



Eidgenössische Technische Hochschule Zürich  
Swiss Federal Institute of Technology Zurich

# Topological Clustering Algorithm ToMATo

Bachelor Thesis

Daniel Hostettler

July 19, 2024

Advisors: Dr. Sara Kališnik Hintz

Department of Mathematics, ETH Zürich



## Abstract

In the age of big data, the ability to analyse large and complex data sets has become a critical component of technological advancement and decision-making across many segments of society. One major technique in this context is clustering. This is a method that involves grouping a set of objects such that those within the same group are more similar to each other than to those in other groups. Clustering schemes like *k-means clustering*, *spectral clustering*, *mean shift clustering* and *hierarchical clustering* are some examples of distinct clustering approaches. Recently, topological ideas have been introduced in clustering. One example of a topological clustering algorithm is the *topological mode analysis tool* ToMATo. It is a mode seeking algorithm with a cluster merging phase which uses topological persistence theory to guide merges between clusters. In this thesis, we start by introducing clustering. We define the concept of complexes and homology. As an extension of homology to point clouds, we introduce persistent homology. We state and prove the structure theorem used to classify persistent homology, which we visually describe in the form of a two-dimensional scatter plot, called a persistence diagram. Last but not least, we describe the ToMATo clustering scheme and present a couple applications of ToMATo.

## **Acknowledgements**

I would like to express my gratitude to Dr. Sara Kališnik Hintz for supervising my thesis. I am especially grateful for her patience and giving detailed feedback on my drafts. Her commitment and support whenever I faced challenges made this whole journey enjoyable and rewarding for me.

---

# Contents

---

<b>Abstract</b>	<b>i</b>
<b>Acknowledgements</b>	<b>ii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Clustering</b>	<b>7</b>
2.1 <i>k</i> -means Clustering . . . . .	7
2.2 Spectral Clustering . . . . .	9
2.3 Mean Shift Clustering . . . . .	15
2.4 Hierarchical Clustering . . . . .	18
<b>3 Simplicial Complexes and Homology</b>	<b>23</b>
3.1 Simplicial Complexes . . . . .	23
3.2 Homology . . . . .	24
<b>4 Persistent Homology</b>	<b>31</b>
4.1 Construction of Different Complexes . . . . .	31
4.2 Persistence Vector Spaces and Structure Theorem . . . . .	35
4.3 Bottleneck Distance and Stability Theorem . . . . .	44
4.4 Computing Persistent Homology . . . . .	46
<b>5 Topological Mode Analysis Tool</b>	<b>55</b>
5.1 Continuous Setting . . . . .	55
5.2 ToMATo Algorithm . . . . .	59
5.3 Experimental Result . . . . .	63

5.3.1	Synthetic Data . . . . .	64
5.3.2	Image Segmentation . . . . .	66
	<b>Bibliography</b>	<b>70</b>

## Chapter 1

---

# Introduction

---

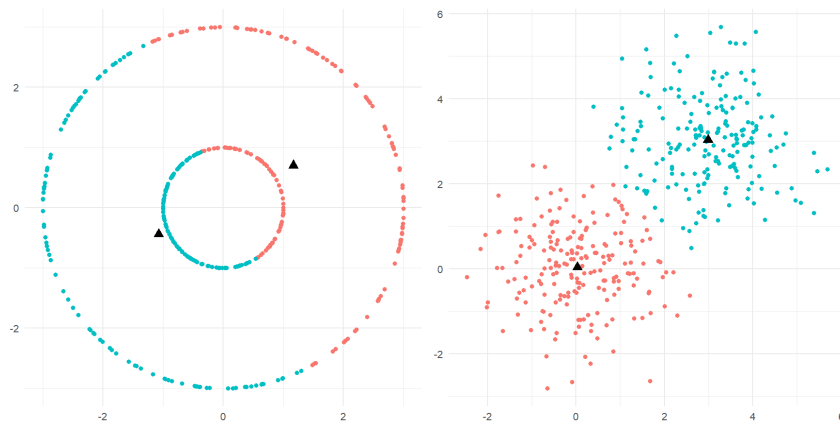
Clustering is a fundamental concept in data analysis. It involves grouping data points into clusters in a way that data points within a cluster are more similar to each other than to those in other clusters.

Some traditional clustering methods include  $k$ -means and hierarchical clustering. The  $k$ -means algorithm is a clustering algorithm that partitions a dataset into  $k$  distinct clusters, where each data point belongs to the cluster with the nearest mean. More precisely, the algorithm iteratively adjusts the cluster centres and reassigns data points to find the minimum of the *sums of squared distances*,

$$\arg \min_{\substack{S_1, \dots, S_k \\ \mu_1, \dots, \mu_k}} \sum_{l=1}^k \sum_{i \in S_l} \|\mathbf{x}_i - \mu_l\|^2.$$

Here,  $\mathbf{x}_i$  denote the data points,  $S_l$  denote the index sets of clusters and  $\mu_l$  denote the corresponding cluster centres for all  $i \in S_l$  and  $l \in \{1, \dots, k\}$ . Because Euclidean open balls are convex,  $k$ -means clustering naturally produces convex clusters. However, convex clusters are not necessarily suitable for accurately modeling the underlying structure of every dataset.

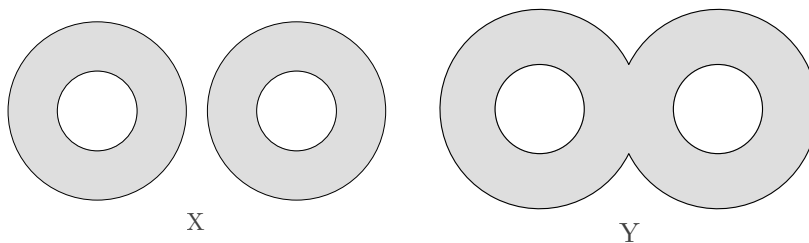
Consider Figure 1.1. We see the  $k$ -means algorithm applied to two different data sets. On the right, we see two clear clusters around two distinct centres.  $k$ -means performs well on these kinds of convex data sets. On the left, we see circular data around the same centre but with two separate radii. Clearly, the clustering does not resemble the underlying structure of the data.



**Figure 1.1:** Two  $k$ -means clusterings applied to circular data and to two distinct normally distributed blobs, respectively.

Recent approaches integrate *topological theory* to data analysis. *Topology* is a branch of mathematics that deals with the properties of spaces that are preserved under continuous transformations. A fundamental concept in topology is *homology*: topological spaces are classified with *homology groups*, which we denote by  $H_n$ . When we use a field for the coefficients,  $H_n$  is a vector space. The dimension of  $H_n$  is called the  $n$ -th *Betti number* and is denoted by  $\beta_n$ . Intuitively,  $\beta_n$  captures the number of  $n$ -dimensional ‘holes’ in a space. For  $\beta_0$  and  $\beta_1$ , these holes correspond to connected components and loops, respectively.

In Figure 1.2 the space  $X$  consists of two annuli that do not touch, and space  $Y$  consists of two annuli that are merged in the middle. For  $X$ , there are two connected components and two loops, thus  $\beta_0 = \beta_1 = 2$ . For  $Y$ , there is one connected component, but there again are two loops, thus  $\beta_0 = 1$  and  $\beta_1 = 2$ . Due to the invariance of homology under continuous transformations, we can tell these two spaces apart.

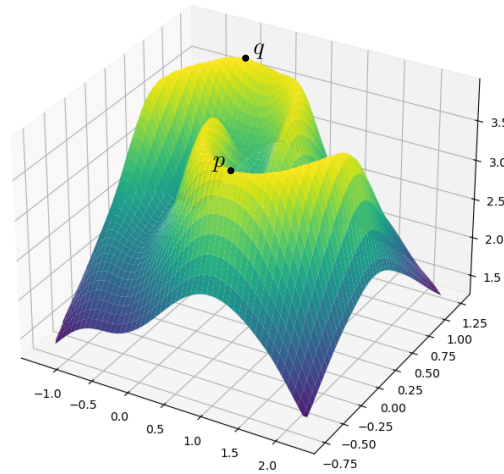


**Figure 1.2:** Spaces  $X$  and  $Y$  as examples of spaces with different homology groups.



Computing  $H_0(X)$  on a data set  $X$ , which mathematically speaking is a finite metric space, gives the 0-th Betti number  $\beta_0 = |X|$ . One way of producing a more interesting result is by viewing a data set  $X$  as a sample drawn from a space  $\mathbb{X}$  for some density function  $f : \mathbb{X} \rightarrow \mathbb{R}$ . The goal is to find a way to compute homology groups that relate to  $X$  through the lens of  $f$ . For this, we consider superlevel-sets. *Superlevel-sets* are the sets of the form  $\mathbb{F}^\alpha = f^{-1}([\alpha, +\infty))$ , where the parameter  $\alpha$  decreases from  $+\infty$  to  $-\infty$ . We call the collection of superlevel-sets  $\{\mathbb{F}^\alpha\}_{\alpha \in \mathbb{R}_+}$ , such that  $\mathbb{F}^\alpha \subseteq \mathbb{F}^{\alpha'}$  for all  $\alpha' \leq \alpha$  a *filtration*. Intuitively, this filtration captures which subsets of  $\Omega$  have higher probability to be drawn than others. More precisely, regions with higher density appear earlier in the filtration.

Consider a function  $f$  whose graph is drawn in Figure 1.3. The superlevel-sets of the corresponding filtration can be distinguished by the colour gradient. Let  $q$  and  $p$  be local maxima such that  $q > p$ . Let  $s$  be a local minimum between  $p$  and  $q$ . For  $\alpha \in [f^{-1}(q), f^{-1}(p))$ ,  $\mathbb{F}^\alpha$  consists of one component  $C_\alpha$ . For  $\alpha \in [f^{-1}(p), f^{-1}(s))$ ,  $\mathbb{F}^\alpha$  consists of two components  $C_\alpha^1$  and  $C_\alpha^2$ . For  $\alpha \in [f^{-1}(p), f^{-1}(-\infty))$  again consists of one component  $C_\alpha$ . Each component  $C$  appears when a local maximum of  $f$  is reached and two components get merged when a local minimum is reached.

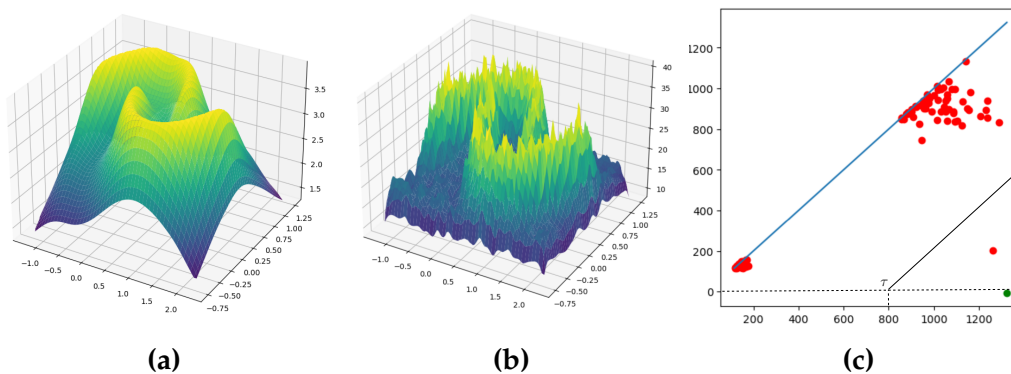


**Figure 1.3:** Filtration of superlevel-sets of density function  $f$  with two peaks.

By computing the homology group of filtrations, we end up with a sequence of homology groups called the *persistent homology group*, where the inclusion  $\mathbb{F}^\alpha \hookrightarrow \mathbb{F}^{\alpha'}$  for  $\alpha' \leq \alpha$  induces a linear map  $H_n(\mathbb{F}^\alpha) \rightarrow H_n(\mathbb{F}^{\alpha'})$ . The name originates from

the fact that they capture the range of parameters over which the topological features persist. This behaviour is summarized in a persistence diagram. *Persistence diagrams* are scatter plots, where each point  $p$  represents a generator of a connected component  $C$  in the homology groups of the filtration. The *lifespan*  $\tau$  of a generator is the duration between the time of appearance  $p_x$  and the time of disappearance  $p_y$ . At  $p_y$ ,  $C$  gets merged into another component generated by some higher peak of  $f$ . The lifespan  $\tau$  is an indicator of the prominence of  $C$ .

Since  $f$  is usually not provided, it must be estimated from the available sample using pairwise distances within the data. This often leads to density estimations with many non-prominent local peaks, as can be observed in Figure 1.4. Persistence diagrams are provably stable, that is points that are further away from the diagonal  $\{(x, y) \mid x = y\}$ , i.e. points with longer lifespans, describe more prominent features of the underlying data, whereas points close to the diagonal, i.e. points with lower lifespans, are often considered noise.



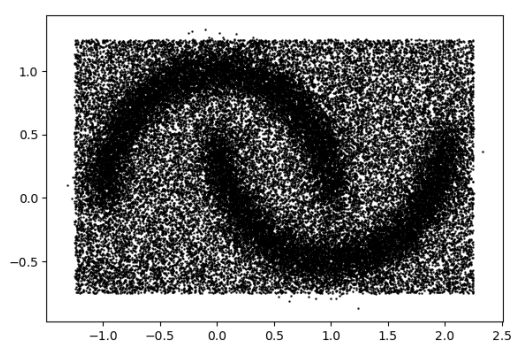
**Figure 1.4:** Evolution of the connectivity of the super-level sets of a function  $f$  on the left and of an approximation  $\tilde{f}$  in the middle. The corresponding persistence diagram on the right.

Consider Figure 1.4. Assume we have a density function  $f : \mathbb{X} \rightarrow \mathbb{R}$  as can be seen on the left. Assume we have a density estimator  $\tilde{f}$  in Figure 1.4(b). We compute the persistent homology group  $H_0(\tilde{\mathbb{F}}^\alpha) = H_0(\tilde{f}^{-1}([\alpha, +\infty)))$  and plot the persistence diagram in Figure 1.4(c). The points near the diagonal  $\{(x, y) \mid x = y\}$  represent the small peaks in the middle plot. We see that  $\tilde{f}$  has two prominent peaks corresponding to the two prominent peaks of  $f$ , by the stability properties of persistence diagrams. Note, the prominences  $\tau = p_x - p_y$  of the components in

$H_0(\mathbb{F}^\alpha)$  correspond to the height of the components in the left plot.

One clustering scheme that makes use of these ideas is the *Topological Mode Analysis Tool* (ToMATo). It provides a flexible and robust approach to clustering high-dimensional and non-linear datasets by taking into account the changes in the topology of the superlevel-sets. In particular, it analyses the zeroth persistent homology groups  $H_0(\mathbb{F}^\alpha)$  of the filtration of superlevel-sets. It achieves this by mimicking gradient flow via iteratively moving points towards regions of higher density, according to the gradient flow estimate of the density function. This process results in points converging into clusters, effectively determining the basins of attraction. Basins of attraction are areas in which points, when moved according to the gradient flow, end up at the same peak, thus defining distinct segments within the data.

As we have seen above, simply analyzing persistent homology of the superlevel-sets could lead to an excessive number of clusters. The occurrence of noise in data can amplify the situation. Thus in the ToMATo algorithm, two basins of attraction get connected if exactly one component is of prominence at least  $\tau > 0$ , for some predefined threshold parameter  $\tau$ . The component generated by the lower peak is said to be merged into the one generated by the higher peak. The output of the ToMATo algorithm is the collection of merged clusters whose prominence is larger or equal to some threshold  $\tau$ . In a first stage, the algorithm is run with  $\tau = +\infty$  in order to compute the persistence diagram of  $\tilde{f}$ . Then, the algorithm is re-run with the value of  $\tau$  defined by analysing the persistence diagram, in order to get the desired clusters.

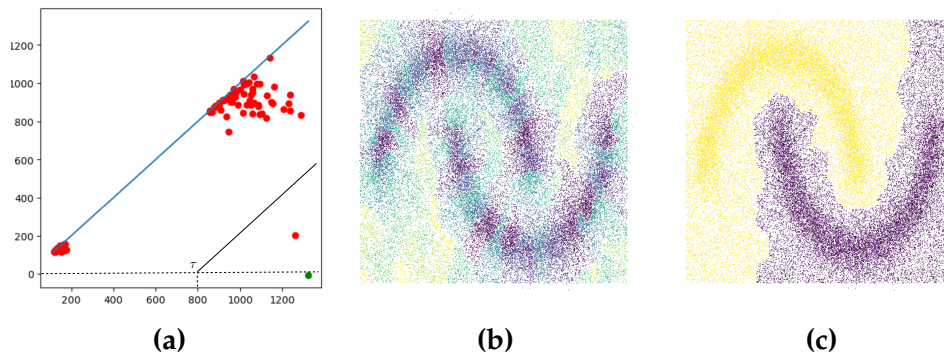


**Figure 1.5:** Halfmoons data set.

Let us now look at an example with highly non-linear synthetic data. The data

set is shown in Figure 1.5. We consider two Halfmoons in Euclidean space  $\mathbb{R}^2$ , each containing 20,000 points. Each Halfmoon follows a uniform distribution with some Gaussian noise. To ensure that the Halfmoons are not completely disjoint, we added some uniform background noise of 30,000 points.

Looking at the persistence diagram in Figure 1.4(c), we can clearly see a gap in the prominences. Therefore, we chose  $\tau = 800$  as threshold parameter. The choice of  $\tau$  is immediate in this example, but as we will see later, this choice will not always be as clear. The resulting choice leaves us with the two most prominent clusters that indeed correspond to the two Halfmoons.



**Figure 1.6:** Halfmoons data set. (a) Persistence Diagram. Two points far off the diagonal correspond to the two prominent peaks of  $f$  (b) Clustering for  $\tau = +\infty$ , i.e. result for the basic clustering algorithm without a merging phase. (c) Clustering for  $\tau = 800$ , i.e. final result after merging the clusters of non-prominent peaks.

## Chapter 2

---

# Clustering

---

When we look at geometric objects, one of the simplest properties to study is its number of connected components. A counterpart to connected components in the context of point cloud data is clustering. Clustering is the concept of partitioning data into groups, where we want to have high similarity within groups and low similarity between groups. Clustering serves as a crucial tool for understanding and interpreting patterns within data. Choosing a suitable algorithm is heavily dependent on the underlying structure of the data. This chapter introduces four popular clustering methods *k-means clustering*, *spectral clustering*, *mean shift clustering* and *hierarchical clustering*.

### 2.1 *k*-means Clustering

*k*-means clustering is one of the most common clustering methods. It serves as the foundation for numerous other clustering techniques. In this section, we closely follow the book 'Data Clustering: Algorithms and Applications' by Charu C. Aggarwal and Chandan K. Reddy [1].

**Definition 2.1 (*k*-means)** Let  $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$  be a finite subset of  $\mathbb{R}^d$ . *k*-means aims to partition the data set into  $k \leq n$  pairwise disjoint clusters  $S_1, \dots, S_k$  with centres  $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k \in \mathbb{R}^d$  such that the sums of squares distance to the centres is minimized:

$$\arg \min_{\substack{S_1, \dots, S_k \\ \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k}} \sum_{l=1}^k \sum_{i \in S_l} \|\mathbf{x}_i - \boldsymbol{\mu}_l\|^2. \quad (2.1)$$

A common algorithm to minimize Equation (2.1) is *Lloyd's algorithm*, often called the *k-means algorithm*. It is an iterative algorithm that starts by choosing  $k$  initial points called centres. Each point then gets assigned to the closest initial centre based on some proximity measure, thus leading to convex clusters for Euclidean metric spaces. Once all points are assigned, the centres get updated. The algorithm then iterates these two steps until the centres remain constant.

---

**Algorithm 1** Lloyd's Algorithm
 

---

**Input:** Data points  $\mathbf{x}_1, \dots, \mathbf{x}_n$ .

- 1: Select  $k$  initial centres  $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k$  of the corresponding clusters  $S_1, \dots, S_k$
- 2: Classify the clusters by identifying each point  $\mathbf{x}_i$  to the nearest centre  $\boldsymbol{\mu}_l$  of the cluster  $S_l$ ,

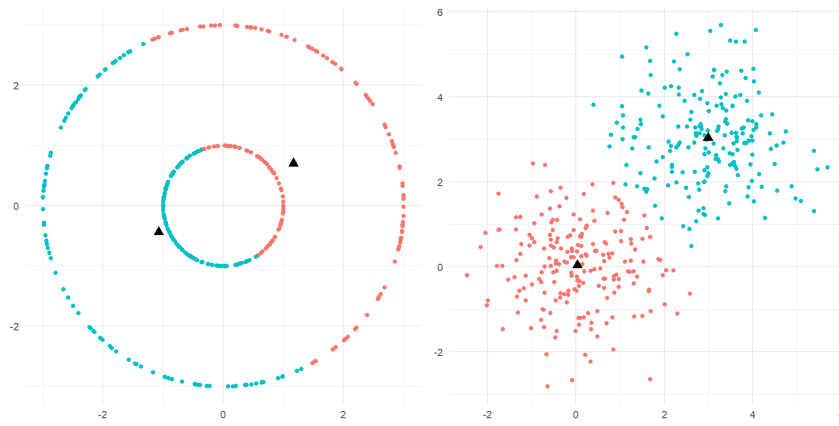
$$l = \min_{j=1, \dots, k} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|;$$

- 3: Update the centres  $\boldsymbol{\mu}_l = \frac{1}{|S_l|} \sum_{i \in S_l} \mathbf{x}_i$ ;
  - 4: Repeat 2. and 3. until the centres do not change;
- 

Lloyd's algorithm is a *greedy* algorithm which is guaranteed to converge to a local minimum of the sums of squares from Equation (2.1). The algorithm performs well if the dataset has spherical clusters. A great feature of the algorithm is that it is not hard to implement. However, if the data has a more complicated underlying geometric structure, it performs badly.

The minimization of the sums of squares is known to be NP-hard, which is why most practical attempts and algorithms to minimize Equation (2.1) converge to a local minimum. A proof of the convergence can be found in the article '*k-means-type algorithms: A generalized convergence theorem and characterization of local optimality*' by Shokri Z. Selim and M. A. Ismail [2].

Intuitively, it is not clear how to optimally choose the number of clusters  $k$ . Just optimizing the sums of squares in Equation 2.1 to find a suitable number of clusters would not lead to a satisfying result, since choosing  $k$  to be the number of points  $n$  in our dataset would lead to the sums of squares to equal zero, as all the points would lie in their own singleton cluster. Another problem arises in how to choose the initial centres. We could choose them at random, but then we could get a bad run time for the  $k$ -means algorithm. Several methods for initializing centres and estimating  $k$  can be found in [1].



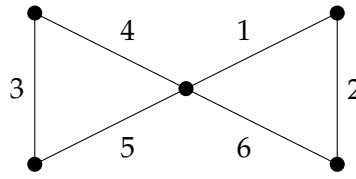
**Figure 2.1:** Two  $k$ -means clusterings. Left  $k$ -means is applied with the Euclidean distance measure on circular data, and right on data with two distinct normally distributed clusters. The left plot visually illustrates the bad performance on non-convex data sets. One can clearly see that  $k$ -means results in convex clusters.

## 2.2 Spectral Clustering

Before presenting spectral clustering, we have to first introduce a few mathematical objects. The spectral clustering algorithm requires similarity graphs and graph Laplacians. For the entire section, we closely follow the article ‘A Tutorial on Spectral Clustering’ by Ulrike von Luxburg [3].

Assume we have a set of data points  $\mathbf{x}_1, \dots, \mathbf{x}_n$  and some measure of *similarity*  $s_{ij} = s_{ji} \geq 0$  between all pairs of data points  $\mathbf{x}_i$  and  $\mathbf{x}_j$ . If there is no additional information than the measure of similarity, one way of representing the data is in the form of a *weighted graph*  $G = (V, E)$ .  $V$  is the *vertex set*, where each *vertex* represents a data point and  $E$  is the set of edges, where each *weighted undirected edge* between vertices  $v_i$  and  $v_j$  has weight  $w_{ij}$ . When the weights  $w_{ij}$  resemble some measure of similarity  $s_{ij}$  such that  $w_{ij} = s_{ij}$  if it is larger than some predefined threshold and zero otherwise, are called *similarity graphs*. The goal of clustering the data can then be reformulated in terms of partitioning the corresponding similarity graph. An example of a similarity graph can be seen in Figure 2.2. The graph depicted is called the *butterfly graph*. Each edge has a given weight, which denotes the similarity between the vertices it connects. The larger the value, the more similar the vertices are.

**Definition 2.2 (Adjacency Matrix)** *The adjacency matrix of a weighted graph  $G$  is the matrix  $W = (w_{ij})_{1 \leq i, j \leq n}$  where  $w_{ij}$  are the non-negative weights from above.*



**Figure 2.2:** Butterfly graph

If  $w_{ij} = 0$  the vertices  $v_i$  and  $v_j$  are not connected by an edge. Since we assume  $G$  to be an undirected graph, the adjacency matrix  $W$  is symmetric, i.e.  $w_{ij} = w_{ji}$ .

**Definition 2.3 (Degree of a Vertex)** The degree of a vertex  $v_i$  is the sum over the weighted edges connected to  $v_i$

$$d_i = \sum_{j=1}^n w_{ij}.$$

**Definition 2.4 (Degree Matrix)** The degree matrix  $D = \text{diag}(d_1, \dots, d_n)$  is defined as the diagonal  $n \times n$  matrix, where  $n$  is the number of vertices.  $D$  has the vertex degrees on the diagonal and zero entries otherwise.

**Definition 2.5 (Indicator Vector)** Let  $A \subset V$  be a subset of vertices. The indicator vector is  $\mathbb{1}_A = (\mathbb{1}_{v_1 \in A}, \dots, \mathbb{1}_{v_n \in A})^T \in \mathbb{R}^n$  where  $\mathbb{1}_{v_i \in A} = 1$  if  $v_i \in A$  and  $\mathbb{1}_{v_i \in A} = 0$  otherwise.

**Definition 2.6** We say a subset  $A \subset V$  is connected if for all pairs of vertices  $v_i, v_j \in A$  there exists a path in  $A$  connecting  $v_i$  and  $v_j$ .  $A \subset V$  is a connected component if it is connected and there exists no edge between  $A$  and  $\bar{A} = V \setminus A$ .

A priori, it is not clear how to construct a graph  $G$ , just knowing the data set and the pairwise distances. We now give a few popular graph constructions given data points  $x_1, \dots, x_n$  and pairwise similarities  $s_{ij}$  or distances  $d_{ij}$ . For this, let  $G = (V, E)$  be an undirected graph with  $V = \{v_1, \dots, v_n\}$  where  $v_1, \dots, v_n$  represent the data points  $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$ :

- **$\epsilon$ -neighbourhood graph:** For this graph, we connect two vertices  $v_i, v_j$  if  $d(v_i, v_j) \leq \epsilon$ . Since all the distances between connected vertices are bounded by  $\epsilon$ , this graph is usually considered an unweighted graph.
- **$k$ -nearest neighbour graph:** We want to connect  $v_i$  with  $v_j$  if  $v_j$  is among the  $k$ -nearest neighbours of  $v_i$ . But since this relation is not symmetric, this would give a directed graph. There are two ways to get an undirected graph.



Firstly, we could ignore the direction and connect  $v_i$  and  $v_j$  if one vertex is among the  $k$ -nearest neighbours of the other vertex. We usually refer to this to be the *k-nearest neighbour graph*. Secondly, we could connect  $v_i$  and  $v_j$  if both are among the  $k$ -nearest neighbours of each other. We usually refer to this to be the *mutual k-nearest neighbour graph*. In both cases, the weights of the edges correspond to the similarity  $s_{ij}$  of the respective vertices.

- **Fully connected graph:** Here we connect all the vertices with positive similarity with each other and weigh them accordingly. This construction only makes sense if the similarity models local relationships. One could use the Gaussian similarity function  $s(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / (2\sigma^2))$ , where  $\sigma$  controls the width of the neighbourhoods.

An important tool for spectral clustering are *graph Laplacian matrices*. We now define the unnormalized graph Laplacian matrix and look at a few properties. For the following, assume that we have an undirected weighted graph  $G = (V, E)$  with adjacency matrix  $W$  where  $w_{ij} = w_{ji} \geq 0$  and the corresponding degree matrix  $D$ . We assume that the eigenvalues are always ordered in an increasing order, and that eigenvectors are not necessarily normalized.

**Definition 2.7 (Unnormalized Graph Laplacian)** *The unnormalized graph Laplacian matrix  $L$  is defined as*

$$L = D - W.$$

where  $D = \text{diag}(d_1, \dots, d_n)$  is the degree matrix and  $W$  is the adjacency matrix.

Note, the unnormalized graph Laplacian does not depend on diagonal entries  $w_{ii}$  of  $W$  since they also appear in  $d_i$  exactly once. This means that two graphs  $G$  and  $G'$ , that have the same adjacency matrix up to self-edges, have the same unnormalized graph Laplacian  $L$ .

**Proposition 2.8** *The unnormalized graph Laplacian matrix  $L$  satisfies the following properties:*

1. For every vector  $f \in \mathbb{R}^n$ ,

$$f^T L f = \frac{1}{2} \sum_{i,j=1}^n w_{ij} (f_i - f_j)^2. \quad (2.2)$$

2.  $L$  is symmetric and positive semi-definite.

3. The smallest eigenvalue of  $L$  is 0, the corresponding eigenvector is the indicator vector  $\mathbb{1} = (1, \dots, 1)^T$ .
4.  $L$  has  $n$  non-negative, real-valued eigenvalues  $0 = \lambda_1 \leq \dots \leq \lambda_n$ .

*Proof.* 1. Since  $d_i = \sum_{j=1}^n w_{ij}$  it follows that

$$\begin{aligned}
 f^T L f &= f^T D f - f^T W f = \sum_{i=1}^n d_i f_i^2 - \sum_{i,j=1}^n f_i f_j w_{ij} \\
 &= \frac{1}{2} \left( \sum_{i=1}^n d_i f_i^2 - 2 \sum_{i,j=1}^n f_i f_j w_{ij} + \sum_{j=1}^n d_j f_j^2 \right) \\
 &= \frac{1}{2} \left( \sum_{i=1}^n \sum_{j=1}^n w_{ij} f_i^2 - 2 \sum_{i,j=1}^n w_{ij} f_i f_j + \sum_{j=1}^n \sum_{i=1}^n w_{ij} f_j^2 \right) \\
 &= \frac{1}{2} \sum_{i,j=1}^n w_{ij} (f_i - f_j)^2.
 \end{aligned}$$

2. The symmetry follows directly from the symmetry of  $W$  and  $D$ . The positive semi-definiteness follows from Equation 2.2.
3. Plugging in the indicator vector  $f = \mathbb{1}$  in Equation 2.2 gives us the zero, thus  $\mathbb{1}$  is an eigenvector with eigenvalue  $\lambda_1 = 0$ .
4. That  $L$  has  $n$  non-negative, real-valued eigenvalues  $0 = \lambda_1 \leq \dots \leq \lambda_n$  follows from the positive semi-definiteness of  $L$  and the fact that all  $n \times n$  symmetric matrices have  $n$  eigenvalues when counted with multiplicity.

□

**Proposition 2.9** *Let  $G$  be an undirected graph with non-negative weights and let  $k$  be the number of connected components. Then the multiplicity of the eigenvalue 0 equals the number of connected components  $A_1, \dots, A_k$  in  $G$ .  $\mathbb{1}_{A_1}, \dots, \mathbb{1}_{A_k}$  span the eigenspace of the eigenvalue 0.*

*Proof.* We prove this using induction over  $k$ . Let  $k = 1$ , i.e.  $G$  is connected. Assume that  $f$  is an eigenvector with eigenvalue 0. From Proposition 2.8 we know

$$0 = f^T L f = \frac{1}{2} \sum_{i,j=1}^n w_{ij} (f_i - f_j)^2.$$

We know that since  $G$  is connected, we get  $w_{ij} > 0$ . Because  $w_{ij}(f_i - f_j)^2$  vanishes, we get that  $f_i = f_j$ . Moreover, as all vertices of a connected component in an

undirected graph can be connected by a path,  $f$  needs to be constant on the whole connected component. We can conclude that we only have the indicator vector  $\mathbb{1}$  as eigenvector with eigenvalue 0, and this is obviously the indicator vector of the connected component.

Let us now consider the case of  $k$  connected components. Without loss of generality, we assume that the vertices are ordered according to the connected components they belong to. In this case, the adjacency matrix  $W$  has block diagonal form, thus the unnormalized graph Laplacian has block diagonal form too

$$L = \begin{pmatrix} L_1 & & \\ & \ddots & \\ & & L_k \end{pmatrix}.$$

Note that each block  $L_i$  is a proper unnormalized graph Laplacian corresponding to the connected component  $A_i$ . For block diagonal matrices, we know the eigenvectors of  $L$  are the eigenvectors of  $L_i$  embedded in the corresponding higher dimensional vector space, i.e. filled with 0 at the position of the other blocks. Using that each  $A_i$  has only one connected component, we get that the eigenvector of  $L_i$  with eigenvalue 0 is  $\mathbb{1}$ , we conclude that for  $L$  the eigenvalue 0 has multiplicity  $k$  and  $\mathbb{1}_{A_1}, \dots, \mathbb{1}_{A_k}$  span the eigenspace of the eigenvalue 0.  $\square$

Let us now consider a set of data points  $\mathbf{x}_1, \dots, \mathbf{x}_n$ , which can be arbitrary objects, and their pairwise similarities  $s_{ij} = s(\mathbf{x}_i, \mathbf{x}_j)$  by some symmetric, non-negative similarity function. Let  $S = (s_{ij})_{1 \leq i, j \leq n}$  be the similarity matrix and  $k$  be the number of clusters we want to construct. The *unnormalized spectral clustering algorithm* works as follows.

---

**Algorithm 2** Unnormalized Spectral Clustering Algorithm

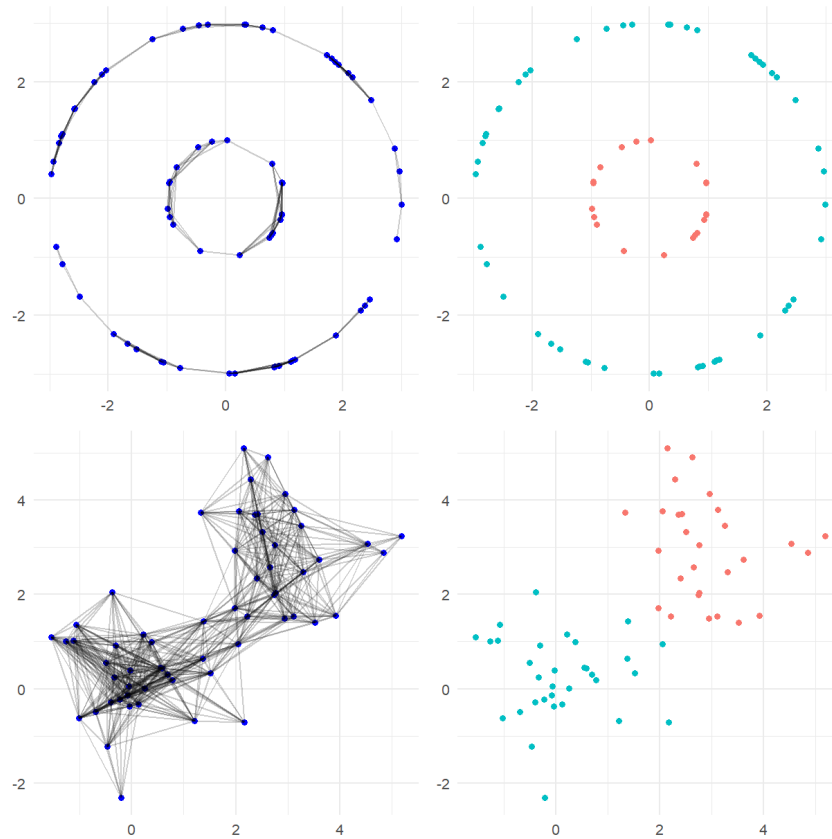
---

**Input:** Data points  $\mathbf{x}_1, \dots, \mathbf{x}_n$ ,  $n \times n$  similarity matrix  $S$ .

- 1: Construct a similarity graph  $G = (V, E)$ . Let  $W$  be its weighted adjacency matrix;
  - 2: Compute the unnormalized graph Laplacian  $L$ ;
  - 3: Compute the first  $k$  eigenvectors  $\mathbf{u}_1, \dots, \mathbf{u}_k$  of  $L$ ;
  - 4: Let  $U = (\mathbf{u}_1, \dots, \mathbf{u}_k) \in \mathbb{R}^{n \times k}$  the matrix containing the first  $k$  eigenvectors;
  - 5: For  $i = 1, \dots, n$  let  $\mathbf{y}_i \in \mathbb{R}^k$  the vector corresponding to the  $i$ -th row of  $U$ ;
  - 6: Cluster the points  $\mathbf{y}_1, \dots, \mathbf{y}_n$ , representing  $\mathbf{x}_1, \dots, \mathbf{x}_n$ , using  $k$ -means;
-

Intuitively, the unnormalized spectral clustering algorithm attempts to partition the similarity graph to have high similarity within groups and low similarity between groups using properties of Propositions 2.8 and 2.9.

**Remark 2.10** *Instead of using the unnormalized graph Laplacian, there are spectral clustering algorithms using a version of a normalized graph Laplacian. It comes with many advantages like stronger convergence results for increasingly bigger data sets and robustness.*



**Figure 2.3:** Spectral clustering. In the top plots we can see circular data and in the bottom plots we can see normally distributed data that is centred around two different centroids. On the top left is the similarity graph using the Gaussian similarity with  $\sigma = 0.5$  and threshold 0.1 and on the bottom left with  $\sigma = 1$  and threshold 0.1. Right is the corresponding clustering.

Spectral clustering can be applied to a wide range of data types, including graphs and non-Euclidean data. This makes it versatile for various applications

beyond standard vector-based data. There are many ways to choose similarity measures, the number of clusters and which graph Laplacian to use. However, a priori, it is not clear which suits a given data set. One can read further into it in [3].

## 2.3 Mean Shift Clustering

Density based clustering methods make the assumption that the given data points follow some underlying probability distribution with unknown density function  $f$ . Clustering then becomes a problem of understanding the empirical density of  $f$ , estimated from the sample. A widely used method called *mode-seeking* involves identifying the local peaks, the so-called *modes*, of  $f$  to serve as cluster centres and then partitioning the data based on their respective *basins of attraction*. *Mean shift* is a popular mode-seeking clustering procedure. In this section, we closely follow the article 'Mean shift: A robust approach toward feature space analysis' by Dorin Comaniciu [4].

The idea of mean shift clustering is to treat the points as a sample from some absolutely continuous distribution. Dense regions correspond to the local maxima of the density function. First one estimates the density function using a multivariate kernel density estimate and then performs gradient ascent to determine the local maxima, the so-called mode. The basins of attraction serve as the clusters, and the modes function as attraction points.

**Definition 2.11 (Univariate Kernel)** *A univariate kernel is a non-negative real valued integrable function  $K$ .  $K$  is considered symmetric if in addition  $K(-x) = K(x)$  for all values  $x$ .*

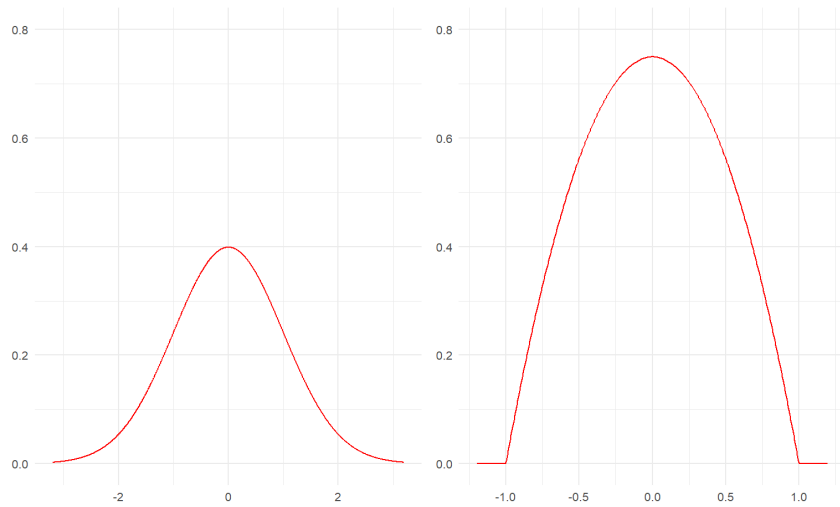
**Definition 2.12 (Radially Symmetric Kernel)** *A radially symmetric kernel  $k$  generated from a symmetric univariate kernel  $K$  by rotating  $K(x)$  in  $\mathbb{R}^d$  is defined as,*

$$k(\mathbf{x}) = c_K K(\|\mathbf{x}\|^2).$$

where  $K$  is a real-valued non-negative integrable function and  $c_K$  is a normalization constant such that

$$\int_{\mathbb{R}^d} c_K K(\|\mathbf{x}\|^2) d\mathbf{x} = 1.$$

Since we are only interested in these types of kernels, we refer to radially symmetric kernels as kernels in Definition 2.12.



**Figure 2.4:** The left kernel is a Gaussian kernel and is described by the function  $K_{\text{Gaussian}}(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}x^2}$ . Another popular kernel is the Epanechnikov kernel on the right, that is described by the function  $K_{\text{Epanechnikov}}(x) = \frac{3}{4}(1 - x^2)$  for  $|x| \leq 1$  and zero otherwise. By rotating these functions around  $x = 0$ , we get radially symmetric kernels.

**Definition 2.13 (Multivariate Kernel Density Estimate)** Let  $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$ , the multivariate kernel density estimate with a radially symmetric kernel  $k(\mathbf{x})$  and a bandwidth  $h$ , is given by

$$\hat{f}_k(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^n k\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right).$$

Intuitively, the bandwidth gives a measure of the radius we want our kernel to be applied on. Thus, it indirectly determines the number of clusters we want our data set to be divided in. There are many ways to select the bandwidth parameter, which are further discussed in [4].

The multivariate kernel density estimate estimates the underlying density function of a given data set  $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$ . The constant  $\frac{1}{nh^d}$  is necessary to normalize the estimator  $\hat{f}_k$  to ensure that it defines a probability density. Calculating the gradient of the multivariate kernel density estimate, we get

$$\nabla \hat{f}_k(\mathbf{x}) = -\frac{2c_K}{nh^{d+2}} \left[ \sum_{i=1}^n K' \left( \left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2 \right) \right] \left[ \frac{\sum_{i=1}^n \mathbf{x}_i K' \left( \left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2 \right)}{\sum_{i=1}^n K' \left( \left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2 \right)} - \mathbf{x} \right]. \quad (2.3)$$

**Definition 2.14 (Mean Shift Vector)** *The second term in the above expression is called the mean shift vector*

$$\mathbf{m}(\mathbf{x}) = \left[ \frac{\sum_{i=1}^n \mathbf{x}_i K' \left( \left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2 \right)}{\sum_{i=1}^n K' \left( \left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2 \right)} - \mathbf{x} \right].$$

**Proposition 2.15** *The mean shift vector is a scalar multiple of the gradient of the multivariate kernel density estimator,*

$$\mathbf{m}(\mathbf{x}) = \frac{1}{2} h^2 \frac{\nabla \hat{f}_k(\mathbf{x})}{\hat{f}_k(\mathbf{x})}.$$

*Proof.* The first term in Equation 2.3 is proportional to the density estimate for the kernel  $\tilde{k} = c_{\tilde{k}} K'(\|\mathbf{x}\|)$ , that is

$$\hat{f}_{\tilde{k}}(\mathbf{x}) = \frac{c_K}{nh^d} \left[ \sum_{i=1}^n K' \left( \left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2 \right) \right].$$

Thus, the mean shift vector can be written as a scalar multiple of the gradient of the multivariate kernel density estimate of  $\hat{f}_k$ .  $\square$

The mean shift vector points towards the closest local maximum of the density estimator, i.e. in the direction of greatest ascent. Intuitively, it shifts the local mean of our data set toward the closest region with the most points. The mean shift clustering algorithm then works as follows. For every data point  $\mathbf{x}_i^t$  in our data set, we calculate the corresponding mean shift vector  $\mathbf{m}(\mathbf{x}_i^t)$  and shift our data point towards the closest mode,  $\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{m}(\mathbf{x}_i^t)$ . We repeat this until we reach  $\nabla f(\mathbf{x}_i^{t+1}) = 0$ . Every data point  $\mathbf{x}_i$  then belongs to the cluster  $S_j$  corresponding to the mode where  $\mathbf{x}_i^t$  converged to.

---

### Algorithm 3 Mean Shift Algorithm

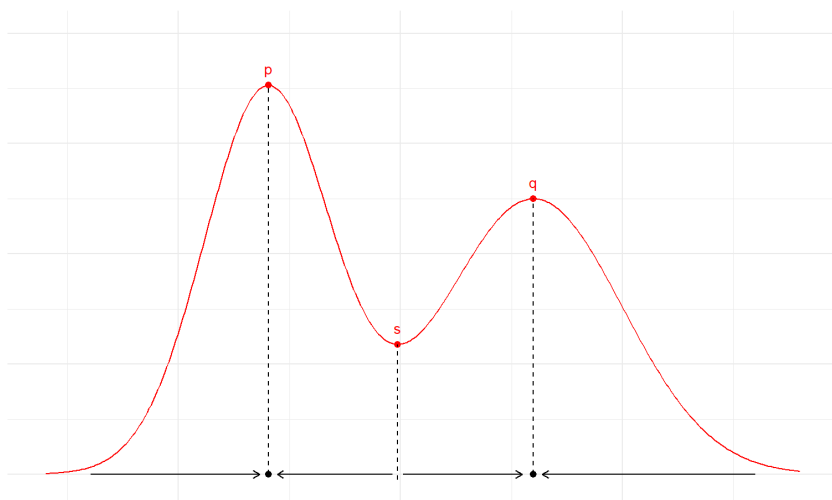
---

**Input:** Data points  $\mathbf{x}_1, \dots, \mathbf{x}_n$ , mean shift vector  $\mathbf{m}$ .

- 1: Compute the mean shift vector  $\mathbf{m}(\mathbf{x}_i^t)$ ;
  - 2: Translate toward the mode:  $\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{m}(\mathbf{x}_i^t)$ ;
  - 3: Repeat 1. and 2. until  $\nabla f(\mathbf{x}_i^{t+1}) = 0$ , i.e.  $\mathbf{x}_i^{t+1}$  has reached a local maximum;
  - 4: The data points  $\mathbf{x}_i$  that converged to the same local maximum belong to the same cluster;
- 

A proof of convergence for the Algorithm 3 mean shift algorithm can be found in [4].

Mean shift clustering does not make any strong model assumptions on the underlying data. Using a probabilistic approach, it is robust towards outliers, and can also be applied on complex structures with non-convex shapes. However, we cannot directly control the number of clusters. Another drawback is its sensitivity to noise. Every mode is in one to one correspondence with a cluster. Peaks that are non-prominent can be seen as nonsensible peaks, since they are often caused by noise and thus should not be clustered separately.



**Figure 2.5:** A density function with peaks  $p$  and  $q$ .

**Example 2.16** Assume we have a data set  $x_1, \dots, x_n \in \mathbb{R}$  that is a sample of some distribution function. Let  $f$  be the estimated density depicted in Figure 2.5. We get that the local maxima of the density estimator are  $p$  and  $q$  and the local minimum in  $s$ . Since  $\nabla f(x)$  points in the direction of greatest ascent, applying the mean shift algorithm gives that all the data points left of  $s$  converge to the cluster belonging to  $p$  and all the data points right of  $s$  converge to the cluster belonging to  $q$ . Since the gradient  $\nabla f(s) = 0$ , a data point at the point  $s$  will not converge using the mean shift algorithm, thus will not belong to any cluster.

## 2.4 Hierarchical Clustering

Until now, the methods discussed formed disjoint clusters. One would say that the data description is *flat* or *partitional based*. However, often clusters have sub-clusters, which then have sub-sub-clusters and so on. *Hierarchical clustering algorithms* were



developed to lead to representations that are more *hierarchical* rather than *flat*. In this section, we closely follow the book 'Data Clustering: Algorithms and Applications' by Charu C. Aggarwal and Chandan K. Reddy [1] and the book 'Pattern Classification' by Richard O. Duda, Peter E. Hart and David G. Stork [5].

**Definition 2.17 (Hierarchical Clustering)** Consider a sequence of length  $n$  of partitions of data points  $\mathbf{x}_1, \dots, \mathbf{x}_n$  into  $c$  clusters, where the  $l$ -th partition in the sequence has  $c = n - l + 1$  clusters. We then say we are at level  $l$ . Given two subsamples of our data set  $S_i$  and  $S_j$ , if they are in the same cluster at some level, then they are in the same cluster in all higher levels. We call such a sequence hierarchical clustering.

**Remark 2.18** Notice the first partition groups the data points into  $n$  clusters containing one point each, namely the singleton clusters. The next is a partition into  $n - 1$  clusters. This continues iteratively until the  $n$ -th partition forms one cluster, containing all data points.

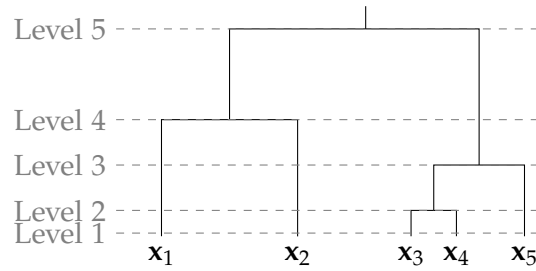
Hierarchical methods can be categorized into *agglomerative* and *divisive* clustering methods. Agglomerative methods are *bottom up* methods and start by taking each data point as a singleton cluster. They then continue by merging two clusters at a time using some proximity rule. Divisive methods are *top down* methods and start with all the data points in one cluster and partitions it continuously into two groups. We will mainly discuss agglomerative clustering.

**Definition 2.19 (Dendrogram)** A dendrogram is a diagram, which resembles a tree. In the case of clustering, it shows how clusters are merged. Every level corresponds to how the data is clustered.

**Remark 2.20** The most natural representation of hierarchical clustering is a dendrogram. In the case of agglomerative clustering, all the levels are represented by the current depth in the tree. Level 1 shows all the singleton clusters. The following levels then group clusters together to get a lower number of clusters than in the previous level. The last level consists of one cluster containing all the data points.

**Definition 2.21** Given a partition of our data set into clusters  $S_1, \dots, S_c$ , the dissimilarity matrix  $D = (d_{ij})_{1 \leq i, j \leq c}$  contains in each entry  $d_{ij}$  the measure of proximity between the two clusters  $S_i$  and  $S_j$  for some given proximity measure.

Commonly used proximity measures are the minimal or the maximal Euclidean distance between the points in two clusters, that is



**Figure 2.6:** This dendrogram shows on the horizontal axis the singleton clusters  $x_1, \dots, x_5$  at the first level and the merged clusters on the higher levels. The vertical axis corresponds to the levels. Here  $x_3$  and  $x_4$  are the most similar, because their singleton clusters merged first.

$$d_{min}(S_i, S_j) = \min_{\substack{\mathbf{x} \in S_i \\ \mathbf{x}' \in S_j}} \|\mathbf{x} - \mathbf{x}'\|,$$

or

$$d_{max}(S_i, S_j) = \max_{\substack{\mathbf{x} \in S_i \\ \mathbf{x}' \in S_j}} \|\mathbf{x} - \mathbf{x}'\|.$$

The agglomerative hierarchical clustering algorithm then works as follows. First, we start with all the data points represented in the form of singleton clusters. They are shown at the bottom of the dendrogram. Then the closest pair of clusters with respect to the dissimilarity matrix, is merged at each level, and the dissimilarity matrix is updated correspondingly. This agglomerative merging process is repeated until the sequence arrives at the cluster containing all the data points. This is represented as the apex of the dendrogram. Different kinds of proximity measures provide for different clustering methods.

---

#### Algorithm 4 Agglomerative Hierarchical Clustering Algorithm

---

**Input:** Data points  $x_1, \dots, x_n$ .

- 1: Compute the dissimilarity matrix;
  - 2: Merge the closest clusters according to the dissimilarity matrix giving  $S_{a \cup b} = S_a \cup S_b$ ;
  - 3: Update the dissimilarity matrix. Replace the rows and columns of  $S_a$  and  $S_b$  with a new row and column containing the distances between the new cluster  $S_{a \cup b}$  and the remaining clusters;
  - 4: Repeat 2. and 3. until there is only one cluster left;
-

One of the most popular agglomerative clustering methods is *single linkage clustering*, also called nearest neighbour clustering. In *single linkage clustering*, the dissimilarity of two clusters is the dissimilarity between the two most similar points.

**Definition 2.22 (Single Linkage Clustering)** *The single linkage clustering algorithm is the hierarchical clustering algorithm applied using a dissimilarity matrix of the form  $D = [d_{\min}(S_i, S_j)]_{i,j}$  for*

$$d_{\min}(S_i, S_j) = \min_{\substack{\mathbf{x} \in S_i \\ \mathbf{x}' \in S_j}} \|\mathbf{x} - \mathbf{x}'\|,$$

where  $S_i$  and  $S_j$  are clusters in the corresponding level.

Intuitively, this method looks at the local structure of the data and ignores the underlying global structure. It gives more weight to regions where clusters are close. Due to the strong local behaviour, single linkage performs well on non-elliptical, elongated data sets. It is very simple to implement and can be solved efficiently by standard linear algebra methods. The main drawback is its sensitivity to noise and outliers.

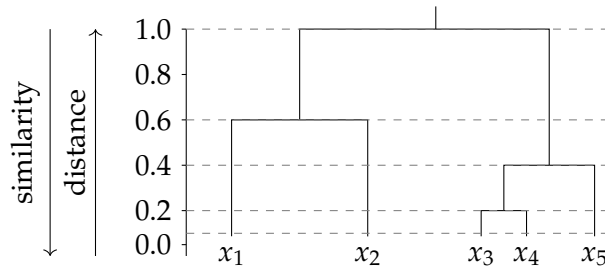
**Example 2.23** *For this example, we revisit the dendrogram from Figure 2.6. Let us consider data points  $x_1 = 0$ ,  $x_2 = 0.6$ ,  $x_3 = 1.6$ ,  $x_4 = 1.8$ ,  $x_5 = 2.2$  in  $\mathbb{R}$ . First, all the data points belong to their corresponding singleton clusters  $S_1, \dots, S_5$  on level 1. Now we construct a dissimilarity matrix  $D$  using the Euclidean distance.*

$$D = \begin{matrix} & S_{x_1} & S_{x_2} & S_{x_3} & S_{x_4} & S_{x_5} \\ \begin{matrix} S_{x_1} \\ S_{x_2} \\ S_{x_3} \\ S_{x_4} \\ S_{x_5} \end{matrix} & \begin{pmatrix} 0.0 & 0.6 & 1.6 & 1.8 & 2.2 \\ 0.6 & 0.0 & 1.0 & 1.2 & 1.6 \\ 1.6 & 1.0 & 0.0 & 0.2 & 0.6 \\ 1.8 & 1.2 & 0.2 & 0.0 & 0.4 \\ 2.2 & 1.6 & 0.6 & 0.4 & 0.0 \end{pmatrix} \end{matrix}.$$

Notice that  $D$  is symmetric. We see that  $d_{\min}(S_i, S_j) = d_{34} = d_{43} = 0.2$  is the smallest distance between all the singleton clusters, thus we merge  $S_3$  and  $S_4$  and update the dissimilarity matrix. We now get

$$\begin{array}{c}
 S_{x_1} \\
 S_{x_2} \\
 S_{x_3 \cup x_4} \\
 S_{x_5}
 \end{array}
 \begin{array}{cccc}
 S_{x_1} & S_{x_2} & S_{x_3 \cup x_4} & S_{x_5} \\
 \left( \begin{array}{cccc}
 0.0 & 0.6 & 1.6 & 2.2 \\
 0.6 & 0.0 & 1.0 & 1.6 \\
 1.6 & 1.0 & 0.0 & 0.4 \\
 2.2 & 1.6 & 0.4 & 0.0
 \end{array} \right)
 \end{array}
 .$$

We repeat this until we get the cluster  $S_{x_1 \cup \dots \cup x_5}$  at level 5. We then get a dendrogram as in Figure 2.7. The grey dashed lines represent the levels.



**Figure 2.7:** Dendrogram for single linkage clustering of the data set  $x_1 = 0$ ,  $x_2 = 0.6$ ,  $x_3 = 1.6$ ,  $x_4 = 1.8$ ,  $x_5 = 2.2$ .

Instead of using  $d_{\min}(S_i, S_j)$  for the dissimilarity matrix  $D$  one could also use the maximal distance between two clusters to get *complete linkage clustering*, the distance between two centroids to get *centroid agglomerative clustering* or the average distance between all the points in set two clusters to get *average linkage clustering*. For details we refer the reader to [1].

## Chapter 3

---

# Simplicial Complexes and Homology

---

Homology is a fundamental concept in algebraic topology. It provides a powerful framework to classify and understand the structure of spaces by counting the number of 'holes' in each dimension. We only focus on homology over the finite field  $k = \mathbb{F}_2$  with two elements. However, one can develop a theory for general coefficients. In this chapter we introduce *simplicial complexes*, define *homology* and state some useful results. This chapter is based on the book 'Topological pattern recognition for point cloud data' by Gunnar Carlsson [6].

### 3.1 Simplicial Complexes

Simplicial complexes provide a geometric framework to study topological properties of spaces. It enables the decomposition of shapes into simpler building blocks like points, line segments, triangles, and higher-dimensional simplices. This approach is useful for the computation and understanding of homology groups.

**Definition 3.1 (General Position)** Let  $S = \{\mathbf{x}_0, \dots, \mathbf{x}_n\} \subseteq \mathbb{R}^d$ . Then  $S$  is said to be in general position, if it is not contained in any affine hyperplane of  $\mathbb{R}^d$  of dimension less than  $n$ .

**Definition 3.2 (Simplex)** Let  $S = \{\mathbf{x}_0, \dots, \mathbf{x}_n\} \subseteq \mathbb{R}^d$  be in general position. The  $n$ -simplex spanned by  $S$  is the convex hull  $\sigma = \sigma(S) \subseteq \mathbb{R}^d$ . The points  $\mathbf{x}_i$  are called vertices. The simplices  $\sigma(T)$  for any non-empty  $T \subseteq S$  are called faces of  $\sigma$ .

**Definition 3.3 (Simplicial Complex)** A (finite) simplicial complex  $\mathcal{X}$  is a collection of simplices in Euclidean space such that:

1. For any simplex  $\sigma$  of  $\mathcal{X}$ , all faces of  $\sigma$  are contained in  $\mathcal{X}$ .
2. For any two simplices  $\sigma, \tau \in \mathcal{X}$ , the intersection  $\sigma \cap \tau$  is a face of both  $\sigma$  and  $\tau$  and is contained in  $\mathcal{X}$ .

Simplicial complexes rely on specific geometric embeddings into Euclidean space. A more flexible and combinatorial approach to studying topological spaces are abstract simplicial complexes.

**Definition 3.4 (Abstract Simplicial Complex)** Let  $X = (V(X), \Sigma(X))$  be a pair, where the vertices  $V(X)$  denote a finite set of  $X$  and the simplices  $\Sigma(X)$  are a non-empty subset of  $\mathcal{P}(V(X))$ , where  $\mathcal{P}$  denotes the power set.  $X$  is called an abstract simplicial complex if for  $\sigma \in \Sigma(X)$  and  $\emptyset \neq \tau \subseteq \sigma$ , called faces,  $\tau \in \Sigma(X)$ . Simplices consisting of exactly two vertices are called edges. Moreover, let  $\Sigma_i(X)$  denote the set of  $i$ -dimensional simplices.

Note that a simplicial complex  $\mathcal{X}$  determines an abstract simplicial complex where  $V(\mathcal{X})$  consists of all vertices of all simplices of  $\mathcal{X}$  and  $\Sigma(X)$  contains all vertices and all their faces. With some abuse of notation, we will write  $n$ -simplices  $\{\{x_0\}, \dots, \{x_n\}\}$  as  $\{x_0 \dots x_n\}$ .

**Example 3.5** Consider the abstract simplicial complex  $X$  in Figure 3.1.  $X$  consists of the vertices

$$V(X) = \{A, B, C, D\}$$

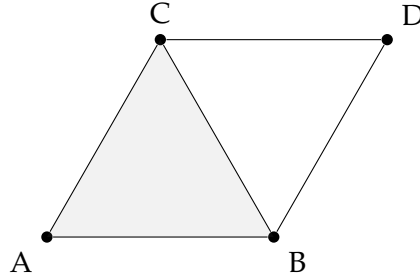
and the simplices

$$\Sigma(X) = \{ABC, AB, AC, BC, BD, CD, A, B, C, D\}.$$

The 0-simplices  $\Sigma_0(X) = \{A, B, C, D\}$  coincide with the vertices. The 1-simplices  $\Sigma_1(X) = \{AB, AC, BC, BD, CD\}$  are the edges that connect the vertices. The 2-simplex  $\Sigma_2(X) = \{ABC\}$  corresponds to the grey triangle.

## 3.2 Homology

Homology provides information about  $n$ -dimensional features. That involves connectivity in dimension 0, loops in dimension 1, voids in dimension 2 and so on. We derive this information by building algebraic objects called chain complexes on abstract simplicial complexes.



**Figure 3.1:** Abstract simplicial complex  $X$

**Definition 3.6 (Free  $k$ -Vector Space on  $X$ )** Let  $k$  be a field and  $X$  be a finite set. Then a free  $k$ -vector space on  $X$  is the vector space  $V_k(X)$  spanned by the elements of  $X$ , with pointwise sum and scalar multiplication.

Note that  $V_k(X)$  has a basis  $B_k(X)$  consisting of all the elements of  $X$ . In particular, the dimension of the vector space  $V_k(X)$  is  $|X|$ . Recall,  $\Sigma_i(X)$  denote the set of  $i$ -dimensional simplices of the abstract simplicial complex  $X$ .

**Definition 3.7 (Boundary Map)** Let  $V_k(\Sigma_i(X))$ , called the space of  $i$ -chains, be a free  $k$ -vector space on the  $i$ -simplices and let  $\sigma \in \Sigma_i(X)$  be an  $i$ -simplex. Then the boundary map is defined by  $\partial_i : V_k(\Sigma_i(X)) \rightarrow V_k(\Sigma_{i-1}(X))$ , with  $\partial_i(\sigma) = \sum_{i \in I} \tau_i$ , where  $I$  is the index set of all  $(i-1)$ -dimensional faces of  $\sigma$ . In addition, we define  $\partial_0 : V_k(\Sigma_0(X)) \rightarrow \{0\}$  to be the zero map.

As a map of finite dimensional free vector spaces,  $\partial_i$  can be written in the form of a matrix, where columns correspond to the  $i$ -dimensional simplices in  $\Sigma_i(X)$  and the rows correspond to the  $(i-1)$ -dimensional simplices in  $\Sigma_{i-1}(X)$ .

**Example 3.8** Let  $X$  be the abstract simplicial complex depicted in Figure 3.2. We have the following setup:  $X = (V(X), \Sigma(X))$  with

$$V(X) = \{A, B, C, D\}$$

and

$$\Sigma(X) = \{A, B, C, D, AB, AC, BC, BD, CD, ABC\}.$$

The 0-simplices  $\Sigma_0(X) = \{A, B, C, D\}$  coincide with the vertices. The 1-simplices are  $\Sigma_1(X) = \{AB, AC, BC, BD, CD\}$  and the 2-simplex is  $\Sigma_2(X) = \{ABC\}$ . The space of

0-chains  $V_k(\Sigma_0(X))$  is the free  $k$ -vector space with basis  $\Sigma_0(X)$ , as elements of  $V_k(\Sigma_0(X))$ . Analogue for  $V_k(\Sigma_1(X))$  and  $V_k(\Sigma_2(X))$ . The boundary maps can be described by

$$\partial_1 = \begin{matrix} & AB & AC & BC & BD & CD \\ \begin{matrix} A \\ B \\ C \\ D \end{matrix} & \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{matrix}, \quad \partial_2 = \begin{matrix} & ABC \\ \begin{matrix} AB \\ AC \\ BC \\ BD \\ CD \end{matrix} & \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \end{matrix}.$$

We see that the boundary map  $\partial_1 : V_k(\Sigma_1(X)) \rightarrow V_k(\Sigma_0(X))$  has  $\text{rank}(\partial_1) = 3$ , thus the kernel has  $\dim(\ker(\partial_1)) = 5 - \text{rank}(\partial_1) = 2$ . A basis for the kernel is given by the elements  $BC + BD + CD$  and  $BC + AB + AC$ .

The map  $\partial_2 : V_k(\Sigma_2(X)) \rightarrow V_k(\Sigma_1(X))$  sends the only 2-simplex  $ABC$  to the sum of its faces  $AB + AC + BC$ .

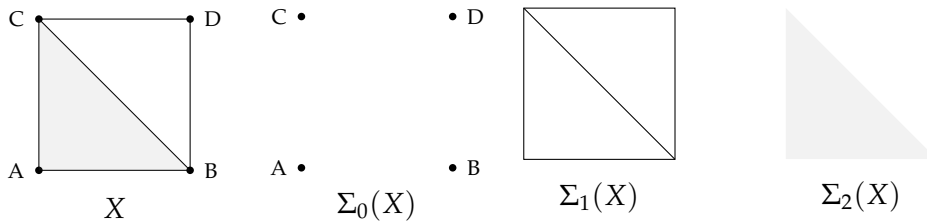


Figure 3.2: Abstract simplicial complex  $X$  and the sets of its  $i$ -simplices.

**Proposition 3.9** *The composition of two consecutive boundary maps vanishes, that is  $\partial_{i-1} \cdot \partial_i \equiv 0$ .*

*Proof.* The columns of  $\partial_{i-1} \cdot \partial_i$  correspond to the  $i$ -dimensional simplices in  $\Sigma_i(X)$  and the rows correspond to the  $(i - 2)$ -dimensional simplices in  $\Sigma_{i-2}(X)$ . The entry in the row of  $\tau'' \in \Sigma_{i-2}(X)$  and the column of  $\tau \in \Sigma_i(X)$  equals the number of  $\tau' \in \Sigma_{i-1}(X)$  such that  $\tau'' \subseteq \tau' \subseteq \tau$ . If  $\tau'' \not\subseteq \tau$  then there is no  $\tau'$  satisfying the condition. Finally, if  $\tau'' \subset \tau$  then there are always two  $\tau'_1, \tau'_2$  satisfying the condition, because any two  $(i - 1)$ -simplices in  $\tau$  share exactly one  $(i - 2)$ -simplex as a face.  $\square$



The boundary maps can be regarded as linear transformations. Considering the following diagram,

$$\dots \xrightarrow{\partial_{i+1}} V_k(\Sigma_i(X)) \xrightarrow{\partial_i} V_k(\Sigma_{i-1}(X)) \xrightarrow{\partial_{i-1}} \dots \xrightarrow{\partial_1} V_k(\Sigma_0(X)) \xrightarrow{\partial_0} \{0\}, \quad (3.1)$$

where the composition of two consecutive maps is the zero map, motivates the following definition.

**Definition 3.10 (Chain Complex)** A chain complex  $C$  is defined as  $k$ -vector spaces  $C_i$  for  $i \geq 0$ , together with linear transformations  $\partial_{i+1} : C_{i+1} \rightarrow C_i$  such that  $\partial_i \cdot \partial_{i+1} \equiv 0$ . The chain complex  $C(X)$  (or just  $C$  if the complex is clear from the context) of an abstract simplicial complex  $X$ , is then defined as the sequence of free  $k$ -vector spaces on  $\Sigma_i(X)$  with the corresponding boundary maps.

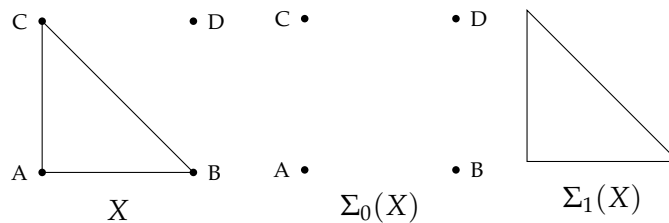
As we are only interested in chain complexes of abstract simplicial complexes, from now on  $C(X)$  denotes the chain complex on the  $i$ -chains  $V_k(\Sigma_i(X))$  and  $\partial_i$  denotes the boundary map  $\partial_i : V_k(\Sigma_i(X)) \rightarrow V_k(\Sigma_{i-1}(X))$ .

**Definition 3.11 (Boundary and Cycles)** For all  $i \geq 0$  define the boundaries as  $B_i = \text{im}(\partial_{i+1})$  and the cycles as  $Z_i = \ker(\partial_i)$ .

**Example 3.12** Note that we have the inclusion  $B_i \subseteq Z_i$ . This follows directly from Proposition 3.9, since  $\partial_i \cdot \partial_{i+1} \equiv 0$  implies that  $\text{im}(\partial_{i+1}) \subseteq \ker(\partial_i)$ .

**Definition 3.13 (Homology Group)** Let  $C(X)$  be as in Diagram 3.1. For  $i \geq 0$ , the  $i$ -th homology group of the abstract simplicial complex  $X$  is the quotient

$$H_i(X) = Z_i / B_i = \ker(\partial_i) / \text{im}(\partial_{i+1}).$$



**Figure 3.3:** Abstract simplicial complex  $X$  and the sets of its  $i$ -simplices.

**Example 3.14** Let  $X$  be the abstract simplicial complex depicted in Figure 3.3. Let  $X = (V(X), \Sigma(X))$  with  $V(X) = \{A, B, C, D\}$  and  $\Sigma(X) = \{A, B, C, D, AB, AC, BC\}$ . The 0-simplices are  $\Sigma_0(X) = \{A, B, C, D\}$ . The 1-simplices are  $\Sigma_1(X) = \{AB, AC, BC\}$ . The boundary map  $\partial_1 : V_k(\Sigma_1(X)) \rightarrow V_k(\Sigma_0(X))$  can be described by

$$\partial_1 = \begin{array}{c} \begin{array}{ccc} & AB & AC & BC \\ A & 1 & 1 & 0 \\ B & 1 & 0 & 1 \\ C & 0 & 1 & 1 \end{array} \\ \left( \begin{array}{ccc} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{array} \right), \end{array}$$

whereas  $\partial_2 : V_k(\Sigma_2(X)) \rightarrow V_k(\Sigma_1(X))$  and  $\partial_0 : V_k(\Sigma_0(X)) \rightarrow \{0\}$  are both zero maps. We compute the cycles  $Z_0 = \ker(\partial_0) = \langle A, B, C, D \rangle$  and the boundaries  $B_0 = \text{im}(\partial_1) = \langle A + B, A + C, B + C \rangle = \langle A + B, A + C \rangle$ . Thus,  $H_0 = Z_0/B_0 \cong \langle B, D \rangle$ . Intuitively, we identify all the vertices that are connected through edges. Furthermore, those vertices that are not connected through edges are not identified. It follows that the dimension of  $H_0(X)$  is the number of connected components of  $X$ .

Analogously, we compute  $H_1 = \langle AB + AC + BC \rangle$ . As to before, we can understand the dimension of  $H_1(X)$  as the number of 1-dimensional holes (loops) and, more generally, the dimension of  $H_n(X)$  as the number of  $n$ -dimensional holes in  $X$ .

Homology is functorial, i.e. it does not only assign vector spaces to complexes but maps to maps as well.

**Proposition 3.15 (Functoriality)** Let  $f : X \rightarrow Y$  be a map of abstract simplicial complexes, then:

1. There are linear transformations  $V_k(\Sigma_n(f)) : V_k(\Sigma_n(X)) \rightarrow V_k(\Sigma_n(Y))$ .
2. The boundary maps  $\partial_n$  and  $V_k(\Sigma_n(f))$  are homomorphisms such that the following diagram commutes.

$$\begin{array}{ccccc} \cdots & V_k(\Sigma_n(X)) & \xrightarrow{\partial_n} & V_k(\Sigma_{n-1}(X)) & \cdots \\ & \downarrow V_k(\Sigma_n(f)) & & \downarrow V_k(\Sigma_{n-1}(f)) & \\ \cdots & V_k(\Sigma_n(Y)) & \xrightarrow{\partial_n} & V_k(\Sigma_{n-1}(Y)) & \cdots \end{array}$$

3.  $V_k(\Sigma_n(f))$  maps boundaries  $B_n(X)$  to boundaries  $B_n(Y)$  and cycles  $Z_n(X)$  to cycles  $Z_n(Y)$ .

4. There is an induced homomorphism  $H_n(f) : H_n(X) \rightarrow H_n(Y)$ .

*Proof.* This proof is inspired by Chapter 2, Homotopy Invariance [7]. Let  $\sigma_i \in \Sigma_n(X)$  and  $\tau_j \in \Sigma_n(Y)$  for  $i \in I$  and  $j \in J$  be bases. Let  $\Sigma_n(f)$  be a map where an element  $\sigma_i$  gets mapped to  $\tau_j$ . Write  $\sigma = \sum_i \sigma_i \in V_k(\Sigma_n(X))$  and  $\tau = \sum_j \tau_j \in V_k(\Sigma_n(Y))$ . Let  $V_k(\Sigma_n(f))$  be the map where a basis element  $\sigma_i \in V_k(\Sigma_n(X))$  get mapped to a basis element  $\tau_j \in V_k(\Sigma_n(Y))$ . As a map between bases, we extend  $V_k(\Sigma_n(f))$  linearly,

$$V_k(\Sigma_n(f))\left(\sum_i \sigma_i\right) = \sum_i V_k(\Sigma_n(f))(\sigma_i).$$

We know the boundary map sends a simplex  $\sigma_i$  to the sum of all its faces  $\sum_j \sigma'_j$ . Because a map of abstract simplicial complexes maps faces to faces,

$$\begin{aligned} V_k(\Sigma_{n-1}(f))(\partial_n(\sigma_i)) &= V_k(\Sigma_{n-1}(f))\left(\sum_j \sigma'_j\right) \\ &= \sum_j V_k(\Sigma_{n-1}(f))(\sigma'_j) = \partial_n(V_k(\Sigma_n(f))(\sigma_i)). \end{aligned}$$

Thus  $V_k(\Sigma(f)) \circ \partial = \partial \circ V_k(\Sigma(f))$ .

Point 2. implies that cycles get mapped to cycles since

$$(V_k(\Sigma(f)) \circ \partial)(Z) = 0 = (\partial \circ V_k(\Sigma(f)))(Z).$$

Also, boundaries get mapped to boundaries, as they are the image of the boundary map,  $(V_k(\Sigma(f)) \circ \partial)(B) = (\partial \circ V_k(\Sigma(f)))(B)$ . This implies that  $(V_k(\Sigma(f)))$  induces a homomorphism  $H_n(f) : H_n(X) = Z_n(X)/B_n(X) \rightarrow H_n(Y) = Z_n(Y)/B_n(Y)$ .  $\square$

Homology groups are invariant under certain deformations of spaces, including homotopy equivalences. To formalize this, let us define homotopy and homotopy equivalence.

**Definition 3.16** Let  $f, g : X \rightarrow Y$  be two maps of topological spaces. We say  $f$  and  $g$  are homotopic, denoted as  $f \simeq g$ , if there is a continuous map  $H : X \times [0, 1] \rightarrow Y$  such that  $H(x, 0) = f(x)$  and  $H(x, 1) = g(x)$  for all  $x \in X$ .

The relation of being homotopic is an equivalence relation. Intuitively, two maps are homotopic if they are the same up to continuous deformation.

**Example 3.17** In  $\mathbb{R}^d$  the identity map on any connected subspace is homotopic to the zero map. In particular,  $H(\mathbf{x}, t) = (1 - t)\mathbf{x}$  is the homotopy between  $id_{\mathbb{R}^d}$  and the constant zero map.

**Definition 3.18 (Homotopy Equivalence [7])** *Two spaces  $X$  and  $Y$  are said to be homotopy equivalent if there are maps  $f : X \rightarrow Y$  and  $g : Y \rightarrow X$  such that  $f \circ g \simeq id_Y$  and  $g \circ f \simeq id_X$ . The maps  $f$  and  $g$  are called homotopy equivalences.*

**Remark 3.19** *Note that homotopy equivalence is an equivalence relation and that two homeomorphic spaces are also homotopy equivalent.*

An important property [7] is that, if we have a homotopy equivalence  $f : X \rightarrow Y$ , the induced maps  $H_n(f) : H_n(X) \rightarrow H_n(Y)$  are isomorphisms for all  $n$ . Using this, we can differentiate between spaces up to homotopy equivalence by computing and analysing their respective homology groups.

## Chapter 4

---

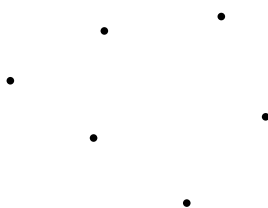
# Persistent Homology

---

Persistent homology theory extends classical homology to point cloud data. By tracking topological features as a proximity parameter between points varies, it provides a description of the data's underlying geometric properties. We present constructions of different *complexes* on which we can construct *filtrations* on, state and prove the structure theorem and show a concrete example. This chapter is based on the book 'Topological pattern recognition for point cloud data' by Gunnar Carlsson [6].

### 4.1 Construction of Different Complexes

If we have point cloud data, it is not immediately clear how to construct a complex that accurately represents the underlying structure. We look at constructions such as *Nerve*, *Rips*,  $\alpha$ -, and *witness complexes*. In addition to [6], we also use results from 'Computational Topology for Data Analysis' by Tamal Krishna Dey [8].



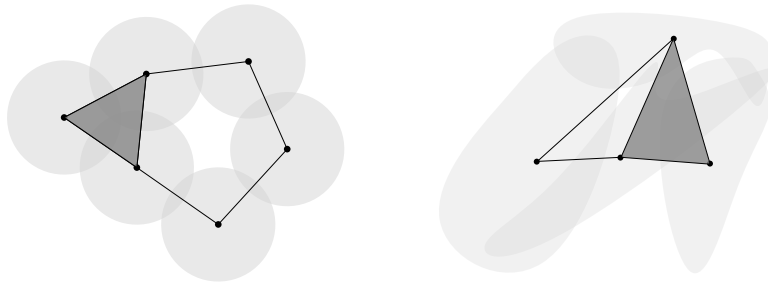
**Figure 4.1:** Dataset.

Consider the data set in Figure 4.1. To capture the shape of the point cloud, we replace points with open sets that contain them, leading to the nerve construction.

**Definition 4.1 (Nerve [8])** Let  $\mathcal{U} = \{U_x\}_{x \in X}$  be a finite collection of sets within a Euclidean space. The nerve  $N(\mathcal{U})$  is the abstract simplicial complex whose vertex set is the index set  $X$ , and a subset  $\{x_0, \dots, x_n\} \subseteq X$  is a  $n$ -simplex if

$$U_{x_0} \cap \dots \cap U_{x_n} \neq \emptyset.$$

**Example 4.2** Consider the two data sets given in Figure 4.2. On the left is the nerve complex  $N(\mathcal{U}_1)$ , constructed using the open balls  $B_r(x)$  with radius  $r$  around each point  $x$ . On the right is the nerve  $N(\mathcal{U}_2)$  using arbitrary open neighbourhoods for each point  $x$ . When two sets coincide, we construct a 1-simplex consisting of the corresponding points, i.e. an edge. If three sets coincide, we construct a 2-simplex consisting of the corresponding points, i.e. a triangle. We do this for all dimensions  $n$ . Note that the dimension of the simplices can be as big as  $|X| - 1$ .



**Figure 4.2:** Nerve  $N(\mathcal{U})$  for open balls and some arbitrary open sets.

While the nerve complex captures the structure of overlapping sets, choosing different sets can lead to different complexes on the same data. There are sufficient conditions on the open cover  $\mathcal{U}$  of a topological space  $X$  for the nerve theorem [9] that guarantee  $H_n(N(\mathcal{U})) \cong H_n(X)$ . To provide an alternative to the nerve complex, we introduce the Rips complex, which simplifies the construction by considering simplices formed from pairwise distances.

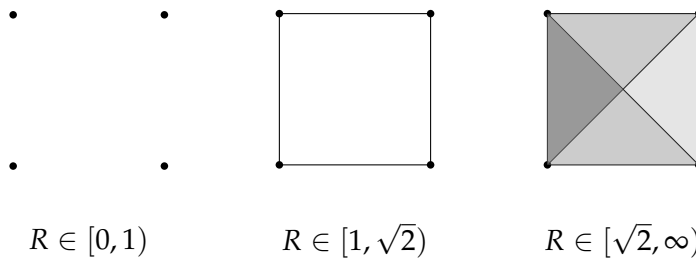
**Definition 4.3 (Vietoris-Rips Complex)** Let  $(X, d)$  be a finite metric space and  $R \geq 0$ . We then define the Vietoris-Rips complex or just Rips complex  $VR(X, R)$ , as the abstract simplicial complex such that any subset  $\{x_0, \dots, x_n\} \subseteq X$  is a simplex if

$$d(x_i, x_j) \leq R \text{ for all } i, j \in \{0, \dots, n\}.$$

In Euclidean space, the 1-simplices of  $VR(X, R)$  coincides with the 1-simplices of the nerve  $N(\mathcal{U})$  with  $\mathcal{U} = \{\overline{B_{R/2}(x_i)}\}_{x_i \in X}$ . For  $R = 0$ , the Rips complex consists

of the points of the metric space  $(X, d)$ . By increasing the radius  $R$ , the Rips complex then consists of sufficiently close points in a given neighbourhood as higher dimensional simplices.

**Example 4.4** Let us look at the Rips complexes in Figure 4.3.  $X = \{A, B, C, D\}$  consists of four points, where  $A = (0, 0)$ ,  $B = (1, 0)$ ,  $C = (1, 1)$ ,  $D = (0, 1)$ . For  $R \in [0, 1)$  none of the points have distance  $d(x_i, x_j) \leq R$ , thus  $VR(X, R) = \{A, B, C, D\}$ . For  $R \in [1, \sqrt{2})$  we add all the edges  $\{AB, BC, CD, AD\}$ . Lastly, for  $R \in [\sqrt{2}, \infty)$ ,  $VR(X, R)$  consists of the simplex  $ABCD$  and all its faces.



**Figure 4.3:** Family of Rips complexes for increasing  $R$ .

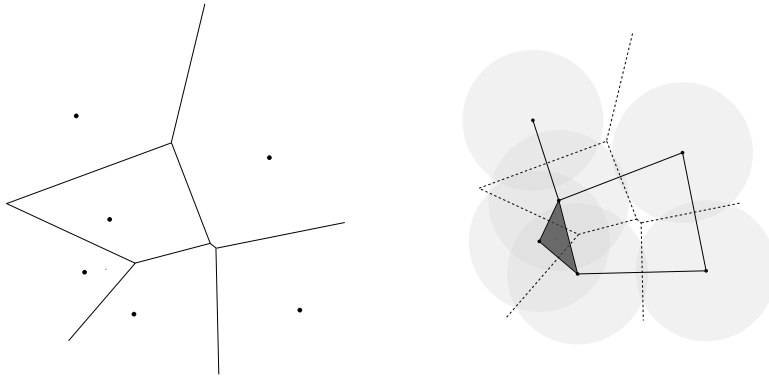
Though nerve and Rips complexes are useful and intuitive constructions, they are often large. The  $\alpha$ -complex is an example of a sparse complex. The construction of  $\alpha$ -complexes is performed on a metric space  $X$  which is a subspace of a metric space  $Y$ . Typically,  $Y$  is Euclidean space  $\mathbb{R}^N$ . Most often  $N$  is small, that is,  $N \in \{2, 3, 4\}$ .

**Definition 4.5 (Voronoi Cell)** Let  $(X, d)$  be a finite metric space and  $X \subseteq Y$  be a subspace of a larger metric space  $(Y, d)$  (typically chosen as  $\mathbb{R}^d$ , for a small  $d$ ). The Voronoi cell  $V(x)$  of the point  $x \in X$  is given by

$$V(x) = \{y \in Y \mid d(x, y) \leq d(x', y) \text{ for all } x' \in X\}.$$

The collection of all Voronoi cells of  $X$  in the Euclidean space  $\mathbb{R}^d$  is called a Voronoi diagram.

**Definition 4.6 ( $\alpha$ -Complex)** We define an  $\alpha$  cell together with a scale parameter  $\varepsilon$  as  $A_\varepsilon(x) = \overline{B_\varepsilon(x)} \cap V(x)$ . The  $\alpha$ -complex with scale parameter  $\varepsilon$  is the nerve  $N(\mathcal{A})$  with  $\mathcal{A} = \{A_\varepsilon(x)\}_{x \in X}$ .



**Figure 4.4:** Voronoi diagram and  $\alpha$ -complex.

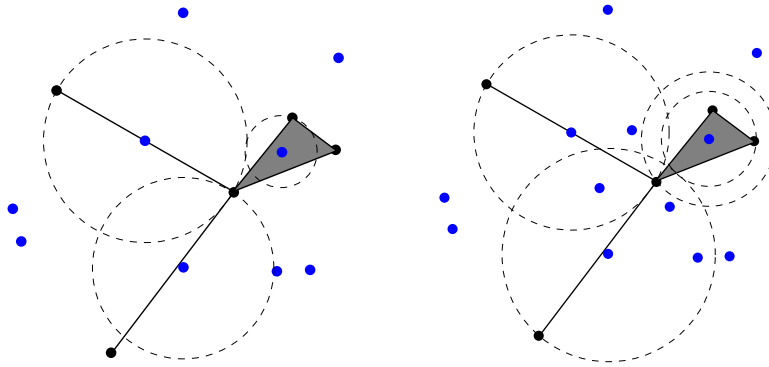
**Example 4.7** Consider the data set  $X$  in Figure 4.4. On the left we see the Voronoi diagram given the points  $X$ . On the right we see the corresponding  $\alpha$ -complex for a given scale parameter  $\epsilon$ . If two Voronoi cells share an edge, we can construct a 1-simplex with the two points  $x$  and  $x'$  when  $B_\epsilon(x) \cap B_\epsilon(x') \neq \emptyset$ . The same holds for higher dimensions. Note that all simplices are of dimension  $\leq d$  for  $Y = \mathbb{R}^d$ .

Another construction that yields smaller complexes is the witness complex in its various forms. The idea is to use a version of the Voronoi diagram on the data set  $X$  itself, rather than on a space in which  $X$  is embedded. The vertex set of the complex constructed is smaller than  $X$ , consisting of a set of *landmark* points within  $X$ . This means, we may select the size of the complex we are willing to work with.

**Definition 4.8 (Strong Witness Complex)** Let  $(X, d)$  be a metric space and suppose  $\mathcal{L} \subseteq X$  is a finite set of points called the landmark set and let  $\epsilon > 0$ . Denote  $m_x$  as the (minimal) distance from a point  $x \in X$  to the landmark set  $\mathcal{L}$ . We define the strong witness complex  $W^s(X, \mathcal{L}, \epsilon)$  as the abstract simplicial complex with vertex set  $\mathcal{L}$ , where  $\{l_0, \dots, l_n\} \subseteq \mathcal{L}$  spans a  $n$ -simplex if there is a point  $x \in X$ , the witness, such that  $d(x, l_i) \leq m_x + \epsilon$  for all  $i \in \{0, \dots, n\}$ .

**Definition 4.9 (Weak Witness Complex)** Let  $(X, d)$ ,  $\mathcal{L}$  and  $\epsilon$  be as above. A point  $x \in X$  is called a weak witness for  $\Lambda = \{l_0, \dots, l_n\} \subseteq \mathcal{L}$  if  $d(x, l) \geq d(x, l_i)$  for all  $i \in \{0, \dots, n\}$  and all  $l \notin \Lambda$ , and it is called a  $\epsilon$ -weak witness if  $d(x, l) + \epsilon \geq d(x, l_i)$  for all  $i \in \{0, \dots, n\}$  and all  $l \notin \Lambda$ . We then define a weak witness complex  $W^w(X, \mathcal{L}, \epsilon)$  to be the abstract simplicial complex with vertex set  $\mathcal{L}$  where  $\Lambda$  spans a  $n$ -simplex if  $\Lambda$  and all its faces admit an  $\epsilon$ -weak witness.





**Figure 4.5:** Left is a strong witness complex and right a weak witness complex.

**Example 4.10** In Figure 4.5 we can see a strong witness complex on the left and a weak witness complex on the right. Landmarks are depicted as black points and the rest of the metric space  $X$  as blue points. The strong witness complex can be accomplished by choosing  $\varepsilon$  to be the distance from the smallest dashed circle to the landmark in the middle. In the left setup, we would not end up with the complex depicted for the same  $\varepsilon$ , when constructing a strong witness complex. The complex depicted is a weak witness complex.

## 4.2 Persistence Vector Spaces and Structure Theorem

We now apply  $H_n$  to families of complexes for increasing threshold parameter. In the case of the Rips complex, we want to apply  $H_n$  to  $\{VR(X, R)\}_{R \in \mathbb{R}}$  to obtain the family  $\{H_n(VR(X, R))\}_{R \in \mathbb{R}}$  of vector spaces. In the following, we define persistence vector spaces and state and prove the structure theorem.

**Definition 4.11 (Filtration)** A family of complexes defined as  $\{X_r\}_r$  such that  $X_r \subseteq X_{r'}$  for  $r \leq r'$  is called a filtration.

**Definition 4.12 (Persistence Vector Space)** Let  $k$  be a field. A persistence vector space over  $k$  is a family of  $k$  vector spaces  $\{V_r\}_{r \in \mathbb{R}}$  together with linear transformations  $L_V(r, r') : V_r \rightarrow V_{r'}$  for  $r \leq r'$  such that  $L_V(r', r'') \cdot L_V(r, r') = L_V(r, r'')$  for all  $r \leq r' \leq r''$ .

**Example 4.13** The family of Rips complexes  $\{VR(X, R)\}_{R \in \mathbb{R}}$  defines a filtration. In fact, all the complexes defined in Section 4.1 define filtrations for increasing scale parameter.

**Definition 4.14 (Sub-Persistence Vector Space)** Let  $U_r \subseteq V_r$  be a choice of  $k$ -subspaces for all  $r \in \mathbb{R}$ . It is called a sub-persistence vector space of  $\{V_r\}_r$  if  $L_V(r, r')(U_r) \subseteq U_{r'}$  for all  $r \leq r'$ .

**Definition 4.15 (Linear Transformation)** A linear transformation  $f$  of persistence vector spaces over  $k$  from  $\{V_r\}_r$  to  $\{W_r\}_r$  is a family of linear transformations  $f_r : V_r \rightarrow W_r$ , such that for all  $r \leq r'$  the diagram below commutes.

$$\begin{array}{ccc} V_r & \xrightarrow{L_V(r, r')} & V_{r'} \\ f_r \downarrow & & \downarrow f_{r'} \\ W_r & \xrightarrow{L_W(r, r')} & W_{r'} \end{array}$$

**Example 4.16** For any linear transformation  $f$  of persistence vector spaces, the images  $\{\text{im}(f)\}_r$  denote a sub-persistence vector space.

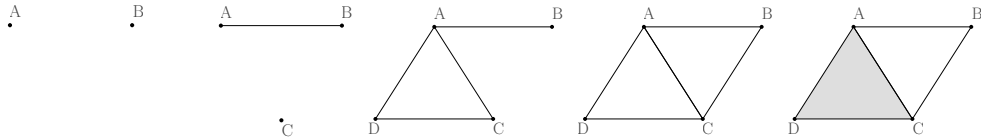
**Definition 4.17 (Quotient Space)** Let  $\{U_r\}_r \subseteq \{V_r\}_r$  be a sub-persistence vector space. The persistence vector space  $\{V_r/U_r\}_r$  together with the induced linear transformation  $L_{V/U}(r, r') : V_r/U_r \rightarrow V_{r'}/U_{r'}$ , given by  $[v] \mapsto [L_V(r, r')(v)]$  for  $v \in V_r$  is called a quotient space.

**Definition 4.18 ( $\mathbb{R}_+$ -Filtered Set)** Let  $X$  be a set and  $\rho : X \rightarrow [0, +\infty)$  be a positive map. The pair  $(X, \rho)$  is called a  $\mathbb{R}_+$ -filtered set. The free persistence vector space  $\{V_k(X, \rho)\}_r$  is then defined as  $V_k(X, \rho)_r \subseteq V_k(X)$ , where  $V_k(X, \rho)_r$  is given by the  $k$ -linear span of the set  $X_r := \rho^{-1}([0, r])$ .

**Remark 4.19** If  $X$  is finite, then for some large enough  $r$  we obtain  $V_k(X, \rho)_r = V_k(X)$ , since  $X_r = X$  holds for  $r = \max \rho(x)$ .

**Example 4.20** Let  $X = (V(X), \Sigma(X))$  be an abstract simplicial complex with vertices  $V(X) = \{A, B, C, D\}$  and simplices  $\Sigma(X) = \{A, B, C, D, AB, AC, AD, BC, CD, ACD\}$  and let  $\rho : X \rightarrow [0, +\infty)$  be a map such that we get a filtration as in Figure 4.6. We now compute the persistence vector spaces for the corresponding  $n$ -chains  $C_n(X)$ :

$$\begin{aligned}
C_0(X)_r &= \begin{cases} \langle A, B \rangle & \text{for } r \in [0, 1) \\ \langle A, B, C \rangle & \text{for } r \in [1, 2) \\ \langle A, B, C, D \rangle & \text{for } r \in [2, +\infty) \end{cases} \\
C_1(X)_r &= \begin{cases} \{0\} & \text{for } r \in [0, 1) \\ \langle AB \rangle & \text{for } r \in [1, 2) \\ \langle AB, AC, AD, CD \rangle & \text{for } r \in [2, 3) \\ \langle AB, AC, AD, BC, CD \rangle & \text{for } r \in [3, +\infty) \end{cases} \\
C_2(X)_r &= \begin{cases} \{0\} & \text{for } r \in [0, 4) \\ \langle ACD \rangle & \text{for } r \in [4, +\infty). \end{cases}
\end{aligned}$$



**Figure 4.6:** Filtration of the complex  $X$  for times  $r = 0, \dots, 4$ .

**Definition 4.21** A persistence vector space  $\{V_r\}_r$  is called *free* if it is isomorphic to a free persistence vector space of the form  $\{V_k(X, \rho)_r\}_r$ . It is called *finitely generated* if  $X$  can be chosen to be finite. It is called *finitely presented* if it is isomorphic to a persistence vector space of the form  $\{W_r\}_r / \text{im}(f)$  for some linear transformation  $f : \{V_r\}_r \rightarrow \{W_r\}_r$  between finitely generated free persistence vector spaces.

**Example 4.22** Recall the persistence vector spaces for the  $n$ -chains  $C_n(X)$  from Example 4.20. Since  $X$  is a  $\mathbb{R}_+$ -filtered set, the  $C_n(X)$  are free. Since  $X$  is a finite set, the  $C_n(X)$  are finitely generated and since we can choose  $f$  to be the zero map, the  $C_n(X)$  are finitely presented.

**Example 4.23 (Interval Persistence Vector Space)** Let  $k$  be a field,  $a \in \mathbb{R}_+$  and  $b \in \mathbb{R}_+ \cup \{+\infty\}$  with  $a < b$ . The persistence vector space  $P(a, b) = \{P(a, b)_r\}_{r \in \mathbb{R}_+}$  with

$$P(a, b)_r = \begin{cases} k & \text{for } r \in [a, b) \\ \{0\} & \text{otherwise,} \end{cases}$$

together with the linear transformations  $L_{P(a,b)}(r, r')$  for  $r \leq r'$  given by

$$L_{P(a,b)}(r, r') = \begin{cases} id_k & \text{for } r \leq r' \in [a, b) \\ 0 & \text{otherwise} \end{cases},$$

resembles a finitely presented persistence vector space. In particular,

$$P(a, b) \cong \{V_k(X, \rho)_r\}_r / \text{im}(f),$$

where  $X = \{x\}$ ,  $\rho(x) = a$  and  $f_r : V_k(X, \rho)_r \rightarrow V_k(X, \rho)_r$ ,  $x \mapsto x$  for all  $r \in [b, +\infty)$  and  $x \mapsto 0$  otherwise.

**Proposition 4.24** *The linear combination  $\sum_x a_x x \in V_k(X)$  lies in  $V_k(X, \rho)_r$  if and only if  $a_x = 0$  for all  $x$  with  $\rho(x) > r$ .*

The proof follows directly from the definition of  $\{V_k(X, \rho)_r\}_r$ , as it is the  $k$ -linear span of the persistence set  $X_r = \rho^{-1}([0, r])$ .

The choice of a basis for vector spaces  $V_r$  and  $W_r$  allow us to represent linear transformations from  $\{V_r\}_r$  to  $\{W_r\}_r$  by matrices. By choosing  $\{V_r\}_r$  and  $\{W_r\}_r$  to be persistence vector spaces of the form  $\{V_k(Y, \sigma)_r\}_r$  and  $\{V_k(X, \rho)_r\}_r$  such matrices can be described as follows.

**Definition 4.25 (( $\rho, \sigma$ )-Adapted)** *Let  $(X, Y)$  be a pair of finite sets and  $k$  a field. An  $(X, Y)$ -matrix is given by an array  $[a_{xy}]$  over  $k$ . For two  $\mathbb{R}_+$ -filtered sets  $(X, \rho)$  and  $(Y, \sigma)$ , a  $(X, Y)$ -matrix is called  $(\rho, \sigma)$ -adapted if  $a_{xy} = 0$  whenever  $\rho(x) > \sigma(y)$ .*

**Example 4.26** *Let us revisit the complex  $X$  from Example 4.20. Denote by  $\rho_i$  the restriction of  $\rho$  to the  $i$ -simplices. The boundary maps of  $X$  to are,*

$$\partial_1 = \begin{matrix} & AB & AC & AD & BC & CD \\ \begin{matrix} A \\ B \\ C \\ D \end{matrix} & \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \end{matrix}, \quad \partial_2 = \begin{matrix} & & & & ACD \\ \begin{matrix} AB \\ AC \\ AD \\ CD \\ BC \end{matrix} & \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} \end{matrix}.$$

Since an  $i$ -simplex  $\tau$  can only appear after all of its faces have appeared in the filtration,  $\rho_i(\tau) \geq \rho_{i-1}(\tau_m)$  for all  $\tau_m \subset \tau$ . Thus, the boundary map  $\partial_1$  is  $(\rho_0, \rho_1)$ -adapted and  $\partial_2$  is  $(\rho_1, \rho_2)$ -adapted. More generally, a boundary map  $\partial_i$  is  $(\rho_{i-1}, \rho_i)$ -adapted for all  $i > 0$ .

For any pair of finite sets  $(X, Y)$  over  $k$  and any  $(X, Y)$ -matrix, we denote the row corresponding to the vertex  $x \in X$  by  $r(x)$  and the column corresponding to the vertex  $y \in X$  by  $c(y)$ . Assume we have finitely generated free persistence vector spaces  $\{V_k(X, \rho)_r\}_r$  and  $\{V_k(Y, \sigma)_r\}_r$ . By Remark 4.19 we know that  $\{V_k(X, \rho)_r\}_r$  and  $\{V_k(Y, \sigma)_r\}_r$  attain their maximum, i.e. for  $r = \max\{\max \rho(x), \max \sigma(y)\}$  we get that  $\{V_k(X, \rho)_r\}_r = V_k(X)$  and  $\{V_k(Y, \sigma)_r\}_r = V_k(Y)$ . Thus, for any linear transformation  $f : \{V_k(Y, \sigma)_r\}_r \rightarrow \{V_k(X, \rho)_r\}_r$  of finitely generated free persistence vector spaces,  $f$  gives a linear transformation  $f_\infty : V_k(Y) \rightarrow V_k(X)$  between finite dimensional vector spaces, that sends the basis elements  $\{x\}_{x \in X}$  to  $\{y\}_{y \in Y}$ . The linear transformation  $f$  can be represented by a  $(X, Y)$ -matrix  $A(f) = [a_{xy}]$ .

**Proposition 4.27** *The  $(X, Y)$ -matrix  $A(f)$  is  $(\rho, \sigma)$ -adapted. Any  $(\rho, \sigma)$ -adapted  $(X, Y)$ -matrix  $A$  uniquely determines a linear transformation of persistence vector spaces*

$$f_A : \{V_k(Y, \sigma)_r\}_r \rightarrow \{V_k(X, \rho)_r\}_r.$$

*In particular, the correspondences  $f \mapsto A(f)$  and  $A \mapsto f_A$  are inverses to each other.*

*Proof.* Notice that the basis vector  $y$  lies in  $V_k(Y, \sigma)_{\sigma(y)}$ . Write

$$f(y) = \sum_{x \in X} a_{xy}x.$$

Using Proposition 4.24 we follow that  $f(y)$  lies in  $V_k(X, \rho)_{\rho(x)}$  if and only if  $a_{xy} = 0$ , for all  $\rho(x) > \sigma(y)$ , i.e.  $A(f)$  is  $(\rho, \sigma)$ -adapted.  $\square$

**Proposition 4.28** *Let  $(X, \rho)$  and  $(Y, \sigma)$  be  $\mathbb{R}_+$  filtered finite sets and let  $A$  be a  $(\rho, \sigma)$ -adapted  $(X, Y)$ -matrix. Then  $A$  determines a finitely presented persistent vector space  $\theta(A)$  via the map  $A \xrightarrow{\theta} V_k(X, \rho) / \text{im}(f_A)$ .*

*Proof.* This is an immediate consequence of Proposition 4.27 and the definition of finitely generated persistence vector spaces.  $\square$

**Example 4.29** *Let  $k$  be a field,  $a \in \mathbb{R}_+$  and  $b \in \mathbb{R}_+ \cup \{+\infty\}$  with  $a < b$ . Let  $(X, \rho)$  and  $(Y, \sigma)$  be  $\mathbb{R}_+$ -filtered sets where  $X = \{x\}$  and  $Y = \{y\}$  together with  $\rho(x) = a$  and  $\sigma(y) = b < +\infty$ . The persistence vector space of  $(X, \rho)$  is of the form  $\{V_k(X, \rho)_r\}_r$  with*

$$V_k(X, \rho)_r \cong \begin{cases} k & \text{for } r \geq a \\ \{0\} & \text{otherwise.} \end{cases}$$

We derive the  $(1 \times 1)$   $(\rho, \sigma)$ -adapted  $(X, Y)$ -matrix

$$A = \begin{matrix} & (y, b) \\ (x, a) & \left( \begin{array}{c} 1 \end{array} \right) \end{matrix},$$

where  $(x, a)$  denotes the row  $x$  appearing at time  $a$  and  $(y, b)$  denotes the column  $y$  appearing at time  $b$ . The matrix  $A$  maps  $y \in Y$  that appears at time  $\sigma(y) = b > a$  to  $x \in X$  that appears at time  $\rho(x) = a$ , thus we obtain

$$\text{im}(f_A)_r = \begin{cases} k & \text{for } r \geq b \\ \{0\} & \text{otherwise.} \end{cases}$$

We follow that

$$\theta(A)_r = (V_k(X, \rho) / \text{im}(f_A))_r = \begin{cases} k & \text{for } r \in [a, b) \\ \{0\} & \text{otherwise.} \end{cases}$$

Conclude that  $\theta(A) \cong P(a, b)$ . If we assume that  $b = +\infty$  then  $\text{im}(f_A)_r = \{0\}$  for all  $r$  and we get that  $\theta(A) \cong P(a, b) \cong V_k(X, \rho)$ .

The next statement is a criterion for when two finitely presented vector spaces are isomorphic. This will be an essential result to determine the persistent homology of filtrations.

**Proposition 4.30** *Let  $(X, \rho)$  and  $(Y, \sigma)$  be  $\mathbb{R}_+$ -filtered sets and let  $A$  be a  $(\rho, \sigma)$ -adapted  $(X, Y)$ -matrix. Let  $B$  be a  $(\rho, \rho)$ -adapted invertible  $(X, X)$ -matrix and  $C$  be a  $(\sigma, \sigma)$ -adapted invertible  $(Y, Y)$ -matrix. Then the matrix  $BAC$  is  $(\rho, \sigma)$ -adapted and the persistence vector space  $\theta(A)$  is isomorphic to  $\theta(BAC)$ .*

*Proof.* For the  $(\rho, \sigma)$ -adaptedness, we just show that  $BA$  is  $(\rho, \sigma)$ -adapted. The rest is calculated analogously. Let  $B = [b_{x'x}]$  and  $A = [a_{xy}]$  for  $x, x' \in X$  and  $y \in Y$ . The matrix  $BA = [d_{x'y}]$  has entries

$$d_{x'y} = \sum_{x \in X} b_{x'x} a_{xy}.$$

Since  $B$  is  $(\rho, \rho)$ -adapted and  $A$  is  $(\rho, \sigma)$ -adapted we have  $b_{x'x} = 0$  for  $\rho(x') > \rho(x)$  and  $a_{xy} = 0$  for  $\rho(x) > \sigma(y)$ . Let  $\rho(x') > \sigma(y)$ . Then  $\rho(x') > \rho(x)$  or  $\rho(x) > \sigma(y)$ , thus  $b_{x'x} = 0$  or  $a_{xy} = 0$ . In particular,  $d_{x'y} = 0$  for  $\rho(x') > \sigma(y)$ . We conclude that  $BA$  is  $(\rho, \sigma)$ -adapted.

We now want to show that  $\theta(A) \cong \theta(BAC)$ . Write  $(f_{BAC})_r = (f_B)_r \cdot (f_A)_r \cdot (f_C)_r$  for all  $r \in \mathbb{R}$ , or just  $f_{BAC} = f_B \cdot f_A \cdot f_C$ . Notice that  $(f_B)_r$  and  $(f_C)_r$  define isomorphisms on  $V_k(X, \rho)_r$  and  $V_k(Y, \sigma)$  respectively. Fix  $r \in \mathbb{R}$  and consider the vector space

$$\theta(A)_r = (V_k(X, \rho) / \text{im}(f_A))_r,$$

and

$$\theta(BAC)_r = (V_k(X, \rho) / \text{im}(f_{BAC}))_r.$$

Notice, it is enough to show that  $\text{im}((f_A)_r) \cong \text{im}((f_{BAC})_r)$ , which holds since

$$\begin{aligned} \text{im}((f_{BAC})_r) &\cong \text{im}((f_B)_r \cdot (f_A)_r \cdot (f_C)_r) \\ &\cong \text{im}((f_B)_r \cdot (f_A)_r) \end{aligned} \tag{1}$$

$$\begin{aligned} &\cong \text{im}((f_B)_{r|_{\text{im}((f_A)_r)}}) \\ &\cong \text{im}((f_A)_r). \end{aligned} \tag{2}$$

Isomorphism (1) holds since  $(f_C)_r$  defines an isomorphism. (2) holds since  $(f_B)_r$  defines an isomorphism.  $\square$

We now define operations that correspond to left-multiplication with matrices like  $B$  and right-multiplication with matrices like  $C$ .

**Definition 4.31 (Adapted Row and Column Operations)** *Let  $(X, \rho)$  and  $(Y, \sigma)$  be  $\mathbb{R}_+$ -filtered sets and let  $A$  be a  $(\rho, \sigma)$ -adapted  $(X, Y)$ -matrix.*

- For  $x, x' \in X$  with  $x \neq x'$  an adapted row operation adds a multiple of row  $r(x)$  to  $r(x')$ , when  $\rho(x) \geq \rho(x')$ , multiplies a row by a non-zero element or exchanges rows.
- For  $y, y' \in Y$  with  $y \neq y'$  an adapted column operation adds a multiple of column  $c(y)$  to  $c(y')$ , when  $\sigma(y) \leq \sigma(y')$ , multiplies a column by a non-zero element or exchanges columns.

The following two results are crucial to uniquely classify finitely presented persistence vector spaces. They serve as a cornerstone in analyzing persistent homology.

**Theorem 4.32 (Structure Theorem)** *Every finitely presented persistence vector space over  $k$  is isomorphic to a finite direct sum of the form*

$$P(a_1, b_1) \oplus \dots \oplus P(a_n, b_n)$$

for some choices of  $a_i \in \mathbb{R}_+$  and  $b_i \in \mathbb{R}_+ \cup \{+\infty\}$  with  $a_i < b_i$  for all  $i \in \{1, \dots, n\}$ .

*Proof.* Let  $A$  be a  $(\rho, \sigma)$ -adapted  $(X, Y)$ -matrix such that every row and every column has at most one non-zero element, which is equal to 1. Up to swapping rows or columns, we can assume that  $A$  is of block diagonal form such that,

$$A = \begin{pmatrix} I_n & 0 \\ 0 & 0 \end{pmatrix}.$$

Let  $\{(x_1, y_1), \dots, (x_n, y_n)\}$  be all the pairs such that  $a_{x_i y_i} = 1$ . We obtain

$$\begin{aligned} \theta(A) &= V_k(X, \rho) / \text{im}(f_A) \\ &= \bigoplus_{i \in \{1, \dots, n\}} V_k(x_i, \rho) / \text{im}(f_{A|_{\langle x_i \rangle}}) \oplus \bigoplus_{x \in X \setminus \{x_1, \dots, x_n\}} V_k(x, \rho) / \text{im}(f_{A|_{\langle x \rangle}}) \\ &\cong \bigoplus_{i \in \{1, \dots, n\}} P(\rho(x_i), \sigma(y_i)) \oplus \bigoplus_{x \in X \setminus \{x_1, \dots, x_n\}} P(\rho(x), +\infty), \end{aligned}$$

where the last isomorphism follows from Example 4.29. Therefore, it suffices to show that there exists a  $(\rho, \rho)$ -adapted invertible  $(X, X)$ -matrix  $B$  and a  $(\sigma, \sigma)$ -adapted invertible  $(Y, Y)$ -matrix  $C$  such that  $BAC$  is a  $(\rho, \sigma)$ -adapted  $(X, Y)$ -matrix such that every row and every column has at most one non-zero element, which is equal to 1. For this, we apply adapted row and column operations on  $A$ . Recall that adapted row and column operations consist of the following operations:

1. Multiply a row or a column by a non-zero element.
2. Add a multiple of row  $r(x)$  to  $r(x')$  when  $\rho(x) \geq \rho(x')$  and  $x \neq x'$ .
3. Add a multiple of column  $c(y)$  to  $c(y')$  when  $\sigma(y) \leq \sigma(y')$  and  $y \neq y'$ .

Over all  $y \in Y$  with at least one non-zero element in  $c(y)$ , we now choose  $y$  such that  $\sigma(y)$  is minimized. If there is no such  $y$ , we are already done. Next, over all  $x \in X$  such that  $a_{xy} \neq 0$ , we choose  $x$  such that  $\rho(x)$  is maximized. The way we have chosen  $x$ , we can add multiples of  $r(x)$  to all other rows. We can perform adapted row operations such that all entries in  $c(y)$  are zero except  $a_{xy}$ . The way we have chosen  $y$ , we can add multiples of  $c(y)$  to all other columns, so that all entries in  $r(x)$  are zero except  $a_{xy}$ . We now multiply  $r(x)$  (or  $c(x)$ ) by  $a_{xy}^{-1}$  to



get that  $r(x)$  and  $c(y)$  has exactly one non-zero element 1. By deleting  $r(x)$  and  $c(y)$ , we obtain a  $(\rho', \sigma')$ -adapted  $(X \setminus \{x\}, Y \setminus \{y\})$ -matrix where  $\rho'$  and  $\sigma'$  are the restrictions of  $\rho$  and  $\sigma$  to  $X \setminus \{x\}$  and  $Y \setminus \{y\}$ , respectively. By induction, we arrive at a matrix with only zero entries. The composition of all the previous operations gives us a  $(\rho, \rho)$ -adapted invertible  $(X, X)$ -matrix  $B$  and a  $(\sigma, \sigma)$ -adapted invertible  $(Y, Y)$ -matrix  $C$  such that  $BAC$  is a  $(\rho, \sigma)$ -adapted  $(X, Y)$ -matrix such that every row and column has at most one non-zero element, which is equal to 1. We conclude using Proposition 4.30.  $\square$

**Theorem 4.33 (Uniqueness)** *Suppose that  $\{V_r\}$  is a finitely presented persistence vector space over  $k$  and that we have two decompositions*

$$\{V_r\} \cong \bigoplus_{i \in I} P(a_i, b_i) \cong \bigoplus_{j \in J} P(c_j, d_j),$$

where  $|I|, |J| < +\infty$ . Then  $|I| = |J|$  and the set of pairs  $(a_i, b_i)$  with multiplicities equal the set of pairs  $(c_j, d_j)$  with multiplicities.

*Proof.* Let  $a_{\min}$  and  $c_{\min}$  denote the smallest value of  $a_i$  and  $c_i$ , respectively. Notice, we can write  $a_{\min} = \min\{r \mid V_r \neq 0\} = c_{\min}$ . Thus  $a_{\min} = c_{\min}$ . Now define the minima  $b_{\min} = \min\{b_i \mid a_i = a_{\min}\}$  and  $d_{\min} = \min\{d_i \mid c_i = c_{\min}\}$ . Notice, we can write  $b_{\min} = \min\{r' \mid \ker(L_V(r, r')) \neq 0\} = d_{\min}$ , thus  $b_{\min} = d_{\min}$ . This means  $P(a_{\min}, b_{\min}) = P(c_{\min}, d_{\min})$  both appear in decompositions. They can be characterized as the sub-persistence vector space  $\{W_r\} := \ker(L(r, b_{\min})|_{L(a_{\min}, r)})$  where

$$L(r, b_{\min})|_{L(a_{\min}, r)} : \text{im}(L(a_{\min}, r)) \rightarrow V_{b_{\min}}.$$

It now follows that the number of summands of the form  $P(a_{\min}, b_{\min})$  in both decompositions are the same, that is for  $I' := \{i \mid a_i = a_{\min} \text{ and } b_i = b_{\min}\}$  and  $J' := \{j \mid c_j = c_{\min} \text{ and } d_j = d_{\min}\}$  we have  $|I'| = |J'|$ . We then obtain the following identification.

$$\{V_r\} / \{W_r\} \cong \bigoplus_{i \in I \setminus I'} P(a_i, b_i) \cong \bigoplus_{j \in J \setminus J'} P(c_j, d_j),$$

We conclude by induction.  $\square$

The isomorphism classes of finitely presented persistence vector spaces can be represented as *barcodes*, which are families of intervals in  $\mathbb{R}$  and *persistence diagrams*, which are generally represented as points in the plane  $\mathbb{R}^2$  above the diagonal, that is as points in  $\{(x, y) \mid x \geq 0 \text{ and } y > x\}$ .

**Definition 4.34 (Barcode)** Let  $\{V_r\} \cong \bigoplus_{i=1}^n P(a_i, b_i)$  be a finitely presented persistence vector space. The barcode on  $\{V_r\}$  is then defined by the family of disjoint intervals  $[a_1, b_1), \dots, [a_n, b_n) \subseteq \mathbb{R}$ .

**Definition 4.35 (Persistence Diagram)** Let  $\{V_r\} \cong \bigoplus_{i=1}^n P(a_i, b_i)$  be a finitely presented persistence vector space. The persistence diagram on  $\{V_r\}$  is then defined by the collection of points  $\{(a_i, b_i) \mid i \in \{1, \dots, n\}\}$ . Points with  $b_i = +\infty$  are plotted above the diagram.

**Definition 4.36 (Persistent Homology Group)** For  $n \geq 0$  the persistent homology group of the filtration  $\{X_{\rho,r}\}_r$ , as a  $\mathbb{R}_+$ -filtered set with filtration function  $\rho : X \rightarrow \mathbb{R}_+$ , is the persistence quotient space

$$\{H_n(X_{\rho,r})\}_r = \{Z_n(X_{\rho,r})/B_n(X_{\rho,r})\}_r,$$

where  $Z_n(X_{\rho,r}) = \ker(\partial_n)$  and  $B_n(X_{\rho,r}) = \text{im}(\partial_{n+1})$ .

Note that, for finite sets  $X$ , we now can apply the structure theorem to obtain a unique identification

$$\{H_n(X_{\rho,r})\}_r \cong \bigoplus_{i \in I} P(a_i, b_i).$$

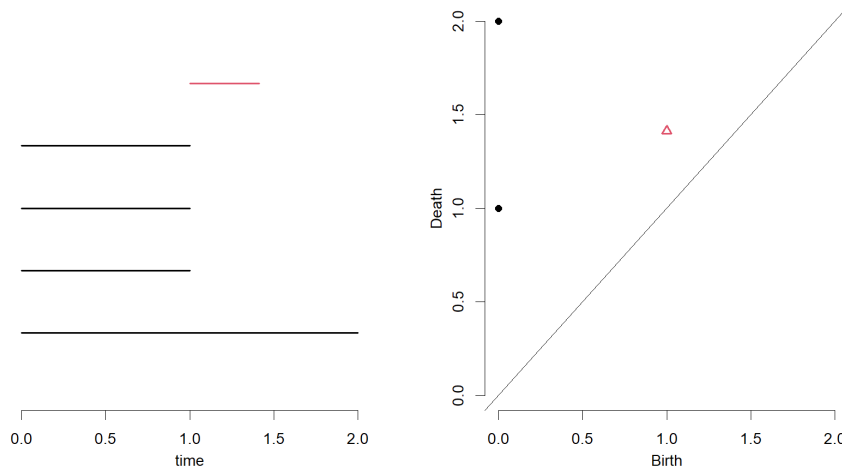
Intuitively, each summand  $P(a_i, b_i)$  represents a  $n$ -dimensional hole that persists in the interval  $[a_i, b_i)$ , meaning it appears at time  $a_i$  and disappears at time  $b_i$ .

**Example 4.37** Consider the Rips complex from Figure 4.3. Computing the persistent homology gives  $\{H_0(X_{\rho,r})\}_r \cong P(0, +\infty) \oplus P(0, 1)^3$ , and  $\{H_1(X_{\rho,r})\}_r \cong P(1, \sqrt{2})$ . The corresponding barcode and persistence diagram are plotted in Figure 4.7. We see that there are three connected components that persist for  $R \in [0, 1)$  and one that persists for  $R \in [0, +\infty)$ . We also see that there is one loop that persists for  $R \in [1, \sqrt{2})$ .

### 4.3 Bottleneck Distance and Stability Theorem

One important property to understand is how barcodes and persistence diagrams change when we have small changes in the data. In order to quantify small changes and the stability of barcodes, we define the bottleneck distance between barcodes and state the stability theorem for tame functions.

For any pair of intervals  $I = [a_1, b_1]$  and  $J = [a_2, b_2]$ , let  $\Delta(\cdot, \cdot)$  denote the  $l^\infty$ -distance, i.e.  $\Delta(I, J) = \max(|a_2 - a_1|, |b_2 - b_1|)$ . For a given interval  $I = [a, b]$ ,



**Figure 4.7:** Left is the barcode and right the persistence diagram of the homology groups computed for the complex in Figure 4.7.  $H_0$  is represented as black and  $H_1$  is represented as red.

let  $\lambda(I) = (b - a)/2$ .  $\lambda(I)$  is the  $l^\infty$ -distance to the closest interval of the form  $[c, c]$  to  $I$ .

**Definition 4.38 (Penalty and Bottleneck Distance)** Given two families  $\mathcal{I} = \{I_\alpha\}_{\alpha \in A}$  and  $\mathcal{J} = \{J_\beta\}_{\beta \in B}$  of intervals, for finite sets  $A$  and  $B$ , and any bijection  $\theta$  from a subset  $A' \subseteq A$  to  $B' \subseteq B$ , the penalty of  $\theta$  is

$$P(\theta) = \max \left( \max_{\alpha \in A'} \left( \Delta \left( I_\alpha, J_{\theta(\alpha)} \right) \right), \max_{\alpha \in A - A'} (\lambda(I_\alpha)), \max_{\beta \in B - B'} (\lambda(J_\beta)) \right)$$

and the bottleneck distance  $d_\infty(\mathcal{I}, \mathcal{J})$  is

$$\min_{\theta} P(\theta),$$

where the minimum is taken over all possible bijections from subsets of  $A$  to subsets of  $B$ .

Let  $X$  be a topological space, and let  $f : X \rightarrow \mathbb{R}$  be a real-valued function on  $X$ . For every non-negative integer  $n$ ,  $a \in \mathbb{R}$ , and  $\varepsilon \in (0, +\infty)$ , we have the induced map

$$j = j_{n,a,\varepsilon} : H_n(f^{-1}[a + \varepsilon, +\infty)) \rightarrow H_n(f^{-1}[a - \varepsilon, +\infty)).$$

**Definition 4.39 (Homological Critical Value and Tame)** We say that  $a$  is a homological critical value of  $f$  if there is a  $n$  such that  $j_{n,a,\varepsilon}$  fails to be an isomorphism for all sufficiently small  $\varepsilon$ . Further, we say that the function  $f$  is tame if it has a finite number of homological critical values and the homology groups  $H_n(f^{-1}[a, +\infty))$  are finite-dimensional for all  $n \in \mathbb{N}$  and  $a \in \mathbb{R}$ .

**Theorem 4.40 (Stability Theorem for Tame Functions)** Let  $X$  be any space homeomorphic to a simplicial complex, and suppose  $f, g : X \rightarrow \mathbb{R}$  are continuous tame functions. Then the persistence vector spaces  $\{H_n(f^{-1}([a, +\infty)))\}_a$  and  $\{H_n(g^{-1}([a, +\infty)))\}_a$  are finitely presented. We denote the barcodes as  $\beta_n f$  and  $\beta_n g$ . Moreover, for any  $n \in \mathbb{N}$ , we have that

$$d_\infty(\beta_n f, \beta_n g) \leq \|f - g\|_\infty.$$

A proof can be found in ‘Stability of Persistence Diagrams’ by Cohen-Steiner et al. [10].

## 4.4 Computing Persistent Homology

We now want to come back to the underlying problem of computing persistent homology. In the following section, we give an algorithm that simultaneously calculates  $\{Z_n(X_{\rho,r})\}_r = \ker(\partial_n)$  and  $\{B_n(X_{\rho,r})\}_r = \text{im}(\partial_{n+1})$  and in addition provides generators for the respective spaces to then compute persistent homology.

**Definition 4.41 (Admissible Operations)** Let  $(X, \rho)$ ,  $(Y, \sigma)$  and  $(Z, \tau)$  be  $\mathbb{R}_+$ -filtered sets. Let  $(A, B)$  be a pair of two matrices such that  $A$  is a  $(\rho, \sigma)$ -adapted  $(X, Y)$ -matrix,  $B$  is a  $(\sigma, \tau)$ -adapted  $(Y, Z)$ -matrix and  $A \cdot B = 0$ . Then admissible operations on such a pair consist of the following:

1. An arbitrary adapted row operation on  $A$ .
2. An arbitrary adapted column operation on  $B$ .
3. Perform a column operation on  $A$  and a row operation on  $B$  simultaneously, with the operations related as follows.
  - If the adapted column operation on  $A$  is a multiplication of the  $i$ -th column by a non-zero constant  $a$ , then the row operation on  $B$  is a multiplication of the  $i$ -th row by  $a^{-1}$ .

- If the adapted column operation on  $A$  is the transposition of two columns, then the row operation on  $B$  is the transposition of the corresponding rows of  $B$ .
- If the adapted column operation on  $A$  is the addition of  $a$  times the  $i$ -th column to the  $j$ -th column, then the adapted row operation on  $B$  is the subtraction of  $a$  times the  $j$ -th row from the  $i$ -th row.

**Proposition 4.42** Let  $(X, \rho)$ ,  $(Y, \sigma)$  and  $(Z, \tau)$  be  $\mathbb{R}_+$ -filtered sets. Let  $(A, B)$  be a pair of two matrices such that  $A$  is a  $(\rho, \sigma)$ -adapted  $(X, Y)$ -matrix,  $B$  is a  $(\sigma, \tau)$ -adapted  $(Y, Z)$ -matrix and  $A \cdot B = 0$ . By performing admissible operations on  $(A, B)$  we can obtain a pair  $(A', B')$  such that

$$(A', B') = \left( \begin{bmatrix} I_n & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & I_m \end{bmatrix} \right),$$

where there are  $n$ ,  $l$  and  $m$  columns in the leftmost, middle, and rightmost blocks of columns of  $A$  and consequently  $n$ ,  $l$  and  $m$  rows in the top, middle, and bottom blocks of rows of  $B$ .

*Proof.* We first perform arbitrary adapted row and column operations to  $A$ , making sure to apply the corresponding adapted row operations to  $B$  whenever an adapted column operation is applied to  $A$ , to obtain a pair  $(A', B')$  of the form

$$(A', B') = \left( \begin{bmatrix} I_n & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} B'_{11} & B'_{12} & B'_{13} \\ B'_{21} & B'_{22} & B'_{23} \\ B'_{31} & B'_{32} & B'_{33} \end{bmatrix} \right).$$

Because of the condition  $A' \cdot B' = 0$ , we follow that  $B'_{11} = B'_{12} = B'_{13} = 0$ . Write

$$B' = \begin{bmatrix} 0 & 0 & 0 \\ B'_{21} & B'_{22} & B'_{23} \\ B'_{31} & B'_{32} & B'_{33} \end{bmatrix}.$$

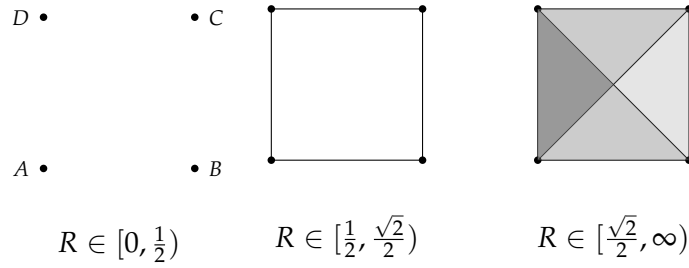
We perform only adapted row operations involving the last  $l + m$  rows, since the upper  $n$  rows are identically zero. Each such adapted row operation has a corresponding adapted column operation on the matrix  $A$  which affects only the rightmost  $l + m$  columns, and therefore has no effect. We now perform adapted operations to get a matrix of the form

$$B'' = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & I_m \end{bmatrix}.$$

That this representation is unique is clear from the fact that  $n$  and  $m$  are the ranks of the matrices  $A$  and  $B$ , respectively. We conclude.  $\square$

If we now have  $A \equiv \partial_i$  and  $B \equiv \partial_{i+1}$  and if there are  $n$ ,  $l$  and  $m$  columns in the leftmost, middle, and rightmost blocks of columns of  $A$  respectively and consequently  $n$ ,  $l$  and  $m$  rows in the top, middle, and bottom blocks of rows of  $B$ , respectively, then the dimension of the homology is  $l$ , since  $H_i = \ker(\partial_i) / \text{im}(\partial_{i+1})$ . Note that here we used that the row and column operations correspond to isomorphisms that imply  $\ker(A) \cong \ker(A')$  and  $\text{im}(B) \cong \text{im}(B')$ .

**Example 4.43** Let us consider the family of Rips complexes for increasing  $R$ , given by the vertices  $A = (0,0)$ ,  $B = (\frac{1}{2},0)$ ,  $C = (\frac{1}{2},\frac{1}{2})$ ,  $D = (0,\frac{1}{2})$ , similar to Figure 4.3.



This means, we have the abstract simplicial complex  $X = (V(X), \Sigma(X))$  with

$$V(X) = \{A, B, C, D\},$$

$$\Sigma(X) = \{A, B, C, D, AB, AC, AD, BC, BD, CD, ABC, ABD, ACD, BCD, ABCD\},$$

and the persistence vector spaces of the  $i$ -chains

$$\begin{aligned}
C_0(X)_r &= \langle A, B, C, D \rangle \quad \text{for } r \in [0, +\infty) \\
C_1(X)_r &= \begin{cases} \{0\} & \text{for } r \in [0, \frac{1}{2}) \\ \langle AB, AD, BC, CD \rangle & \text{for } r \in [\frac{1}{2}, \frac{\sqrt{2}}{2}) \\ \langle AB, AC, AD, BC, BD, CD \rangle & \text{for } r \in [\frac{\sqrt{2}}{2}, +\infty) \end{cases} \\
C_2(X)_r &= \begin{cases} \{0\} & \text{for } r \in [0, \frac{\sqrt{2}}{2}) \\ \langle ABC, ABD, ACD, BCD \rangle & \text{for } r \in [\frac{\sqrt{2}}{2}, +\infty) \end{cases} \\
C_3(X)_r &= \begin{cases} \{0\} & \text{for } r \in [0, \frac{\sqrt{2}}{2}) \\ \langle ABCD \rangle & \text{for } r \in [\frac{\sqrt{2}}{2}, +\infty). \end{cases}
\end{aligned}$$

In order to compute the persistent homology, we now perform admissible operations from Definition 4.41 as in Proposition 4.42 and 4.32. We perform operations on the boundary maps  $\partial_i$ , by first bringing  $\partial_i$  into block diagonal form  $\text{diag}(I_n, 0)$  and then bringing  $\partial_{i+1}$  to block diagonal form  $\text{diag}(0, I_m)$ . Note that we only update the generators for the 1-simplices on  $\partial_1$  and 2-simplices on  $\partial_2$ , since for calculating the persistent homology, we are only interested in them. But keep in mind that for getting the correct representation of our maps, we would also have to update all the 0- and 1-simplices of both matrices. When referring to the row of a 1-simplex, we mean the corresponding row of  $\partial_2$ . The calculations work as follows:

$$[\partial_1, \partial_2] =$$

$$\left[ \begin{array}{c} (A) \\ (B) \\ (C) \\ (D) \end{array} \begin{pmatrix} (AB, \frac{1}{2}) & (AD, \frac{1}{2}) & (BC, \frac{1}{2}) & (CD, \frac{1}{2}) & (AC, \frac{\sqrt{2}}{2}) & (BD, \frac{\sqrt{2}}{2}) \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \right],$$

$$\left[ \begin{array}{c} (ABC, \frac{\sqrt{2}}{2}) & (ABD, \frac{\sqrt{2}}{2}) & (ACD, \frac{\sqrt{2}}{2}) & (BCD, \frac{\sqrt{2}}{2}) \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{array} \right] \xrightarrow{r(B) \rightarrow r(B) + r(A)}$$

$$\left[ \begin{array}{c} (A) \\ (B) \\ (C) \\ (D) \end{array} \begin{pmatrix} (AB, \frac{1}{2}) & (AD, \frac{1}{2}) & (BC, \frac{1}{2}) & (CD, \frac{1}{2}) & (AC, \frac{\sqrt{2}}{2}) & (BD, \frac{\sqrt{2}}{2}) \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \right],$$

$$\left[ \begin{array}{c} (ABC, \frac{\sqrt{2}}{2}) & (ABD, \frac{\sqrt{2}}{2}) & (ACD, \frac{\sqrt{2}}{2}) & (BCD, \frac{\sqrt{2}}{2}) \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{array} \right] \xrightarrow{\begin{array}{l} c(AD) \rightarrow c(AD) + c(AB) \\ c(AC) \rightarrow c(AC) + c(AB) \\ \text{consequently} \\ r(AB) \rightarrow r(AD) + r(AC) \end{array}}$$

$$\left[ \begin{array}{cccccc} (AB, \frac{1}{2}) & (AD+AB, \frac{1}{2}) & (BC, \frac{1}{2}) & (CD, \frac{1}{2}) & (AC+AB, \frac{\sqrt{2}}{2}) & (BD, \frac{\sqrt{2}}{2}) \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix} & & & & & \end{array} \right],$$

$$\left[ \begin{array}{cccc} (ABC, \frac{\sqrt{2}}{2}) & (ABD, \frac{\sqrt{2}}{2}) & (ACD, \frac{\sqrt{2}}{2}) & (BCD, \frac{\sqrt{2}}{2}) \\ \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} & & & \end{array} \right] \xrightarrow{r(D) \rightarrow r(D)+r(B)}$$

$$\left[ \begin{array}{cccccc} (AB, \frac{1}{2}) & (AD+AB, \frac{1}{2}) & (BC, \frac{1}{2}) & (CD, \frac{1}{2}) & (AC+AB, \frac{\sqrt{2}}{2}) & (BD, \frac{\sqrt{2}}{2}) \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} & & & & & \end{array} \right],$$

$$\left[ \begin{array}{cccc} (ABC, \frac{\sqrt{2}}{2}) & (ABD, \frac{\sqrt{2}}{2}) & (ACD, \frac{\sqrt{2}}{2}) & (BCD, \frac{\sqrt{2}}{2}) \\ \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} & & & \end{array} \right] \xrightarrow{\begin{array}{l} c(BC) \rightarrow c(BC)+c(AD+AB) \\ c(AC+AB) \rightarrow c(AC+AB)+c(AD+AB) \\ c(BD) \rightarrow c(BD)+c(AD+AB) \\ \text{consequently} \\ r(AD+AB) \rightarrow r(AD+AB)+r(BC) \\ \quad +r(AC+AB)+r(BD) \end{array}}$$

Define  $c_0 := CD + BC + AD + AB$ ,  $c_1 := AC + BC + AB$  and  $c_2 := BD + AD + AB$ .

$$\left[ \begin{array}{cccccc} (AB, \frac{1}{2}) & (AD+AB, \frac{1}{2}) & (BC+AD+AB, \frac{1}{2}) & (CD, \frac{1}{2}) & (AC+AD, \frac{\sqrt{2}}{2}) & (c_2, \frac{\sqrt{2}}{2}) \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} & & & & & \end{array} \right],$$

$$\left[ \begin{array}{cccc} (ABC, \frac{\sqrt{2}}{2}) & (ABD, \frac{\sqrt{2}}{2}) & (ACD, \frac{\sqrt{2}}{2}) & (BCD, \frac{\sqrt{2}}{2}) \\ \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} & & & \end{array} \right] \xrightarrow{r(D) \rightarrow r(D)+r(C)}$$



$$\left[ \begin{array}{cccccc} (AB, \frac{1}{2}) & (AD+AB, \frac{1}{2}) & (BC+AD+AB, \frac{1}{2}) & (CD, \frac{1}{2}) & (AC+AD, \frac{\sqrt{2}}{2}) & (c_2, \frac{\sqrt{2}}{2}) \\ \left( \begin{array}{cccccc} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right) \end{array} \right],$$

$$\left( \begin{array}{cccc} (ABC, \frac{\sqrt{2}}{2}) & (ABD, \frac{\sqrt{2}}{2}) & (ACD, \frac{\sqrt{2}}{2}) & (BCD, \frac{\sqrt{2}}{2}) \\ \left( \begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{array} \right) \end{array} \right) \xrightarrow{\begin{array}{l} c(CD) \rightarrow c(CD) + c(BC+AD+AB) \\ c(AC+AD) \rightarrow c(AC+AD) \\ \quad + c(BC+AD+AB) \\ \hline \text{consequently} \\ r(BC+AD+AB) \rightarrow r(BC+AD+AB) \\ \quad + r(CD) + r(AC+AD) \end{array}} \right]$$

$$\left[ \begin{array}{cccccc} (AB, \frac{1}{2}) & (AD+AB, \frac{1}{2}) & (BC+AD+AB, \frac{1}{2}) & (c_0, \frac{1}{2}) & (c_1, \frac{\sqrt{2}}{2}) & (c_2, \frac{\sqrt{2}}{2}) \\ \left( \begin{array}{cccccc} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right) \end{array} \right],$$

$$\left( \begin{array}{cccc} (ABC, \frac{\sqrt{2}}{2}) & (ABD, \frac{\sqrt{2}}{2}) & (ACD, \frac{\sqrt{2}}{2}) & (BCD, \frac{\sqrt{2}}{2}) \\ \left( \begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{array} \right) \end{array} \right) \xrightarrow{\text{rearrange columns of } \partial_{i+1}} \right]$$

Now that  $\partial_1$  is of the desired form, we perform adapted operations on  $\partial_{i+1}$  that do not change  $\partial_i$ , apart from having to update generators of the kernel.

$$\left[ \begin{array}{cccccc} (AB, \frac{1}{2}) & (AD + AB, \frac{1}{2}) & (BC + AD + AB, \frac{1}{2}) & (c_0, \frac{1}{2}) & (c_1, \frac{\sqrt{2}}{2}) & (c_2, \frac{\sqrt{2}}{2}) \\ \left( \begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right) & & & & & \end{array} \right],$$

$$\left[ \begin{array}{cccc} (BCD, \frac{\sqrt{2}}{2}) & (ACD, \frac{\sqrt{2}}{2}) & (ABC, \frac{\sqrt{2}}{2}) & (ABD, \frac{\sqrt{2}}{2}) \\ \left( \begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{array} \right) & & & \end{array} \right] \xrightarrow{c(BCD) \rightarrow c(BCD) + c(ACD) + c(ABC) \\ + c(ABD) \quad c(ACD) \rightarrow c(ACD) + c(ABC)}$$

Define  $b_0 := BCD + ACD + ABC + ABD$

$$\left[ \begin{array}{cccccc} (AB, \frac{1}{2}) & (AD + AB, \frac{1}{2}) & (BC + AD + AB, \frac{1}{2}) & (c_0, \frac{1}{2}) & (c_1, \frac{\sqrt{2}}{2}) & (c_2, \frac{\sqrt{2}}{2}) \\ \left( \begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right) & & & & & \end{array} \right],$$

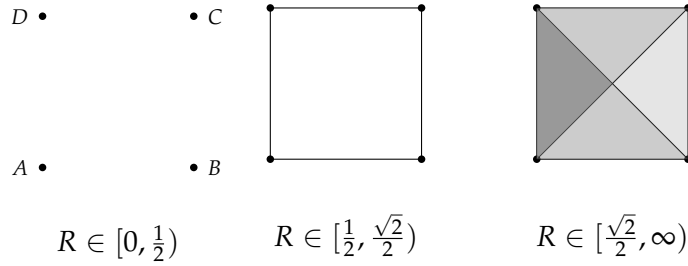
$$\left[ \begin{array}{cccc} (b_0, \frac{\sqrt{2}}{2}) & (ACD + ABC, \frac{\sqrt{2}}{2}) & (ABC, \frac{\sqrt{2}}{2}) & (ABD, \frac{\sqrt{2}}{2}) \\ \left( \begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array} \right) & & & \end{array} \right].$$

Recall that  $H_1 = \ker(\partial_1) / \text{im}(\partial_2)$ , thus, we can now read off the persistence vector space of the first homology group of the Rips-filtration of  $X$  by looking at the restriction  $\partial_2|_{\ker(\partial_1)}$ :

$$\begin{array}{c} (c_0, \frac{1}{2}) \\ (c_1, \frac{\sqrt{2}}{2}) \\ (c_2, \frac{\sqrt{2}}{2}) \end{array} \left( \begin{array}{cccc} (b_0, \frac{\sqrt{2}}{2}) & (ACD + ABC, \frac{\sqrt{2}}{2}) & (ABC, \frac{\sqrt{2}}{2}) & (ABD, \frac{\sqrt{2}}{2}) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array} \right).$$

Recall  $c_0 := CD + BC + AD + AB$ ,  $c_1 := AC + BC + AB$ ,  $c_2 := BD + AD + AB$  and  $b_0 := BCD + ACD + ABC + ABD$ . Since we are only interested in the image of

$\partial_2|_{\ker(\partial_1)}$ , we can simply ignore the column corresponding to  $b_0$ . We can now read off, when the cycles in the kernel of  $\partial_1$  appear and when they disappear. The cycle  $c_0$  appears at time  $\frac{1}{2}$  and corresponds to the sum of the edges of the square below. It disappears at time  $\frac{\sqrt{2}}{2}$  as it gets filled in by the sum of the two 2-simplices  $ACD + ABC$ . The two other cycles  $c_1$  and  $c_2$  appear at the same time as they get filled in by  $ABC$  and  $ABD$  respectively.



We conclude that

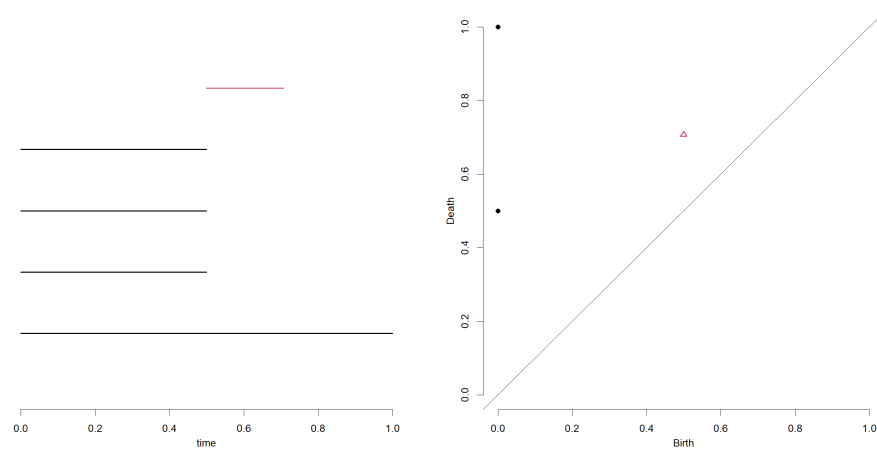
$$\{H_1(X_{\rho,r})\}_r \cong P\left(\frac{1}{2}, \frac{\sqrt{2}}{2}\right).$$

Similarly, we can calculate that

$$\{H_0(X_{\rho,r})\}_r \cong P(0, +\infty) \oplus \bigoplus_{i=0}^2 P\left(0, \frac{1}{2}\right) \cong P(0, +\infty) \oplus P\left(0, \frac{1}{2}\right)^3,$$

and for  $n > 1$

$$\{H_n(X_{\rho,r})\}_r \cong \{0\}.$$



**Figure 4.8:** Left is the barcode and right the persistence diagram of the homology groups computed in Example 4.43.  $H_0$  is represented as black and  $H_1$  is represented as red.

## Chapter 5

---

# Topological Mode Analysis Tool

---

After looking at examples of clustering schemes and introducing persistent homology, we now discuss a clustering scheme, called *topological mode analysis tool*, in short ToMATo. It combines ideas from the previous chapters, as it is a graph-based hill-climbing algorithm with a cluster merging step guided by persistence. This chapter is based on ‘Persistence-Based Clustering in Riemannian Manifolds’ by Frédéric Chazal, et al. [11]. The results we provide can be generalized from Euclidean spaces to Riemannian manifolds, together with the corresponding Riemannian metric and Hausdorff measure.

### 5.1 Continuous Setting

To give a better intuition, we first consider the continuous setting. Our goal is to find a suitable partitioning of the space  $U$  (in the discrete setting a clustering), given some density function  $f$ , that represents how likely it is that data points appear in some region of our space  $U$ . Now, let  $f : U \rightarrow \mathbb{R}$ , with  $U \subseteq \mathbb{R}^d$  be a  $C^2$ -continuous density function with non-degenerate critical points, that is, the Hessian

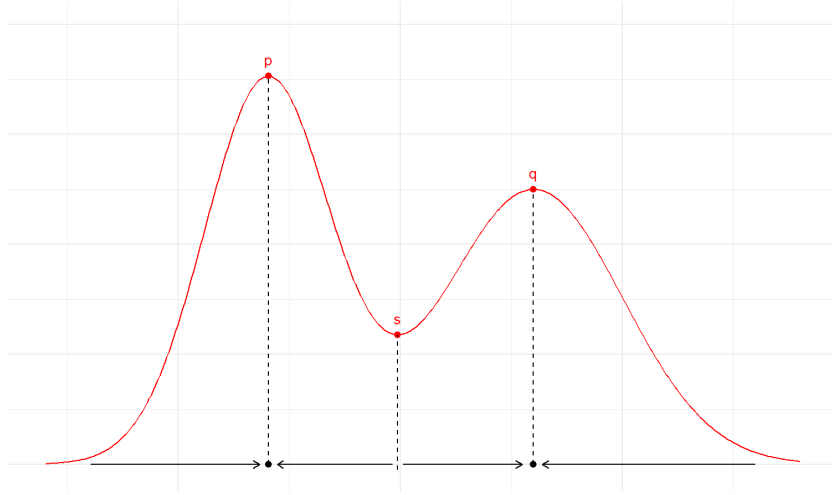
$$H_f(\mathbf{x}) := \left( \frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x}) \right)_{1 \leq i, j \leq d},$$

is invertible at all critical points. Assume  $f$  has a finite number of critical points. The goal now is to find a suitable partitioning of  $U$  that represents how we would cluster points that are sampled by the distribution according to  $f$ .

**Definition 5.1 (Ascending Region)** *The ascending region of a critical point  $m_p$ , denoted as  $A(m_p) \subseteq U$ , is the subset of points in Euclidean space that reach  $m_p$  by moving along*

the flow induced by the gradient vector field of  $f$ . For all  $\mathbf{x} \in A(m_p)$ , we call  $m_p$  the root of  $\mathbf{x}$ , denoted as  $r(\mathbf{x}) = m_p$ .

**Example 5.2** Revisiting Figure 2.5 the ascending regions are depicted with arrows, indicating the flow induced by the gradient vector field to the roots  $p$  and  $q$ . Note that  $s$  does not belong to any ascending region, since  $\nabla f(s) = 0$ . Assuming  $U$  does not have a boundary yields that it can be covered by ascending regions up to a set with measure zero.



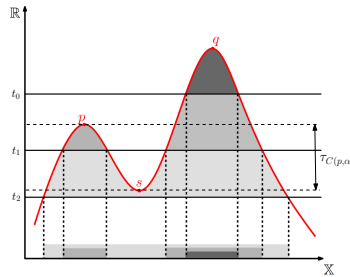
**Definition 5.3 (Superlevel-Sets)** The set  $\mathbb{F}^\alpha := f^{-1}([\alpha, +\infty))$  is called the superlevel-set of index  $\alpha \in \mathbb{R}$ . For any  $\mathbf{x} \in U$  and  $\alpha \in \mathbb{R}$  let  $C(\mathbf{x}, \alpha) \subseteq \mathbb{F}^\alpha$  denote the path-connected component of  $\mathbb{F}^\alpha$  containing  $\mathbf{x}$ .

**Definition 5.4 (Superlevel-Set Filtration)** We call the collection of superlevel-sets  $\{f^{-1}([\alpha, +\infty))\}_{\alpha \in \mathbb{R}_+}$  a superlevel-set filtration.

Intuitively, this filtration captures which subsets of  $U$  have higher probability to be drawn than others. More precisely, regions with higher density appear earlier in the filtration. Note, in order for  $\{\mathbb{F}^\alpha\}_\alpha$  to resemble a filtration, we have to let  $\alpha$  go from  $+\infty$  to  $-\infty$ . So for the scope of this chapter, we will let  $\alpha$  flow backwards.

**Example 5.5** In Figure 5.1 the superlevel-sets in the filtration can be distinguished by different shades of grey. Let  $m_p$  be the local maximum of  $f$  at point  $p$  and  $f(m_p) = \alpha_1$ . Looking at the homology group  $H_0(\{\mathbb{F}^\alpha\}_\alpha)$ , that counts the number of path-connected components, we observe that  $m_p$  is a generator of  $C(m_p, \alpha_1)$ , which appears at time  $\alpha_1$

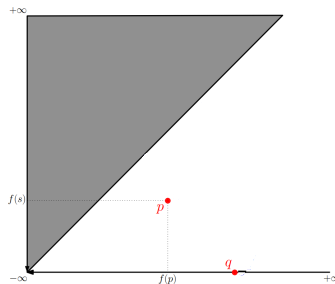
in the superlevel-set filtration  $\{\mathbb{F}^\alpha\}_\alpha$ . This component disappears when it gets connected to another path-connected component, generated by the higher peak  $m_q$ . This process is called merging. More precisely, this merging process merges  $C(p, \alpha)$  into  $C(q, \alpha)$  for  $\alpha = f(s)$ , while  $q$  still remains the generator, and  $p$  ceases to be a generator. Moreover, for  $\alpha \in [f^{-1}(q), f^{-1}(p))$ ,  $\mathbb{F}^\alpha$  consists of one component  $C(q, \alpha)$ . For  $\alpha \in [f^{-1}(p), f^{-1}(s))$ ,  $\mathbb{F}^\alpha$  consists of two components  $C(q, \alpha)$  and  $C(p, \alpha)$ . For  $\alpha \in [f^{-1}(p), f^{-1}(-\infty))$  again consists of one component  $C(q, \alpha)$ .



**Figure 5.1:** Filtration of superlevel-sets of density function  $f$  with two peaks [11].

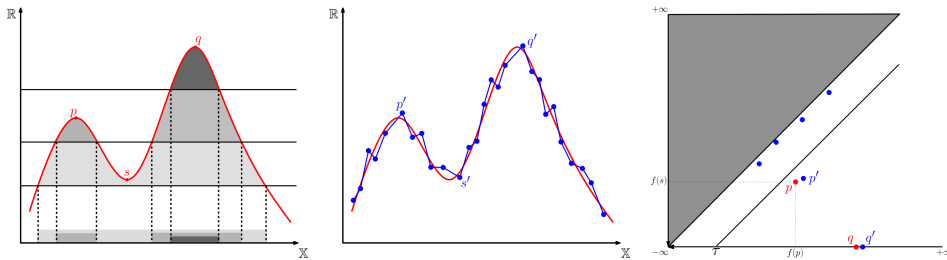
**Definition 5.6 (Prominence)** Assume we have a persistent homology group with the decomposition arising from the superlevel-set filtration  $\{H_0(\mathbb{F}^\alpha)\}_\alpha \cong \bigoplus_{i \in I} P(a_i, b_i)$ , where  $P(a_i, b_i)$  that persists in the interval  $[a_i, b_i)$ . The prominence  $\tau_i$  of  $P(a_i, b_i)$  is given by  $\tau = b_i - a_i \geq 0$ .

**Example 5.7** Consider the persistence diagram from Figure 5.2.  $p = (p_x, p_y)$  represents a persistence vector space  $P(p_x, p_y)$ , the prominence is then given by  $\tau = p_x - p_y > 0$ , since we have an inverted flow of time. Equivalently, the prominence is the height of the corresponding component as can be seen in Figure 5.1.



**Figure 5.2:** Persistence diagram of the filtration of superlevel-sets of density function  $f$  from Figure 5.1 [11].

**Example 5.8** Consider Figure 5.3. Assume we have a density function  $f : U \rightarrow \mathbb{R}$  as can be seen on the left. Assume we have some estimator  $\tilde{f}$  in the middle in blue, inducing the filtration  $\mathbb{F}^\alpha = \{\tilde{f}^{-1}([\alpha, +\infty))\}_\alpha$ . We compute the persistent homology groups  $H_0(\mathbb{F}^\alpha)$  and  $H_0(\tilde{\mathbb{F}}^\alpha)$  and plot the overlapping persistence diagrams on the right.  $H_0(\mathbb{F}^\alpha)$  is plotted in red and  $H_0(\tilde{\mathbb{F}}^\alpha)$  is plotted in blue. The points near the diagonal  $\{(x, y) \mid x = y\}$  represent the small peaks in the middle plot. We see that  $\tilde{f}$  has two prominent peaks corresponding to the two prominent peaks of  $f$ , by the stability properties of persistence diagrams. Now, given a threshold parameter  $\tau \geq 0$ , we restrict our focus to the peaks  $\mathbf{x}$  of  $f$  of prominence at least  $\tau$ , points right from the diagonal  $\tau$  and consider peaks of prominence lower than  $\tau$ , left from the diagonal  $\tau$  noise. Intuitively, the points of  $U$  attracted by  $m_p$  are the ones belonging to  $A(m_p)$  and that are eventually merged into  $C(m_p, \alpha)$  before being merged into the component of any other peak of prominence at least  $\tau$ . This is illustrated in Figure 5.4.



**Figure 5.3:** Evolution of the connectivity of the super-level sets of a function  $f$  on the left and of an approximation  $\tilde{f}$  in the middle. The corresponding persistence diagram on the right [11].

**Definition 5.9 (Iterated Root Map)** Let  $m_q \in U$  be a point with  $C(m_q, \alpha)$  having prominence less than  $\tau$ . Let  $m_p$  be the peak with  $C(m_p, \alpha)$  of prominence at least  $\tau$ , such that  $C(m_q, \alpha)$  gets merged into  $C(m_p, \alpha)$  before merging into any other component of prominence at least  $\tau$ . The iterated root map  $r_\tau^*$  is then given by  $m_q \mapsto r_\tau^*(m_q) = m_p$ .

**Definition 5.10 (Basin of Attraction)** The basin of attraction of the peak  $m_p$  with prominence at least  $\tau$ , is given by

$$B_\tau(m_p) = \bigcup_{r_\tau^*(m_q)=m_p} A(m_q).$$

Note that  $B_\tau(m_p)$  contains  $A(m_p)$  since  $m_p$  is a fixed point of  $r_\tau^*$ . More precisely, we have  $A(m_p) = B_0(m_p) \subseteq B_\tau(m_p)$ . In addition, since the iterated root map



$m_q \mapsto r_\tau^*(m_q)$  is uniquely defined, the basins of attraction form a partition of the union of all ascending regions. These basins form our target clusters.

**Example 5.11** In Figure 5.4 we can see the basins of attraction, with  $\tau$  as in Figure 5.3, indicated by arrows. Note that all the smaller peaks are merged into the two most prominent peaks.

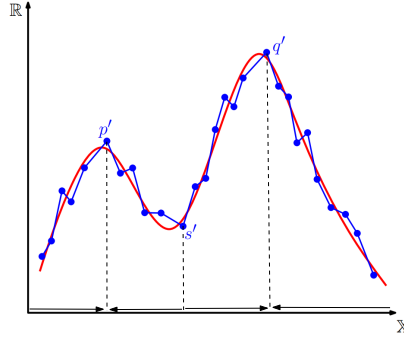


Figure 5.4: Basins of attraction for the estimator of  $f$  [11].

## 5.2 ToMATo Algorithm

The ToMATo algorithm is applied in a discrete setting given some point cloud in Euclidean space. The goal is to find a suitable clustering using results from persistent homology. We only look at the zeroth homology group  $H_0$ , but the algorithm can be adapted to study any  $H_n$  to identify and cluster based on the  $n$ -dimensional features of the data set. We first provide a few definitions.

**Definition 5.12 (Rips graph)** Given a data set  $L = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$  in a metric space  $(X, d)$  and a parameter  $\delta > 0$ , the Rips graph  $R_\delta(L)$  is the graph with a vertex set  $L$ , where the edges are the 1-simplices of the Rips complex.

Note that the Rips graph can be viewed as an  $\varepsilon$ -neighbourhood graph introduced in Section 2.2, Spectral Clustering, where we looked at the construction of similarity graphs.

**Definition 5.13 (Trace of a Superlevel-Set)** Given a distribution function  $f : X \rightarrow \mathbb{R}$  and a parameter  $\alpha \in \mathbb{R}$ ,

$$L^\alpha = L \cap \mathbb{F}^\alpha$$

denotes the trace of the superlevel-set  $\mathbb{F}^\alpha$  over the data set  $L$ .

**Example 5.14** Consider Figure 5.4.  $L^\alpha$  corresponds to the points in  $L$  that have density  $f(l) \geq \alpha$ . If we choose  $\alpha = f^{-1}(p')$  then  $L^\alpha$  corresponds to the points with larger density than  $p'$ .

**Definition 5.15 (Upper-Star Rips Filtration)** The upper star rips filtration  $\mathcal{R}_\delta^f(L)$ , is the family Rips graphs

$$\mathcal{R}_\delta^f(L) = \{R_\delta(L^\alpha)\}_{\alpha \in \mathbb{R}}.$$

The name upper star rips filtration stems from the fact that, when a vertex  $v \in L$  enters the filtration, the set of edges of  $R_\delta(L)$  connecting  $v$  to other vertices with higher function values, i.e. its *upper star*, enters at the same time. Observe that, even though  $\alpha$  ranges over the whole of  $\mathbb{R}$ ,  $L$  still is a finite data set and since  $L^\alpha \subseteq L$ , this will give a finite family of graphs  $R_\delta(L^\alpha)$ .

Let  $L = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^d$  be  $n$  data points,  $f \in \mathbb{R}^n$  be a  $n$ -dimensional vector,  $D \in \mathbb{R}^{n \times n}$  be a symmetric matrix with non-negative real coefficients and  $\delta, \tau > 0$  be two parameters. The dimension  $n$  represents the  $n$  data points, the vector  $f$  represents the density estimator for each point in the corresponding coordinate, the entries  $d_{ij} = d_{ji}$  of the matrix  $D$  represent the distance between the  $i$ -th and the  $j$ -th data point,  $\delta$  represents the parameter for the Rips graph  $R_\delta(L)$  and  $\tau$  represents the parameter for our basins of attraction  $B_\tau(\cdot)$ . Details on how these quantities can be estimated are provided later in this section.

In a preprocessing step, the ToMATo algorithm computes the Rips graph  $R_\delta(L)$  from the input  $D$  and  $\delta$ . The algorithm then mimics within the Rips graph the construction of the basins of attraction of parameter  $\tau$  described in the previous section.

**Clustering:** First, we iterate over the points of  $L$  by decreasing function values  $f(\mathbf{x}_i)$ . At each vertex  $\mathbf{x}_i$ , we approximate the gradient of the underlying probability density function by connecting  $\mathbf{x}_i$  to its neighbour in the graph  $R_\delta(L)$  with the highest function value. If all neighbours of  $\mathbf{x}_i$  have lower function values, then  $\mathbf{x}_i$  is declared a peak of  $f$  with gradient  $\nabla f(\mathbf{x}_i) = 0$ . The collection of these gradients forms a collection of spanning trees of the graph  $R_\delta(L)$ , where each tree can be viewed as the analogue of the ascending region in the continuous setting. An illustration of a clustered data set can be seen in 5.5(c).

**Algorithm 5** ToMATo Clustering

---

**Input:**  $n$ -dimensional vector  $f$ ,  $n \times n$  symmetric matrix  $D$ , parameters  $\delta, \tau > 0$ .

- 1: Sort the index set  $L$  so that  $f_1 \leq f_2 \leq \dots \leq f_n$ ;
- 2: Initialize the union-find data structure  $\mathcal{U}$ ;
- 3: **for**  $i = n$  to 1 **do**
- 4: compute the upper star  $S_i = \{(i, j_1), \dots, (i, j_k)\}$  of vertex  $i$  in  $R_\delta(L)$ ;
- 5: **if**  $S_i = \emptyset$  **then**  $\{\text{vertex } i \text{ is a local maximum of } f \text{ within } R_\delta(L)\}$
- 6:  $g(i) \leftarrow \text{null}$ ;  $\{g(i) \text{ stores the approximate gradient at vertex } i\}$
- 7: Create a new entry in  $\mathcal{U}$  containing the tree  $\{i\}$ ;
- 8: **else**  $\{\text{vertex } i \text{ is not a local maximum of } f \text{ within } R_\delta(L)\}$
- 9:  $g(i) \leftarrow \arg \max_{j \in \{j_1, \dots, j_k\}} f(j)$ ;
- 10: Attach vertex  $i$  to the tree  $t$  containing  $g(i)$ ;
- 11:  $\mathcal{U} \leftarrow \text{Merge}(f, \mathcal{U}, i, S_i, \tau)$ ;
- 12: **end if**
- 13: **end for**

**Output:** The set of entries  $e$  of  $\mathcal{U}$  satisfying  $f_{r(e)} \geq \tau$ .

---

**Merge:** To manage merges between trees, we use a *union find* data structure [12], where each entry corresponds to a union of trees of the spanning forest. We denote the *root* of an entry  $e$  (element of the union find data structure, i.e. a collection of spanning trees), or just  $r(e)$ , as the vertex contained in  $e$  whose function value is highest. By construction,  $r(e)$  is a peak of  $f$  in the Rips graph  $R_\delta(L)$ . The merge of an entry in the union find data structure into another entry is the analogue to the merge of a basin of attraction into another basin in the continuous setting. Merges are performed in the order prescribed by persistence. More precisely, we iterate once again over the vertices of  $R_\delta(L)$  by decreasing order of function values, considering at each vertex  $x_i$  the edges of the upper star of  $x_i$  in  $R_\delta(L)$ . Letting  $e_i$  be the entry of the union-find data structure containing  $x_i$ , if any edge of the upper star of  $x_i$  connects  $e_i$  to some other entry  $e_j$  whose root  $r(e_j)$  has lower function value than the root  $r(e_i)$ , then the algorithm prescribes that  $e_j$  be merged into  $e_i$ . We depart from this prescription and perform the merge only if the prominence of  $r(e_j)$ , viewed as a peak of  $f$  in the graph  $R_\delta(L)$ , is less than the threshold  $\tau$ . This condition comes down to checking whether  $f_{r(e_j)} - f_i < \tau$ . Once all non-prominent neighbouring clusters have been merged into  $e_i$ , we check whether  $e_i$  itself should be merged. Letting  $\bar{e}$  be the neighbouring cluster with the highest root, we merge  $e_i$  into  $\bar{e}$  if and only if the prominence of  $r(e_i)$  is less than  $\tau$ , i.e. if  $f_{r(e_i)} - f_i < \tau$ . An illustration of merged clusters can be seen in 5.5(d).

The algorithm outputs the collection of entries of the union-find data structure,

**Algorithm 6** ToMATo Merge

---

**Input:**  $n$ -dimensional vector  $f$ , union find data structure  $\mathcal{U}$ , integer  $i$ , integer list  $S = \{j_1, \dots, j_k\}$ , parameter  $\tau > 0$ .

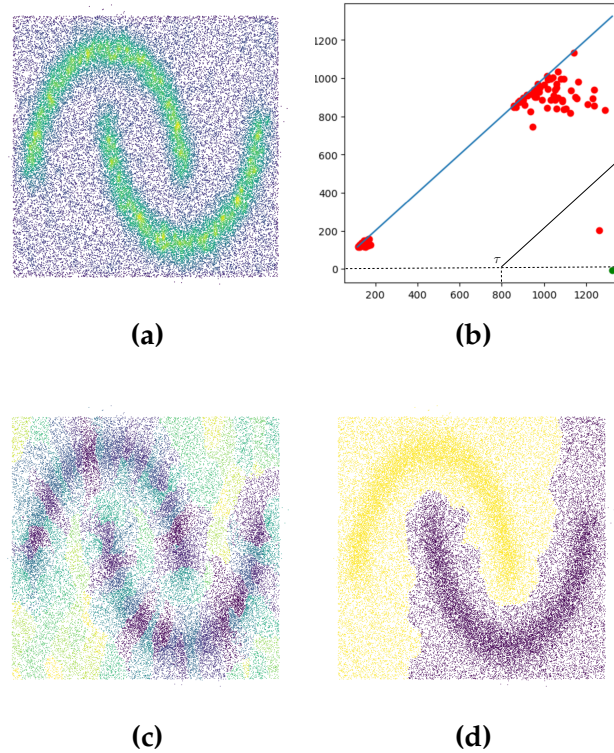
- 1: Let  $e_i$  be the entry of  $\mathcal{U}$  containing  $i$ ;  
*{find entries of  $\mathcal{U}$  intersecting  $S$  whose roots are less than  $\tau$ -prominent; merge those into  $e_i$ }*
- 2: **for**  $j \in \{j_1, \dots, j_k\}$  **do**
- 3:     Let  $e_j$  be the entry of  $\mathcal{U}$  containing  $j$ ;
- 4:     **if**  $e_j \neq e_i$  and  $f_{r(e_j)} - f_i < \tau$  **then**
- 5:         Remove entry  $e_j$  from  $\mathcal{U}$  and attach it to  $e_i$ ;
- 6:     **end if**
- 7: **end for**  
*{find entry  $\bar{e}$  of  $\mathcal{U}$  intersecting  $S$  whose root is highest}*
- 8:  $\bar{e} \leftarrow \text{null}$
- 9: **for**  $j \in \{j_1, \dots, j_k\}$  **do**
- 10:     Let  $e_j$  be the entry of  $\mathcal{U}$  containing  $j$ ;
- 11:     **if**  $\bar{e} = \text{null}$  or  $f_{r(e_j)} > f_{r(\bar{e})}$  **then**
- 12:          $\bar{e} \leftarrow e_j$ ;
- 13:     **end if**
- 14: **end for**
- 15: *{merge  $e_i$  into  $\bar{e}$  if the prominence of the root of  $e_i$  is less than  $\tau$ }*
- 16: **if**  $\bar{e} \neq e_i$  and  $f_{r(e_i)} - f_i < \tau$  **then**
- 17:     Remove entry  $e_i$  from  $\mathcal{U}$  and attach it to  $\bar{e}$ ;
- 18: **end if**

**Output:** updated union find data structure  $\mathcal{U}$ .

---

which partitions the input point cloud  $L$  into clusters. It only outputs those entries  $e$  whose root  $r(e)$  satisfies  $f_{r(e)} \geq \tau$ . This additional filtering step is motivated by the fact that some outliers in the point cloud  $L$  may form independent connected components in the graph  $\mathcal{R}_\delta(L)$  that cannot be merged.

The merging step of the algorithm provides additional feedback in the form of a collection of intervals, representing the lifespan of an entry in the union find structure  $\mathcal{U}$ . The endpoints of the intervals represent the points in the entry appears and disappears. When the parameter  $\tau$  is set to  $+\infty$ , the merging step becomes the standard persistence algorithm computing the persistent homology  $H_0$  of the upper star filtration  $\mathcal{R}_\delta^f(L)$ , thus the output collection of intervals corresponds to the 0-th persistence diagram of this filtration. An illustration of this can be seen in 5.5(b).



**Figure 5.5:** Halfmoons data set. (a) Density estimation for underlying density  $f$  for the Halfmoons data set. (b) Persistence Diagram. Two points far off the diagonal correspond to the two prominent peaks of  $f$  (c) Clustering for  $\tau = +\infty$ , i.e. result for the basic clustering algorithm without a merging phase. (d) Clustering for  $\tau = 800$ , i.e. final result after merging the clusters of non-prominent peaks.

### 5.3 Experimental Result

In this section we present results of ToMATo on different data sets. We discuss experimental results on synthetic data in the form of regular clustering and on images in the form of image segmentation. For this, we first introduce the estimators we use and discuss how we choose the parameters.

A widely used density estimator is the *truncated Gaussian estimator*. The truncated Gaussian estimator is given by

$$f(\mathbf{x}) = \frac{1}{|L|} \sum_{l \in L} K(d(\mathbf{x}, p_l)),$$

where the summand is defined as

$$K(d(\mathbf{x}, p_l)) = \begin{cases} e^{-\frac{d^2(\mathbf{x}, p_l)}{2h}} & d(\mathbf{x}, p_l) \leq h \\ 0 & \text{otherwise.} \end{cases}$$

where  $d(\cdot, \cdot)$  is the Euclidean distance, which can also be used to compute the matrix  $D$ .

Another estimator is the *distance to measure* to estimate the density. It computes the root-mean-squared distance to the  $k$  nearest neighbours:

$$f(\mathbf{x}) = \sqrt{\frac{1}{k} \sum_{i=1}^k d^2(\mathbf{x}, p_i)},$$

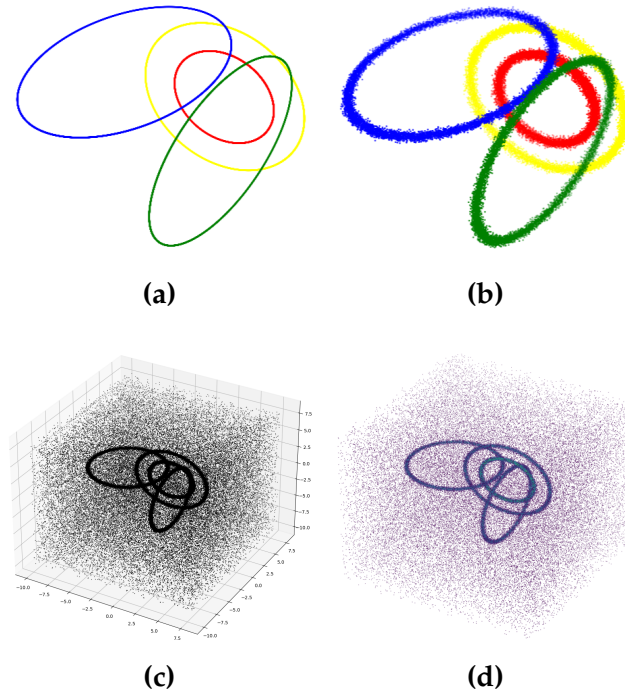
where  $p_i$  denotes the  $i$ -th nearest neighbour of  $x$  among the point set  $L$ . In contrast to the Gaussian estimator, it has fixed complexity, since we only consider the nearest  $k$  neighbours. Note that the distance to measure is a distance rather than a density, so we take  $(-f)$  as input for the algorithm.

To set  $\delta$ , one can compute a single linkage clustering and chose a relevant scale from the resulting dendrogram. In the synthetic data example, we used a  $k$ -nearest neighbour graph as introduced in the Spectral Clustering section 2.2, instead of a Rips graph. Then we ran the ToMATo algorithm with the precomputed  $k$ -nearest neighbour graph (or the Rips graph with the chosen value for  $\delta$ ) and with  $\tau = +\infty$ , to compute the persistence diagram of the estimated density, from which we chose a relevant value for  $\tau$ . A relevant value for  $\tau$  can be chosen between the most distant prominences in the persistence diagram, as we will see later. We then run the ToMATo algorithm a second time with the chosen parameters to compute the final clustering.

### 5.3.1 Synthetic Data

Let us now look at an example with synthetic data of four interlocked rings in Euclidean space  $\mathbb{R}^3$ . The data set is shown in Figure 5.6. The clusters are highly non-linear and non-seperable. The rings are samples of 10,000 points each. Each ring follows a circular uniform distribution with some Gaussian noise. Note that, for the larger rings, the points are spaced farther apart, since we used the same number of points in all four rings. Thus, the differences in lengths of the rings result

in differences in sampling densities. To ensure that the rings are not completely disjoint in space, or rather the resulting graph, we added some uniform background noise of 50,000 points.



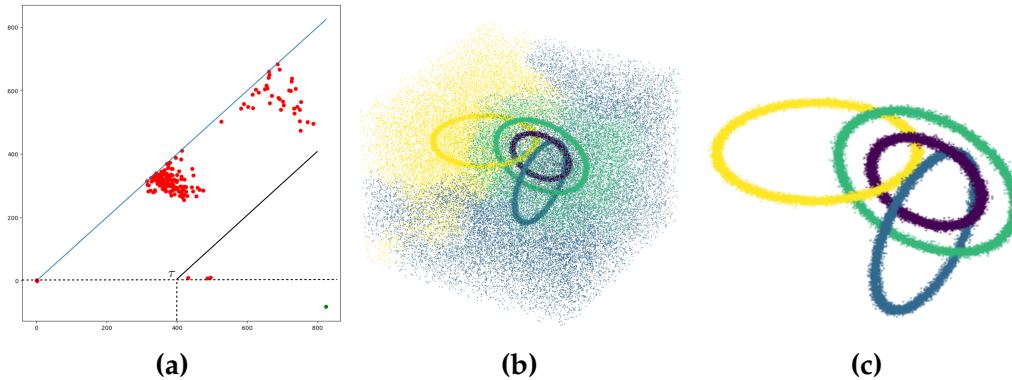
**Figure 5.6:** Synthetic data set. (a) Four interlocked rings. (b) Data consisting of four interlocked rings, as samples of 10,000 points each from a circular uniform distribution with some Gaussian noise. (c) Additional background noise of 50,000 points as a sample from a uniform distribution. (d) Density estimate for the synthetic data set in Figure 5.6(c).

We used a distance to measure density estimator with  $k = 100$ . Note, when using distance to measure, the input to the algorithm is a vector of negative numbers ( $-f$ ), where the larger values, i.e. lower absolute value, resembles a higher probability. The reason for this is that regions with higher density have more points close to each other and thus

$$f(\mathbf{x}) = \sqrt{\frac{1}{k} \sum_{i=1}^k d^2(\mathbf{x}, p_i)},$$

is small, and vice versa for sparser regions. Considering this fact, we end up with the density estimate in Figure 5.6(d), where the colours range from light blue to

violet. Instead of a Rips graph, we used a  $k$ -nearest neighbour graph with  $k = 100$ . For this, we used the standard Euclidean metric.



**Figure 5.7:** (a) Persistence diagram for the data set in 5.6(c). (b) Resulting clustering. (c) Resulting clustering, ignoring points with low distance to measure.

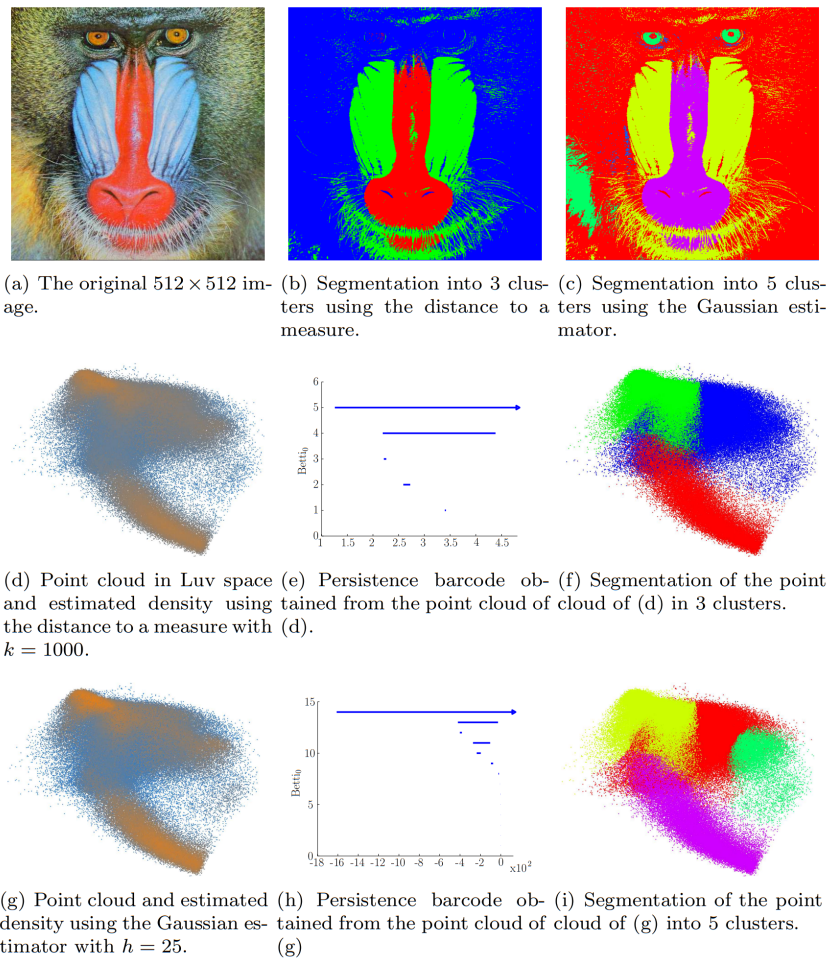
Looking at the persistence diagram in Figure 5.7(a), we can clearly see a gap in the prominences. Therefore, we chose  $\tau = 400$  as threshold parameter. The choice of  $\tau$  is immediate in this example, but as we will see for image segmentation, this choice will not always be as clear. The resulting choice leaves us with the four most prominent clusters that indeed correspond to the four interlocked rings. Note that since there is only one infinitely persistent component, all the data points are connected in the  $k$ -nearest neighbour graph. In particular, the four rings are connected to each other. The four rings are recovered almost perfectly. In Figure 5.7(c) we removed all the background noise with high distance to measure, to see a clearer picture of the clustering.

### 5.3.2 Image Segmentation

We perform image segmentation on a  $512 \times 512$  pixel picture of a mandrill by projecting each pixel into Luv colour space. The experiments are run in two ways. The first one is performed in the Luv colour space. In the second one, spatial information is incorporated into the point cloud data to show how the spatial context can influence the segmentation results. For the test image, we show the original and the corresponding point cloud in Luv space, as well as the resulting segmentation. Along with the histogram, we also show the persistence diagram in the form of a barcode. In the following, we did not reconstruct any findings. All



results, including all figures, can be found in [11].

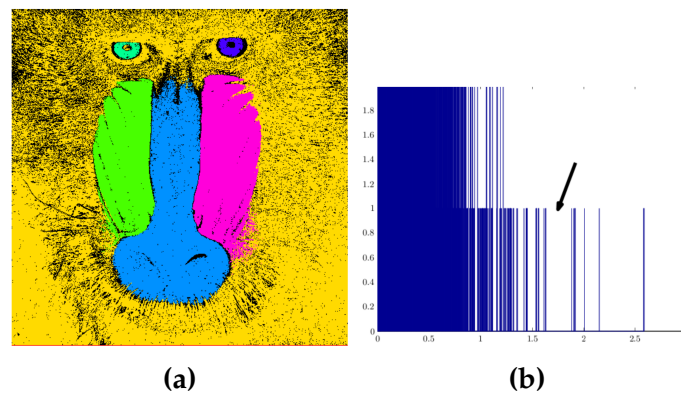


**Figure 5.8:** Results of the ToMATo approach on the Mandrill data set [11].

In the first experiment, the points clouds are clustered in the three-dimensional Luv space, ignoring any spacial information of the pixels in the image. Initially, areas of uniform colour proved challenging because many points were in close proximity in Luv space, producing many edges in the Rips graph. In order to speed up computation, the following downsampling is done: We begin by making all points unmarked. Considering each point  $p$  in order: if  $p$  is unmarked, then we detect all the data points within Euclidean distance  $\delta$  of  $p$ , and we create Rips edges as usual. Then, we mark all the data points within a smaller distance  $\delta/m$  of  $p$ : these points are to be removed from the data set for the clustering phase. Their cluster center will be the same as the one from  $p$ . Typically, we chose  $m$  between

10 and 20 in our experiments. This downsampling procedure can be shown to induce a small additive error (in the order of  $\delta/m$ ) on the persistence diagram approximation, therefore it is provably safe.

The results are shown in Figure 5.8. We use both the distance to measure estimator with  $k = 1000$  and a truncated Gaussian estimator with bandwidth parameter  $h = 25$ . In the case of the distance to measure, we show the result with 3 clusters. They can be clearly seen in the point cloud of Figure 5.8(f). In the original image these clusters correspond to the nose, cheeks and the fur. Using the Gaussian estimator, we see 5 clusters, where further features can be identified, including the eyes and the yellow part of the fur. The 5th cluster corresponds to very dark colours. It is barely visible in the point cloud of Figure 5.8(c). It is important to note that in the Gaussian case, black is the third most prominent cluster, whereas in the case of the distance to measure it is the fourth one.



**Figure 5.9:** (a) Mandrill segmented with colour and spatial information. (b) Persistence histogram for the mandrill. A persistence histogram is a histogram of the prominences, that shows how many times prominences appear in the persistence diagram. The arrow indicates the merging parameter.

Clustering in the Luv space allows pixels that are far apart in the image to end up in the same cluster. Remove this requires to involve spatial information during the clustering phase. The most naive way to do this is by appending the two pixel coordinates to the three colour coordinates. This leaves us with a 5-dimensional space. The disadvantage is that the colour and spatial coordinates may not be balanced appropriately since the scales of the colour channels and the spatial coordinates are unrelated. To avoid this, we consider the point clouds in the Luv

space and compute the density estimates as in the first experiment. However, for two data points to be connected in the Rips graph, we now require them to be close both in the Luv space and in the image domain. This boils down to pruning the Rips graph of the first experiment. In practice, we equivalently proceed the opposite order. We first connect points that are close in the image domain, and then prune out the edges whose vertices are far apart in the Luv space. The results are shown in Figure 5.9. We are able to distinguish between the left cheek from the right cheek and the left eye from the right eye. Dark pixels correspond to very small clusters due to the texture of the fur. In the persistence histogram, we see a clear gap, within which we chose the value of the merging parameter  $\tau$ .

---

## Bibliography

---

- [1] C. C. Aggarwal and C. K. Reddy, *Data Clustering: Algorithms and Applications*. Chapman & Hall/CRC, 2013, pp. 89–93, 100–107.
- [2] S. Z. Selim and M. A. Ismail, “K-means-type algorithms: A generalized convergence theorem and characterization of local optimality,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 81–87, 1984.
- [3] U. von Luxburg, *A tutorial on spectral clustering*, 2007.
- [4] D. Comaniciu and P. Meer, “Mean shift: A robust approach toward feature space analysis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 603–619, 2002.
- [5] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. Wiley, 2012, ch. 10, pp. 37–41.
- [6] G. Carlsson, “Topological pattern recognition for point cloud data,” *Acta Numerica*, May 2014.
- [7] A. Hatcher, *Algebraic Topology (Algebraic Topology)*. Cambridge University Press, 2002.
- [8] T. Dey and Y. Wang, *Computational Topology for Data Analysis*. Cambridge University Press, 2022.
- [9] P. Gillespie, *A homological nerve theorem for open covers*, 2022.
- [10] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer, “Stability of persistence diagrams,” *Discrete and Computational Geometry - DCG*, pp. 263–271, Jun. 2005.
- [11] F. Chazal, L. J. Guibas, S. Y. Oudot, and P. Skraba, “Persistence-Based Clustering in Riemannian Manifolds,” Nov. 2013.

- [12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 4th edition. The MIT Press, 2001, ch. 21.

## Declaration of originality

The signed declaration of originality is a component of every written paper or thesis authored during the course of studies. In consultation with the supervisor, one of the following three options must be selected:

- I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used no generative artificial intelligence technologies<sup>1</sup>.
- I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used and cited generative artificial intelligence technologies<sup>2</sup>.
- I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used generative artificial intelligence technologies<sup>3</sup>. In consultation with the supervisor, I did not cite them.

Title of paper or thesis:

Topological Clustering Algorithm ToMATo

Authored by:

*If the work was compiled in a group, the names of all authors are required.*

Last name(s):

Hostettler

First name(s):

Daniel

With my signature I confirm the following:

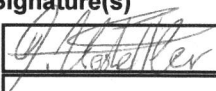
- I have adhered to the rules set out in the Citation Guide.
- I have documented all methods, data and processes truthfully and fully.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for originality.

Place, date

8185 Winkel, 19.07.2024

Signature(s)



*If the work was compiled in a group, the names of all authors are required. Through their signatures they vouch jointly for the entire content of the written work.*

<sup>1</sup> E.g. ChatGPT, DALL E 2, Google Bard

<sup>2</sup> E.g. ChatGPT, DALL E 2, Google Bard

<sup>3</sup> E.g. ChatGPT, DALL E 2, Google Bard