Dijkstra's algorithm*, finding the shortest path between

nodes of a weighted graph.

We will use both the normalized distances given to us in the data set and a simpler representation of these distances as the weights between vertices (i.e. airports), the latter being a distance of 1 between airports with direct flights and 0 between airports that do not have direct flights rather than a specific distance for any flight. This use of simplified distances will assist in flight pattern analysis. With there now being a distance between every airport to another, we can create a weighted graph to use within our Mapper algorithm and see what interesting results are available to us.

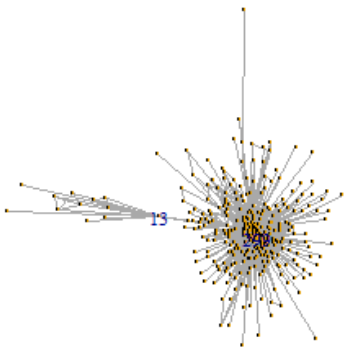

Figure 4.1: USAIR97 weighted graph
using given distances from data set

We see in Figures 4.1 and 4.2 the different weighted graphs based on our USAIR97 data set, the first using the given distances and the second using the simplified distances.

We will examine two separate implementations in R: the first with the distance object being the normalized distances (denoted as d) and the second with the distance object of the simplified

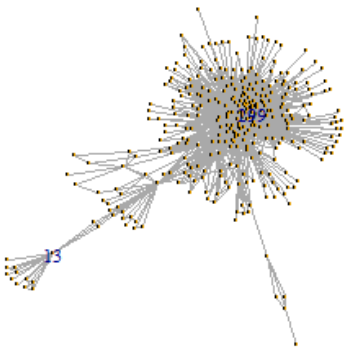

Figure 4.2: USAIR97 weighted graph
using simplified distances

distances (denoted d.s). In both cases, we select the filter d.s[,13], which focuses specifically on flight patterns originating from Bethel Airport.

We see in Figure 4.3 the histogram representation of our chosen filter specific to Bethel Airport. The histogram displays the frequency of the number of flights necessary to get from Bethel Airport to the rest.

The algorithm requires the specifying of the number of intervals, percent overlap, and number of bins when clustering, but these are easily manipulated to analyze different results. As discussed earlier, any manipulation of these variables can change the results drastically, but the values chosen for this implementation are considered very standard. With the selection of the number of intervals, we simultaneously are selecting a cover spaced out evenly over those intervals of equal length.

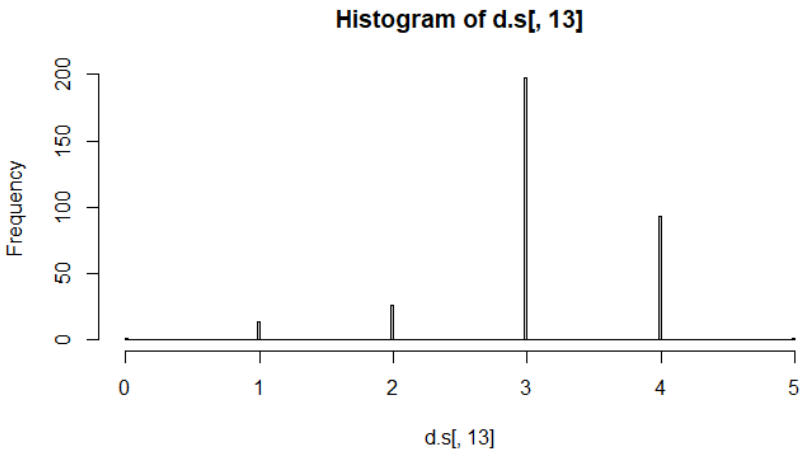

Figure 4.3: Histogram of filter function d.s
with a focus on Bethel Airport

Once input into Mapper, we can analyze the clusters formed. See Figure 4.4 for the output of our Mapper algorithm in the first case with the normalized distances as the distance metric and the filter d.s focusing on Bethel Airport.

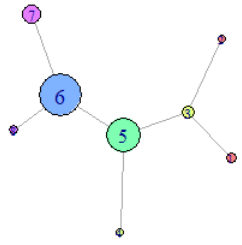

Figure 4.4: Mapper output using distance metric d and filter d.s[,13]

The colors within the graph represent the five different level sets, separating the airports into five groups based on how many flights necessary to get from Bethel Airport to a specific airport. Level set one is colored red, two is yellow, three is green, four is blue, and five is purple; this coloring remains consistent in both of our Mapper graphs.

We specifically focus on our analysis on vertices 1 through 4, for convenience. Vertices 1 and 2 contain all airports with a direct flight from Bethel Airport, and Bethel Airport itself. This is clear as vertices 1 and 2 are within the first level set, and as we are using the d.s filter, the first level set will contain all airports within a one flight distance from Bethel Airport. Using that same logic, we can conclude that vertices 3 and 4 contain all flights within a two flight distance.

Vertex 1 contains Bethel Airport and all airports within a direct flight distance, with the exception of Anchorage International Airport; vertex 2 contains Anchorage International Airport by itself. Anchorage International Airport acts as a "hub" to get from Bethel International to most other airports in the United States, but we see from the Mapper graph that this idea is not properly demonstrated.

With this implementation of the first case, we attempt to conduct edge analysis of the Mapper graph, yet find that using the distance metric as the normalized distances gives inconclusive results. This is clear as every trip to an airport that requires more than a direct flight from Bethel Airport must pass through Anchorage International Airport, but within Figure 4.4, Anchorage International Airport is contained in vertex 2 by itself and vertex 2 is not connected in a way that demonstrates this fact. For analyzing flight patterns, this first choice of distance metric is not the best.

However, with the second case of the d.s distance metric, we refer to Figure 4.5 for the Mapper output. In this graph, we have Anchorage International Airport contained in vertex 2 by itself once again, but it is now reflected that any flight from Bethel Airport (contained alone in vertex 1) to another airport more than a direct flight away must pass through Anchorage International Airport.

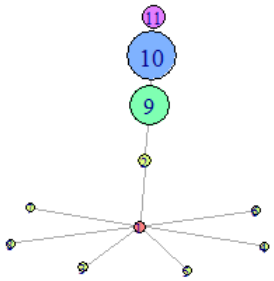

Figure 4.5: Mapper output using distance metric d.s and filter d.s[,13]

As this is a simple data set, this conclusion is not difficult to arrive upon with a focused eye on Bethel Airport and its possible departing flights, so this is not meant to be a claim of incredible findings through the use of the Mapper algorithm with this particular data set.

The choice for such a simple data set was strategic, in that logical conclusions on flight patterns can be gathered from just looking at the data set USAIR97 itself, as it is one-dimensional and does not require any background knowledge to understand the numerical data within (which we know to be the normalized distance between airports and the average person is aware of the concept of air travel).

Mapper is used today to draw conclusions that are not necessarily obvious when looking at a point cloud data set on its own, but for our purpose today, we verified the effectiveness of the algorithm and successfully arrived upon results that matched the logic behind the data set. Through the implementation of the Mapper algorithm and the corresponding cluster analysis of the graphical output, we can pick up on any patterns or relationships between subjects and determine significance

for ourselves depending on the parameters chosen for the algorithm and the background of the data set itself.

CHAPTER 5: CONCLUSION

The Mapper algorithm has many applications and its use is continuing to grow. We have discussed the topological background necessary to comprehend the algorithm's contents, the information on the algorithm itself, and a few examples of the algorithm in action, including an analysis of a data set in R using the TDAmapper package. Throughout this thesis, we have explored the concept of cluster analysis, after inputting selections of the many free parameters in the algorithm, to see that mathematicians and data analysts are able to visualize relationships from high-dimensional data sets in a practical way with the use of the Mapper algorithm.

This algorithm provides simply a framework based on choice of filter function, covers of co-domain, and clustering method, but its output is not always meaningful in practice. The difficulty surrounding this algorithm does not lie in its implementation, rather in its analysis afterwards. With the amount of freedom in the statistical version, a lot of effort may be necessary to find the necessary combination of parameters for the desired result.

A future improvement for the algorithm lies in the selection of the above choices: filter function, covers of the co-domain, and clustering method. The use of persistent homology may aid in this selection. If a feature stands to persist when moving through maps of increasing coarseness, it is more likely to be substantial in meaning and result. As the purpose behind topological data analysis lies in the extraction of qualitative properties, we will find improved results with this extraction in analyzing features that persist.

LIST OF REFERENCES

[1] Singh, G., Mémoli, F., and Carlsson, G. "Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition." *Eurographics Symposium on Point-Based Graphics*. The Eurographics Association, (2007).

[2] Carlsson, G. and Vejdemo-Johansson, M. *Topological Data Analysis with Applications*. Cambridge University Press, (2021).

[3] Li, L., Cheng, W., Glicksber, B.S., Gottesman, O., Tamler, R., Chen, R., Bottinger, E.P., and Dudley, J.T. "Identification of type 2 diabetes subgroups through topological data analysis of patient similarity." *Sci Transl Med*, (2015).

[4] Hatcher, A. *Algebraic Topology*. Cambridge University Press, (2002).

[5] Munkres, J.R. *Elements of Algebraic Topology*. CRC Press, (2018).

[6] Matsumoto, Y. *An Introduction to Morse Theory*. American Mathematical Society, (2002).

[7] Hajij, M., Rosen, P., and Wang, B. "Mapper on Graphs for Network Visualization." *arXiv preprint arXiv:1804.11242*, (2019).

[8] Carriere, M., Bertrand, M., and Oudot, S. "Statistical analysis and parameter selection for Mapper." *The Journal of Machine Learning Research*, Volume 19, Issue 1, (2018).

[9] Abdel-All, N. "Computer geometry and encoding the information on a manifold." *Journal of the Egyptian Mathematical Society*, Volume 19, Issues 1-2, (2011).

[10] Luxburg, U.V. "A Tutorial on Spectral Clustering." *Statistics and Computing*, Volume 17, Issue 4, (2007).

[11] Batagelj, V. and Mrvar, A. Pajek datasets. URL: http://vlado.fmf.uni-lj.si/pub/networks/data/, (2006).

[12] Reaven, G.M. and Miller, R.G. "An attempt to define the nature of chemical diabetes using a multidimensional analysis." *Diabetologia*, Volume 16, (1979).

[13] Ristovska, D. and Sekuloski, P. "Mapper algorithm and it's application." *Mathematical Modeling*, Volume 3, Issue 3, (2019).