Application of Topological Data Analysis to the Hand Written Digits Classification Problem

Felipe González Casabianca

Advisor: Andrés Ángel

University of the Andes
Science Faculty
Department of Mathematics
Bogotá, Colombia
November 2017

Contents

Chapter 1

Introduction

In recent years, new techniques have emerged that allow us to study data and information from their underlying geometric and topological structure [Lum+13]. Particularly, there is an algorithm called *Mapper* [SMC07], that lets us visualize data as a 1-simplex (a graph), determining clusters and their interactions. This algorithm has been successfully applied to real world problems, for example, it enabled the discovery of a new type of breast cancer by P.Y Lum et.al in [Lum+13]. These same authors also analyzed the voting tendencies for each member of the house of representatives in the USA along 22 years.

This work focuses in further understanding this algorithm using modern tools in its parameters to analyze the well known MNIST test set. As we will see in chapter 2, this algorithm depends heavily on the set of inputs, which in our specific case will be:

- **Data Representation:** We will represent the data as elements inside $\mathbb{RP}^\infty$, using the Multi-Scale Projective Coordinate scheme, developed by José A. Perea in [Per16] as we will see in chapter 4.
- **Distance Matrix:** In order to make *Mapper* more efficient, we will approximate the distance between records, using a Local Sensitive Hashing scheme as explained in chapter 5.
- **Filter Function:** We will discriminate records using the algorithm t-Distributed Stochastic Neighbor Embedding, for its performance in separating MNIST samples.

We will compare these results against more usual inputs, showing the obtained graphs and some numeric values that will help us asses the quality of the result. Our aim is also to show what topological data analysis has to offer in understanding the MNIST test, giving some interpretation to the obtained results from the specific context of digit classification.

1.1 MNIST data set

As was mentioned before, we will use the MNIST data set, a tagged collection of handwritten digits from 0 to 9. This data set was compiled by Yann LeCunn, Corinna Cortes and Christopher J.C Burges and consists of a training sample of 60,000 digits and a testing sample of 10,000 made publicly accessible at: `http://yann.lecun.com/exdb/mnist/`

Each element corresponds to a 28 by 28 pixel grid with values from 0 to 255 indicating the gray value. In turn, each image can be represented as a vector in $\mathbb{R}^{784}$. Here are some examples:

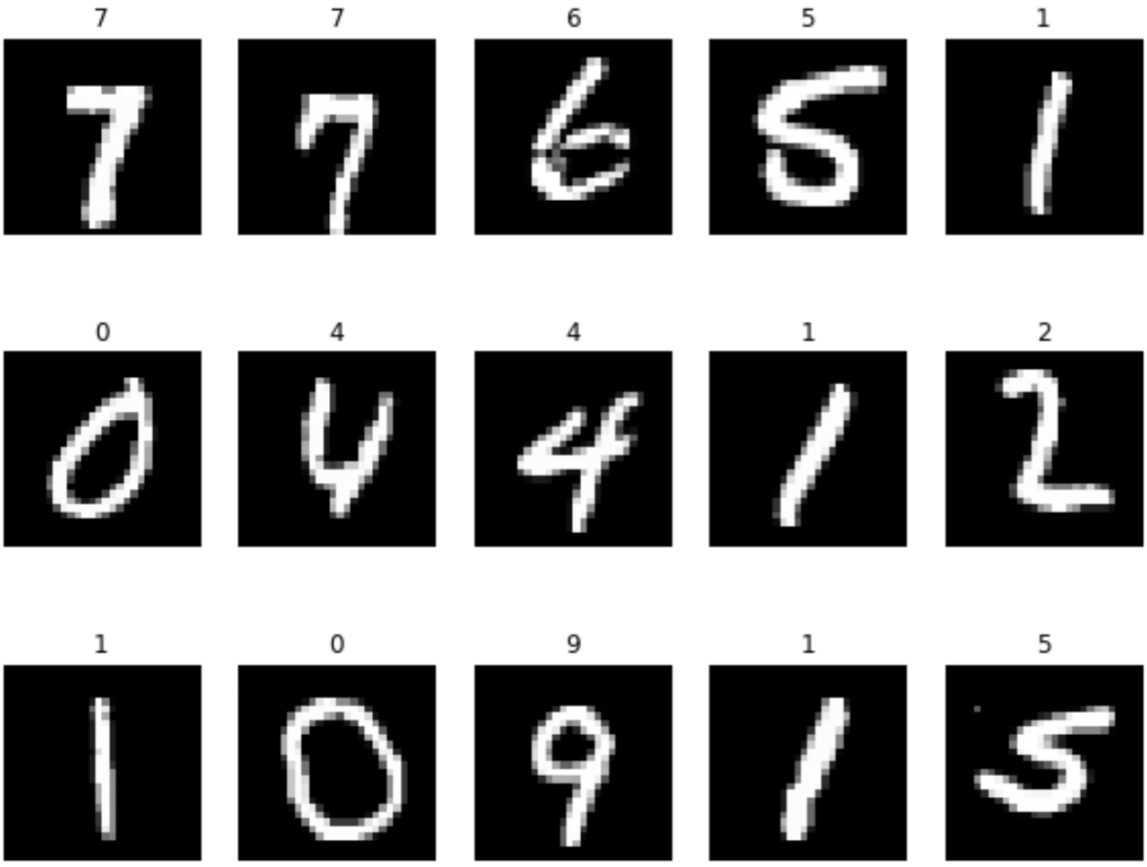


Figure 1.1: Some examples from the MNIST data set.

Chapter 2

Mapper

In this chapter we will define the *Mapper* algorithm, the main procedure of this work.

2.1 Definition and Explanation

The *Mapper* algorithm was first proposed by Gurjeet Singh, Facundo Mémoli and Gunnar Carlsson in [SMC07], and it's a method for extracting information of high dimensional sets organizing them as a simplicial complex (see definition 8) and verifying the result by visualizing it as a graph. So ultimately, *Mapper* receives a data set and outputs a corresponding simplicial complex.

Starting from some data records, this procedure requires the following elements:

- **Distance Matrix:** *Mapper* requires a scheme for calculating distances between records. Hence, the data can be in any format as long as we are able to compute a distance function among the records.
- **Filter Function:** One of the core ideas behind *Mapper* is to send the data set onto another space (usually of lower dimension), split this space into specific sections, and use their preimages to construct the simplicial complex. Thus, we must provide the filter function value for each one of the records in the data set.
- **Clustering Algorithm:** *Mapper* uses a clustering scheme, based on the distance between records, to detect clusters among the data. These clusters will serve as 0-dimensional simplexes for constructing the desired simplicial complex. No restriction is made on the provided clustering procedure, other than it must use a distance scheme as its only input.

We will illustrate the algorithm's scheme with a graphic example. In the following images, we apply the *Mapper* algorithm to a sample of S^1 with some

noise.

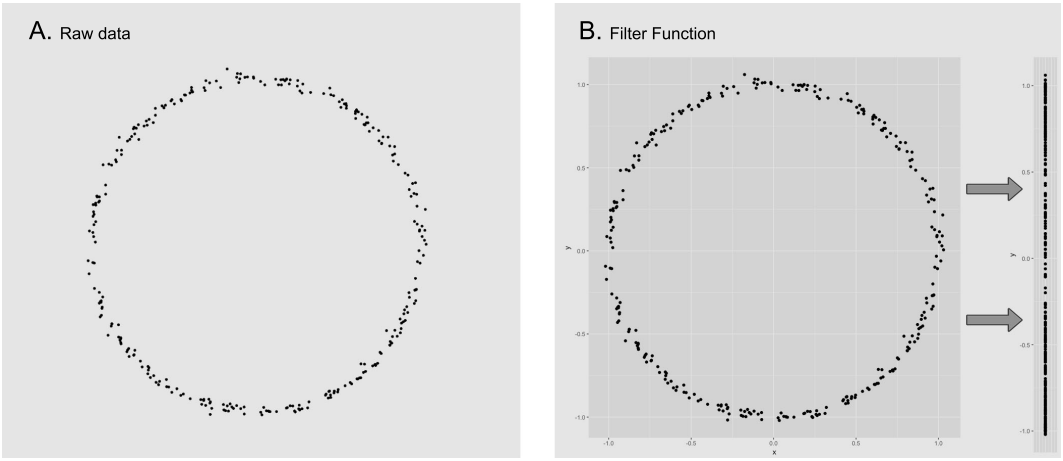


Figure 2.1: We start with a sample of 300 points from S^1 with some noise (slide A). For this example we will use the euclidean distance as the means to relate points in the data. We proceed with the filter function (slide B), that will be the y-coordinate of each point. So in turn, our filter space will be $[-1, 1]$ and notice how the filter function is not injective.

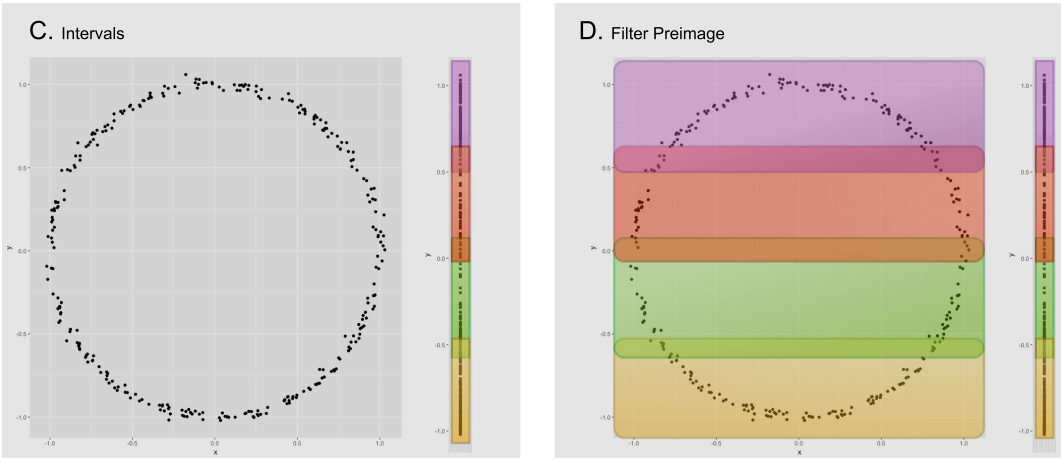


Figure 2.2: We then split the filter space $[-1, 1]$ into overlapping sections (slide C). Both the number of sections and overlapping percentage can be tweaked in the algorithm's configuration. In this example we use 4 sections with an overlapping of 15% highlighted with different colors. After this, we compute each of the sections preimages (Slide D).

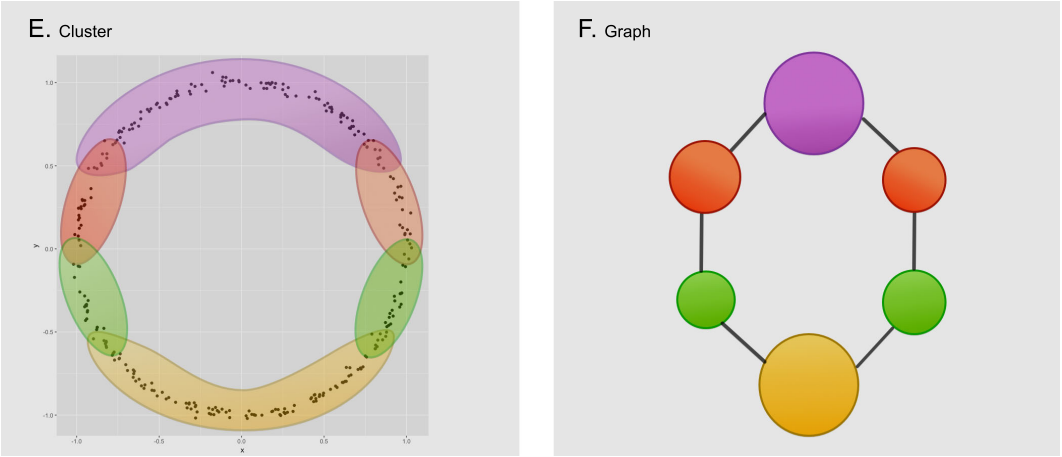


Figure 2.3: For each one of the colored sections of points we execute our cluster algorithm (slide E), obtaining isolated groups of points inside every one of the section's preimages which we will then use to construct the corresponding simplicial complex. The procedure ends when the simplicial complex is built (slide F). For this example the resulting simplicial complex has dimension one, therefore we obtain a graph representation of the circle. Notice how each cluster became a node of the resulting graph and edges are placed between clusters that have points in their intersection.

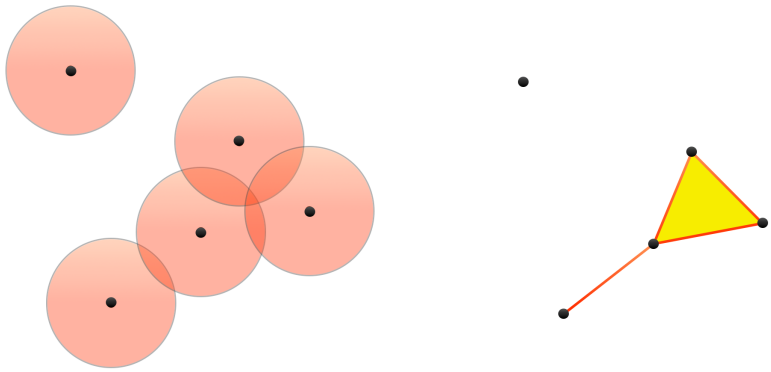
Theoretically, one can build a simplicial complex of higher dimension following this procedure. In our example, notice how a single point only belonged to a maximum of two clusters, but it possible to belonged to a more numerous intersection, creating a higher dimensional simplex. This procedure corresponds to the Čech Complex:

Definition 1. Given a metric space $(\mathbb{X}, \mathbf{d})$ and some sample $X \subset \mathbb{X}$, fix some $\varepsilon > 0$. The **Čech Complex** for radius ε is defined as

$$\check{C}_\varepsilon(X) = \left\{ \{x_1, \ldots, x_n\} \subset X \ : \ B_{\frac{\varepsilon}{2}}(x_0) \cap \cdots \cap B_{\frac{\varepsilon}{2}}(x_n) \neq \varnothing, \ n \in \mathbb{N} \right\}$$

where: $B_{\frac{\varepsilon}{2}}(x_0) = \left\{ x \in X \ : \ \mathbf{d}(x, x_0) < \frac{\varepsilon}{2} \right\}$

Here is an example of a two dimensional Čech Complex:



(a) 4 points and their corresponding ε open balls

(b) Respective Čhech complex

Figure 2.4: Construction of the 2 dimensional Čech complex from 4 points.

For this work, we implemented an extension of the R library: **TDAmapper**, developed by Paul Pearson, Daniel Muellner and Gurjeet Singh [PMS15]. Our extension, called **mapperKDLowRAM**, allows filter functions of any dimension and has the option to calculate parts of the distance matrix as they are needed, instead of computing it entirely from the start. This allows the procedure to run in machines with low RAM capacity. This extension can be downloaded from: `https://github.com/minigonche/tda_mapper`.

For complexity reasons, our library only checks for double intersections, ignoring the construction of higher dimensional simplexes. Therefore, for this project, the simplicial complex obtained as output will always be one dimensional (a graph), regardless of the intersections that might arise.

2.2 Examples

We include a toy example of the *Mapper* algorithm and finish with an application to real world data.

Definition 2. The k **Moment Curve** is the curve in $\mathbb{R}^{2k}$ defined as:

$$\xi_k(t) = \Big(cos(t), sin(t), cos(2t), sin(2t), \dots cos(kt), sin(kt)\Big)$$

Notice that the function:

$$\begin{aligned} g : S^1 &\longrightarrow \xi_k \\ t &\mapsto \xi_k(t) \end{aligned}$$

is a homeomorphism (inyective continuous function from a compact space to a Hausdorff space). Hence, ξ_k is in fact a "twisted" circle in higher dimension.

Our first example is a sample of 3000 points from the $\xi_2 \subset \mathbb{R}^4$, here are the four projections into $\mathbb{R}^3$ of the sampled points:

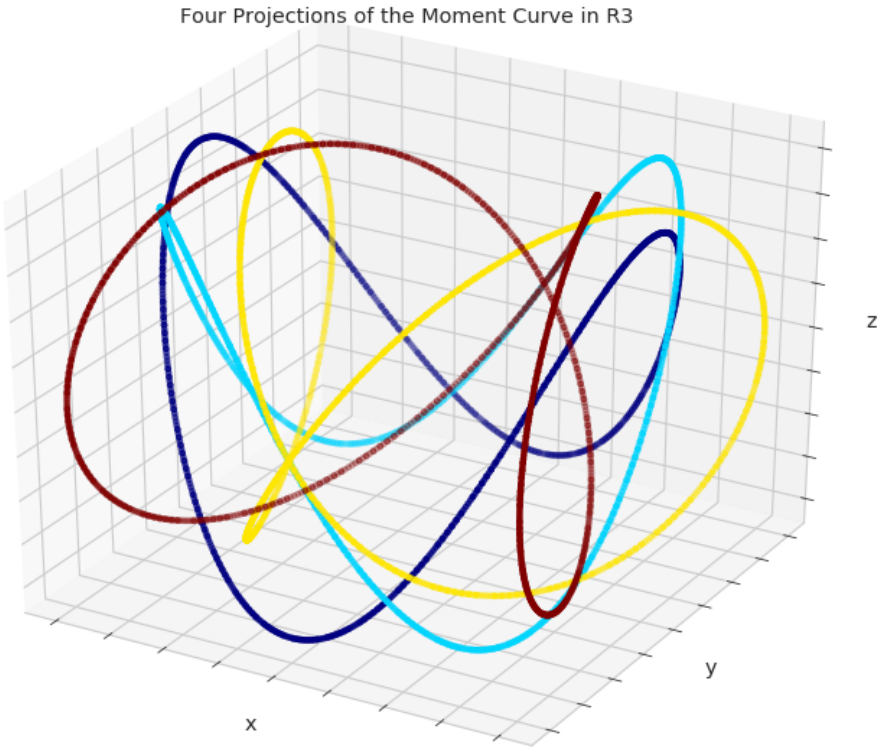


Figure 2.5: Projections of the second moment curve in $\mathbb{R}^3$. Notice that all of them are twisted circles in 3 dimensions.

Now, we apply *Mapper* algorithm to this sample of points, using the following configuration:

- **Distance Matrix:** Euclidean distance matrix.
- **Filter Function:** π_1. This means the filter values will be the first component of each point in the sample.
- **Clustering Algorithm:** Hierarchical Clustering [Ble08].

Here is the resulting graph:

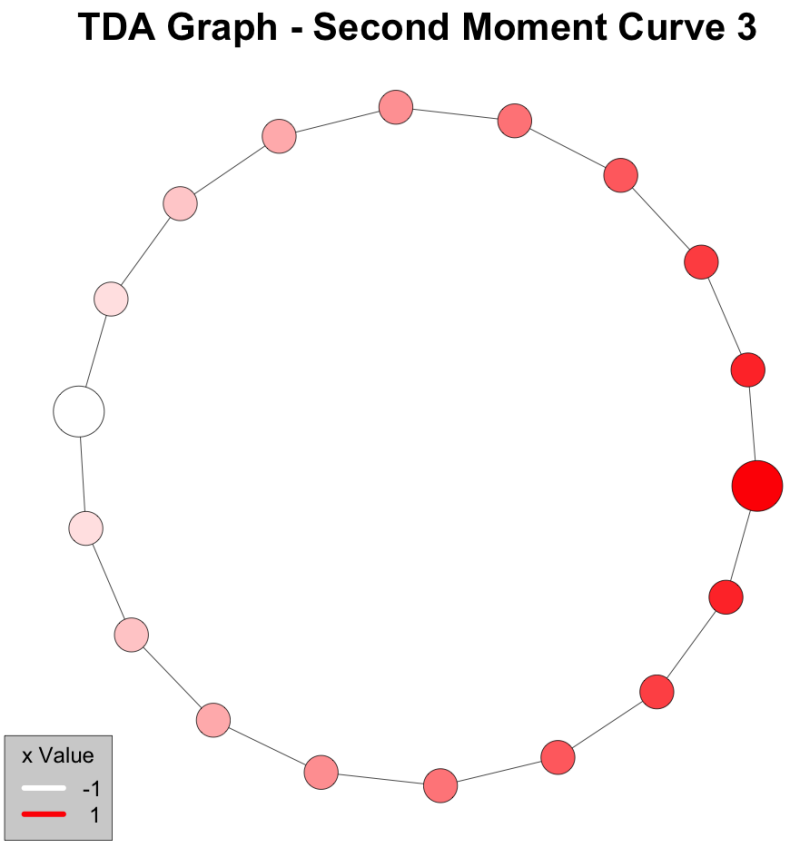


Figure 2.6: Resulting *Mapper* graph for the second moment curve filtered over its first component.

Obtaining a one dimensional simplicial complex of S^1, as was expected.

We finish with an example of the *Mapper* algorithm applied to real world data. The following results are taken from a submitted article in collaboration with Andrés Ángel, Alejandro Rivadeneira and Camilo Rivera: *Malaria intensity in Colombia by regions and populations*

We extracted from the Colombian Ministry of Health's public records, laboratory confirmed malaria (both *Plasmodium vivax* and *Plasmodium falciparum*) cases from 2007 to 2015. All of these cases are geocoded to a municipality level and include some social and demographic information of the patient. We wished to study how malaria behaves around the Colombian territory and its relation to social and demographic values. A Kulldorf's spatial scan statistic procedure

[Kul97], lets us study how malaria behaves locally around municipalities and gives us the locations where epidemic outbreaks of the disease occurred.

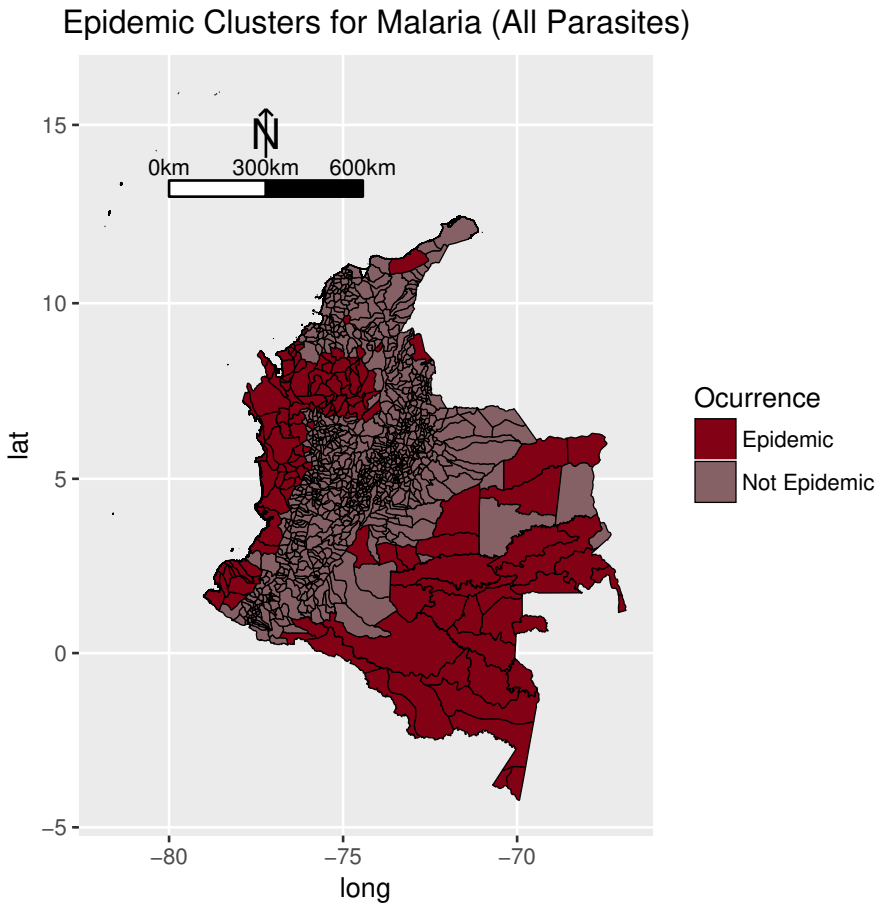


Figure 2.7: Epidemic municipalities identified using Kulldorf's spatial scan statistic.

We then turned to topological data analysis and *Mapper*, to study this outbreak's sincronicity. To do this we grouped the obtain records by weeks and executed for each one a Kulldorf spatial scan statistic, thus obtaining which municipalities showed a malaria outbreak in a given week. With this, we constructed a time series for each one of the 1,156 municipalities in the data set,

consisting of all weeks between 2007 and 2015. Each time series was composed of zeros and ones, indicating whether or not there was a malaria outbreak in the municipality for that certain week.

To these new set of 1,156 records of zeros and ones, we applied the *Mapper* algorithm, with the objective of detecting clusters of municipalities that showed sinchronic malaria outbreaks. We used the cosine "distance" (see equation 5.2.1) as the similarity notion among records, projecting over the first two principal components of the time series using PCA. The following figures show the obtained results.

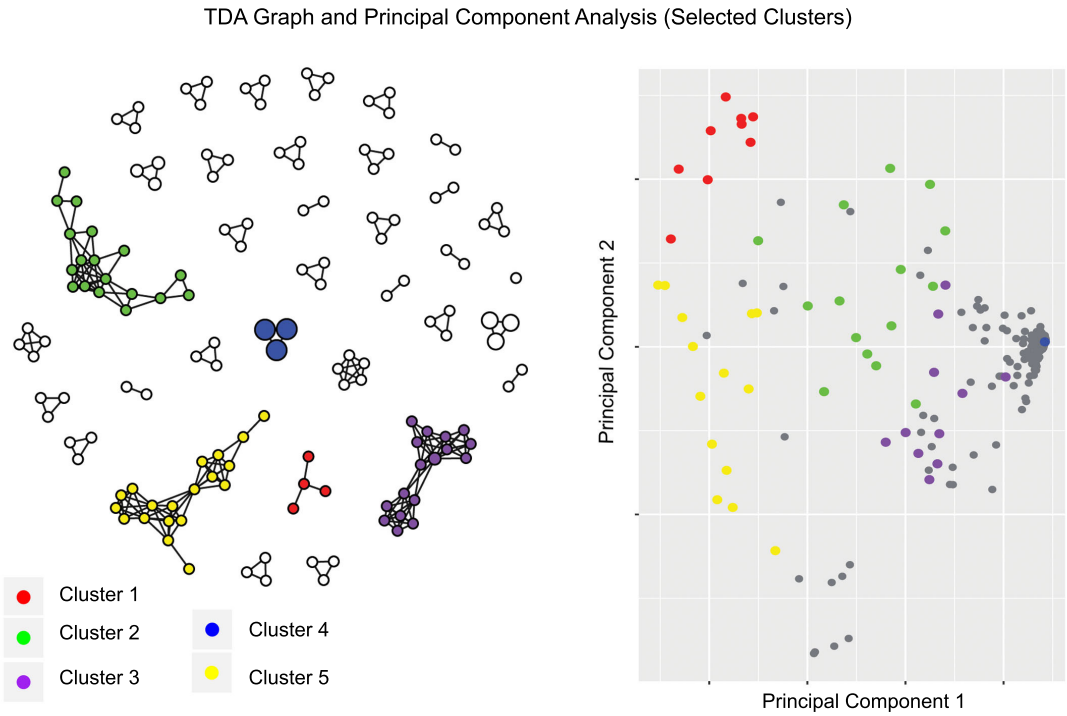


Figure 2.8: Resulting graph after executing the *Mapper* algorithm over the malaria time series and the corresponding filter space, where selected clusters where found by size and intensity. Here malaria intensity can be defined as the proportion of diagnosed infants (age 5 or less) among the set of cases. With malaria, if an infant has been diagnosed, it probably means his or her environment is saturated with the disease.

The cluster detection was done through an application developed using the Shiny framework from R, the same one we will use for this work's results. Screenshots of its interface can be seen in Figure 8.1 and 8.2. We then decomposed

each node in the cluster and mapped the corresponding municipalities over the Colombian territory.

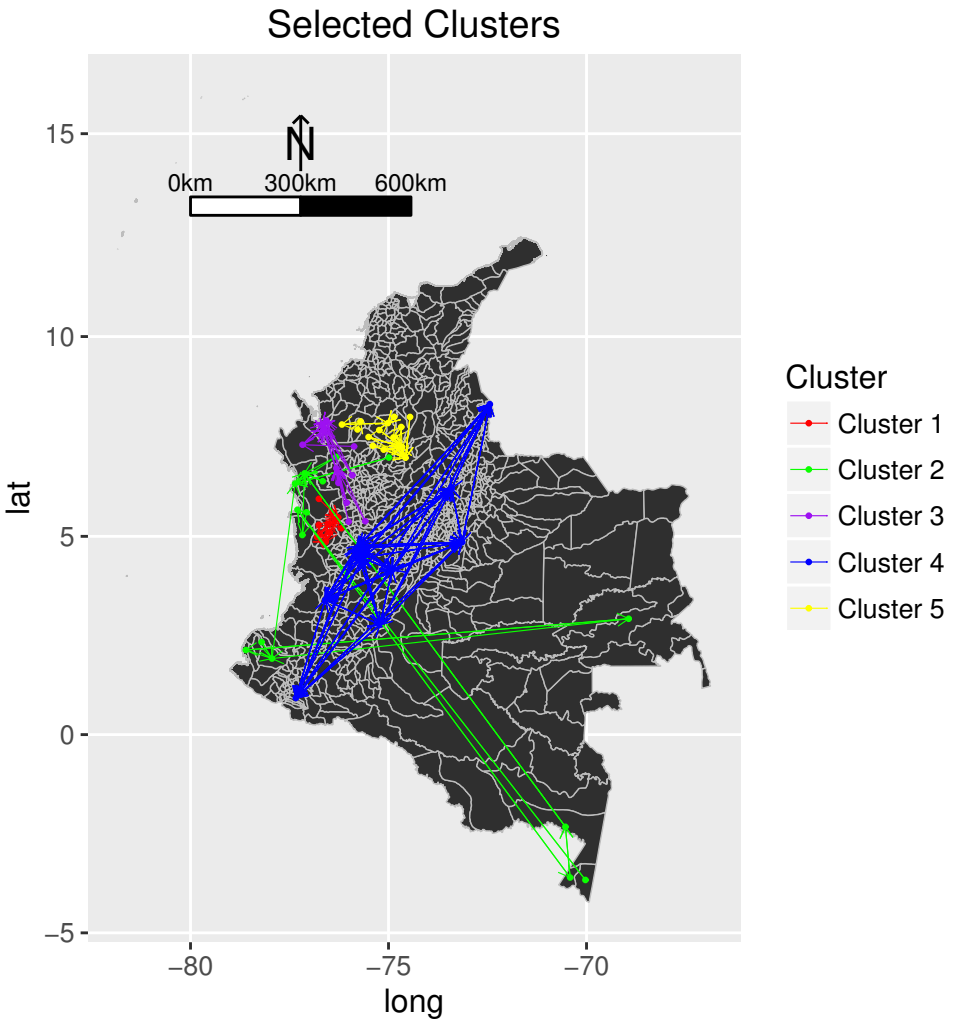


Figure 2.9: Selected clusters visualized over the Colombian Territory. Here, each colored dot corresponds to a municipality inside one of the nodes of the graph, for the corresponding cluster.

Notice how there are arrows between municipalities in Figure 2.9, they rep-

resent connections between municipalities (not to be confused with the connections between nodes in the previous graph). To build these connections, we defined a centrality for a municipality as follows and devised a connection scheme: given a certain municipality, its centrality corresponds to the number of nodes in the graph in which it appears. This means that if we remove a municipality with high centrality, it is very possible that the resulting graph will have less arches and in turn be disconnected. Now, the connection scheme is as follows:

- Only municipalities that appear together in a nodes in the graph can have a connection.
- No municipality will be connected to itself.
- Municipalities will be connected towards the municipalities in its node with the highest centrality. Note that there could be municipalities with multiple outgoing connections.

Mapper algorithm enabled us to detect sets of municipalities that have a similar malaria outbreak time series across several parts of the Colombian territory. We then proceeded with further inspection of these clusters.

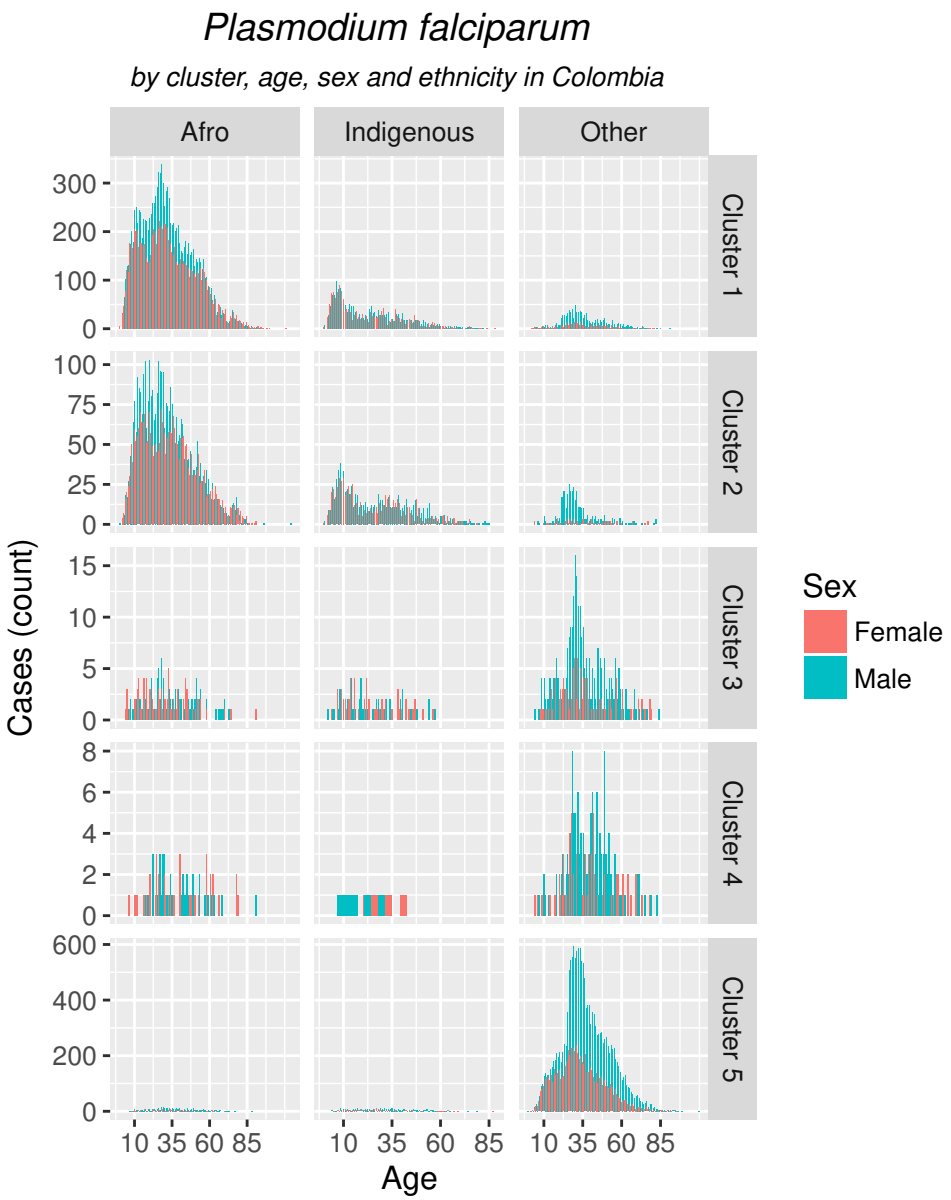


Figure 2.10: Malarial infection by *Plasmodium falciparum* by cluster, age, sex and ethnicity in Colombia. Histograms for the indigenous and Afrocolombian populations in clusters 1 and 2 suggest that these populations experience intense exposure to malarial infection, with the Afrocolombian population showing occupational hazard transmission. Histograms for the population with no ethnic denomination in clusters 1,2 and 5 suggest malarial infection is associated with occupational hazard, with cluster 5 being the strongest. Clusters 3 and 4 have too few cases to infer either transmission intensity or occupational hazard.

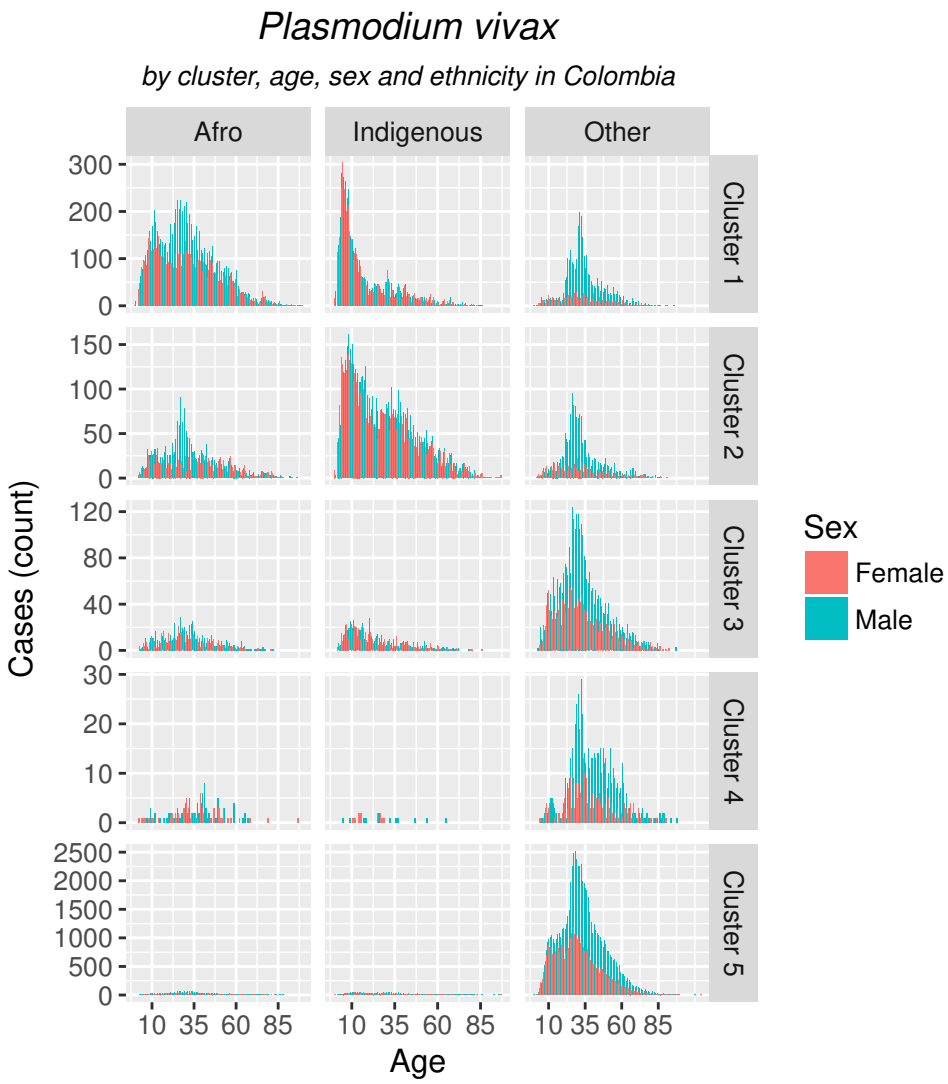


Figure 2.11: Malarial infection by *Plasmodium vivax* by cluster, age, sex and ethnicity in Colombia. Histograms for the indigenous population in clusters 1, 2, and 3 suggest intense exposure to malarial infection among these populations, with cluster 1 experiencing the most intense exposure. Histograms for the Afrocolombian populations of clusters 1, 2, and 3 suggest some degree of occupational hazard transmission. The population with no ethnic denomination experiences malarial infection as an occupational hazard in all clusters. It is interesting that clusters 4 and 5 show an existing occupational hazard without another population experiencing endemic malaria in the same region.

Most interestingly, our analysis identifies, without explicitly addressing it, locations where the proportion of cases of *Plasmodium falciparum* to *Plasmodium vivax* changed over the past few years Fig 2.12. This is one of the open

questions in tropical malaria, and our implementation has findings: most of the reduction in *Plasmodium vivax* cases was seen in Cluster 5, in the lower Cauca Basin, where an 8-fold reduction was observed. Most of the increase in *Plasmodium falciparum* happened in Cluster 1, in the Pacific Coast (although not explicitly on the coastal areas, but rather in the lower-altitude Andes on the Pacific), where an almost 3-fold increase was observed. Overall, cases of *Plasmodium vivax* reduced by almost half from the period 2008-2012 to the period 2013-2016 in municipalities that were not part of any cluster.

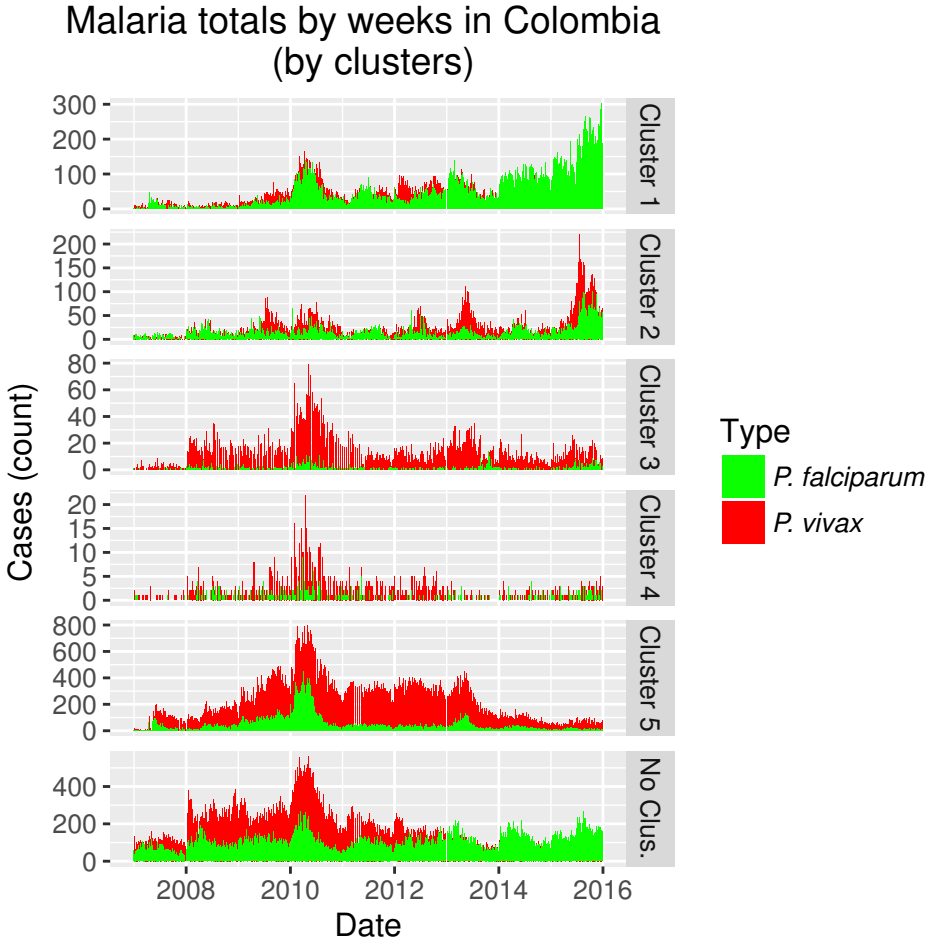


Figure 2.12: Total cases of malaria by weeks, between the years 2007 and 2015. Cluster 1 shows a change in the proportion of the type of malaria as time progresses and cluster 5 shows a significant reduction in the amount of cases as time progresses.

Chapter 3

Filter Function

3.1 Visualizing High Dimensional Data

As humans, we are obsessed with extracting information. Each year newer and finer methods are developed for the extraction of data (there are now temporary-tattoo like patches for monitoring heart rate, skin hydration and body temperature [Kim+16]). This means that data is not only increasing in size but in dimension and complexity, since we are now able to extract more specific features for every type of data. This leads to more insight and opportunity in data analysis, but also highlights the physical limitations of our visualization machines and our biological threshold to only grasp three dimensions simultaneously.

Which in turn, motivates the development of different approaches to faithfully portray high dimensional data structures into lower dimensional spaces, by means of projections or abstractions. We will mention a few of these techniques covered and reviewed by S. Liu et. al [Liu+17] and focuse on the technique we will use for our *Mapper* filter function.

- **Linear Projections**: as their name suggests, they are linear functions to lower dimensional spaces and are very popular. Probably the best known is Principal Component Analysis (PCA), which aims to construct an orthogonal linear transformation that maximizes the variance of the resulting coordinates. Another popular scheme is Linear discriminate analysis (LDA), which tries to construct a linear projection for labeled data, that maximizes class separation.
- **Non-linear Dimension Reduction**: these type of techniques embed the given set into a lower dimensional space, using non linearly functions, trying to preserve some characteristic the points share in their original space. One example of these methods is Locally Linear Embedding (LLE), a dimensionality reduction algorithm that aims to preserve distances locally,

that is, looking at a certain number of neighrest neighbors for each element. Another very popular method (and the one we will be focusing on) is t-Distributed Stochastic Neighbor Embedding (t-SNE), this method projects point sampled form a metric space into a lower dimensional $\mathbb{R}^n$, trying to preserve the distance between them.

- **Reeb Graph**: given a real valued function, the Reeb Graph aims to describe the level sets (pre-images of the given function) and their interaction. So, this method for visualizing high dimensional data, aims to locate the connected components among the data and how they interact. We mention this, because the *Mapper* algorithm, restricted to simplicial complex of dimension 1, can be used to correctly estimate the Reeb Graph of a set [CMO17].

3.2 t-Distributed Stochastic Neighbor Embedding

As mentioned before, this is a non-linear dimension reduction technique used for visualizing high dimensional data. It was developed by Laurens van der Maaten and Geoffrey Hinton in 2008 [MH08] based on Stochastic Neighbor Embedding (SNE) [HR03], and since then, it has been used in applications and has won several competitions [Hun12]. We will explain the t-SNE algorithm by first covering its initial version SNE and finish with a visualization of a section of the MNIST data base.

3.2.1 Stochastic Neighbor Embedding

This algorithm receives as an entry a distance matrix M between the high dimensional set $X = \{x_1, \dots, x_n\}$ and its objective is to map it into a set $Y = \{y_1, \dots, y_n\}$, embedded in a lower dimensional set. What SNE does first is convert distance entries of the matrix M to conditional probabilities. The idea is that the entry M_{ij} will no longer hold the value $d(x_i, x_j)$ but the value $p_{j|i}$. This value corresponds to the probability that x_j is picked as a neighbor of x_i if the its neighbors are picked accordingly to their probability density under a Gaussian distribution centered at x_i. This means that points close to x_i have higher chances of being picked whereas points far from this point will have almost zero probability of being picked. This value is calculated by the formula:

$$p_{j|i} = \frac{exp\big(-d(x_i, x_j)^2/2\sigma_i^2\big)}{\sum_{k \neq i} exp\big(-d(x_i, x_k)^2/2\sigma_i^2\big)}$$

Notice how the variance of the Gaussian depends on the current center, the authors determine this value by means of binary search to match a certain parameter (perplexity) given by the user. We refer the reader to the original paper: [HR03] to understand the details of how each of these σ_i are determined.

Also, since the actual base point for the SNE technique is a conditional probability matrix, this method can be applied to any data where similarities can be interpreted as conditional probabilities (like the probability that a user likes a movie after enjoying its prequel).

Now, the idea is that each $p_{i|j}$ will have a corresponding conditional probability $q_{i|j}$ analogous in its meaning but in the lower dimensional space, which means is computed using only the y_i. This conditional probability is in turn defined as:

$$q_{j|i} = \frac{exp\big(-d(y_i, y_j)^2\big)}{\sum_{k \neq i} exp\big(-d(y_i, y_k)^2\big)}$$

Notice that this formula is the same Gaussian conditional probability used before but with $\sigma = 1/\sqrt{2}$.[1] If the procedure correctly selects each y_i for each x_i, then $p_{i|j} = q_{i|j}$ for all $i, j \in \{1, \ldots, n\}$. By construction, it makes sense to define $p_{i|i} = q_{i|i} = 0$ for all $i \in \{1, \ldots, n\}$.

Now, the way the authors propose to progressively select the values of Y so that the difference between each pair of $(p_{i|j}, q_{i|j})$ is reduced, is by minimizing the sum of all Kullback-Leibler divergences among each of this pairs, producing a cost function that looks like this:

$$C = \sum_i \sum_j p_{i|j} \, \log\left(\frac{p_{i|j}}{q_{i|j}}\right)$$

So in summary, after receiving a set X of points, SNE constructs a set Y of the same size and in a lower dimension by reducing the cost function C by means of gradient descent.

3.2.2 t-SNE

The authors of this technique make the following changes to its original version:

- **Symmetrize the cost function**. They propose a new way to describe the similarities between points in the lower dimensional space, stated as follows

  $$q_{ji} = \frac{exp\big(-d(y_i, y_j)^2\big)}{\sum_{k \neq l} exp\big(-d(y_l, y_k)^2\big)}$$

  While preserving the notion of similarities by probabilities notice, that unlike its first version, these ones are now symmetric: $q_{ij} = q_{ji}$.

[1] By choosing a constant variance for the different $q_{i|j}$, we sacrifice the property that under this procedure the map renders a perfect model of the original sample, if the sample and target dimensions are the same.

The analogous way of defining:

$$p_{ji} = \frac{exp\big(-d(x_i, x_j)^2/2\sigma^2\big)}{\sum_{k \neq l} exp\big(-d(x_l, x_k)^2/2\sigma^2\big)}$$

Causes problems when x_i is an outlier, since the values p_{ij} are extremely small for all other x_j and will have almost no effect on the cost function. The authors in turn define p_{ij} as:

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}$$

allowing us to have that:

$$\sum_i p_{ij} > \frac{1}{2n}$$

Ensuring that all points (even the outliers) make a significant contribution to the cost function.

- **Use Student t-distribution instead of Gaussian in the lower dimensional space** . In SNE, we converted the distances between points into probabilities (in both spaces) using a Gaussian distributions. For this version, the authors redefine the $q_{i|j}$ using a Student t-distribution with one degree of freedom. The formula for these values is now:

$$q_{ij} = \frac{\big(1 + d(y_i, y_j)^2\big)^{-1}}{\sum_{l \neq k} \big(1 + d(y_l, y_k)^2\big)^{-1}}$$

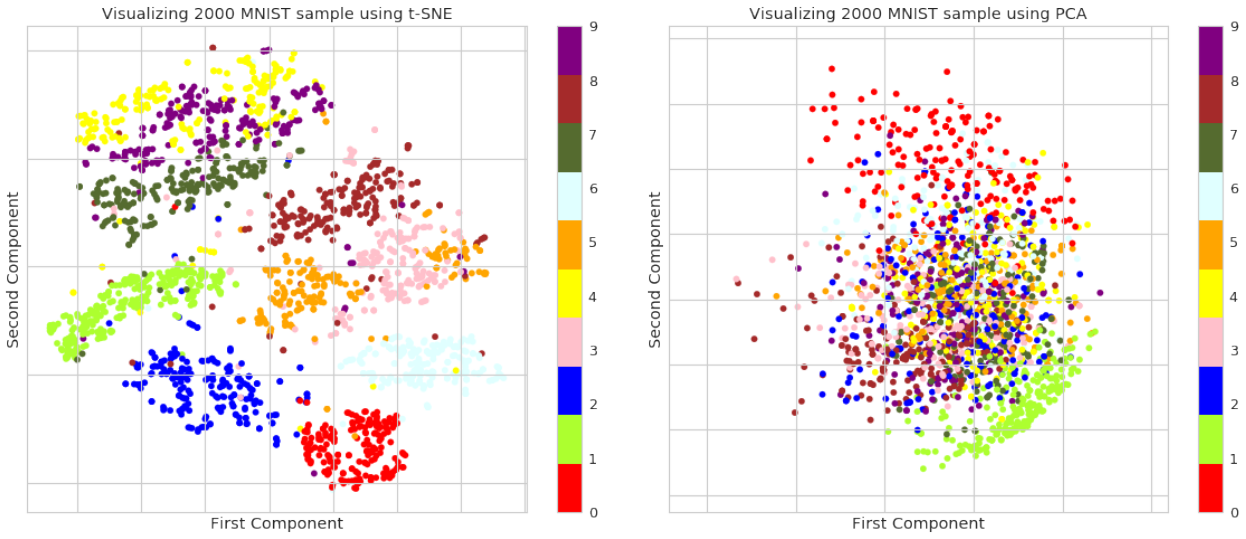
This selection alleviates the *crowding* problem, which happens when the map tries to faithfully represent closely situated points in high dimensional space by setting them to far apart, causing the points in the center of the map to crush together[2]. Having a crowded center prevents the creation of gaps among groups of points and thus the formation of clusters.

This changes and other tweaks, enabled this technique to win the *Merck Molecular Activity Challenge* in 2012.

[2]The optimization process in SNE can be viewed as if every point in the lower dimensional space exerts a force onto each point proportional to is similarity on the higher dimensional space. So, even though the points situated far from the center exert little force, all of them combined cause the center to collapse in a small crowded place

3.3 t-SNE applied to MINST

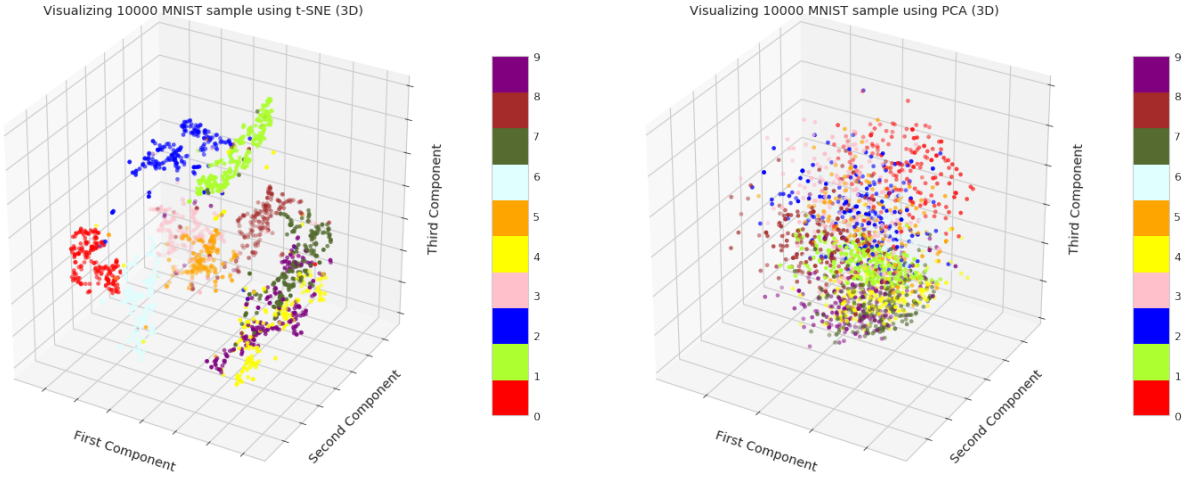
In this section we visualize a small sample of the MNIST test set, using the t-SNE technique and compare them with the corresponding PCA procedure. The following images were constructed using the Python script available at one of the author's site: `https://lvdmaaten.github.io/tsne/`. We simply made the few alterations needed to make the script run on Python3.



(a) t-SNE visualization

(b) PCA visualization

Figure 3.1: 2000 random sample of the MINST test database, visualized using the t-SNE and PCA techniques over a two dimensional space



(a) t-SNE visualization

(b) PCA visualization

Figure 3.2: 2000 random sample of the MINST test database, visualized using the t-SNE and PCA techniques over a three dimensional space

Chapter 4

Data Representation

In this chapter we will explain the data representation method used for the MNIST data frame. We will use a projection of sets that admit a distance onto $\mathbb{RP}^\infty$ developed by Jose A. Perea in [Per16]. We will give some theoretical background for his proposed function and will give a step by step construction of the procedure.

This chapter summarizes some of the content of [Per16], following most of its notation. Please refer to it for a more complete layout of the multi-scale projective coordinates.

4.1 Theoretical Background

Our main objective is to get from a finite sample of points X a function $f_X : X \longrightarrow \mathbb{RP}^\infty$, getting a representation of X in the infinite protective space. This function will come from relations between mathematical elements and from a crucial assumption: the given sample X comes from a metric ambient space $(\mathbb{X}, \mathbf{d})$, a space that will be used in the theoretical construction of the function.

4.1.1 Definitions

As mentioned before, the construction of f comes from the relations between three specific elements. Let us give the necessary definitions and notations for these elements.

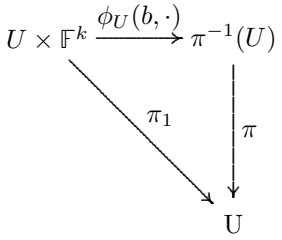
Definition 3. Let E and B be topological spaces and $\pi : E \longrightarrow B$ a surjective continuous map. Then the tripe $\zeta = (E, B, \pi)$ is a **Vector Bundle** of dimension $k \in \mathbb{N}$ over a field $\mathbb{F}$, if each fiber $\pi^{-1}(b)$ is an $\mathbb{F}$-vector space of dimension k and has the following property:

For every $b_0 \in B$, there exist an open neighborhood $b_0 \in U \subset B$ and an homeomorphism $\phi_U : U \times \mathbb{F}^k \longrightarrow \pi^{-1}(U)$ called a **local trivialization**,

satisfying:

- $\pi(\phi_U(b,v)) = b$ for every $(b,v) \in U \times \mathbb{F}^k$
- $\phi_U(b,\cdot) : \mathbb{F}^k \longrightarrow \pi^{-1}(b)$ is an isomorphism of F-vector spaces for each $b \in U$

This means that these functions are such that, for the $b_0 \in U$, the following diagram commutes:



We say that a triple ζ satisfying the previous condition is locally trivial.

Now, two vector bundles $\zeta = (E, B, \pi)$ and $\hat{\zeta} = (\hat{E}, \hat{B}, \hat{\pi})$ are isomorphic if there is a homeomorphism $T : E \longrightarrow \hat{E}$ such that: $\hat{\pi} \circ T = \pi$ and for each $b \in B$, the restriction $T|_{\pi^{-1}(b)}$ is an isomorphism.

The collection of isomorphism classes of $\mathbb{F}$-vector bundels of rank k of a B, will be denoted as $Vec^k_{\mathbb{F}}(B)$.

Definition 4. Let $\mathbb{F} = \mathbb{R}, \mathbb{C}$. The **Grassmannian of k-planes** in $\mathbb{F}^m$, for $k \in \mathbb{N}$, $m \in \mathbb{N} \cup \{\infty\}$ and $k \leq m$, is the manifold consisting of the collection of k-dimensional linear subspaces in $\mathbb{F}^m$, and is denoted as $Gr_k(\mathbb{F}^m)$.

Definition 5. Let A and B be topological spaces. Given two continuous functions $f_0, f_1 : A \longrightarrow B$, we say that f_0 and f_1 are **Homotopic** if there exists a continuous map $F : A \times I \longrightarrow B$ such that: $F(a,0) = f_0(a)$ and $F(a,1) = f_1(a)$, where $I = [0,1]$. If these functions are homotopic, we write $f_0 \backsimeq f_1$

The homotopy relation is in fact an equivalence relation, so we will denote with $[A; B]$ the set of classes of continuous maps from A to B under homotopy.

Definition 6. Given a topological space A and an element $a \in A$, fix some $n \in \mathbb{N}$ and some $x_0 \in S^n$. Then the **n-th Homotopy Group** of A (denoted $\pi_n(A)$) are the set of classes of continuous maps $\phi_a : S^n \longrightarrow A$ such that $\phi_a(x_0) = a$, under homotopy.[1]

Definition 7. Given $n \in N$ and G a group (if $n > 1$, G must be abelian), an **Eilenberg-MacLane space**, denoted $K(G,n)$, is a topological space with

[1] Here, each homotopy must also map x_0 to a (Pointed Homotopy)

$$\pi_m\big((K(G,n))\big) = \begin{cases} G & m = n \\ 0 & m \neq n \end{cases}$$

where 0 is the trivial group.

Some known examples of these spaces are:

- $K(\mathbb{Z}, 1) = S^1$
- $K(\mathbb{Z}, 2) = \mathbb{CP}^\infty$
- $K(\mathbb{Z}/2, 1) = \mathbb{RP}^\infty$

Our final element is simplicial cohomology, which can be defined, very broadly, as follows.

Definition 8. Given a non empty set S, an **Abstract Simplicial Complex** K with elements (vertices) in S, is a set of nonempty finite subsets that satisfies: if $\tau \subset \sigma \in K$, with $\tau \neq \emptyset$, then we have that $\tau \in K$.

An element $\sigma \in K$ is called a **n-simplex**, where $|\sigma| = n + 1$ with n the number of vertices in σ. Now, if we let $K^{(n)}$ denote the set of n-simplexes of K and if G is an abelian group, we have that the set of functions $\phi : K^{(n)} \longrightarrow G$ which are only non zero in finitely many elements, are an abelian group which we will be denoted $C^n(K; G)$.

There is an important function related to these sets called the n-coboundary $\delta^n : C^n(K; G) \longrightarrow C^{n+1}(K; G)$, where given an element $\phi \in C^n(K; G)$, we have that $\delta^n(\phi)$ belongs to $C^{n+1}(K; G)$ and operates on a simplex $\sigma = \{s_0, \ldots, s_{n+1}\}$ like this

$$\delta^n(\phi)(\sigma) = \sum_{i=0}^{n+1} (-1)^i \phi(\sigma \diagdown \{s_i\})$$

δ^n does not only define an homomorphism between $C^n(K; G)$ and $C^{n+1}(K; G)$, but we have that $\delta^{n+1} \circ \delta^n = 0$ for all $n \in \mathbb{N}$. This very important quality of δ implies for $Z^n(K; G) = ker(\delta^n)$ and $B^n(K; G) = img(\delta^{n-1})$ that:

$$B^n(K; G) \subseteq Z^n(K; G) \subseteq C^n(K; G)$$

Therefore, we can define the **n-th Simplicial Cohomology Group** of K, with coefficients in G as the quotient:

$$H^n(K; G) = Z^n(K; G)/B^n(K; G)$$

if G is a field, we have that $H^n(K, G)$ is a vector space.

4.1.2 Elements and their Interactions

We con now mention the three interacting elements that give us the foundation for our desired function $f_X : X \longrightarrow \mathbb{RP}^\infty$, they are:

$$Vect^1_{\mathbb{R}}(\mathbb{X}) \qquad [\mathbb{X}; \mathbb{RP}^\infty] \qquad H^1(\mathbb{X}; \mathbb{Z}/2)$$

Now, from [May99] if $\mathbb{X}$ is a CW-complex and G an abelian group, we have the group isomorphism:

$$H^n(\mathbb{X}; G) \cong [\mathbb{X}, K(G, n)]$$

So for our particular case, we have that:

$$H^1(\mathbb{X}; \mathbb{Z}/2) \cong [\mathbb{X}, K(\mathbb{Z}/2, 1)] \implies H^1(\mathbb{X}; \mathbb{Z}/2) \cong [\mathbb{X}; \mathbb{RP}^\infty] \tag{4.1}$$

On the other hand if $\mathbb{X}$ is paracompact, from [Spa+75], we have a bijection between:

$$Vect^k_{\mathbb{F}}(\mathbb{X}) \longleftrightarrow [\mathbb{X}; Gr_k(\mathbb{F}^\infty)]$$

where $k \in \mathbb{N}$ and $\mathbb{F} = \mathbb{C}, \mathbb{R}$. So again, for our particular case:

$$Vect^1_{\mathbb{R}}(\mathbb{X}) \longleftrightarrow [\mathbb{X}; Gr_1(\mathbb{R}^\infty)] \implies Vect^1_{\mathbb{R}}(\mathbb{X}) \longleftrightarrow [\mathbb{X}; \mathbb{RP}^\infty] \tag{4.2}$$

Now, for our final and less straightforward relation between $Vect^1_{\mathbb{R}}(\mathbb{X})$ and $H^1(\mathbb{X}, \mathbb{Z}/2)$ we need a couple of more things. Let $\mathcal{U}$ be an open cover of $\mathbb{X}$:

Definition 9. Given a collection of maps $\phi = \{\phi_{UV}\}_{U,V \in \mathcal{U}}$, where:

$$\phi_{UV} : U \cap V \longrightarrow GL_k(\mathbb{F})$$

where $GL_k(\mathbb{F})$ is the general linear group, we say they satisfy the **Cocycle Condition** if:

$$\phi_{UW}(x) = \phi_{UV}(x) \circ \phi_{VW}(x) \text{ for every } x \in U \cap V \cap W$$

So, given an $\mathbb{F}-$vector bundle $(A, \mathbb{X}, \pi)$ of rank k over $\mathbb{X}$ and two local trivializations ϕ_U, ϕ_V with $U \cap V \neq \emptyset$, then for $b \in U \cap V$ we have that the composition:

$$\mathbb{F}^k \xrightarrow{\phi_V(b,\cdot)} \pi^{-1}(x) \xrightarrow{\phi_U(b,\cdot)^{-1}} \mathbb{F}^k$$

gives us an element: $\phi_{UV} : U \cap V \longrightarrow GL_k(\mathbb{F})$ and a set $\{\phi_{UV}\}$ that satisfies the cocycle condition.

This procedure lets us get from a vector bundle, to a set of maps satisfying the cocycle condition. But what is crucial from this construction is that it can be reversed as follows:

Given an open cover $\mathcal{U} = \{U_r\}_{r\in\Delta}$ of $\mathbb{X}$ and a set of maps

$$w = \{w_{rt} : U_r \cap V_r \longrightarrow GL_k(\mathbb{F})\}$$

satisfying the cocycle condition, define the quotient space:

$$E(w) = \left(\bigcup_r \Big(U_r \times \{r\} \times \mathbb{F}^k\Big)\right)/\sim$$

where $(x, r, v) \sim (x, t, w_{rt}^{-1}(v))$ for $x \in U_r \cap U_t$ and define the map:

$$\pi(w) : E(w) \longrightarrow \mathbb{X}$$

as the projection onto the first coordinate. Then we have that the triple: $\Big(E(w), \mathbb{X}, \pi(w)\Big)$ is an $\mathbb{F}-$vector bundle of rank k over $\mathbb{X}$.

Recall that we are trying to establish a relation between $Vect^1_{\mathbb{R}}(\mathbb{X})$ and $H^1(\mathbb{X}, \mathbb{Z}/2)$ and with the previous procedure in mind, we continue defining the missing elements that will fill the gap.

Definition 10. A **Presheaf** F of abelian groups over some topological space is a contravariant functor between the category of open sets with inclusions and category of abelian groups. We have an abelian group $F(U)$ for each open set U and a group homomorphisms:

$$F_U^V : F(U) \longrightarrow F(V)$$

for each pair of open sets $V \subset U$, that satisfy:

1. F_U^U is the identity hommomorphism
2. $F_U^W = F_U^V \circ F_V^W$ for every $W \subset V \subset U$

If a presheaf satisfies the gluing and locality axioms:

- Given a cover $\{U_j\}_{j\in\Delta}$ of an open set U and elements: $\{s_j \in U_j\}$ that satisfy:

  $$F^{U_j}_{U_j\cap U_i}(s_j) = F^{U_i}_{U_j\cap U_i}(s_i)$$

  for every non empty intersection $U_i \cap U_j \neq \emptyset$, then there exists a unique element $s \in F(U)$ such that:

  $$F^U_{U_j}(s) = s_j$$

then it's called a **Sheaf**.

With presheafs, we are able to define the Čech Cohomology:

Definition 11. Let $n \in \mathbb{N}$, $\mathcal{U} = \{U_j\}$ an open cover of $\mathbb{X}$ and F a preasheaf over $\mathbb{X}$. The group of Čech n-cochains is defined as:

$$\check{C}^n(\mathcal{U}; F) = \prod_{(j_0,\dots j_n)} F(U_{j_0} \cap \cdots \cap U_{j_n})$$

As with cohomology, we need a coboundary homomorphism between cochains. For this, we let $(j_0, \dots, \hat{j_i}, \dots j_n)$ be the tuple of length n obtained when removing the i-th coordinate, write $U_{(j_0,\dots j_n)} = U_{j_0} \cap \cdots \cap U_{j_n}$ and denote η_{j_r} the homomorphism form the presheaf, obtained from the inclusion $U_{(j_0,\dots j_n)} \subset U_{(j_0,\dots,\hat{j_i},\dots j_n)}$. Then if $\left\{f_{j_0,\dots j_n}\right\}_{(j_0,\dots j_n)}$ denotes an element of $\check{C}^n(\mathcal{U}; F)$, with $f_{j_0,\dots j_n} \in F(U_{(j_0,\dots j_n)})$, then we have that the function:

$$\delta^n : \check{C}^n(\mathcal{U}; F) \longrightarrow \check{C}^{n+1}(\mathcal{U}; F)$$

that sends $\left\{f_{j_0,\dots j_n}\right\}_{(j_0,\dots j_n)} \mapsto \left\{g_{k_0,\dots k_{n+1}}\right\}_{(k_0,\dots k_{n+1})}$, with:

$$g_{k_0,\dots k_{n+1}} = \eta_{k_0}\left(f_{\hat{k_0},\dots k_{n+1}}\right)^{(-1)^0} \circ \cdots \circ \eta_{k_r}\left(f_{k_0,\dots,\hat{k_r},\dots k_{n+1}}\right)^{(-1)^r} \circ \cdots \circ \eta_{k_{n+1}}\left(f_{k_0,\dots \hat{k_{n+1}}}\right)^{(-1)^{n+1}}$$

(with $\circ$ is the operation of the respective group and x^{-1} denotes the inverse element of x)

serves as the desired coboundary, satisfying:

$$\delta^{n+1} \circ \delta^n = 0$$

Letting us define the **Čech Cohomology Group** of F with respect to $\mathcal{U}$ as:

$$\check{H}^n(\mathcal{U}; F) = \check{Z}(\mathcal{U}; F)/\check{B}(\mathcal{U}; F)$$

with $\check{Z}(\mathcal{U}; F)$ the kernel of δ^n and $\check{B}(\mathcal{U}; F) \subset \check{Z}(\mathcal{U}; F)$ the image of δ^{n-1}.

Now, denote the sheaf of continuous $\mathbb{R}^\times$ valued functions on $\mathbb{X}$ with $\mathscr{C}_{\mathbb{R}^\times}$[2] and consider the representative element w of the a class: $[w] \in \check{H}^1(\mathcal{U}; \mathscr{C}_{\mathbb{R}^\times})$. Since:

$$w \in \prod_{(r,t)} \mathscr{C}_{\mathbb{R}^\times}(U_r \cap U_t)$$

this element gives us an indexed family of functions $\{w_{rt} : U_r \cap U_t \longrightarrow \mathbb{R}^\times\}$, one for each pair of open sets $U_r, U_t \in \mathcal{U}$.

[2]Here the operation between functions is the multiplication and $\mathscr{C}_{\mathbb{R}^\times}{}_V^U$ maps each $f : U \longrightarrow \mathbb{R}^\times$ to $f|_V$

Furthermore, since $w \in \check{Z}^1(\mathcal{U}; \mathscr{C}_{\mathbb{R}^\times})$, we have that, for every k_0, k_1, k_2 :

$$\begin{aligned} 0 &= \delta^{n+1}\Big(\big\{w_{j_0,j_1}\big\}_{(j_0,j_1)}\Big)_{(k_0,k_1,k_2)} \\ &= \eta_{k_0}(w_{k_1,k_2})^{(-1)^0} \circ \eta_{k_1}(w_{k_0,k_2})^{(-1)^1} \circ \eta_{k_2}(w_{k_0,k_1})^{(-1)^2} \\ &= w_{k_1,k_2}|_{U_{k_0,k_1,k_2}} \circ w_{k_0,k_2}^{-1}|_{U_{k_0,k_1,k_2}} \circ w_{k_0,k_1}|_{U_{k_0,k_1,k_2}} \\ &= w_{k_0,k_2}^{-1}|_{U_{k_0,k_1,k_2}} \circ w_{k_0,k_1}|_{U_{k_0,k_1,k_2}} \circ w_{k_1,k_2}|_{U_{k_0,k_1,k_2}} \end{aligned}$$

then, restricted to U_{k_0,k_1,k_2}, we have that:

$$w_{k_0,k_2} = w_{k_0,k_1} \circ w_{k_1,k_2} \tag{4.3}$$

Hence, we have that w satisfies the cocycle condition.

Since $GL_1(\mathbb{R}) \cong \mathbb{R}^\times$, the cocycle condition together with the initial procedure of constructing vector bundles from maps, lets us close our objective gap from $\check{H}^1(\mathcal{U}; \mathscr{C}_{\mathbb{R}^\times})$ to $Vect^1_{\mathbb{R}}(\mathbb{X})$. So we only need to relate the Čech cohomology with simplicial cohomology, which can be done with the nerve of an open cover and the nerve theorem:

Definition 12. With $\mathcal{U}$ an nonempty open cover of $\mathbb{X}$, the **Nerve** of $\mathcal{U}$, denoted $\mathcal{N}(\mathcal{U})$, is an abstract simplicial complex, where each one of the open sets in $\mathcal{U}$ corresponds to a vertex and set of vertices $\{v_0, \dots v_n\}$ compose a simplex if:

$$U_{v_0} \cap \cdots \cap U_{v_n} \neq \emptyset$$

(notice from definition 1 from chapter 2 that: if $\mathcal{U}$ is the cover of open neighboorhoods of radii $\varepsilon > 0$ around all elements of X, then $\mathcal{N}(U) = \check{C}_\varepsilon(X)$)

Let U be an nonempty open cover of $\mathbb{X}$ and let $\tau \in Z^1(\mathcal{N}U; \mathbb{Z}/2)$. Then τ is a function of arches (r,t) to $\mathbb{Z}/2$, so we can define $\phi^\tau = \{\phi^\tau_{rt}\}_{rt}$ as the collection of constant functions:

$$\begin{array}{rccc} \phi^\tau_{rt} & U_r \cap U_t & \longrightarrow & \mathbb{R}^\times \\ & b & \mapsto & (-1)^{\tau_{rt}} \end{array}$$

with τ_{rt} the image of the arch (r,t). Since each ϕ^τ_{rt} is constant, hence is continuous and we have that $\phi^\tau \in \check{C}(\mathcal{U}; \mathscr{C}_{\mathbb{R}^\times})$. Moreover, since τ is a cocycle we have that $\phi^\tau \in \check{Z}(\mathcal{U}; \mathscr{C}_{\mathbb{R}^\times})$ and the association:

$$\begin{array}{rccc} \Phi_{\mathcal{U}}: & H^1(\mathcal{N}U; \mathbb{Z}/2) & \longrightarrow & \check{H}^1(\mathcal{U}; \mathscr{C}_{\mathbb{R}^\times}) \\ & [\tau] & \mapsto & [\phi^\tau] \end{array} \tag{4.4}$$

is an homomorphism. Furthermore, if each U_r is connected then by proposition 4.3 from [Per16] we have that $\Phi_{\mathcal{U}}$ is also injective.

Theorem 1. *(Collorary 4G.3 [Hat02]) If $\mathcal{U}$ is an open cover of a paracompact space $\mathbb{X}$ such that every nonempty intersection of finitely many sets in $\mathcal{U}$ is contractible, then $\mathbb{X}$ is homotopy equivalent to the nerve $\mathcal{N}(\mathcal{U})$*

By the homotopy invariance of cohomology, this means that under the previous conditions and for G an abelian group:

$$H^n(\mathcal{N}(\mathcal{U}); G) = H^n(\mathbb{X}; G)$$

So, with the association from equation 4.4 and the previuos statement, we get our missing homomorphism:

$$\Phi_{\mathcal{U}} : H^1(\mathbb{X}; \mathbb{Z}/2) \longrightarrow \check{H}^1(\mathcal{U}, \mathscr{C}_{\mathbb{R}^\times})$$

letting us define the function we where after:

$$\Psi : H^1(\mathbb{X}; \mathbb{Z}/2) \longrightarrow Vect^1_{\mathbb{R}}(\mathbb{X}) \tag{4.5}$$

To wrap up, we have that if $\mathbb{X}$ is a paracompact CW-Complex, with $\mathcal{U}$ a nonempty open set like in Theorem 1, then we have the following three relations:

1. $H^1(\mathbb{X}; \mathbb{Z}/2) \cong [\mathbb{X}; \mathbb{RP}^\infty]$ (4.1)
2. $Vect^1_{\mathbb{R}}(\mathbb{X}) \longleftrightarrow [\mathbb{X}; \mathbb{RP}^\infty]$ (4.2)
3. $H^1(\mathbb{X}; \mathbb{Z}/2) \longrightarrow Vect^1_{\mathbb{R}}(\mathbb{X})$ (4.5)

Recall our objective function $f_X : X \longrightarrow \mathbb{RP}^\infty$ and notice how:

$$[f_X] \in [\mathbb{X}; \mathbb{RP}^\infty]$$

so the interaction between these three elements will serve as the theoretic ground for the construction of this function.

4.2 Persistent Cohomology

This section focuses on understanding elements in the first cohomology group $H^1(\mathbb{X}; \mathbb{Z}/2)$, which from the previous section, we know there is a theoretic road we can take to get a representative of a class in $[\mathbb{X}; \mathbb{RP}^\infty]$. This can be done through persistent cohomology.

Suppose we are given a finite sample $X \subset \mathbb{X}$ from some metric space of dimension n, and we are given the task of computing the cohomology groups of $\mathbb{X}$ by only looking at X. Of course the elements of X will have gaps between them, since we are only taking a finite sample. So we will probably need to fill in the spaces in order to get a good sense of the topological features of $\mathbb{X}$. One way to do this is through $n-$dimensional balls with shared radii. By turning each point in X into a ball, we might close the appearing gaps and get a sense of real space.

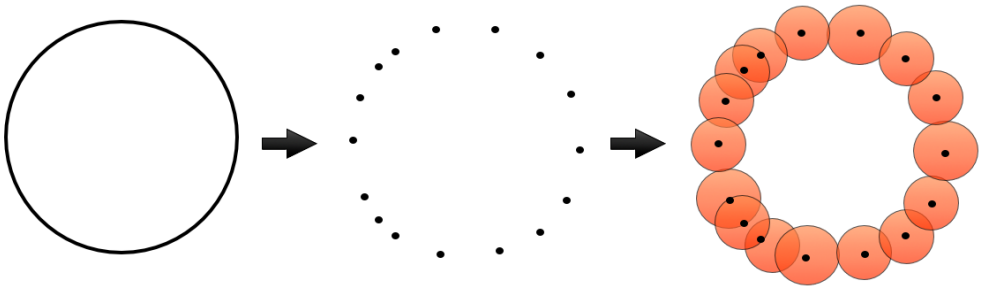


Figure 4.1: Example of how to approximate cohomology through balls, given only a finite sample of points of the original set.

To this set of open balls, we can assigned a corresponding simplicial complex, called the Nerve (definition 12) to which we can compute the corresponding cohomology groups [DSMVJ11]. The key idea behind persistent cohomology is to repeat this procedure for different growing radii, and see how cohomology changes, with the hope that persistent elements across longer radii, give us elements in the cohomology group.

Definition 13. A **Filtered Simplicial Complex** is a family of abstract simplicial complex $K = \{K_\varepsilon\}_{\varepsilon \geq 0}$, where: $K_0 = \emptyset$ and for every $\varepsilon_1 \leq \varepsilon_2$ we have that $K_{\varepsilon_1} \subset K_{\varepsilon_2}$.

So if we have a collection: $0 = \varepsilon_0 < \varepsilon_1 < \cdots < \varepsilon_m \ldots$, for a field $\mathbb{F}$ and $n \in N$, we get a chain of vector spaces and linear transformations:

$$H^n(K_{\varepsilon_0}, \mathbb{F}) \xleftarrow{T_1} H^n(K_{\varepsilon_1}, \mathbb{F}) \xleftarrow{T_2} \ldots \xleftarrow{T_m} H^n(K_{\varepsilon_m}, \mathbb{F}) \xleftarrow{T_{m+1}} \ldots$$

where

$$T_m\Big([\phi]\Big) = \Big[\phi|_{K^{(n)}_{\varepsilon_{m-1}}}\Big]$$

recall that we are trying to track persistent elements across radii, which means we want to track elements $0 \neq [\phi] \in H^n(K_{\varepsilon_m}, \mathbb{F})$ that $T_m\Big([\phi]\Big) \neq 0$ for several sequential values of m. If each vector space in the previous chain is finite dimensional and we have that T_m is an isomorphism for an m sufficiently large, then from the **Basis Lemma** [EJM15], we can choose a basis $V^m = \{V^m_1, \ldots, V^m_{d_m}\}$ for each $H^n(K_{\varepsilon_m}, \mathbb{F})$ such that the following holds:

- $T_m(V^m) \subset V^{m-1} \cup \{0\}$ for all $m \in \mathbb{N}$
- if $T_m(V^m_l) = T_m(V^m_r) \neq 0$ then $l = r$

this lets us trace elements along the filtration. A common way to visualize and detect these persistent classes is through bar codes, associating with each encountered class a pair of numbers: $[\alpha, \beta]$ with $\alpha < \beta$. This means that a class $[\phi]$ is "alive" between $[\alpha, \beta]$, in sense that for any $\alpha \leq \varepsilon_i < \varepsilon_j \leq \beta$:

$$T_i \circ T_{i+1} \circ \cdots \circ T_{j-1} \circ T_j\Big([\phi]\Big) \neq 0$$

We say a class is born in β and dies in α. We can plot this pair of numbers as horizontal bars to detect the longest ones and in turn detect the most persistent classes. For example:

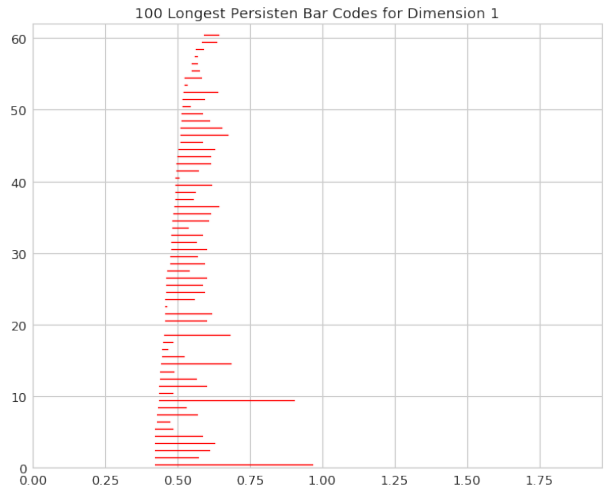


Figure 4.2: Persistence bar plot for a sparse filtration of 100 points with $\varepsilon = 0.5$, obtained from a 3000 sample from the two dimensional torus. Notice that there are two predominantly longer bars (classes).

Calculation of persistent cohomology will be done with the software: **Dionysus 2** available at:

`http://www.mrzv.org/software/dionysus2/`

Explicitly, this software does not compute the cohomology for each radii via the Čech Complex (Nerve), but instead uses another simplicial complex:

Definition 14. Given a metric space $(\mathbb{X}, \mathbf{d})$ and some sample $X \subset \mathbb{X}$, fix some $\varepsilon > 0$. The **Vietoris-Rips Complex** for radius ε is defined as

$$VR_\varepsilon(X) = \Big\{\{x_1, \ldots, x_n\} \subset X \ : \mathbf{d}(x_i, x_j) < \varepsilon, \ \ \forall i, j \leq n \ \text{ with } \ n \in \mathbb{N}\Big\}$$

Although similar to the Čech Complex, this simplicial complex does not require all open balls to intersect, but rather that they have non empty intersections by pairs. Even though this new scheme makes the Vietoris-Rips

complex have equal or larger dimension than the Čech complex[3], it is computationally less expensive to compute [Le+15]. The problem is that the nerve theorem (Theorem 1), does not apply to this new simplicial complex. Never the less, most software (including Dionysus 2) use this complex because of the following fact:

Observation 1. Let X be a sample of a metric space $(\mathbb{X}, \mathbf{d})$, then we have the following inclusions:

$$\check{C}_\varepsilon(X) \subset VR_\varepsilon \subset \check{C}_{2\varepsilon}(X)$$

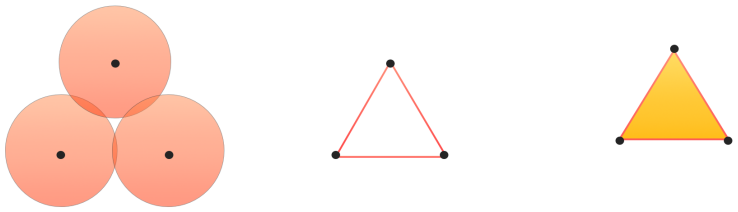
Furthermore:

Theorem 2. *[DSG07] Let $X \subset \mathbb{R}^d$ be a finite sample. Then for any ε we have the following chain of inclusions:*

$$VR_{\varepsilon'}(X) \subset \check{C}_\varepsilon(X) \subset VR_\varepsilon(X) \quad \textit{where} \quad \frac{\varepsilon}{\varepsilon'} = \sqrt{\frac{2d}{d+1}}$$

Moreover, this ratio is the smallest for which the inclusions holds in general.

Hence, making Vietoris-Rips the ideal choice when computing persistent cohomology.



(a) Initial layout. (b) Čech Complex. (c) Vietoris-Rips Complex.

Figure 4.3: Comparison between the Čech and the Vietoris-Rips Complexes

4.3 Sparse Filtrations

Computation of the respective simplicial complex for a specific radius can be computationally expensive, since one has to check for multiple intersections to determine the dimensionality of the output simplex. One way to alleviate this burden is through sparse filtrations, a scheme designed to stop the growth of radius for certain points that are no longer contributing to the change in cohomology.

[3] If n $\frac{\varepsilon}{2}$ balls intersect, all of their centers are a distance less than ε of each other (see figure 4.3)

Explained in detail by Nicholas J. Cavanna, Mahmoodreza Jahanseir and Donald R. Sheehy in [CJS15], the scheme has two main ideas

- For some selected points, their radii has a maximum value.
- Elements whose corresponding open balls is completely contained in another element's ball, are removed from the filtration.

Notice that leaving a radius constant at a certain value and removing only elements whose open neighborhood is completely inside another open ball, still gives us an embedded filtration of simplicial complexes which is what we need for the computation of persistent cohomology.

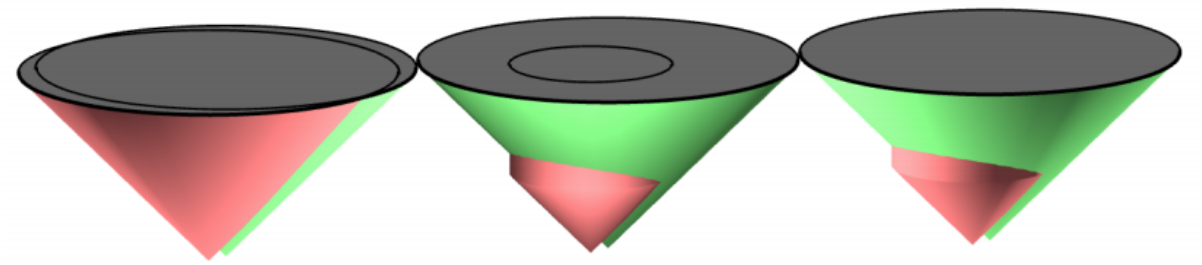


Figure 4.4: Example of growing radii behaviour. Here each color correspond to a point and we can visualize the growing radii scheme as a cone. On the left we have the usual scheme, where both radii grow without limitations. In the center, notice how the red cone stops its growth at a point and eventually resides entirely inside the green cone. On the right, the red ball has been removed from the filtration, since it resides completely inside the green ball and does not give any new simplexes. Image taken from: [CJS15]

To follow this scheme, we first need a greedy permutation of the given set:

Definition 15. If X is a set of points from some metric space $(\mathbb{X}, \mathbf{d})$, a **Greedy Permutation** (landmark sets, farthest point sampling or discrete center set) is an ordering: $\{x_0, \ldots, x_n\}$ such that:

$$x_{i+1} = \arg\max_{x \in X} \mathbf{d}\Big(x, \{x_0, \ldots, x_i\}\Big)$$

with $0 < i \leq n$ and x_0 chosen randomly

from a greedy permutation of X, we can compute the descending values of the insertion radii:

Definition 16. Given a greedy permutation of X, the insertion radius λ_i for each element x_i is defined as:

$$\lambda_i = \begin{cases} \infty & i = 0 \\ \mathbf{d}\Big(x_i, \{x_0, \ldots, x_{i-1}\}\Big) & i > 0 \end{cases} \tag{4.6}$$

The way we can emulate the behaviour in Figure 4.4, is to create a functions $r_i : \mathbb{R}^+ \longrightarrow \mathbb{R}^+$ that adjust the radius for the element x_i as the global radius α grows, like this:

Fix $0 < \varepsilon < 1$, then the radius if the i-th element in X for a given $\alpha \geq 0$ is:

$$r_i(\alpha) = \begin{cases} \alpha & \alpha < \lambda_i(1+\varepsilon)/\varepsilon \\ \lambda_i(1+\varepsilon)/\varepsilon & \lambda_i(1+\varepsilon)/\varepsilon \leq \alpha \leq \lambda_i(1+\varepsilon)^2/\varepsilon \\ 0 & \alpha > \lambda_i(1+\varepsilon)^2/\varepsilon \end{cases} \tag{4.7}$$

Notice how the radius for every element first grows equally with the global radius α, stalls at a maximum and then drops to zero, making the ball and point disappear. By **Lemma 1** [CJS15], for any $\alpha \geq 0$ we have that:

$$X \subset \bigcup_{i=0}^{n} B_{r_i(\alpha)}(x_i)$$

where $B_\varepsilon(x)$ is the open ball of radius ε centered at x. And also by **Theorem 5** [CJS15] the bar codes obtained from the sparse filtration, are an $(1+\varepsilon)$ approximation of the usual constant growth filtration.

Recall that our persistent cohomology computation will be done with the software Dionysus 2, which currently does not support sparse filtrations. To bypass this limitation we can use the **Algorithm 3** [CJS15] to compute, under the sparse filtration scheme, the α responsible for the birth of each edge between pair of points in the set. We can then input into Dionysus 2 these birth times as distances between points, tricking the software into fulfilling the sparse filtration schema.

4.4 Step by Step Procedure

We end the chapter with a step by step procedure to compute the function to the new space.

We start with a finite set X of size $n+1$ and a fix value of $0 < \varepsilon < 1$.

1. Compute a greedy permutation of the given set. One way to do this is thought maxmin sampling ([DP13]) of the entire set, obtaining an ordering:

$$X = \{x_0, \ldots, x_n\}$$

2. Calculate the corresponding λ_i values for the previous permutation, using equation 4.6.

3. Calculate birth time of the edge between each pair of points x_i and x_j through **Algorithm 3** [CJS15].

4. Input the previous values as a distance matrix into Dionysus 2, to obtain the corresponding filtration and persistent bar diagram over $\mathbb{Z}/2$

5. Determine the class in dimension 1 with the longest persistence and select a corresponding cocycle representative τ, with its cohomological birth α. Recall that τ is a function from 1 dimensional simplexes (edges between points) to $\mathbb{Z}/2$. We will denote τ_{ij} as the image of the edge between x_i and x_j.

6. Compute the corresponding radii $r_i(\alpha)$ for each x_i as in equation 4.7 and define:

$$\bigcup B^\alpha = \bigcup_{j=0}^{n} B_{r_j(\alpha)}(x_j)$$

7. Define $\phi_j^\alpha : \bigcup B^\alpha \longrightarrow \mathbb{R}^+$ as:

$$\phi_j^\alpha(b) = \frac{|r_j(\alpha) - \mathbf{d}(b, x_j)|_+^2}{\sum_{t=0}^{n} |r_t(\alpha) - \mathbf{d}(b, x_t)|_+^2}$$

   where $|x|_+ = \max(0, x)$.

8. Then, by **Theorem 7.4** [Per16], we get the well defined map:

$$\begin{aligned} f_\tau^\alpha : \bigcup B^\alpha &\longrightarrow \mathbb{RP}^n \\ b \in B_{r_j(\alpha)}(x_j) &\mapsto \left[(-1)^{\tau_{0j}} \sqrt{\phi_0^\alpha(b)} : \cdots : (-1)^{\tau_{nj}} \sqrt{\phi_n^\alpha(b)}\right] \end{aligned} \tag{4.8}$$

   Giving us a restriction of the function $f : X \longrightarrow \mathbb{RP}^\infty$ we where after. Something to notice is that it is possible to construct f_τ^α for a finite sample M, using a subsample $X \subset M$. We only need that, for the obtained parameter α, the cover satisfies:

$$M \subset \bigcup B^\alpha$$

4.4.1 An Example

For sanity check, we include an example of the mentioned procedure. We will sample 3000 points from the second moment curve $\xi_2 \subset \mathbb{R}^4$ (see definition 2). We produce 51 points by means of maxmin sampling and compute the cohomology of the sparse filtration with $\varepsilon = 0.5$. The following figure shows the corresponding barcodes for dimension 1:

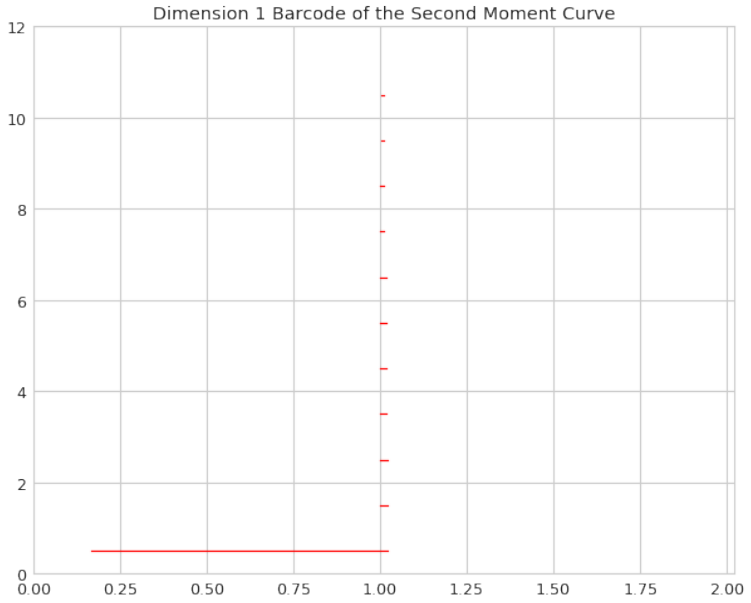


Figure 4.5: Persistence bar plot for a sparse filtration of 51 points with $\varepsilon = 0.5$, obtained from a 3000 sample from the second moment curve. Notice that there is a predominant line (classes), as expected since the curve is homeomorphic to a circle.

From the previous image, we select the cocycle corresponding to the longest line, obtaining $\alpha = 1.02$. We apply function 4.8 to the 3000 points to obtain a representation in $\mathbb{RP}^{50}$. The following figure plots the coordinates obtained from executing t-SNE (chapter 3.2) over a two dimensional space:

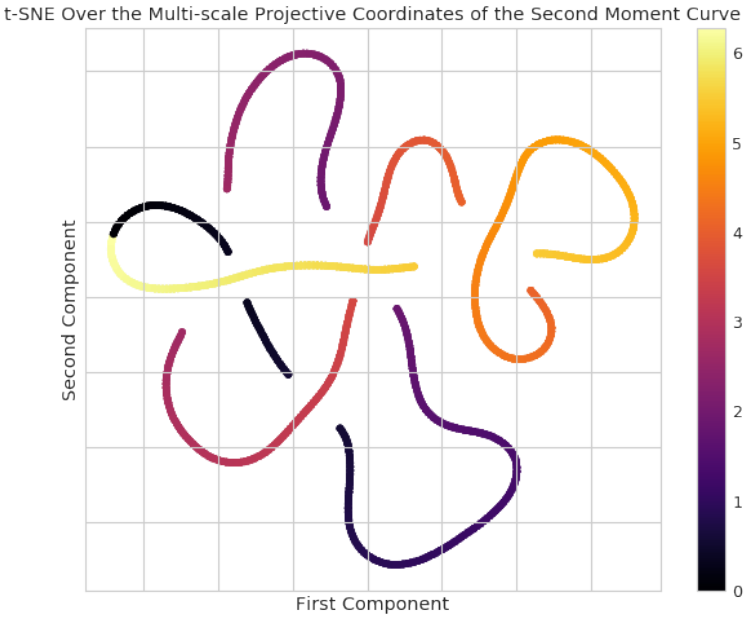


Figure 4.6: Two dimensional t-SNE over the coordinates in $\mathbb{RP}^{50}$ of the second moment curve, where the colors represent the angle t in curve $\xi_2(t)$. Notice how the structure of the circle is preserved. Recall that the t-SNE procedure requires a distance between the elements. For this, we used as a distance between two lines the smallest angle between them. This distance is defined in the next chapter in equation 5.3

4.5 Representation of MNIST in $\mathbb{RP}^{\infty}$

Recall that we want to find a representation of the 10,000 points in $\mathbb{R}^{784}$ (the MNIST test set) in $\mathbb{RP}^{\infty}$. As we saw earlier, it is possible to obtain this representation using only a small subset of database, as long as the obtained α is enough for the cover $\bigcup B^{\alpha}$ to contain the entire set. Although theoretically the case, computationally finding the minimum number of points that gives us a complete cover of the MNIST test set is not an easy task.

In the search for this minimum, we rely entirely on Dionysus 2 to obtain the corresponding α for the cover and with a subsample of 750 points, after 3 hours of computing, the corresponding $\bigcup B^{\alpha}$ covered of only 9% of the desired set. Dionysus 2 kept crushing after that for larger subsets. These types of problems can always be solved with better machines, more time and specific implementations, but we devised another solution.

We executed the procedure by selecting 501 points by means of max-min

sampling (to ensure a greedy permutation). Fixed $\varepsilon = 0.5$ and computed the corresponding cocycle representative and α from Dionysus 2. Since this α was not enough to cover the entire set, by means of binary search while keeping fixed the ε and the insertion radii of the sample, we calculated the minimum value α should take to give us a complete cover. With this new α we where able to project the entire MNIST test set into $\mathbb{RP}^{500}$.

Here is an image of the persistence barcodes in dimension 1 from the previous scenario:

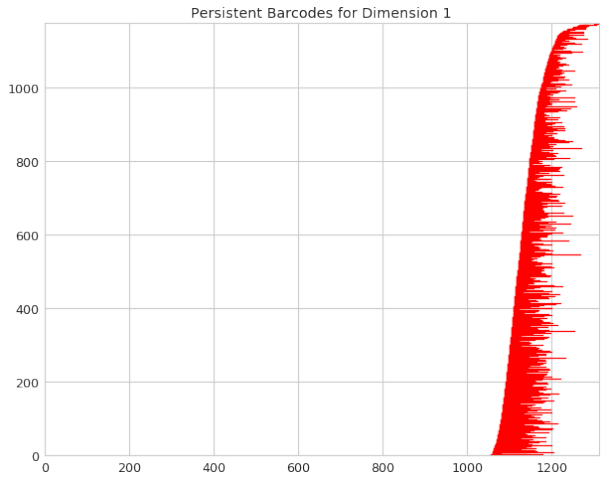


Figure 4.7: Persistence bar plot for a sparse filtration of 501 points with $\varepsilon = 0.5$, sampled from MNIST test set using max-min procedure.

In the absence of a predominant line (cohomology class) we construct the candidate for our function 4.8 using all lines. Since each line represents an independent class, by adding them we obtain a new cohomology class with the longest barcode, which we will us as τ for our function.

Chapter 5

Distance Matrix

5.1 Distance Matrix as a Computational Limitation

As was mentioned in chapter 2, one of the inputs for the *Mapper* algorithm is a distance matrix between the points. Although technically the computation demands of calculating such a matrix are outside the scope of the algorithm in question, in practical manners and taking into account how most of the data is given to us, we need to calculate such a matrix from the input dataset. Therefore, calculating the distances between records of data is one of the algorithm's computational limitations. In this chapter we will explain Locality-Sensitive Hashing, a technique normally used for detecting duplicates among high dimensional datasets. We will explain how this technique can be used to approximate a distance matrix in an efficient way, giving some degree of scalability to the algorithm.

5.2 Locality-Sensitive Hashing

Local Sensitive Hashing (LSH) is a method proposed by Aristides Gionis et al. in 1999 [GIM+99], as a way to approximate the solution of the similarity problem among high dimensional data sets: given a large amount of records, one could be interested in finding duplicates among these records. Clearly the brute force approach for this problem does not behave well as the dimension and size of the sample increases, since we are forced to compare every pair of records. LSH bypasses the dimension problem by efficiently finding good candidates for identical records (nearest neighbors), so we only need to check for similarities among these smaller sets.

The key idea behind LSH is to design a hash function that is *locality-sensitive* with the metric used to find exact duplicates. This means that it should be

more probable for two elements that are close together to have the same hash key (same image under the hash function) than for elements very far apart.

5.2.1 Hyperplane Hashing Function

We will give a small example using the Hyperplane Hashing Function.

Suppose we are given a set $X = \{x_1, \ldots, x_n\}$ sampled from the D-dimensional sphere. Recall that the idea for LSH is to assign a hash key to each element, so we can later check for duplicates only among elements with same hash key. In this case, the hash function will be built as follows:

- Select a random D dimensional vector r whose entries are i.i.d standard Gaussian.
- For each vector x_i, compute the sign (0 negative and 1 positive) of the dot product: $\langle x_i, r \rangle$.
- Repeat k times.

The final hash key for the element x_i is the concatenation of the resulting sign of each of the k iterations.

It's usually referred to the group of elements that share the same hash key as buckets. It's inside each bucket that we will search for identical elements, instead of looking around the whole sample. And if we use as a metric for identifying similar or identical elements the **Cosine Distance**[1]:

$$\mathbf{d}_{cos}(x_i, x_j) = 1 - \frac{\langle x_i, x_j \rangle}{\|x_i\| \|x_j\|}$$

We get that the probability for two points with angle θ between them to have the same hash function value is [And+15]:

$$\left(1 - \frac{\theta}{\pi}\right)^k$$

So we have that as the distance (angle) between the points increases, the probability of sharing a hash key reduces, hence the function is in fact locality-sensitive.

5.3 Approximating the Distance Matrix using LSH

Before we continue we must make the following remark: since the distance approximation is done by performing LSH many times on the data set, we will

[1] Notice that this function is only a distance if we normalize the entire set.

assume that the has function is not deterministic but random. So as long as the function remains locality-sensitive, this allows us to get better approximations as the number of times we preform LSH increases.

Now, given a set $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^D$ and let d be a certain distance function on $\mathbb{R}^D$, there are two ways in which LSH can be used to approximate the real distance matrix between points.

5.3.1 Lower Approximation

The first strategy will approximate the distance from below, that is, we will get a "metric" d_l that allows the distance between two different elements to be zero.

The key idea is to select an element that will act as a representative for each one of the buckets. So suppose the previous procedure returned a set of buckets: $B = \{B_1, \ldots, B_k\}$, we then choose an element from each of these buckets[2] and get a set of representatives: $b = \{b_1, \ldots, b_k\}$. Let $x, y \in X$ with $x \in B_i$ and $y \in B_j$ for some $i, j \in \{1, \ldots, k\}$, then the approximate distance d_l for these two elements is defined like this:

$$d_l(x, y) = \begin{cases} 0 & i = j \\ d(b_i, b_j) & i \neq j \end{cases}$$

As was mentioned at the beginning of the section, the idea is to execute LSH many times over the set. So, let L be that number of times and d_{lm} the distance obtained at the m-th iteration. Our final approximate distance will be:

$$d_l = \underset{i}{\text{mean}}(d_{lm})$$

Recall that this is not technically a distance since it's aloud for $d_l(x, y) = 0$ without $x = y$.

5.3.2 Higher Approximation

The second strategy is to approach the distance from above, constructing a "distance" d_h such that:

$$d_h \geq d$$

Notice how in the lower approximation we assigned zero to the uncomputated distances, insuring a lower bound for these unknown relations (since distances have to be positive, it's the obvious choice to assign zero). We are now faced with the same problem of assigning uncomputed distances, but this time from above. An immediate choice is to simply assing infinity to such tuples, but clearly we can do better. The next idea would probably be to assign the diameter of the

[2]For example, we could select the element that is closer to the mean of the elements inside the bucket

set to these distances (a less conservative bound) but to compute this value we would need the distance between all the records, the precise computation we are trying to avoid.

The value $\delta(X)$ we will assign will be the diameter of the smallest D dimensional cube containing all of X. Which can be calculated as follows:

$$\delta_{max}(X) := \Big(\max(\pi_1(X)), \ldots, \max(\pi_d(X)) \Big)$$
$$\delta_{min}(X) := \Big(\min(\pi_1(X)), \ldots, \min(\pi_d(X)) \Big)$$
$$\delta(X) = d\big(\delta_{max}(X), \delta_{min}(X)\big)$$

where $\pi_j(X)$ is the projection of X onto its j-ith coordinate. This way we get a better upper bound than infinity without having to compute the pairwise distances. Notice that $\delta(X)$ will be equal to the set's diameter if $\delta_{max}(X), \delta_{min}(X) \in X$.

Now, following the same scheme as the previous approximation, suppose we ran LSH for the m-th time over X, obtaining the set of buckets: $B = \big\{B_1, \ldots, B_k\big\}$. So, for $x, y \in X$ with $x \in B_i$ and $y \in B_j$, with $i, j \in \{1, \ldots, k\}$, we define d_{hm} as:

$$d_{hm}(x, y) = \begin{cases} d(x, y) & i = j \\ \delta(X) & i \neq j \end{cases}$$

and after L iterations, we have that:

$$d_h = \min_m(d_{hm})$$

As with the previous approximation, d_h is not a distance (this time the triangle inequality does not hold) but we get that:

$${d_h}^L(x, y) \xrightarrow{L \to \infty} d(x, y)$$

in probability for every $x, y \in X$:

Proof 1. Assume that $x \neq y$ and that $x, y \notin \big\{\delta_{max}(X), \delta_{min}(X)\big\}$, because in both those cases we have that ${d_h}^L(x, y) = d(x, y)$. So if we let p be the probability that x and y have the same hash function, we have that:

$$P\Big({d_h}^L(x, y) \neq d(x, y))\Big) = (1 - p)^L$$

.

since now ${d_h}^L(x, y) \neq d(x, y))$ will only happen if the elements never share a bucket over the L iterations. So for $\varepsilon > 0$, we have that

$$P\Big(|{d_h}^L(x,y) - d(x,y)| > \varepsilon\Big) \;<\; P\Big({d_h}^L(x,y) \neq d(x,y)\Big) = (1-p)^L$$

which tends to zero as L tends to infinity, as we wanted. $\square$

5.3.3 Complexity Trade-off

The whole idea behind working with either d_l or d_u, is that it's computationally cheaper to calculate them. Recall that we are working with a set set $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^D$, a distance d and a locality-sensitive hash function H. So let:

- $g_d(N, D)$ be the computational cost for calculating d between two records of X.
- $g_H(N, D)$ be the computational cost for calculating H over all records in X.
- $E_H(N, D)$ be the expected number of elements per bucket. Hence the expected number of buckets is in turn: $N/E_H(N, D)$

Notice that with these values, the computational cost for $\delta(X)$ is $O(N^2)$ (which is better than the real diameter that costs $O(N^2\, g_d)$)

Given the previous values, the computational cost (CC) for calculating the approximate distances for all pair of elements are:

$$\begin{aligned} CC(d_l) &= O\Big(L\,(\text{Hashvalues} + \text{Dist. between buckets} + \text{Mean})\Big) \\ &= O\Big(L\,g_H + L\,g_d\left(\frac{N}{E_H}\right)^2 + L\,N^2\Big) \end{aligned} \tag{5.1}$$

$$\begin{aligned} CC(d_l) &= O\Big(L\,(\text{Hashvalues} + \text{Dist. inside buckets} + \text{Calc. } \delta(X) + \text{Max})\Big) \\ &= O\Big(L\,g_H + L\,g_d\Big(E_H\Big)^2 + N\,g_d + L\,N^2\Big) \end{aligned} \tag{5.2}$$

So if we use the Hyperplane Hashing Function (with k random hyperplanes for each LSH iteration), the cosine distance and assume our data is sampled from the D dimensional sphere we have that:

- $g_d(N, D) = D$
- $g_H(N, D) = D\,N\,k$

- $E_H(N, D) = N/2^k$

And consequently we get that:

$$\begin{aligned} CC(d_l) &= O\Big(L\,D\,N\,k + L\,D\,\Big(\frac{N\,2^k}{N}\Big)^2 + L\,N^2\Big) \\ &= O\Big(L\,D\,N\,k + L\,D\,2^{2k} + L\,N^2\Big) \end{aligned}$$

$$\begin{aligned} CC(d_u) &= O\Big(L\,D\,N\,k + L\,D\,\frac{N^2}{2^{2k}} + N\,D + L\,N^2\Big) \\ &= O\Big(L\,D\,N\,k + D\,N\,\Big(L\,\frac{N}{2^{2k}} + 1\Big) + L\,N^2\Big) \\ &= O\Big(L\,D\,N\,k + L\,D\,N\,\Big(\frac{N}{2^{2k}}\Big) + L\,N^2\Big) \end{aligned}$$

So by letting $k = log_2(\sqrt{N})$ the costs are:

$$CC(d_l) = CC(d_h) = O\left(L\left(D\,N\,log_2(\sqrt{N}) + N^2\right)\right)$$

which, depending on the value of L, can be lower than computing the whole matrix:

$$CC(d) = O\big(D\,N^2\big)$$

5.4 LSH Applied to MNIST

As we saw earlier, we will have two different representations of the MNIST test set: in $\mathbb{R}^{784}$ and $\mathbb{RP}^{500}$. In the former we will use as a distance: the cosine distance ($\mathbf{d}_{cos}$), and approximate it using the hyperplane hash function as was discussed in section 5.2.1. With the latter we can't use this same scheme, notice that $\mathbf{d}_{cos}(x, y) \neq \mathbf{d}_{cos}(-x, y)$ and since in $\mathbb{RP}^{500}$ elements are invariant under scalar multiplication, we can't use $\mathbf{d}_{cos}$ as our distance.

We would idealy like the distance between two lines in $\mathbb{R}^{500}$ to be the smallest angle between them. This corresponds to the geodesic distance in $\mathbb{RP}^{500}$, induced by the Fubini-Study metric, calculated as follows:

$$\mathbf{d}_{\mathbb{RP}}\Big([x], [y]\Big) = arcos\left(\frac{|\langle x, y\rangle|}{\|x\|\|y\|}\right) \tag{5.3}$$

Notice that this distance gives us a value between $[0, \frac{\pi}{2}]$ and is invariant under scalar multiplications of the element, giving us the angle between two

lines in $\mathbb{R}^{501}$. Both Python and R, lack internal libraries that compute this distance, but it can still be calculated somewhat natively using the fact that:

$$\mathbf{d}_{\mathbb{RP}}\Big([x],[y]\Big) = \arccos\Big(|-\mathbf{d}_{cos}(x,y)+1|\Big)$$

Now, we need a corresponding hash function for this distance in order to properly execute LSH. Let $\hat{X}$ be the MNIST test set in some $\mathbb{RP}^n$ in spherical coordinates, that is for $[x] \in \hat{X}$:

$$x \in \left(\prod_{1}^{n-1}[-\pi,\pi]\right) \times [0,2\pi)$$

(we assume that $r = 1$)

Let $[x],[y] \in \hat{X}$, with θ be the angle between these two lines that is:

$$\mathbf{d}_{\mathbb{RP}}([x],[y]) = \theta$$

Now for $j \in \{0,\dots,n\}$, consider the projections $x^j = \pi_j(x)$ and $y^j = \pi_j(x)$. Since the angle between this two lines is θ, depending on the value of j, we must have one of the following scenarios:

- if $j < n$:
$$|x^j - y^j| \leq \theta \quad \text{or} \quad |(-x^j) - y^j| \leq \theta$$
- if $j = n$:
$$|x^j - y^j| \leq \theta \quad \text{or} \quad |(x^j - \pi) - y^j| \leq \theta$$

Recall that the objective of a hash function is to group together element that are close to one another (giving a certain distance). With this in mind and given a random element $\alpha_j \in [0, \frac{\pi}{2})$, let us divide the unitary circle equally in following four segments:

1. $I_{\alpha_j} := \Big[\alpha_j, \alpha_j + \frac{\pi}{2}\Big)$
2. $II_{\alpha_j} := \Big[\alpha_j + \frac{\pi}{2}, \alpha_j + \pi\Big)$
3. $III_{\alpha_j} := \Big[\alpha_j + \pi, \alpha_j + \frac{3\pi}{2}\Big)$
4. $IV_{\alpha_j} := \Big[\alpha_j + \frac{3\pi}{2}, 2\pi\Big) \cup \Big[0, \alpha_j\Big)$

Basically, the usual four quadrants turned α_j radians clockwise. This gives us that either:

$$x^j \in I_{\alpha_j} \cup III_{\alpha_j} \quad \text{or} \quad x^j \in II_{\alpha_j} \cup IV_{\alpha_j}$$

letting us state the well defined function:

$$H^j_{\mathbb{RP}}\Big([x]\Big) = \begin{cases} 0 & x^j \in I_{\alpha_j} \cup III_{\alpha_j} \\ 1 & x^j \in II_{\alpha_j} \cup IV_{\alpha_j} \end{cases}$$

Now, we have a way to calculate a binary element for every dimension of data in $\mathbb{RP}^n$, so we use these individual functions to construct our final hash function which we will use to approximate $\mathbf{d}_{\mathbb{RP}}$. The direct choice will be to concatenate the results of the individual functions for every dimension, which is not a good idea if n is too large. It makes sense then to look only at a random sample of size k of the dimensions of the data and repeat this process K times. This gives us Kk binary values which we can concatenate into our final hash function.

So, fix some $k \leq n$, $K \in \mathbb{N}$ and let $\Lambda^j_k = \{\lambda^j_1, \ldots, \lambda^j_k\} \subseteq \{1, \ldots, n\}$ chosen at random without replacement for $1 \leq j \leq K$.

Then our hash function for this representation of the MNIST test will be:

$$H_{\mathbb{RP}}([x]) = \{H^{\lambda^1_1}_{\mathbb{RP}}\Big([x]\Big), \ldots, H^{\lambda^1_k}_{\mathbb{RP}}\Big([x]\Big), H^{\lambda^2_1}_{\mathbb{RP}}\Big([x]\Big), \ldots, H^{\lambda^K_k}_{\mathbb{RP}}\Big([x]\Big)\}$$

In order for this hash function to work, we need it to be locality-sensitive with respect to $\mathbf{d}_{\mathbb{RP}}$. Again let $[x], [y] \in \hat{X}$ with $\theta = \mathbf{d}_{RP}([x], [y])$, assume that $y^j > x^j$ and let:

$$\theta_j = \min\{|x^j - y^j|, |(-x^j) - y^j|\}$$

Notice that $H^j_{\mathbb{RP}}\Big([x]\Big) \neq H^1_{\mathbb{RP}}\Big([y]\Big)$, means that the random α_j was such that:[3]

$$x^j \leq \alpha_j \leq y^j \quad \text{or} \quad x^j \leq \alpha_j + \frac{\pi}{2} \leq y^j$$

thus creating one of the new quadrants in the middle of the two lines. Hence, if each α_j is chosen uniformly, we have that:

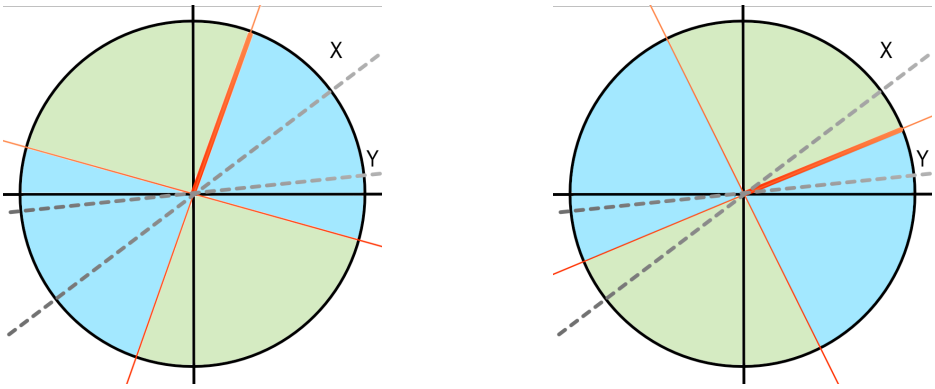
$$P\left(H^j_{\mathbb{RP}}\Big([x]\Big) = H^j_{\mathbb{RP}}\Big([y]\Big)\right) = \left(\frac{\pi/2 - \theta_j}{\pi/2}\right) = \left(1 - \frac{2\theta_j}{\pi}\right) \geq \left(1 - \frac{2\theta}{\pi}\right)$$

So we get that:

$$P\left(H_{\mathbb{RP}}\Big([x]\Big) = H_{\mathbb{RP}}\Big([y]\Big)\right) \geq \left(1 - \frac{2\theta}{\pi}\right)^{Kk}$$

making the elements less likely to have the same hash function as their distance increases, rendering $H_{\mathbb{RP}}$ locality-sensitive with respect to $\mathbf{d}_{\mathbb{RP}}$, like we wanted.

[3]The case where j = n, is similar



(a) Same hash value.

(b) Different hash value.

Figure 5.1: Grafical example of the hash function $H^j_{\mathbb{RP}}$ for $[x]$ and $[y]$

5.4.1 Experiment

We now compare d_l, d_h for our both representation:

- Representation in $\mathbb{R}^{784}$: We will use $\mathbf{d}_{cos}$ with the Hyperplane Hash Function.
- Representation in $\mathbb{RP}^{\infty}$: We will use $\mathbf{d}_{\mathbb{RP}}$ with the $H_{\mathbb{RP}}$ as the hash function. The specific representation will be in $\mathbb{RP}^{50}$, as explained in chapter 4.

For both representations, we sampled 1000 points uniformly from the MNIST test to compute the corresponding lower and higher distance matrix approximations: M_L, M_H and the complete distance matrix M. The following figure reports the value for $\|M - M_L\|_\infty$ and $\|M - M_H\|_\infty$ as the parameters L (iterations), K (number of hyperplanes or times $H_{\mathbb{RP}}$ is executed) and k (number of coordinates in $H_{\mathbb{RP}}$) increase.

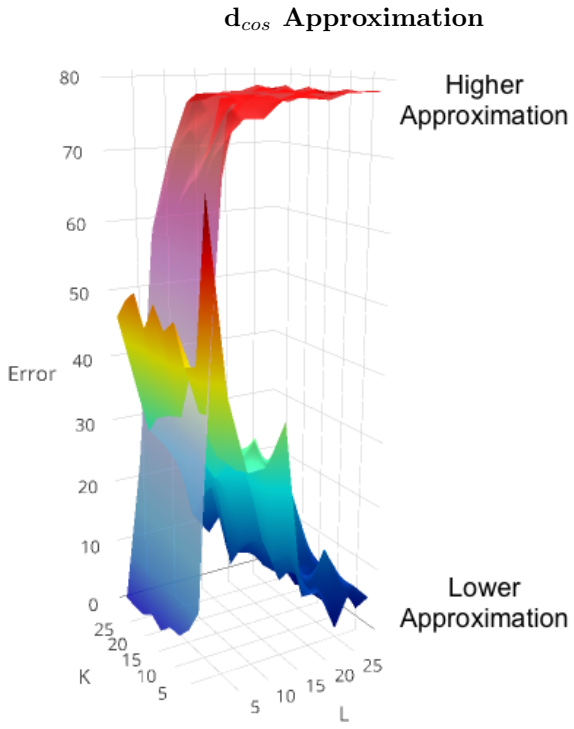


Figure 5.2: Lower and higher approximations of $\mathbf{d}_{cos}$, using the hyperplane hash funciton with a sample of 1000 elements of the MNIST test set in its $\mathbb{R}^{784}$ representation.

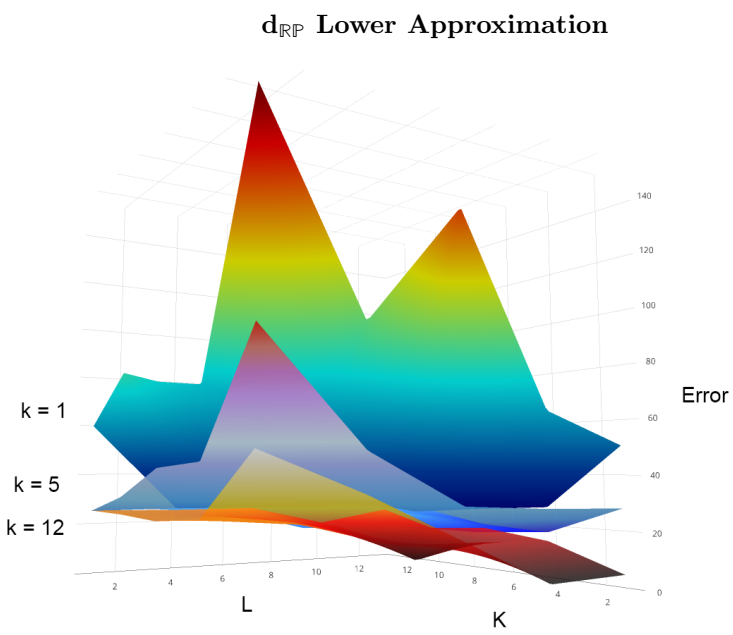


Figure 5.3: Lower approximation of $\mathbf{d}_{\mathbb{RP}}$, using $H_{\mathbb{RP}}$ with a sample of 1000 elements of the MNIST test set in its $\mathbb{RP}^{50}$ representation.

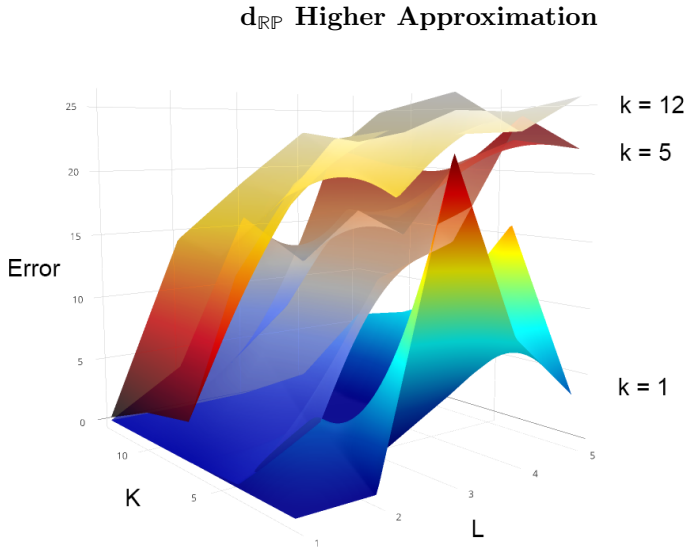


Figure 5.4: Higher approximation of $\mathbf{d}_{\mathbb{RP}}$, using $H_{\mathbb{RP}}$ with a sample of 1000 elements of the MNIST test set in its $\mathbb{RP}^{50}$ representation.

Chapter 6

Results

The following chapter shows the obtained graphs from applying *Mapper* to the MNIST test set.

For each of the possible inputs of *Mapper* we will have two scenarios as follows:

- **Data Representation:**
  1. The data representation will be in its original format, each image will correspond to an element in $\mathbb{R}^{784}$.
  2. The data will be represented in $\mathbb{RP}^{500}$, following the procedure outlined in section 4.5, with $n = 500$ and $\varepsilon = 0.5$.
- **Distance Matrix:** For the similarity between records we have:
  1. The distance matrix will be in its complete form, using $\mathbf{d}_{cos}$ for the R^{784} representation and $\mathbf{d}_{\mathbb{RP}}$ for the $\mathbb{RP}^{500}$ representation.
  2. We used the lower approximation of the distance matrix. For the $\mathbb{R}^{784}$ representation, we approximate $\mathbf{d}_{cos}$ using the Hyperplane Hash Function as explained in section 5.2.1 with:

     $$K = \lceil \log_2(N) \rceil = \lceil \log_2(10,000) \rceil = 14 \qquad L = \lceil \log_2(D) \rceil = \lceil \log_2(784) \rceil = 10$$

     where $\lceil \cdot \rceil$ is the ceiling function. And for the $\mathbb{RP}^{500}$ representation, we approximate $\mathbf{d}_{\mathbb{RP}}$ using $H_{\mathbb{RP}}$ for our hash function as described in section 5.4, with:

     $$L = 2 \qquad K = \lceil \log_2(N) \rceil = \lceil \log_2(10,000) \rceil = 14 \qquad k = \frac{D}{5} = \frac{500}{5} = 100$$

We use the lower approximations since it has given good results in conjunction with the *Mapper* algorithm [MS16].

- **Filter Function:**

  1. The filter function will be the first 3 components obtained by Principal Component Analysis (PCA), over the data in its $\mathbb{R}^{784}$ representation. The filter space will be then: $\mathbb{R}^3$
  2. The filter function will be the resulting projection after executed t-SNE over a three dimensional space over the data in its $\mathbb{R}^{784}$ representation, as explained in section 3.2. Thus, our filter space will also be $\mathbb{R}^3$

We will execute all possible combinations of the previous inputs (giving us a total of 8 scenarios) and compare the obtained results. Although *Mapper* technically allows the configuration of the clustering algorithm, we will use the one selected from the original *TDAmapper* library: Hierarchical Clustering [Ble08], for all our executions.

6.1 Obtained TDA Graphs

The following figures show the resulting graphs from the *Mapper* algorithm for each one of our scenarios. Each plot will be graph with colored nodes under the following scheme, to asses its quality:

1. **Size of the Node:** The size of the node will be calculated accordingly to the amount of elements it groups. Bigger nodes will then hold more digits.

2. **Color of the Node:** The color of each node will be determined by its most common digit. This means that if a node is composed of: 40% digit 1, 30% digit 4 and 30% digit 7, the node will be filled with the color corresponding to: digit 1.

3. **Transparency of Color:** To asses the purity of each node (if its composed only by a single type of digit), the color of the nodes will have a transparency value according to the formula:

$$transparency(N) = \frac{\#\text{ of elements of predominant digit in N}}{\text{total elements in N}}$$

   this way a transparent node means that, although the color of the node indicates a majority, its still a node with mixed digits. Whereas a none transparent node means that its composed almost entirely by a single type of digit.

Although it is hard to generalize what a good outcome graph will look like after executing the algorithm, for this particular set there are a few aspects from which we can judge quality. First, we would like most of the nodes to be pure in the sense that they should only contain a single type of digit. Notice we say most instead of all, because it is also interesting to see interaction between subgraphs. So unless we want 10 isolated subgraphs, there has to be some mixed nodes to build arches between clusters.

Second, we want medium size nodes. Clearly *medium* is a little ambiguous, but what is clear is that we don't want an entire type of digits to belong only to one huge node, or have it completely scattered in nodes composed of singletons or pair. We want considerable grouped sized interacting nodes to have some insight about the data.

Finally, we would like the graph to have a good set of connections. Again, as with the previous aspects, is hard to define what it means to have a *good* amount, but we have that both a set of 10 isolated nodes and a full 10,000 clique, give us no insight of the data.

With the previous three considerations in mind, we've included some numeric values to help us asses the quality of the outcome graph. Let $X = \{x_1, \ldots, x_{10,000}\}$ be the MNIST test set:

- **Purity of a Digit**: for each x_j we compute the value p^j, corresponding to the percentage its digit type holds inside the node containing x_j. For example, if a node V with elements inside $\{x_1, \ldots, x_5\}$ with corresponding digit labels: $\{1, 1, 1, 1, 2\}$, then we have that:

  $$p^1 = p^2 = p^3 = p^4 = 0.8 \quad \text{and} \quad p^5 = 0.2$$

  Recall that a certain element can belong to more than one node (hence the connections between them), so for an element x_j we will have a set of values:

  $$\{p_1^j, \ldots, p_m^j\}$$

  The **Purity** of a digit will be the average of all values p_i^j corresponding to elements with that type of digit.

- **Groupness of a Digit**: for a certain digit j, its corresponding **Groupness** will be the average of amount of elements of type j per node, discarding the nodes with no j elements in them. So for example, if the corresponding groupness for digit our digit is 50, it means that on average, you will find groups of 50 elements of type j grouped together inside a node.

- **Connectivity of the Graph:** We will asses connectivity of the graph through edge connectivity and vertex connectivity. Let's give the necessary definitions for theses concepts

  Definition 17. Given an undirected graph $g = G(V, E)$, we say g is **Connected** if for every $v_1, v_2 \in V$, there are paths of arches connecting v_1 to v_2.

  Definition 18. Given an undirected graph $g = G(V, E)$, the **Edge Connectivity** of g is minimum number of edges that need to be removed in order to make g no longer connected.

  Definition 19. Given an undirected graph $g = G(V, E)$, the **Vertex Connectivity** of g is minimum number of nodes that need to be removed in order to make g no longer connected.

  It is very likely that our resulting graphs will not be connected, making both of the previous defined values equal zero. Hence we will report for each graph the mean edge and vertex connectivity for all it's connected components, disregarding isolated nodes as components.

  Also, we will report the number of connected components for each graph, again, not considering isolated nodes as components.

We will give each one of these values for every proposed configuration.

6.1.1 Scenario 1

Configuration:

- **Data Representation:** $\mathbb{RP}^{500}$
- **Distance Matrix:** Lower approximation
- **Filter Function:** t-SNE

	Value	0	1	2	3	4	5	6	7	8	9	Avg.
1	Purity (%)	96.7	93.7	95	96.6	91.5	97.6	96.4	91.3	97.3	91	94.7
2	Groupness	16	28.3	19.6	20.1	22	18.3	18.6	20.4	20.7	23.7	20.8

Components: **24** Edge Connectivity: **1.17** Vertex Connectivity: **1.12**

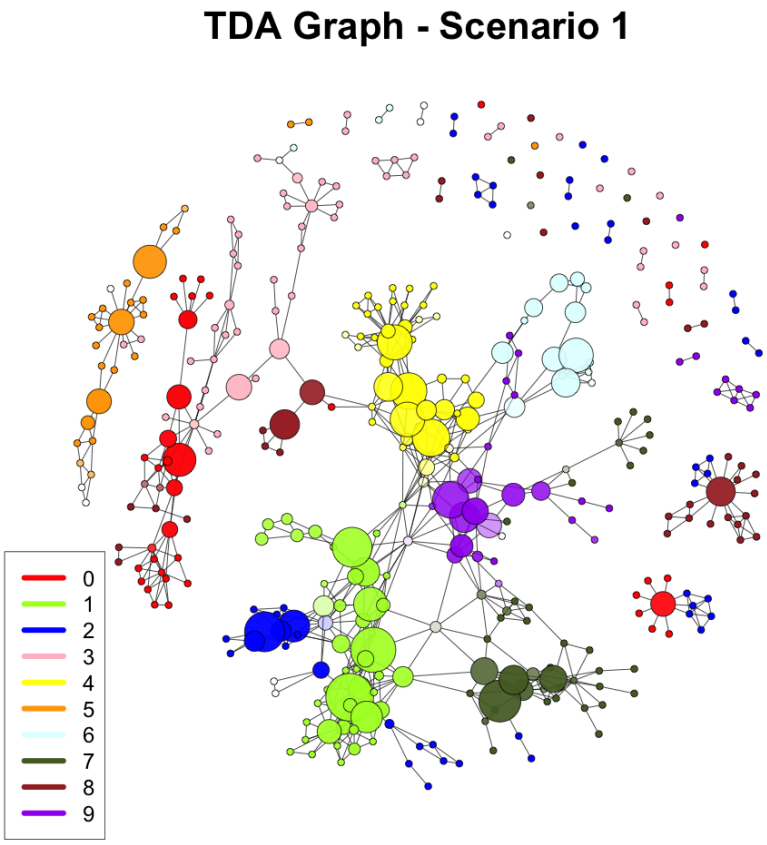


Figure 6.1: Resulting TDA graph. The filtered space was divided into a grid of $8 \times 8 \times 8$, with an overlap percentage of 40%.

6.1.2 Scenario 2

Configuration:

- **Data Representation:** $\mathbb{RP}^{500}$
- **Distance Matrix:** Complete distance matrix
- **Filter Function:** t-SNE

	Value	0	1	2	3	4	5	6	7	8	9	Avg.
1	Purity (%)	96.5	93.4	94.9	95.7	86.5	96.2	95.3	88.9	95.5	83.8	92.7
2	Groupness	20.1	35.6	24.6	25.4	27.6	23	23.4	25.6	26	29.8	26.1

Components: **14** Edge Connectivity: **1.21** Vertex Connectivity: **1.21**

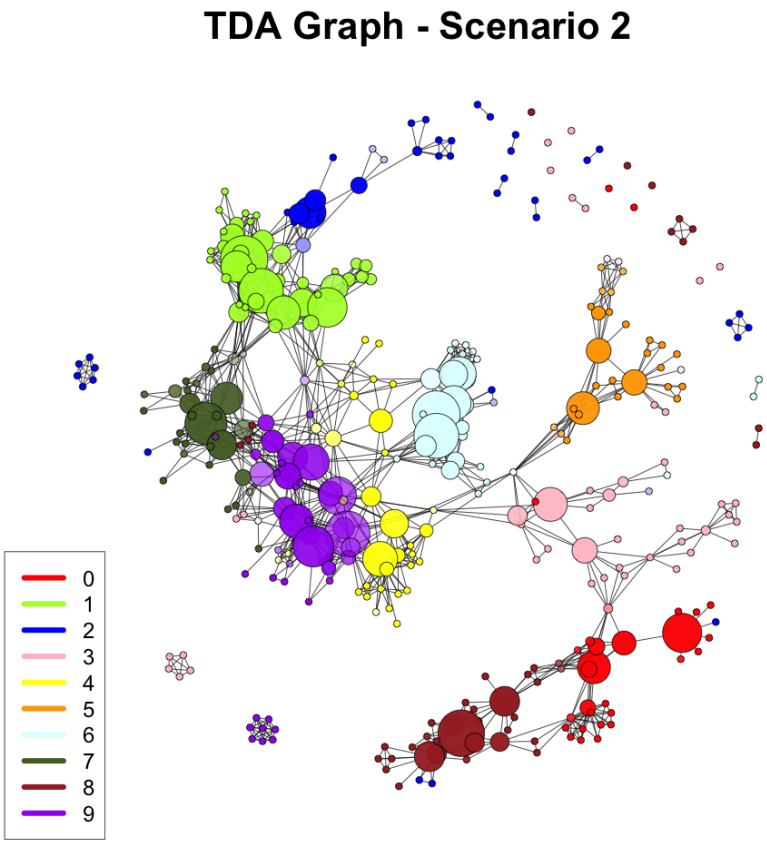


Figure 6.2: Resulting TDA graph. The filtered space was divided into a grid of $8 \times 8 \times 8$, with an overlap percentage of 40%.

6.1.3 Scenario 3

Configuration:

- **Data Representation:** $\mathbb{RP}^{500}$
- **Distance Matrix:** Lower Approximation
- **Filter Function:** PCA

	Value	0	1	2	3	4	5	6	7	8	9	Avg.
1	Purity (%)	99.5	90.6	94.2	94.3	81.8	95.4	97.6	78.3	95.8	82.2	91
2	Groupness	12.2	14	13.2	13.3	12.7	11.4	12.7	12.9	12.4	12.4	12.7

Components: **33** Edge Connectivity: **1.21** Vertex Connectivity: **1.21**

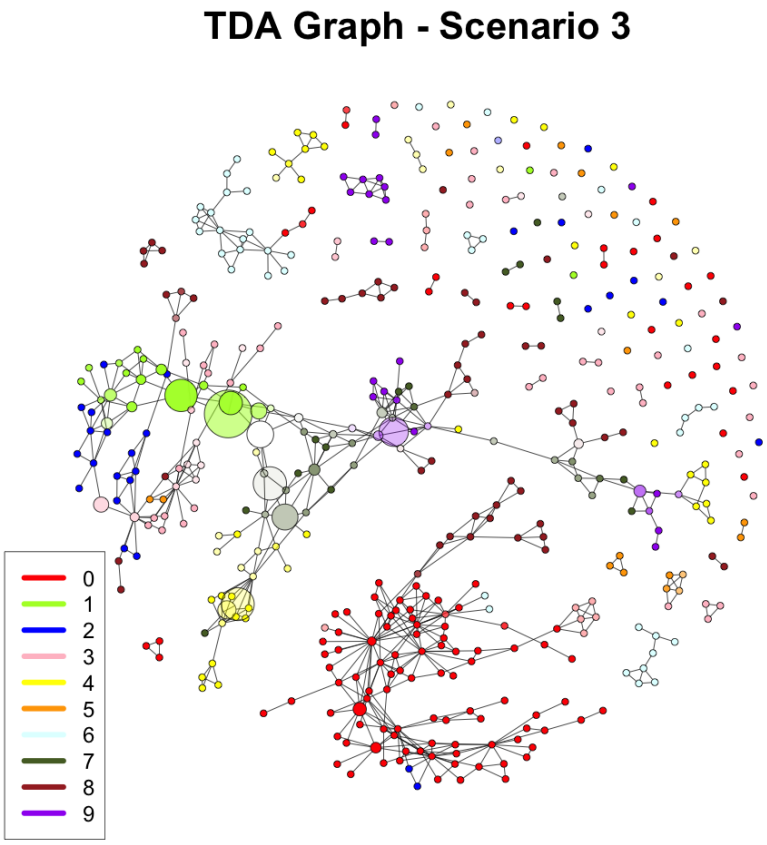


Figure 6.3: Resulting TDA graph. The filtered space was divided into a grid of $8 \times 8 \times 8$, with an overlap percentage of 40%.

6.1.4 Scenario 4

Configuration:

- **Data Representation:** $\mathbb{RP}^{500}$
- **Distance Matrix:** Complete distance matrix
- **Filter Function:** PCA

	Value	0	1	2	3	4	5	6	7	8	9	Avg.
1	Purity (%)	99.1	90.7	94	93.5	86.4	96.4	97.1	81.5	94.8	85.7	91.9
2	Groupness	12.1	13.8	13	13.1	12.5	11.3	12.5	12.7	12.3	12.3	12.6

Components: **25** Edge Connectivity: **1.52** Vertex Connectivity: **1.48**

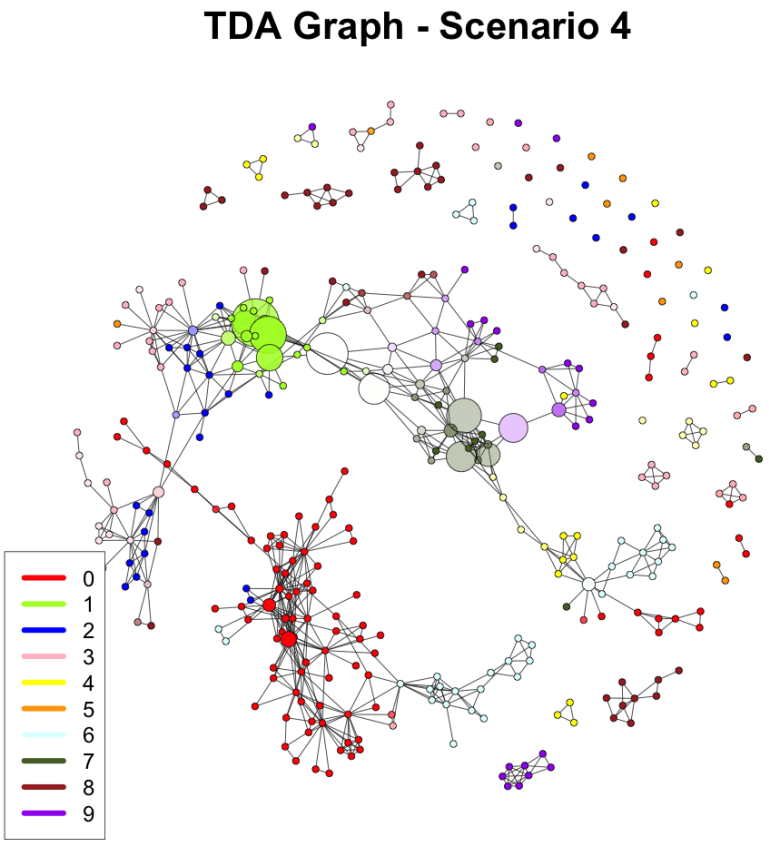


Figure 6.4: Resulting TDA graph. The filtered space was divided into a grid of $8 \times 8 \times 8$, with an overlap percentage of 40%.

6.1.5 Scenario 5

Configuration:

- **Data Representation:** $\mathbb{R}^{784}$
- **Distance Matrix:** Lower approximation
- **Filter Function:** t-SNE

	Value	0	1	2	3	4	5	6	7	8	9	Avg.
1	Purity (%)	93.8	91.5	87.2	86.2	83.3	84.9	93.4	84.6	87.1	79.6	87.2
2	Groupness	295.8	522.9	362	372.5	405.9	337.7	344.1	376.5	382.2	437.5	383.7

Components: **1** Edge Connectivity: **1** Vertex Connectivity: **1**

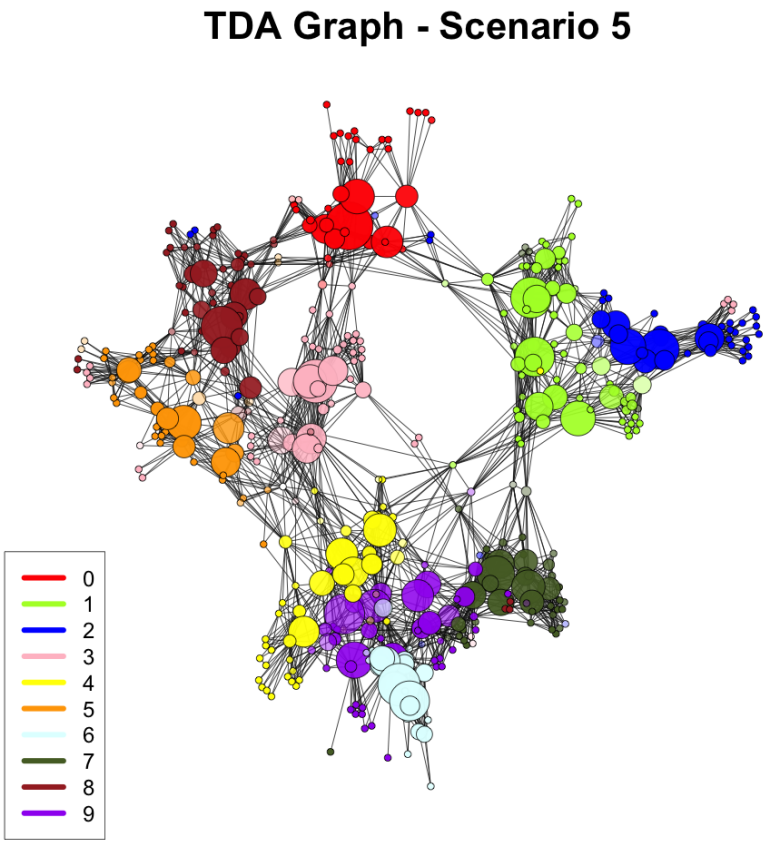


Figure 6.5: Resulting TDA graph. The filtered space was divided into a grid of $8 \times 8 \times 8$, with an overlap percentage of 40%.

6.1.6 Scenario 6

Configuration:

- **Data Representation:** $\mathbb{R}^{784}$
- **Distance Matrix:** Complete distance matrix
- **Filter Function:** t-SNE

	Value	0	1	2	3	4	5	6	7	8	9	Avg.
1	Purity (%)	93.8	91.3	86.8	86.3	83.2	84.4	93.1	83.9	86.6	79	86.8
2	Groupness	629.7	1113	770.4	792.8	863.8	718.8	732.4	801.3	813.4	931.2	816.7

Components: **1** Edge Connectivity: **1** Vertex Connectivity: **1**

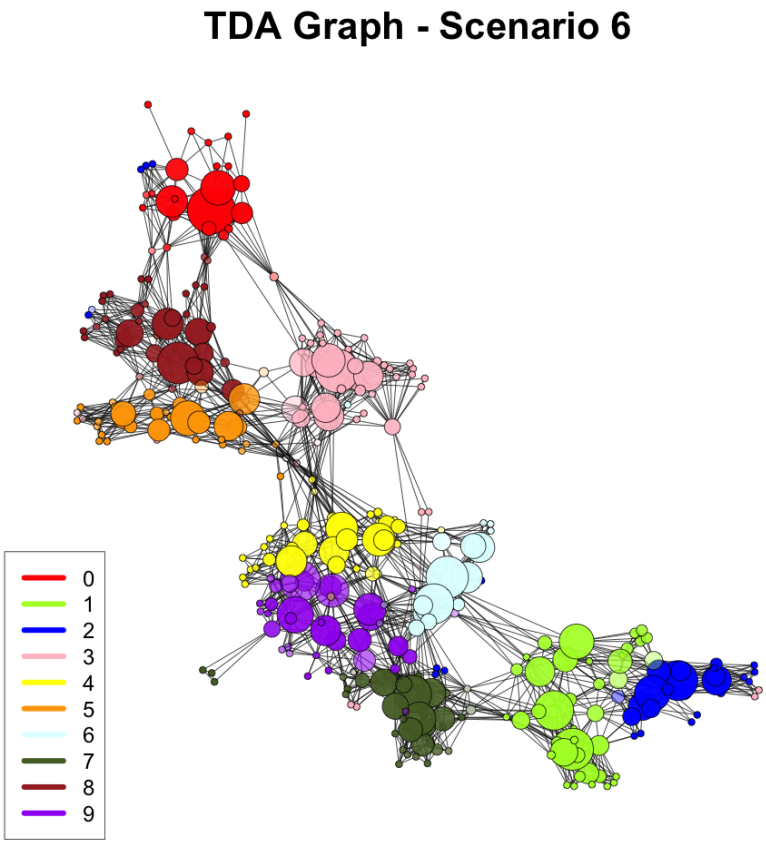


Figure 6.6: Resulting TDA graph. The filtered space was divided into a grid of $8 \times 8 \times 8$, with an overlap percentage of 40%.

6.1.7 Scenario 7

Configuration:

- **Data Representation:** $\mathbb{R}^{784}$
- **Distance Matrix:** Lower approximation
- **Filter Function:** PCA

	Value	0	1	2	3	4	5	6	7	8	9	Avg.
1	Purity (%)	69.8	71.8	24.3	37	32.8	20.9	32.6	33.8	33.2	34.4	39.1
2	Groupness	236.4	270.4	254.5	256.9	246.1	221.2	244.9	249.8	240.4	240.7	246.1

Components: **1** Edge Connectivity: **1** Vertex Connectivity: **1**

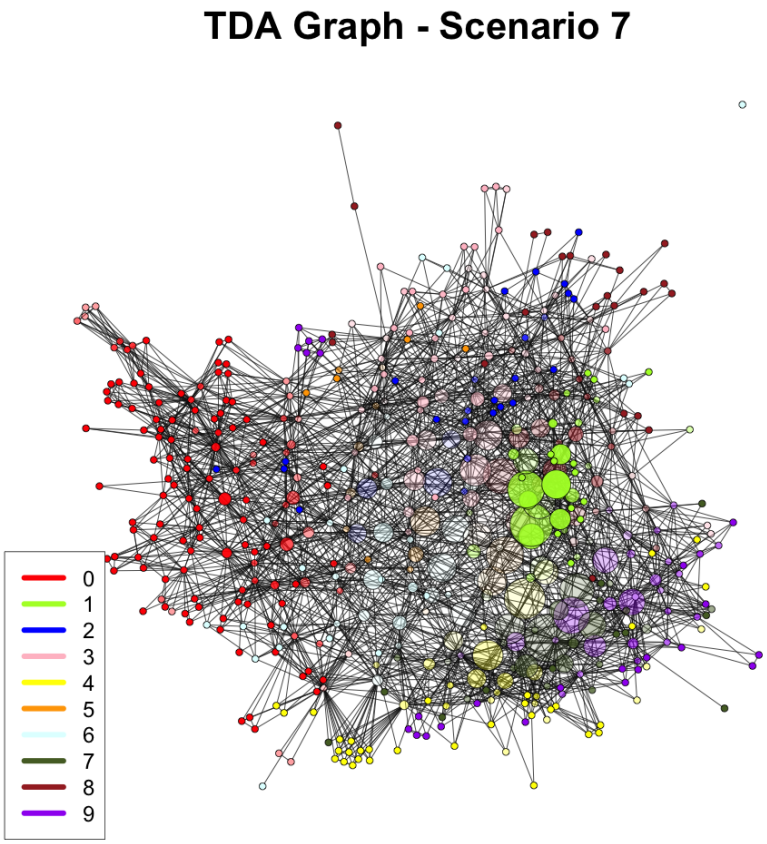


Figure 6.7: Resulting TDA graph. The filtered space was divided into a grid of $8 \times 8 \times 8$, with an overlap percentage of 40%.

6.1.8 Scenario 8

Configuration:

- **Data Representation:** $\mathbb{R}^{784}$
- **Distance Matrix:** Complete distance matrix
- **Filter Function:** PCA

	Value	0	1	2	3	4	5	6	7	8	9	Avg.
1	Purity (%)	68.1	71.8	23.6	36.1	30.3	19.9	30.1	32.7	31.7	32.6	37.7
2	Groupness	627.5	717.6	675.6	681.9	653.1	587.1	650	663	637.9	638.8	653.2

Components: **1** Edge Connectivity: **1** Vertex Connectivity: **1**

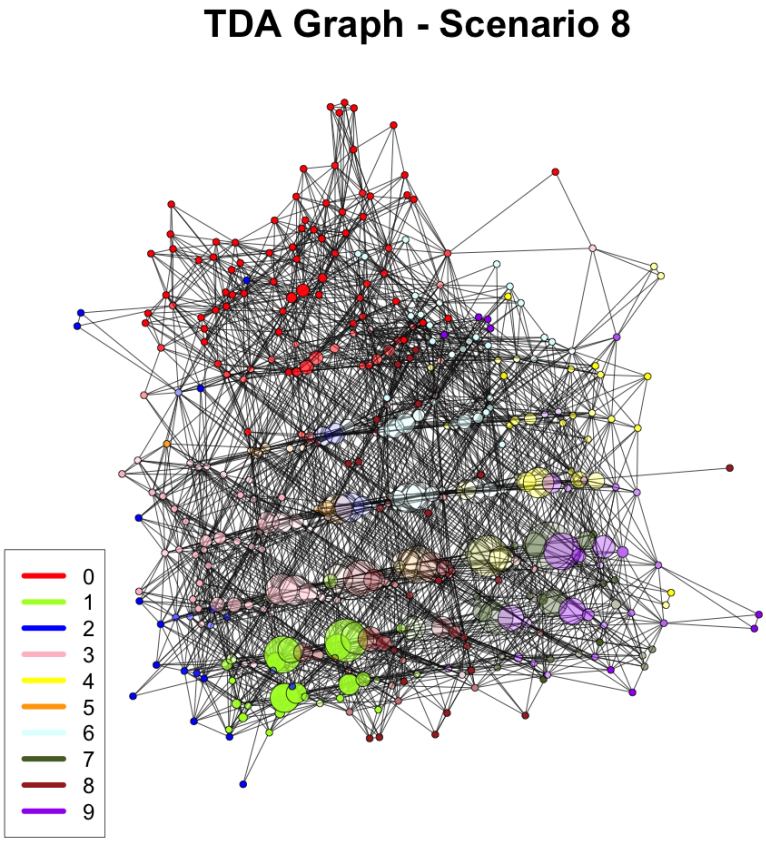


Figure 6.8: Resulting TDA graph. The filtered space was divided into a grid of $8 \times 8 \times 8$, with an overlap percentage of 40%.

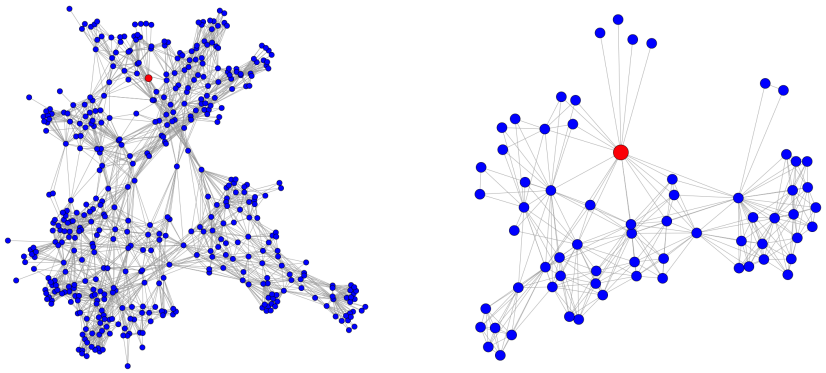
6.2 Results Summary

Table 6.1: Consolidated results for the different scenarios.

Representation	Filter Type	Distance Matrix	Scenario	Comp.	Edge Con.	Vertex Con.	Avg. Purity (%)	Groupness
$\mathbb{RP}^{500}$	t-SNE	Approx.	1	24	1.17	1.12	87.3	20.8
		Complete	2	14	2.21	2.21	92.7	26.1
	PCA	Approx.	3	33	1.21	1.21	91	12.7
		Complete	4	25	1.52	1.48	91.9	12.6
$\mathbb{R}^{784}$	T-SNE	Approx.	5	1	1	1	87.2	383.7
		Complete	6	1	1	1	86.2	816.7
	t-SNE	Approx.	7	1	1	1	39.1	246.1
		Complete	8	1	1	1	37.7	653.2

From the previous table and the resulting plots, some immediate things come to our attention:

- One is immediately disappointed by the connectivity values. Aside from the amount of components that gives us strong differences among the results, the edge and vertex connectivity are almost all practically 1 (except in the second scenario where it's 2). One can even be tempted to doubt these results (specially for scenarios 5 or 6 where difficult to believe that removing a single node will disconnect it, but a closer look into the layout of the graphs shows what is really happening:



(a) Layout of the graph from scenario 5.

(b) Corresponding zoomed layout

Figure 6.9: Layout of the graph from scenario 5. Although visually very dense in connections, it's vertex connectivity is still one.

In the previous scenario, although common in the obtained graphs from *Mapper*, one clearly does not obtain much insight from noticing that re-

moving the red node gives us four isolated vertices and a massive cluster. But this is not always the case. Take another look at the graph from scenario 2, this time focusing only on the layout of the main connected component:

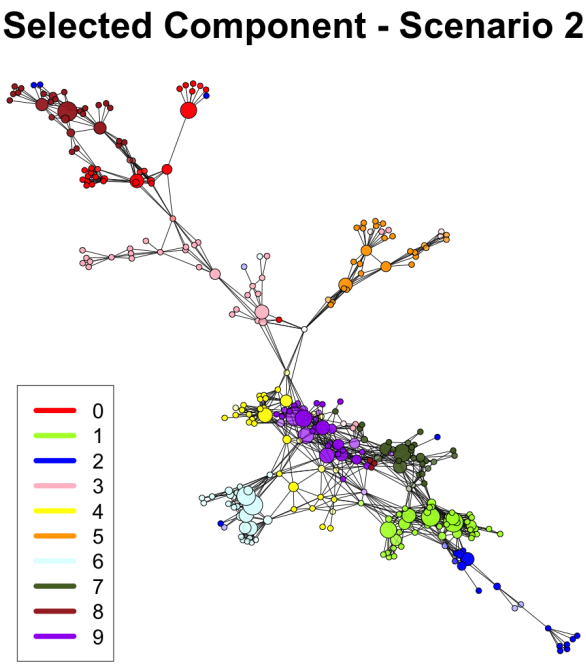


Figure 6.10: Principal connected component from the obtained graph in scenario 2

Notice how we have a "long" graph where several nodes have the condition that if removed, the resulting graph will no longer be connected, thus the subgraph has vertex connectivity equal to 1. But this time, the resulting subgraphs aren't isolated nodes and still have some structure This type of layout is known as **Flares** [Lum+13], and can give us insight above the data as we will see in the following section.

- The most evident impact the representation has is in the groupness of the graph. In the $\mathbb{R}^{784}$ representation, the average groupness is 523.2, whereas in the $\mathbb{RP}^{500}$ it is only 17.5. This means that the latter representation, together with the $\mathbf{d}_{\mathbb{RP}}$ distance, is a lot sparser than its original $\mathbb{R}^{784}$ version. This is also seen through the number of connected components the representations yield: $\mathbb{R}^{784}$ always computed connected graphs, against that $\mathbb{RP}^{500}$ that had an average of 25 connected components per graph.

- Even though the $\mathbb{RP}^{500}$ coordinates of the MNIST test set gave us a sparse

representation sacrificing groupness, it's still a good representation in the sense that it clustered labelled elements together. The average purity of this representation is 90.5% vs the 62.3% of the $\mathbb{R}^{784}$ representation. Even more, the high purity values of scenarios 5 and 6 are probably due to the filter function used, since the t-SNE filtration (that already separates the MNIST test set into labelled clusters: figure 3.2) doubles the purity of its PCA counterpart.

- The distance approximation works. We obtained very similar graphs between using the lower approximation of the distance and the complete distance. The main difference between the approximation and complete scheme is in terms of groupness, since the complete use of the metric yields larger values[1], than the approximate procedure. Recall that we used the lower approximation that assigns a fixed value between elements in different buckets. This value can be higher than the actual value between the two elements, causing them to belong to different nodes in the approximation scheme, lowering the value for the groupness.

6.3 Mapper applied to MNIST

The accuracy in classifying MNIST is almost perfect for specific solutions[2], so there is little that the *Mapper* can do in that regard. However, this algorithm serves as exploration tool for a given data set and it's in that aspect that it has something to offer to MNIST and other real world databases. Ultimately, *Mapper* gives us interaction between clusters of elements and it depends on the context what this interaction means. Like in figure 2.9, where interactions meant synchronicity among the malaria outbreaks for different municipalities across Colombia. In the MINST context, interaction between nodes enable us to find digits that can be confused for another type, for instance in figure 6.6 from scenario 6. We can identify the clusters with the corresponding digit, creating the following flare:

[1] Double in the $\mathbb{R}^{784}$ representation

[2] The current record is held by Dan Ciresan, Ueli Meier and Jurgen Schmidhuber with an accuracy of 99.77% on the test set [CMS12]

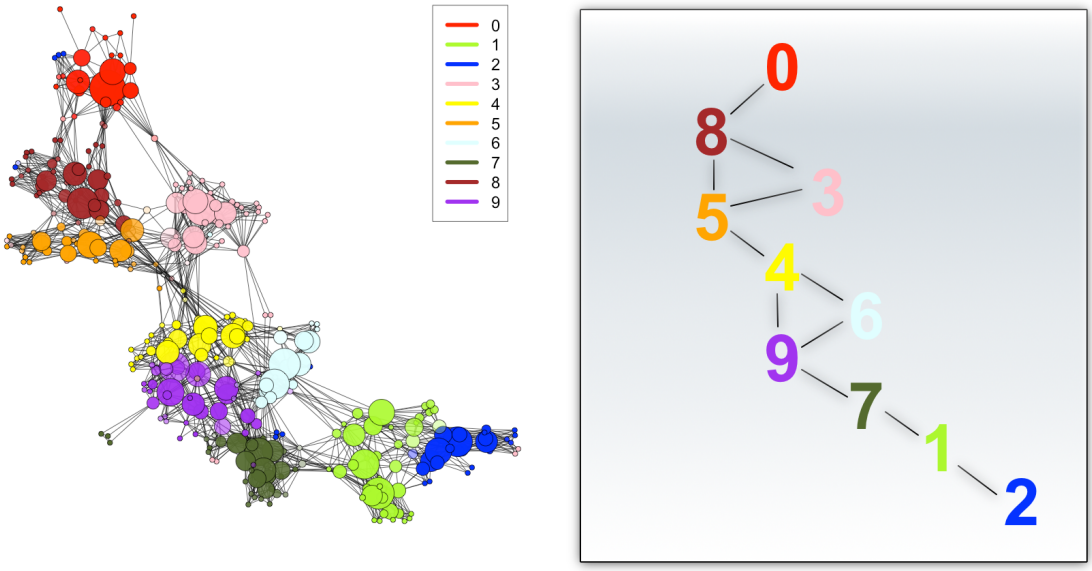
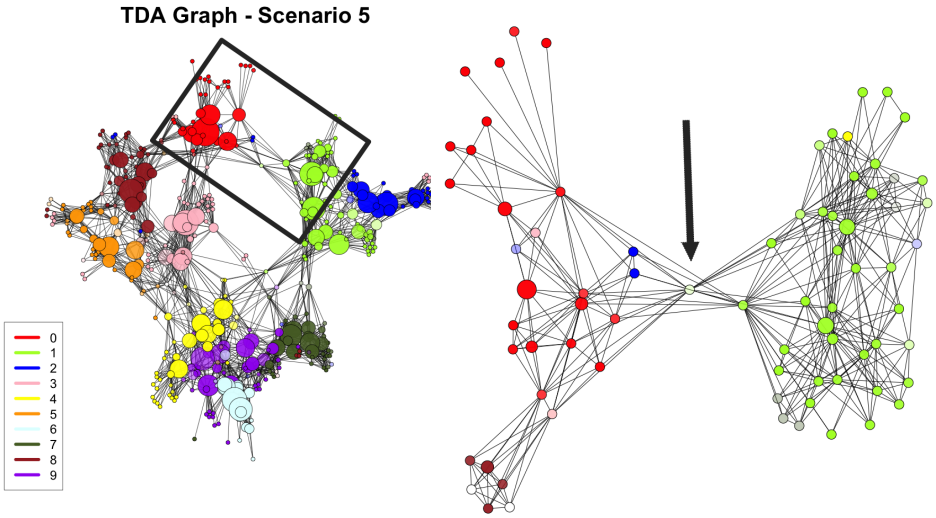


Figure 6.11: Graph from scenario 6 and interaction between predominant digit clusters.

On the right we have a flare of the corresponding digits, seeing how they can be clustered together and the way they interact. It comes to our attention how digits that can be confused with one another are together and progressively turn from one type to another, following the pattern:

$$2 \longrightarrow 1 \longrightarrow 7 \longrightarrow 9 \longrightarrow \{4,6\} \longrightarrow 5 \longrightarrow \{3,8\} \longrightarrow 0$$

As we mentioned before, low vertex connectivity can in some cases lead to insight about the data. A specific example can be seen in scenario 5.



(a) Output graph from scenario 5. (b) Corresponding zoomed subgraph.

Figure 6.12: Resulting graph from scenario 5. Here, notice the pale green dot connecting three different colored clusters.

Notice how there is a small pale green node at the top of the graph connecting the red (digit 0), green (digit 1) clusters with some isolated blue (digit 2) nodes. A closer look inside this node and it's neighbor reveals the elements responsible for this bridge between the digits $\{0, 1, 2\}$:

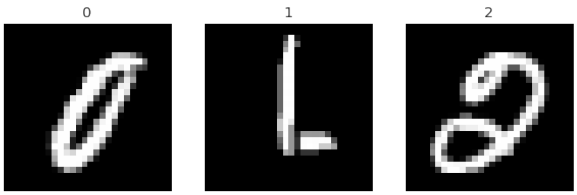


Figure 6.13: Selected digits from scenario 5.

a bridge that exists because of the potential ambiguity between the labels for these specific elements. This same argument lets us find the bridge between the digits 0 and 3 in the main connected component from scenario 2 (figure 6.10), looking at the node between the red and pink clusters:

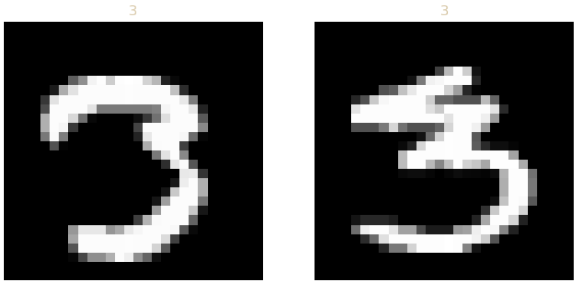


Figure 6.14: Selected digits from scenario 2.

Clearly, these elements where found thanks to the colored clusters in the graph and although this specific "pathological" digits can be found via any classification procedure (simply check for wrongly classified digits and it's very probable that this specific examples will arise) what would happen if we had unlabelled data?

Here is an advantage that *Mapper* has over other tools. Since the ultimate result of the procedure is a graph, we can use specific graph theory tools to further explore the data. Take for instance the Betweeness of a node:

Definition 20. Given a graph $g = G(V, E)$ the **Betweenness** of a vertex $v \in V$ is the amount of shortest path between all elements of V that cross through v.

Notice that the visually selected nodes from figures 6.10 and 6.12 have high betweenness. Hence, it makes sense to select nodes with high values of betweenness to search for this unique digits. Take for example the graph from scenario 6 and color the nodes depending on their betweenness value.

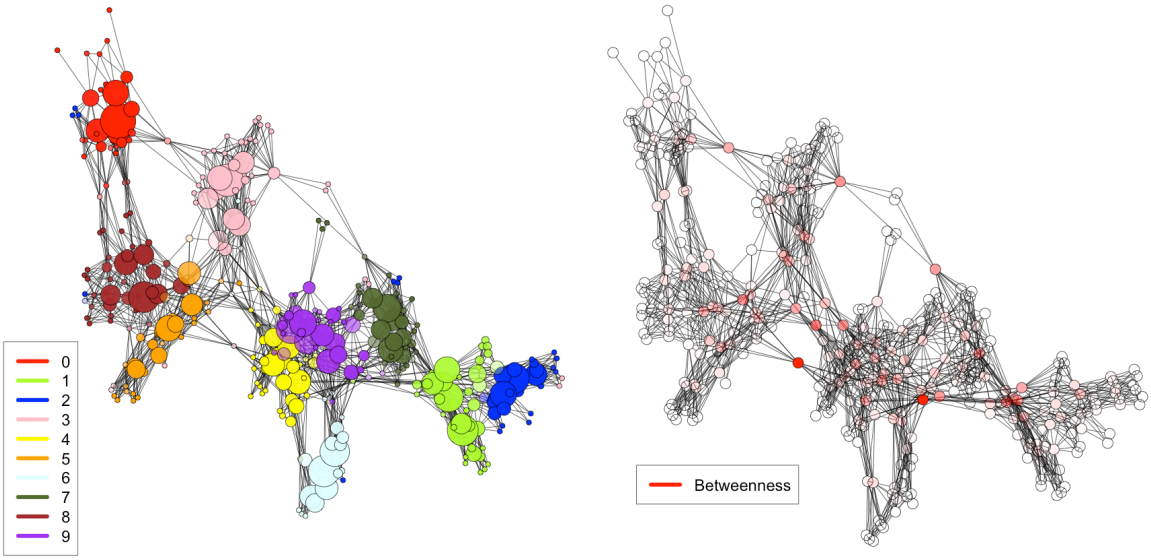


Figure 6.15: Obtained clusters from scenario 6 and its layout with coloring based on the betweenness of the node. Notice that the nodes with intense color unite several clustered of nodes.

The node with highest betweenness (the red one on the left) contains a single element:

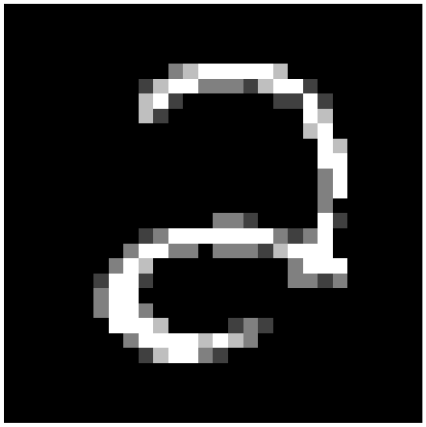


Figure 6.16: Element inside the node with highest betweenness value in scenario 6. Although labelled as a 2, this element is responsible for the union of the 3,4 and 5 digit clusters

Clearly an element than any human would have trouble labelling using only

the digits from 0 to 9 (since is looks like an a). Notice that this element was found without the use of the labels of the data.

We can further investigate the results via graph clustering. For example, we used the community detection algorithm developed by Aaron Clauset, M.E.J. Newman and Cristopher Moore [CNM04] and applied it to the obtained graph from scenario 6:

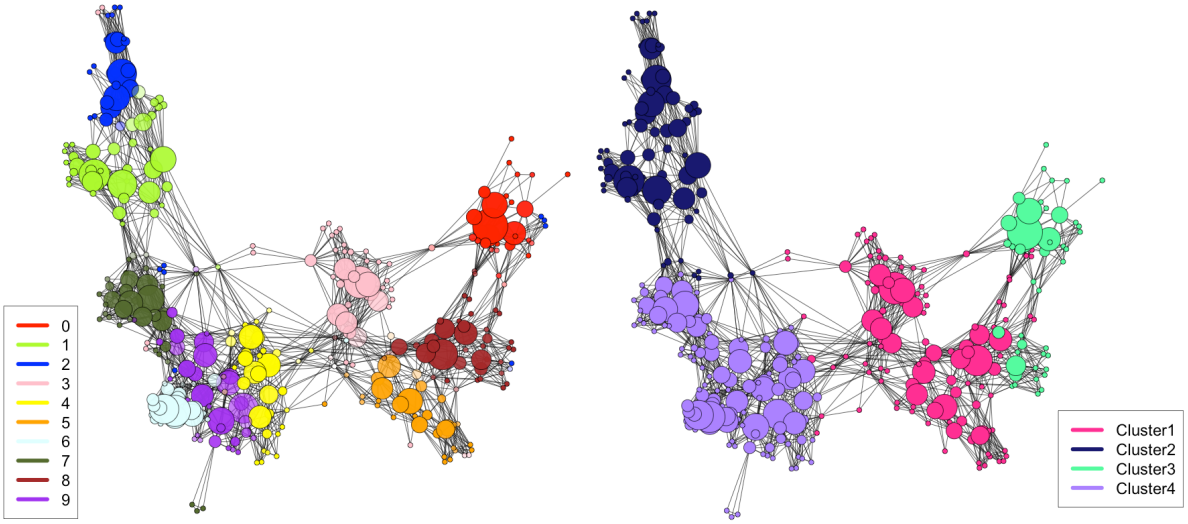


Figure 6.17: Obtained clusters from scenario 6.

We can see that the cluster analysis identifies 4 major groups:

$$\{1,2\} \quad \{7,9,4,6\} \quad \{5,3,8\} \quad \{0\}$$

Grouping digits that are similar in the way they are drawn and in turn, might be confused with one another.

Recall that our library has the limitation that it only calculates double intersections, creating only 1-simplices (arches) between nodes. But we can potentially check for more numerous intersections, giving us simplices of larger dimensions. This means that we can explore our data with more complex interactions among the elements, giving *Mapper* even more potential for data exploration.

Chapter 7

Improving the $\mathbb{RP}^\infty$ Representation on MNIST (Attempts)

As we saw in section 4.5, because of the lack of dominant bars in the resulting barcode, we where forced to use all of them in the construction of the $\mathbb{RP}^\infty$ representation. We now mention some other procedures we followed in order to improve the obtained barcodes and int turn, the cocycle representatives used to calculate the function to $\mathbb{RP}^\infty$.

7.1 Sliding Windows

One approach was to select smaller patches of images, in the hopes of capturing more significant features. We sampled a total of 49 patches from each 28 by 28 pixel image, following the procedure:

1. Add a padding of black pixels (zero value entries) around the image, to make it an 30×30 image.
2. Start at the top left corner and extract the 6×6 image at this position.
3. Move horizontally 4 pixels and repeat extract the corresponding 6×6. That is, make adjacent windows to overlap in 2 pixels.
4. Repeat this process until reaching the other end of the image. Return to the left side and go down 4 pixels to start again horizontally.

We are shifting a "window" of 6×6 across the original image with a 4 pixel stride. Extracting fixed sized patches with a certain stride along the entire image is a common scheme when analyzing images through convolutional neural

networks [Kar17]. Thus, we can concatenate the resulting patches for each image and take it as a vector in $\mathbb{R}^{49\times6\times6} = \mathbb{R}^{1764}$.

Here is an example of how the sliding window extracts features of a digit:

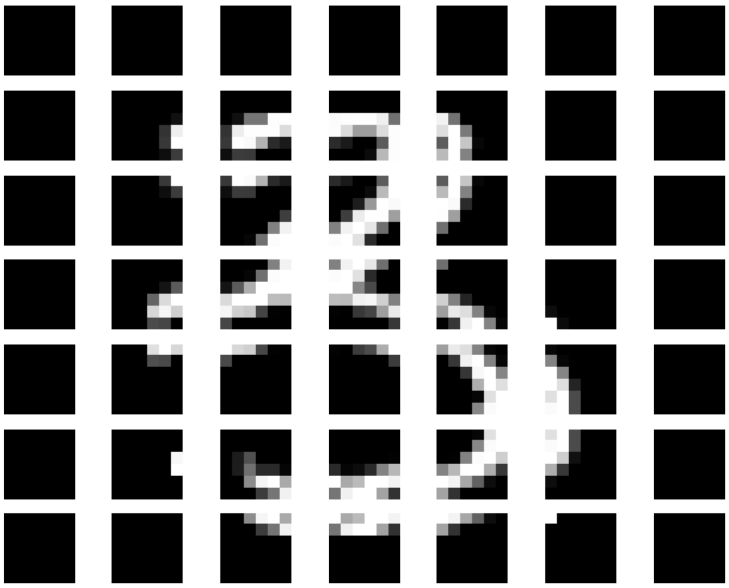


Figure 7.1: The 49 patches of dimensions 6×6 obtained from an image of the digit 3 by a sliding window.

To this new sample of 10,000 elements in $\mathbb{R}^{1764}$, we applied the procedure outlines in chapter 4, and obtained the following barcodes from a sparse sample of 750 points.

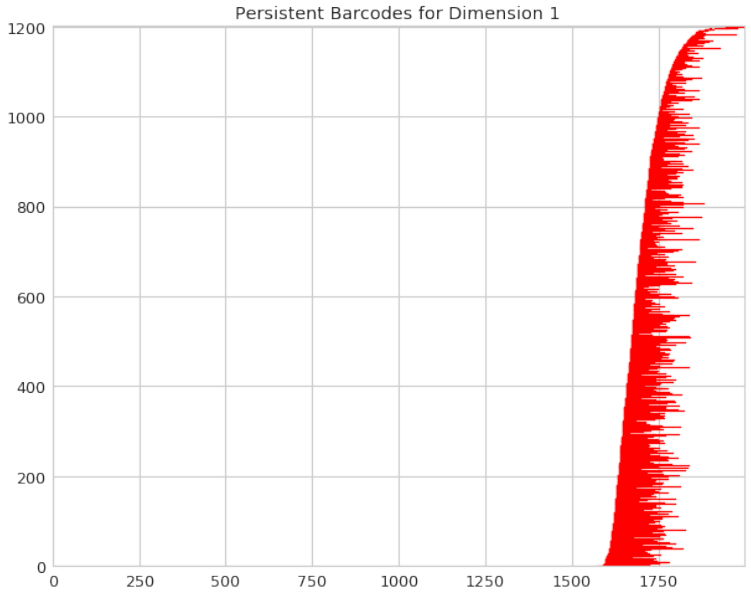


Figure 7.2: Corresponding barcode for dimension 1 of the $\mathbb{R}^{1764}$ representation with 750 elements.

A barcode very similar to the original one in figure 4.7.

7.2 Fixed Window Sub-sampling

Maybe the problem with our previous approach was that we where mixing to many features together. One can imagine that the sliding window at a certain step might extract images that are similar for different digits (like in the center of digits 7 and 1), but we where loosing this advantage when we concatenated them together. Hence, our next procedure focused on a fixed window and sampled only elements that where extracted from the same location of the image.

Recall that our images have a 28×28 dimension. We fixed a window of size 7×7 and extracted the 16 images corresponding to the 4×4 grid formed by them in each original image. Each image in turn, was divided like this:

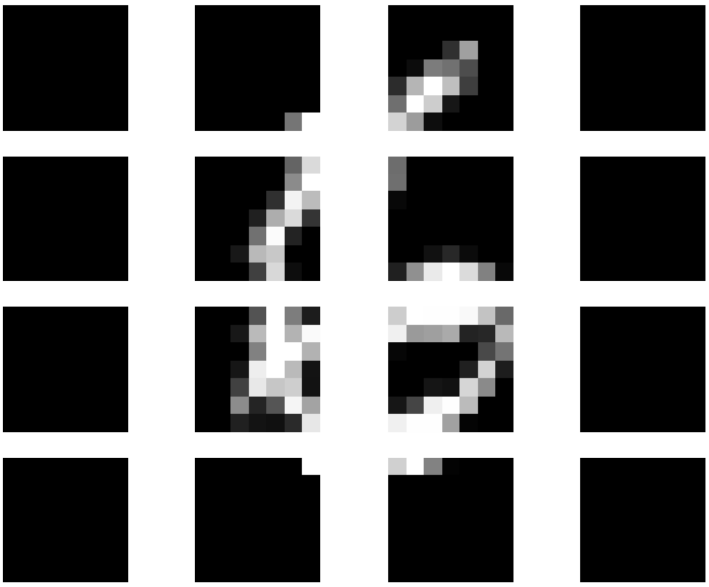


Figure 7.3: The 16 patches of dimensions 7×7 obtained from an image of the digit 6 obtained by a fixed window

Thus, for each element $x_i \in \mathbb{R}^{784}$, we obtained the collection of patches $\{y_1^i, \ldots, y_{16}^i\}$. Then we have the 16 sets:

$$X_j = \{y_j^1, y_j^2, \ldots y_j^{10000}\} \subset \mathbb{R}^{49}$$

To each one of these 16 new sets, we applied the procedure outlined in chapter 4.

Theoretically we obtaining, for each X_j a representation of in $\mathbb{RP}^\infty$. Hence, after executing the procedure 16 times, we obtain a representation of the MNIST test set inside:

$$\underbrace{\mathbb{RP}^\infty \times \cdots \times \mathbb{RP}^\infty}_{16 \text{ times}}$$

The following image shows the 16 barcodes obtained from this procedure, positioned at the location of its corresponding patch.

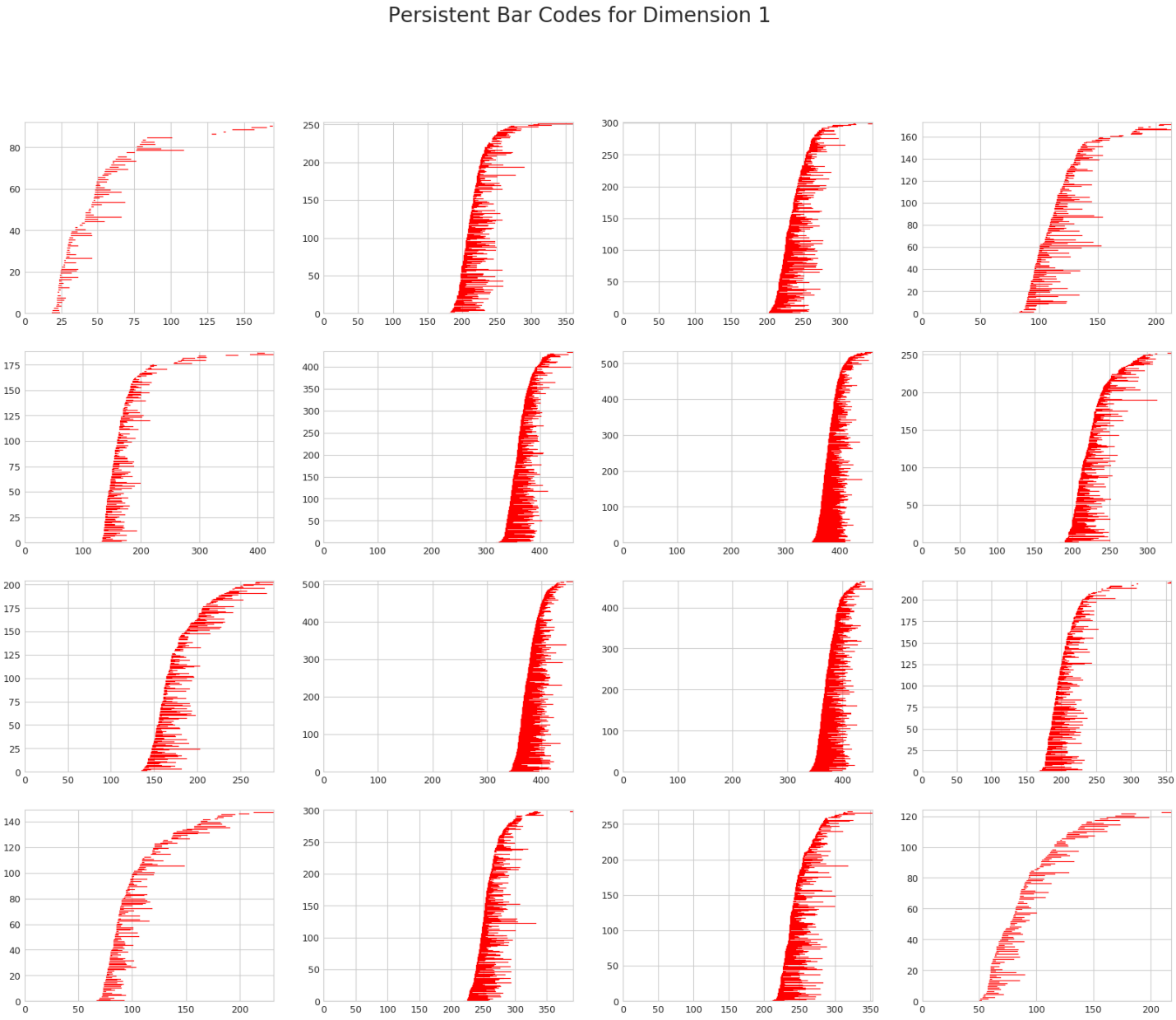


Figure 7.4: Corresponding barcodes for dimension 1 of the $\mathbb{R}^{49} \times \ldots \mathbb{R}^{49}$ representation with 300 elements. Barcodes are positioned at the location of their corresponding patch.

There is still no sight of a persistent lines in any of the barcodes obtained.

Chapter 8

Conclusions and Future Work

We have successfully applied the *Mapper* algorithm to the MNIST test set, using the multi-scale projective coordinates as our representation, approximating the distance $\mathbf{d}_{\mathbb{RP}}$ in $\mathbb{RP}^n$ using a corresponding locality-senstive hash function and filtered the data using the three dimensional embedding provided by t-SNE. We compared this adjustments with more usual methods, obtaining several graphs that together with some numeric quality values, gave us some insight about the MNIST test set.

Although the final barcode in figure 4.7 did not show any predominant lines, the representation of MNIST in $\mathbb{RP}^\infty$ still gave us the following advantages:

- Isolated some groups of labeled elements together. The choice of filter function had little effect on the quality of the obtained graph when the data was taken in its $\mathbb{RP}^\infty$ representation, a very different result from what was obtained when the data was in its $\mathbb{R}^{784}$ form. When the MNIST test set was in its original form, the quality of the result was entirely dependant on the filter function used (to the point where PCA rendered the obtained graph completely uninterpretable) leading us to believe that this representation together with the choice of metric was not distinguishing very well between labels. At least for this specific example, the representation of the MNIST test set in $\mathbb{RP}^\infty$ made the *Mapper* algorithm somewhat robust against the choice in metric approximation and filter.
- The specific representation also gave us a minor dimensionality reduction, making the distance computations and approximations easier.

Nevertheless, even with greedy permutations and sparse filtrations, persistent cohomology is still an expensive procedure to execute making the computation of this representation tricky to adjust to the complete MNIST set. There is still the lingering question of how many elements where needed to find a complete cover of the set and if this selection will have a notorious impact on the

quality of the resulting barcode.

On the other hand, approximating the distance via LSH is a feasible technique to adapt *Mapper* to larger data sets. Even though we obtained somewhat different results from using the approximate metric, the output graphs still preserves much of the structure seen in the complete metric case, making the approximation of distances via locality-sensitive hash functions a viable procedure to reduce the computational cost of *Mapper*.

Finally, t-SNE proved to be a very good candidate function. Regardless on the little impact in had when working in $\mathbb{RP}^{500}$, it made the original representation of the MNIST work as an input to the *Mapper* algorithm. The fact that t-SNE somewhat clustered the labeled elements together enabled this procedure to produce good quality results regardless of the metric, a quality much appreciated when working with fresh and unlabeled data.

Although still little bit obscure, *Mapper* is a powerful tool for data exploration. The few restrictions it imposes on its inputs makes this algorithm suitable for any real world data, giving us a first step in understanding and visualizing the information given.

8.1 Future Work

There are several important items left unexplored by this work:

- The only input in the *Mapper* procedure we did not tweak was the clustering algorithm. Understanding how different types of clustering algorithms affects the outcome graph, can give us some insight in the selection of this parameter. Furthermore, recall that LSH was included as a way to approximate the metric in order to make *Mapper* more scalable. Well, LSH can also be used to improve the performance on hierarchical clustering, so it would also be interesting to see how this approximation affects *Mapper's* performance [KIW07] [CM15].

- Notice that the filter function was calculated over the $\mathbb{R}^{784}$ representation, regardless of whether the images where inside $\mathbb{R}^{784}$ or $\mathbb{RP}^{500}$. José A Perea already developed a scheme to preform PCA in $\mathbb{RP}^n$ [Per16], so it would be interesting to find a way to adapt t-SNE to this new space and repeat the procedures with the filter functions calculated over the used representation.

- Although we included some numeric quantities to asses the quality of the outcome graph, we still relied strongly on the visualization of the result in order to asses the procedure's performance. It is essential to develop a way to identify "good" graphs abstractly, in order to search the metaparameter space computationally without any visualization aid.

Although the convergence of the resulting 1-simplex to the corresponding reeb graph has been studied [CMO17], it's still not clear what exactly composes a "good" graph in topological data analysis.

Bibliography

[And+15] Alexandr Andoni et al. "Practical and optimal LSH for angular distance". In: *Advances in Neural Information Processing Systems*. 2015, pp. 1225–1233.

[Ble08] David M Blei. "Hierarchical clustering". In: (2008).

[CJS15] Nicholas J Cavanna, Mahmoodreza Jahanseir, and Donald R Sheehy. "A geometric perspective on sparse filtrations". In: *arXiv preprint arXiv:1506.03797* (2015).

[CM15] Michael Cochez and Hao Mou. "Twister tries: Approximate hierarchical agglomerative clustering for average distance in linear time". In: *Proceedings of the 2015 ACM SIGMOD international conference on Management of data*. ACM. 2015, pp. 505–517.

[CMO17] Mathieu Carrière, Bertrand Michel, and Steve Oudot. "Statistical Analysis and Parameter Selection for Mapper". In: *arXiv preprint arXiv:1706.00204* (2017).

[CMS12] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. "Multi-column deep neural networks for image classification". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 3642–3649.

[CNM04] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. "Finding community structure in very large networks". In: *Physical review E* 70.6 (2004), p. 066111.

[DP13] Ding-Zhu Du and Panos M Pardalos. *Minimax and applications*. Vol. 4. Springer Science & Business Media, 2013.

[DSG07] Vin De Silva and Robert Ghrist. "Coverage in sensor networks via persistent homology". In: *Algebraic & Geometric Topology* 7.1 (2007), pp. 339–358.

[DSMVJ11] Vin De Silva, Dmitriy Morozov, and Mikael Vejdemo-Johansson. "Dualities in persistent (co) homology". In: *Inverse Problems* 27.12 (2011), p. 124003.

[EJM15] Herbert Edelsbrunner, Grzegorz Jabłoński, and Marian Mrozek. "The persistent homology of a self-map". In: *Foundations of Computational Mathematics* 15.5 (2015), pp. 1213–1244.

[GIM+99] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. "Similarity search in high dimensions via hashing". In: *VLDB*. Vol. 99. 6. 1999, pp. 518–529.

[Hat02] Allen Hatcher. *Algebraic topology*. Cambridge University Press, 2002.

[HR03] Geoffrey E Hinton and Sam T Roweis. "Stochastic neighbor embedding". In: *Advances in neural information processing systems*. 2003, pp. 857–864.

[Hun12] No Free Hunch. *t-Distributed Stochastic Neighbor Embedding Wins Merck Viz Challenge*. 2012. URL: `http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm` (visited on 08/18/2017).

[Kar17] Andrej Karpathy. *Convolutional Neural Networks (CNNs / ConvNets)*. 2017. URL: `http://cs231n.github.io/convolutional-networks/#conv` (visited on 10/30/2017).

[Kim+16] Jeonghyun Kim et al. "Battery-free, stretchable optoelectronic systems for wireless optical characterization of the skin". In: *Science advances* 2.8 (2016), e1600418.

[KIW07] Hisashi Koga, Tetsuo Ishibashi, and Toshinori Watanabe. "Fast agglomerative hierarchical clustering algorithm using Locality-Sensitive Hashing". In: *Knowledge and Information Systems* 12.1 (2007), pp. 25–53.

[Kul97] Martin Kulldorff. "A spatial scan statistic". In: *Communications in Statistics-Theory and methods* 26.6 (1997), pp. 1481–1496.

[Le+15] Ngoc-Khuyen Le et al. "Construction of the Generalized Czech Complex". In: *Vehicular Technology Conference (VTC Spring), 2015 IEEE 81st*. IEEE. 2015, pp. 1–5.

[Liu+17] Shusen Liu et al. "Visualizing high-dimensional data: Advances in the past decade". In: *IEEE transactions on visualization and computer graphics* 23.3 (2017), pp. 1249–1268.

[Lum+13] PY Lum et al. "Extracting insights from the shape of complex data using topology". In: *Scientific reports* 3 (2013), p. 1236.

[May99] J Peter May. *A concise course in algebraic topology*. University of Chicago press, 1999.

[MH08] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE". In: *Journal of Machine Learning Research* 9.Nov (2008), pp. 2579–2605.

[MS16] Anshuman Mishra and Lawrence Spracklen. "Enterprise Scale Topological Data Analysis Using Spark". Spark Summit 2016. 2016. URL: `https://spark-summit.org/2016/events/enterprise-scale-topological-data-analysis-using-spark/`.

[Per16] Jose A Perea. "Multi-Scale Projective Coordinates via Persistent Cohomology of Sparse Filtrations". In: *arXiv preprint arXiv:1612.02861* (2016).

[PMS15] Paul Pearson, Daniel Muellner, and Gurjeet Singh. *TDAmapper: Analyze High-Dimensional Data Using Discrete Morse Theory*. R package version 1.0. 2015. URL: `https://CRAN.R-project.org/package=TDAmapper`.

[SMC07] Gurjeet Singh, Facundo Mémoli, and Gunnar E Carlsson. "Topological methods for the analysis of high dimensional data sets and 3d object recognition." In: *SPBG*. 2007, pp. 91–100.

[Spa+75] EH Spanier et al. "John W. Milnor and James D. Stasheff, Characteristic classes". In: *Bulletin of the American Mathematical Society* 81.5 (1975), pp. 862–865.

Appendix

R Shiny Application for Results Exploration

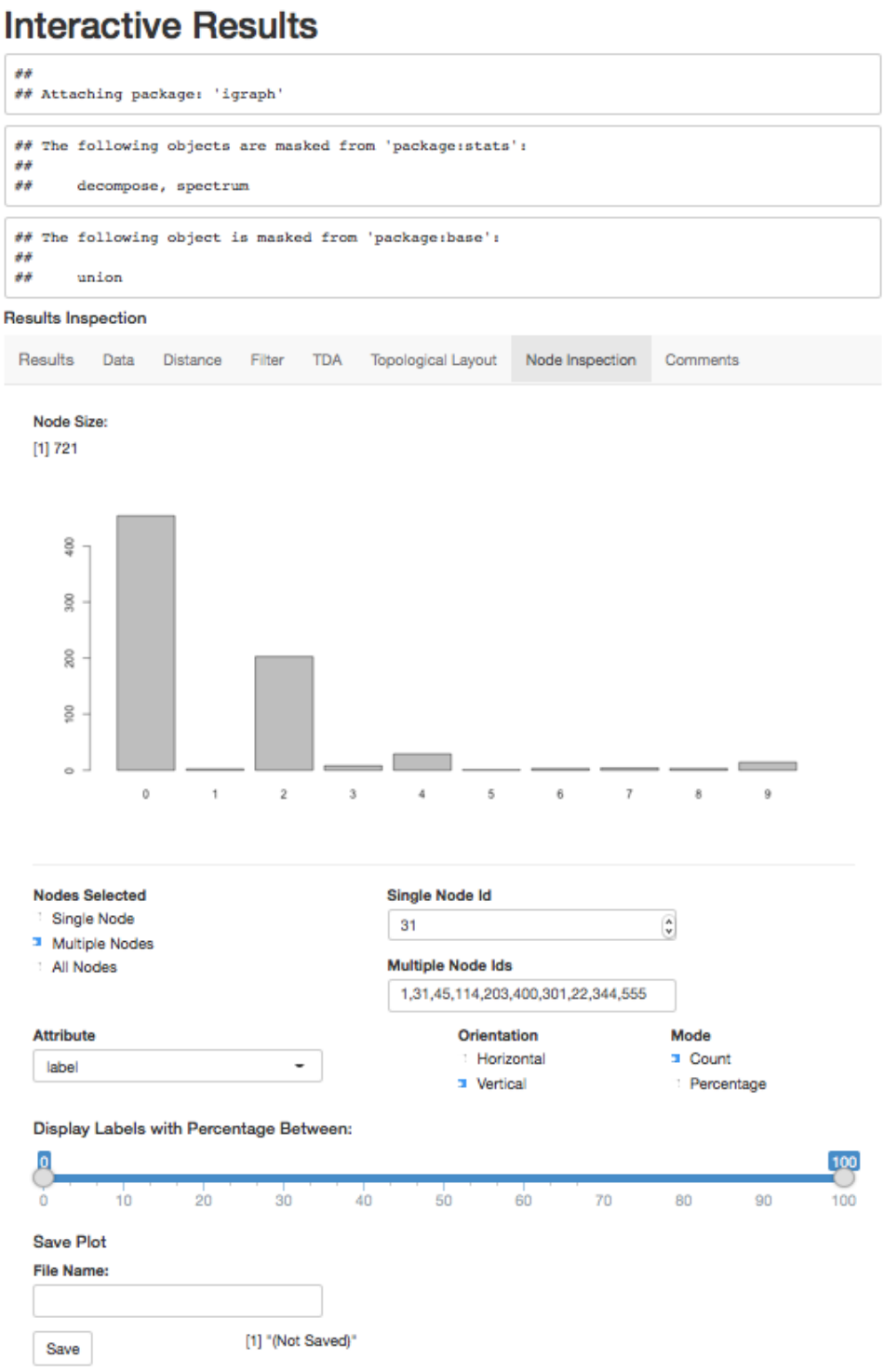


Figure 8.1: Screenshot of the Shiny app's interface created for graphs and result exploration

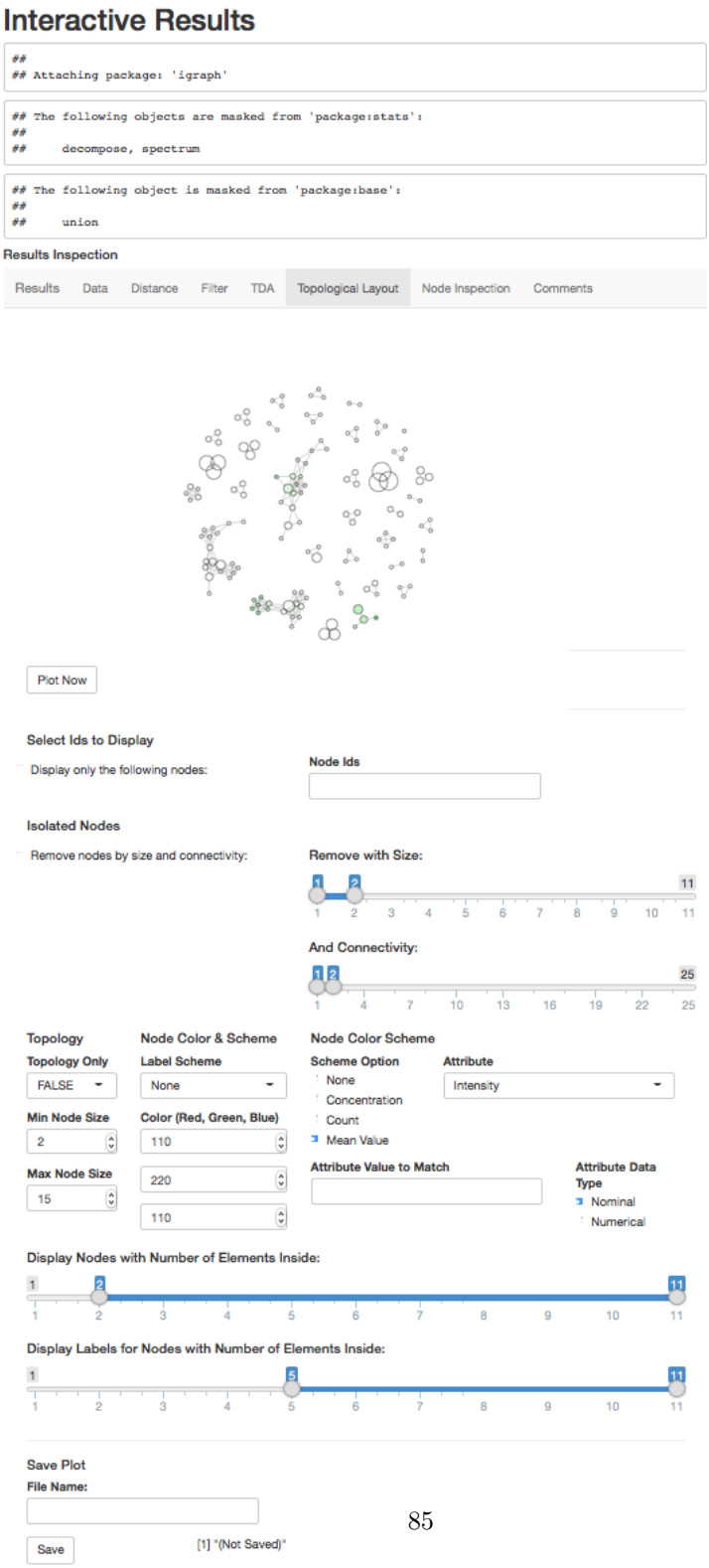


Figure 8.2: Screenshot of the Shiny app's interface created for graphs and result exploration