Applications of Topology and Geometry in Data Science

ZIVA URBANCIC

Applications of Topology and Geometry in Data Science

Živa Urbančič

A thesis presented for the degree of
Doctor of Philosophy at Durham University

Department of Mathematics
Durham University
United Kingdom

19th May 2025

Abstract

In the last decades we have experienced a boom in computing power, closely followed by the emergence of novel analytic methods which rely on more expensive computations. Part of this wave have been methods that build models of the processes at play based on the data collected during observation, and view them through the lens of topology and geometry. The primary assumption of these methods is that topological and geometrical properties of the model reflect the underlying structure in data, and so probing them can help uncover laws governing the phenomena under study.

The versatility and wide range of use of these methods are illustrated in this thesis. We present four works of completely different flavors covering topics from theory to practice, from mathematics and computer science to cell biology. In one, we contribute new insights into persistent homology: a method whose output is an algebraic object called persistence module, which counts and relates topological features of the data at different scales. Our results give guarantees of when two persistence modules are close enough algebraically, so that we can pair the entries encoding the same underlying feature at each scale, and track the evolution of said pairs as the scale is increased. We then switch the setting to one of neural networks, where we evaluate if their plasticity relates to how they partition the input space. In particular, we set out to answer if choosing their initial parameters with the aim of obtaining finer partitions speeds up their learning and increases the accuracy of their final predictions. Next, we provide a framework for topological modeling of spaces, which are characterized by both their metric and directed structure, and appear naturally when the phenomenon under study has a non-reversible component. In particular, we provide two similarity measures that can be used to compare such spaces. As the final curtain, we explore gene expression data obtained from a specific class of neurons, namely monoaminergic neurons, in the brains of fruit flies. Within the data set we identify structure

in the form of denser subsets, which is related to division of neurons into several subtypes. In addition, we uncover the genes that drive this division, and repeat the analysis on a subtype corresponding to dopaminergic neurons.

Declaration

The work in this thesis is based on research carried out within the Pure Mathematics Group at the Department of Mathematics at Durham University, UK. No part of this thesis has been submitted elsewhere for any other degree or qualification. Some work in this thesis is drawn from papers published alongside others, as detailed below.

Parts of this thesis were done in collaboration with Jeffrey Giansiracusa and published in [1]. In particular, some introductory material from the paper is restated in Sections 2.2.3 to 2.2.5, while the original results are stated verbatim in Chapter 3. The author of the thesis contributed all the results and writing to the aforementioned publication.

Chapter 4 contains work done in collaboration with Yue Ren and Iolo Jones. We share authorship of the code used to run the experiments, and the writing was carried out by the author of the thesis. It has not been submitted for publication at the time of writing.

In Chapter 5 we present parts of the work which was, at the time of writing, submitted to the proceedings of the Women in Computational Topology Third Workshop, entitled Research in Computational Topology 3 and published as a volume of Association for Women in Mathematics – Springer series. It is a result of a collaboration with Lisbeth Fajstrup, Brittany Terese Fasy, Wenwen Li, Lydia Mezrag, Tatum Rask and Francesca Tombari. The definitions of notions we introduced are the result of group discussions. While significant and invaluable input was given by the collaborators, this author claims main authorship of Examples 5.1.7, 5.2.6 and 5.3.2 and Theorems 5.2.3 and 5.3.5. Sections 5.4 and 5.5 are summarized as examples in the submitted paper, but the author decided to extended and develop them further in this thesis.

Chapter 6 summarizes results of a collaboration with Vincent Croset. He proposed the problem and provided the data. Data analysis and writing was carried out by the author of the thesis. No content has been submitted for publication at the time of writing.

Acknowledgements

I would like to express my gratitude to

My supervisors, Yue Ren and Jeffrey Giansiracusa, without whose guidance and support, I would not have found my footing in the world of academia. You helped me recognize the importance of taking small steps, that failure is a blessing in disguise and contribution to science can take many forms. Thank you for devoting your time to me and for providing perspective the many times I needed it. Thank you for sharing your thoughts on how to connect and organize projects, which at first glance seemed mostly unrelated, into a thesis, and giving me feedback on all of its versions.

Members of the viva committee, Vidit Nanda and Fernando Galaz García. Thank you for sharing your knowledge and expertise with me through invaluable feedback and insightful questions during the viva. Your input has significantly contributed to the improvement of my thesis.

My family, who are always there for me. Thank you for anticipating my needs, being patient through all my rebellions and moments of despair, and for being unbearably proud of me. I am in particular grateful to Jasna, for sharing with me her excitement about the new subject she took at the university all those years ago. Who knew that conversation would lead to our paths converging once again, and I would follow in your footsteps for the millionth time in our lives. (I know, how annoying!) Since you kindly excused me from reading your thesis, you are excused from reading mine.

Dino, who is my source of strength and had my back in every fight, especially the ones I fought with myself. Thank you for navigating life with me, for being my safe space, my confedan, my light. I could not have done it without you.

My collaborators. Lisbeth, Brittany, Wenwen, Lydia, Tatum, Francesca, you are all fantastic. I am grateful I was able to learn valuable lessons about team work and combining different points of view in such a welcoming group. Vincent and Sophie, thank you for kindly sharing data with me, for answering all of my many questions, and for giving me invaluable insight into the world of genes. Herbert, your infectious enthusiasm helped me gain momentum for the rest of my PhD.

My academic family and my friends. The blessing of your friendship helped me, quite frankly, to stay sane. Your company has been a respite in these years, so I thank you for adjusting to my crazy schedule and being there whenever I asked you to.

Lastly, I would like to express my gratitude to the TDA research community and all the organizations and institutions that supported me throughout my journey. In particular, the work presented in this thesis was greatly supported by the Centre for TDA, funded by EPSRC under grant EP/R018472/1. I am also deeply thankful to Primož Škraba and Omer Bobrowski for generously sharing an implementation of k-cluster with me, which significantly contributed to this work.

Contents

List of Figures

List of Tables

List of Acronyms

Adam adaptive moment estimation. 6, 88, 89, 100–102, 104–106

cDNA copy DNA. 35, 36

DNA deoxyribonucleic acid. 33–35, 129

GABA gamma-aminobutyric acid. 131

kNN k-nearest neighbors. 42

MNN mutual nearest neighbors. 38

mRNA messenger RNA. 34–36, 39, 40

NB nagative binomial. 36

p.f.d. pointwise finite dimensional. 24–27, 44, 136

PCA principal component analysis. 42

PCR polymerase chain reaction. 35

ReLU rectified linear unit. 6, 86–91, 100

RNA ribonucleic acid. 34, 129

RPM reads per million. 40

scRNA-seq single-cell RNA sequencing. 33, 34, 36–40, 42

SGD stochastic gradient descent. 6, 88, 100, 102–104, 106

t-SNE t-distributed stochastic neighbor embedding. 42

TDA topological data analysis. 135

ToMATo topological mode analysis tool. 136

UMAP uniform manifold approximation and projection. 42, 131, 144, 145, 169, 170

VR Vietoris–Rips. 17

ZINB zero-inflated negative binomial. 36

ZIP zero-inflated Poisson. 36

Chapter 1

Introduction

Data is everywhere. When skillfully processed, it becomes an invaluable asset, boosting our understanding of the world around us. However, in its raw state, data is essentially "informationless" – merely a collection of points scattered across feature space. The journey to extract meaningful insights from data has traditionally leaned heavily on statistics and probability theory. In its simplest form, data analysis started with calculating basic statistical metrics like mean and variance. As computing power increased, more sophisticated methods such as hypothesis testing and regression analysis emerged and became staples in the data analysis toolkit. The most recent developments in form of artificial intelligence have revolutionized the field and have significantly expanded our ability to interpret complex data sets. In this thesis, however, we explore a somewhat different approach offered by the fields of geometry and topology.

Geometry arose from studying practical problems in the physical world, using measurements of lengths and angles. Over time, it has moved away from our early understanding of reality, which was limited to (at most) three-dimensional, flat space. However, the interest into how lengths and angles are measured within mathematical objects, and the properties derived from these measurements, has remained central. Topology, geometry's younger sister, on the other hand, is relatively more abstract. It does not rely on the concept of length. Instead, it develops a notion of "close enough" in a more general setting, allowing for the study of continuity. In fact, the properties of mathematical objects studied within topology are those preserved under continuous transformations.

Tools of geometry and topology can be applied to data to study its shape – either on the level of individual entries or the data set as a whole. In the latter case, it is often assumed that the "shape of the sample space" reflects the inherent properties and laws governing the processes at play. These can be uncovered by modeling the relationships between points on several scales and inferring which objects said models represent based on the computed topological and geometric invariants. On the other hand, when each observation in the data set is a shape from the get go (for example in image analysis), these invariants can be used to summarize the shape. Using the resulting compressed features in further analysis, including with machine learning methods, facilitates or speeds up the otherwise expensive computations. We support these claims with the following motivating examples:

- There are many examples of imaging data sets whose shape information has been compressed using topological and geometric summaries. In [3], the authors represent the wing vein structure in Drosophila melanogaster as an embedded planar graph. In [4], topological features are used in classification of hepatic lesions.
- In time series analysis, changes in modes of behavior are reflected in the topological properties of the model (for example sliding window-embedding). By monitoring such properties given the raw neurophysiological recordings of epilepsy patients, we can detect seizures in real time [5]. In statistical physics systems, phase transitions can be identified by topological study of the lattice model configuration and its change with increasing temperature [6].
- When the studied phenomenon is periodic, circles, tori, or other non-trivial topological objects may be found within the data. For example, grid cells are neurons, which are part of the brain's spatial awareness centers, and fire in a characteristic hexagonal pattern corresponding to specific locations. They are organized into modules, with the firing patterns of cells within each module being translations of one other. When the patterns of different cells within a single module are (flattened and) embedded, they reside on a toroidal manifold, which reflects their 2D-lattice structure [7].

By choosing good models for the data and rephrasing the questions in terms of their geometric and topological properties, one can develop novel methods to tackle problems whose nature makes them inaccessible to the existing data-science toolkit. Different methods can also be used in combination to enhance the predictive power of our analysis.

The aim of the following chapters is to further convince the reader that topology and geometry are interesting; not purely because they are so inherently, but because of their many applications.

We include a collection of four works, spanning many topics in mathematics, computer science, and even cell biology and genomics, all centered around leveraging geometry and topology in data analysis. As some experience reported in the following is individual to the author, let me switch between the more formal and more personal style of writing as I present to you the projects filling this thesis.

With only a few, clearly indicated exceptions, Chapters 3 to 6 include original work. My contributions to the projects that are summarized in these chapters are detailed in the Declaration.

Chapter 3: Ladder Decomposition for Morphisms of Persistence Modules

The work described in Chapter 3 began with Jeffrey Giansiracusa stating the following problem. Take a multi-parameter persistence module; that is a covariant functor $V\colon \mathbb{R}^n \to \mathbb{V}\mathrm{ect}(\mathbb{F})$ where n is the number of parameters, $\mathbb{R}^n$ is equipped with the product order, and $\mathbb{V}\mathrm{ect}(\mathbb{F})$ is the category of modules over a chosen field $\mathbb{F}$. For simplicity sake, let $n = 2$ and consider the restrictions to increasing lines in the parameter space. For example, let $L\colon \mathbb{R} \to \mathbb{R}$ be an affine function mapping $x \mapsto ax + b$ with $a > 0$ and observe that

$$V|_L : \mathbb{R} \to \mathbb{V}\mathrm{ect}(\mathbb{F})$$
$$V|_L : t \mapsto V(t, L(t))$$

is a one-parameter persistence module. Restrictions to different lines can be related [8, 9], especially if the lines in questions are parallel to each other. Then, a distinguished type of morphism, which measures algebraic similarity and is called an interleaving morphism, exists between the restrictions. This interleaving morphism is part of the inner structure of V, and may, as a consequence, have even nicer properties. Under mild assumptions each one-parameter persistence module admits a direct sum decomposition into intuitive building blocks. The question is, can we relate the building blocks of the restrictions to parallel lines via the interleaving morphism that exists between them.

A similar question for a general, not necessarily interleaving morphism has received a lot of attention from the topological data analysis community in the recent years. It is interesting for two main reasons. Firstly, a similar direct sum decomposition known to exist for one-parameter persistence modules does not exist in the multi-parameter case under the same mild assumptions.

Thus, approximate results, such as those which might sprout from these studies, can give us new descriptors for the structure of multi-parameter persistence modules. Secondly, morphisms may encode relations between two settings in which a phenomenon is studied. Comparing commonly used topological summaries via morphisms therefore has many potential applications in the field.

First construction of a partial bijection between the summands of the decompositions defined via a morphism, which also provided important stability related guarantees, appeared in [10]. Its main shortcoming, that it is determined by the image of the morphism and not the morphism itself, was addressed independently in [11, 12, 13]. We base our work on the results of [11], where it was shown that a partial bijection induced by a direct sum decomposition of the morphism exists under relatively strict assumption on the summands constituting the decompositions of the persistence modules. We show that this assumption can be relaxed significantly whenever the morphism in question is close to being an isomorphism. Similar results in the setting of vineyard modules appear in [14].

Chapter 4: Initialization Strategy for Deep Neural Networks with ReLU Activation

Chapter 4 includes a partial report on an ongoing collaboration with two members of my research group: my supervisor, Yue Ren, and my academic sibling, Iolo Jones. Following Yue's initiative, we began investigating deep neural networks and, specifically, their connection to tropical geometry.

Tropical geometry is a branch of algebraic geometry, in which the classical study of polynomials and their solutions is carried out within an unconventional arithmetic framework, where addition, $\oplus$, and multiplication, $\otimes$, are defined as

$$x \oplus y = \max\{x, y\}$$
$$x \otimes y = x + y.$$

An example of a tropical polynomial is $a \otimes x^2 \ \oplus \ b \otimes x \otimes y^2$ or $\max\{a + 2x, b + x + 2y\}$. Of course, the degree can be arbitrary and polynomials can be multivariate. Each polynomial in n variables defines a geometric object called a tropical hypersurface, which is the set of points in $\mathbb{R}^n$ at which the maximum is attained at least twice.

Arrangements of tropical hypersurfaces appear naturally in machine learning as decision boundaries of neural networks with maxout activation [15]. Maxout of rank $r \in \mathbb{N}$ is a piece-wise linear function, defined as

$$x \mapsto \max\{A_1 x + B_1,\ \ldots,\ A_r x + B_r\}, \tag{1.1}$$

where the maximum is applied coordinate-wise, and each $A_i x + B_i$ is an affine function. These r affine functions are in practice channels of a layer of a neural network. Such a network is piecewise linear: there are regions in its input space on which it is an affine function. Within each of these regions, the maxout activations are consistent in their decision: which channel achieves the maximum in which coordinate does not change. The set of points where their decision changes is called the decision boundary. In particular, where the decision changes in coordinate j of (1.1), the maximum

$$x \mapsto \max\{(A_1 x + B_1)_j,\ \ldots,\ (A_r x + B_r)_j\},$$

is attained twice. The set of such points is exactly a tropical hypersurface and the entire decision boundary is their arrangement.

Analogous regions of linearity for networks with other piece-wise linear activations have been shown to carry interesting information about the network and the task it is trained for, and have been studied in connection with network expressivity and its potential for generalization to unseen input, with networks with larger number of regions having more desirable properties. Inspired by this knowledge, we set out to develop a novel initialization technique for maxout neural networks, which would prioritize maximizing the number of linear regions before the network is trained.

This turned out to be a very ambitious goal. As this was the first time any of us have manipulated neural networks manually, the learning curve was steep. Even so, we implemented the first version of the methods for initialization and keeping track of the regions rather quickly. The real bottleneck was the abundance of choice available in the machine learning community, which completely overwhelmed three novices. Which data set do we train the networks on? What architecture should we choose for them? Which maxout rank, loss function, optimizer? What should the learning rate be and should we use momentum? How do we make our answers to these questions consistent? We wanted to be good researchers and provide a thorough overview comparing many combinations, but we quickly realized that would require us to adjust the

methods more or less for every single one. Thus, we began by choosing the most popular data set for classification tasks, MNIST [16], a fixed, relatively small, network architecture, cross-entropy loss, and maxout of rank 2, which is equivalent to the better known and extensively studied rectified linear unit (ReLU) activation. We chose to compare stochastic gradient descent (SGD) [17, Chap. 8.1.3] and adaptive moment estimation (Adam) [18] optimizers with their default settings. Three years and many (many!) bugs later, we have some preliminary results, which I summarize in Chapter 4.

Chapter 5: Gromov–Hausdorff Distance for Directed Metric Spaces

The project presented in Chapter 5 was born at the Third Workshop for Women in Computational Topology, held in July 2023 at the Bernoulli center in Lausanne. I joined a group of six brilliant women, Lisbeth Fajstrup, Brittany Terese Fasy, Wenwen Li, Lydia Mezrag, Tatum Rask and Francesca Tombari. Initially, we intended to study the connections between dynamic programming and directed topology. The second has been used ingeniously to study concurrent processes [19] by modeling them as subspaces of directed hypercubes and comparing the equivalence classes of paths up to endpoint-preserving homotopy. The idea was, that the recurrence relations between path equivalence classes with fixed starting point and varying end points could be leveraged to compute them dynamically. While this is true, we agreed gains of dynamic implementation would be minimal if any, and thus decided to pivot.

Our new inspiration came from the work of Lim, Mémoli and Smith [20]. They used Gromov–Hausdorff distance, a metric on compact metric spaces up to isometry, to study the difference between spheres endowed with geodesic distance. Although computing this distance exactly is notoriously difficult, a framework they developed, cleverly using one of its definitions in combination with conventional topological tools, such as the Borsuk–Ulam theorem, allowed them to derive significant results. Alongside other remarkable applications of Gromov–Hausdorff distance, this inspired us to explore its potential in the context of directed spaces. Immediately, we stumbled across two questions:

- What is a natural way to define a metric on a directed space?
- What is a natural way to generalize the definition of the Gromov–Hausdorff distance to directed spaces equipped with a metric?

After many failed attempts, we arrived to some definitions answering these questions. They are listed in Chapter 5 along with some examples of directed spaces. Notably, we define two generalizations of the Gromov–Hausdorff distance: the directed Gromov–Hausdorff distance and the distortion distance, each capturing different properties of the spaces. While directed Gromov–Hausdorff distance recognizes the differences in directed structure that are reflected in the metric only, the distortion distance is almost an extended metric on the set of directed metric spaces up to directed isometry and thus captures the difference in directed structure well. As similarity measures that account for both the metric and the directed structure of the object under study, they have potential applications in many contexts (for example when the problem is modeled by a directed weighted graph). However, to make them readily applicable, the theory needs to be developed further to include results that approximate or bound these distances on interesting examples of directed spaces. The exact computation unfortunately requires searching the spaces of maps between directed metric spaces and consequently has very high complexity, much like the exact computation of Gromov–Hausdorff distance.

Chapter 6: Classification of Gene Expression Data

Two years into my doctoral study, I became frustrated with the fact that I had not yet acquired any practical experience of working with real world data sets. I find applications to biology and medicine incredibly interesting, which is why I reached out to the Biophysical Sciences Institute in Durham and began attending their lunchtime mixer events. At one of these events I met Vincent Croset, an assistant professor at the Department of Biosciences, Durham University. He and his collaborators have been studying thirst-induced changes in gene expression and collated a data set for this purpose [21]. As part of their study, they ran several clustering steps at different resolutions, restricting the data set to the cells of interest at each step and clustering again. When restricting the data set to the neurons identified to be monoaminergic (this restriction is called the Thirsty Fly data set in this thesis), the standard methods of their field that have been serving them well thus far could no longer produce coherent clusters. Vincent attributed this to the small size of Thirsty Fly data set and asked me whether topological methods might be able to identify substructures in such a setting. I was optimistic, and promised to play around with the data set to give him a more informed answer. This was the beginning of a project described in Chapter 6.

It became clear immediately, that the gene expression data sets are peculiar. The simplest questions, such as “How do I know if a specific gene is expressed in a specific cell?”, had unexpectedly

complicated answers, and my initial attempts in clustering were disappointing. Soon after, however, Jeff brought a clustering method inspired by persistent homology, called k-cluster [22], to my attention. We reached out to its authors, Omer Bobrowski and Primož Škraba, who kindly shared its implementation with me. To our surprise, the first clusters I obtained with it were already really impressive, and were only tweaked slightly since (see upper left plot in Figure 6.5).

First success called for further discussions with Vincent in which the aims of the project were more properly defined. He disclosed that they are particularly interested in a subtype of monoaminergic neurons known as the dopaminergic neurons. They wished to find finer substructures, related to different subtypes of dopaminergic neurons, and identify them, despite not knowing which genes are the drivers of this sub-specialization. The task, in short, was to identify dopaminergic neurons, cluster them further, and reason about why those clusters form. We leveraged the clustering task on the monoaminergic neurons, where we tested a collection of methods, as a proof of concept for the subsequent work on dopaminergic neurons.

To summarize the results, fully detailed in Chapter 6, k-cluster consistently performed well and emerged as the most versatile and user-friendly method in experiments on monoaminergic neurons. Further and more detailed comparison of clustering methods should be carried out, but this suggests k-cluster has great potential for use within various communities studying transcriptome data sets. In addition, comparing the obtained sets of clusters I was able to compile a short list of genes whose role in cell specialization into subtypes of monoaminergic neurons should be investigated. Although many methods gave satisfactory results for clustering monoaminergic neurons, this success was not matched on dopaminergic neurons, likely due to the small size of the data set relative to its expected heterogeneity. Fortunately, Vincent's group has acquired a much larger dataset of dopaminergic neurons, which is, at the time of writing, being cleaned and prepared for analysis. Thus, this project is still ongoing.

Chapter 2

Prerequisites

This chapter includes the preliminary theory needed for later work. In particular, Section 2.2 is needed in Chapters 3 and 6, Sections 2.1 and 2.3 in Chapter 5, and Section 2.4 in Chapter 6.

2.1 Measuring Distances

There are several (related) notions of how one can measure distance between points in a set. Most common notions, metrics and their generalizations, are presented in Section 2.1.1. On the other hand, one may choose a distinguished set of paths on a space, define a length of a path, and measure the distance between two points via the lengths of paths between them. This point of view is described in Section 2.1.2. Lastly, we can also measure distances between metric spaces themselves. This can, for example, be done using the Gromov–Hausdorff distance, detailed in Section 2.1.3.

2.1.1 Metric Spaces

We list definitions of a metric and some of its generalizations, which we collate from many sources [23, 24, 25].

Definition 2.1.1. A binary function $d\colon X \times X \to [0, \infty)$ on a set X is a **metric** on X if it has the following properties:

1. **identity**: $d(x,x) = 0$ for all $x \in X$,
2. **positivity**: $d(x,y) > 0$ for all $x \neq y \in X$,
3. **symmetry**: $d(x,y) = d(y,x)$ for all $x, y \in X$, and
4. **triangle inequality**: $d(x,z) \leq d(x,y) + d(y,z)$ for all $x, y, z \in X$.

A pair (X, d) of a set with a metric on it is called a **metric space**.

There are many generalizations of the notion of a metric, where we skip one or more of the requirements from its definition.

Definition 2.1.2. A binary function $d\colon X \times X \to [0, \infty)$ is a

a) **pseudometric** on X if it satisfies identity, symmetry and triangle inequality, but not positivity: for distinct $x \neq y$, $d(x,y)$ might be zero.

b) **quasimetric** on X if it satisfies identity, positivity and triangle inequality, but not symmetry.

c) **semimetric** on X if it satisfies identity, positivity and symmetry, but not triangle inequality.

We can further allow d to not be finite. A binary function $d\colon X \times X \to [0, \infty]$ that satisfies identity, positivity, symmetry and triangle inequality is called an **extended metric** on X. The prefixes "extended", "pseudo-", "quasi-", and "semi-" can be combined when dropping multiple assumptions from Definition 2.1.1. For example, an **extended pseudometric** is a function $d\colon X \times X \to [0, \infty]$ that satisfies identity, symmetry and triangle inequality, but not positivity.

Definition 2.1.3. A map $f\colon X_1 \to X_2$ where (X_1, d_1) and (X_2, d_2) are metric spaces is called K**-Lipschitz** for $K > 0$ if for all $x, y \in X_1$,

$$d_2(f(x), f(y)) \leq K d_1(x,y).$$

It is **distance-preserving** if for all $x, y \in X_1$

$$d_2(f(x), f(y)) = d_1(x,y).$$

A distance-preserving map is always injective, and it is called an **isometry** when it is bijective. Metric spaces between which an isometry exists are **isometric**.

Every metric space (X, d) can be endowed with a metric-induced topology, which can be defined by giving a base $\{B_r(x) \mid r \in (0, \infty), x \in X\}$, where $B_r(x)$ is the open ball of radius r centered at x in the metric d, $B_r(x) = \{x' \in X \mid d(x, x') < r\}$. The converse is not true: not every topological space X can be endowed with a metric, and the ones that can are called **metrizable**.

2.1.2 Paths and Length Structures

The main source for this section is [26], although many notions are standard in the fields of topology and mathematical analysis. The authors study how the notions of distance and length of a path are related to each other. We rephrase some of their results here.

A **path** on a topological space X is a continuous map $\gamma : [a, b] \to X$, where $[a, b] \subset \mathbb{R}$ is an arbitrary interval. Common operations on paths can be defined as follows.

Definition 2.1.4. Let $\gamma\colon [a, b] \to X$ be a path. The **restriction** of γ to the interval $[c, d] \subseteq [a, b]$ is the path $\gamma|_{[c,d]} : [c, d] \to X$. If $\delta\colon [b, c] \to X$ is a path with $\gamma(b) = \delta(b)$, the **concatenation** of γ and δ is a path $\gamma \star \delta\colon [a, c] \to X$ satisfying $\gamma \star \delta|_{[a,b]} = \gamma$ and $\gamma \star \delta|_{[b,c]} = \delta$. A **linear reparametrization** of the path γ is a path $\gamma'\colon [\frac{a-k}{h}, \frac{b-k}{h}] \to X$ defined as $\gamma'(s) = \gamma(hs + k)$ for some $h > 0$ and $k \in \mathbb{R}$.

Let $\mathcal{C}$ be a family of paths $\gamma : [a, b] \to X$ where a and b are variable, and let $\ell : \mathcal{C} \to [0, \infty]$ be some function we call **length**.

Definition 2.1.5. The pair $(\mathcal{C}, \ell)$, where the family $\mathcal{C}$ contains all singleton paths $\{\star\} \to X$ and is closed under restriction, concatenation and linear reparametrization, and length ℓ is

- monotone, *i.e.* $\ell(\gamma|_{[c,d]}) \leq \ell(\gamma)$ for any $\gamma\colon [a, b] \to X$ and $[c, d] \subseteq [a, b]$,
- additive, *i.e.* $\ell(\gamma \star \delta) = \ell(\delta) + \ell(\gamma)$,
- zero on singleton paths,
- independent of linear reparametrizations, *i.e.* $\ell(\gamma) = \ell(\gamma')$ for any reparametrization γ' of γ,

is a **length structure** on X.

For a simple example of a length structure, let $X = \mathbb{R}$ and $\mathcal{C}$ be a family of non-decreasing paths in X. Then the pair $(\mathcal{C}, \ell)$, with length ℓ defined as $\ell(\gamma) = \gamma(b) - \gamma(a)$ for any path $\gamma\colon [a, b] \to X$, is a length structure on $\mathbb{R}$.

Note that in [26] a path is any (not necessarily continuous) map $J \to X$, where X is a non-empty set and J is an arbitrary interval. However, in our work such level of generality is not necessary, so we only consider topological spaces and continuous paths. Further, we often take $\mathcal{C}$ to be the family of rectifiable paths for the chosen length function.

Definition 2.1.6. A path γ is **rectifiable** with respect to the length structure ℓ if $\ell(\gamma) < \infty$.

Notice that the family of rectifiable paths satisfies the assumptions of the path family from Definition 2.1.5. To be specific, constant paths, finite concatenations, restrictions and linear reparametrizations of rectifiable paths are all rectifiable paths as well.

Given a length structure $(\mathcal{C}, \ell)$ on X, we can define $d_\ell\colon X \times X \to [0, \infty]$ by setting

$$d_\ell(x, y) := \inf_{\xi \in \mathcal{C}} \{\ell(\xi) \mid \xi(0) = x, \xi(1) = y\}.$$

Note that when there is no path in $\mathcal{C}$ that starts at x and ends at y, $d_\ell(x, y) = \infty$. Since d_ℓ satisfies identity, positivity and triangle inequality, it is an extended quasimetric. It is induced by the length structure $(\mathcal{C}, \ell)$, and is called **induced distance** accordingly. Similarly, given an extended quasimetric (or even a metric) d on a topological space X, a length function can be defined as below.

Definition 2.1.7. The **length by total variation** of a path $\gamma\colon [a, b] \to X$ is defined as

$$\ell^d(\gamma) = \sup \sum_{i=1}^{N} d(\gamma(t_{i-1}), \gamma(t_i)),$$

where the supremum ranges over all the sequences $a \le t_0 < t_1 < \cdots < t_N \le b$.

If we choose a path family $\mathcal{C}$ on X that contains singleton paths and is closed under restrictions, concatenations and linear reparametrizations (requirements of Definition 2.1.5), then pairing it with length by total variation gives a length structure [24, Section 2.3.2].

The interplay between notions of extended quasimetric and length structure is studied extensively in [26]. In particular, they are interested in the sequences

$$d \longrightarrow \ell^d \longrightarrow d_{\ell^d} \longrightarrow \dots$$
$$\ell \longrightarrow d_\ell \longrightarrow \ell^{d_\ell} \longrightarrow \dots$$

and their idempotency. We restate some of their results here.

Proposition 2.1.8 (Proposition 2.7 of [26])**.** *Let d be an extended quasimetric, $\gamma\colon [a,b] \to X$ a path, and $\varphi\colon [a',b'] \to [a,b]$ a weakly increasing, continuous function that satisfies $\varphi(a') = a$ and $\varphi(b') = b$. Then*

$$\ell^d(\gamma \circ \varphi) = \ell^d(\gamma).$$

Whenever the length functional is a length by total variation, we can use Proposition 2.1.8 to justify restricting the family of paths $\mathcal{C}$ to only paths $I \to X$, where I is the unit interval. Proposition 2.1.9 is contained in Theorem 2.19 and Proposition 3.10 of [26].

Proposition 2.1.9. *For an extended quasimetric d and a length structure $(\mathcal{C}, \ell)$ on X*

$$\begin{aligned} \ell^{d_\ell}(\gamma) &\leq \ell(\gamma), \\ d(x,y) &\leq d_{\ell^d}(x,y), \end{aligned} \tag{2.1}$$

for every path $\gamma \in \mathcal{C}$ and every pair of points $x, y \in X$.

2.1.3 Gromov–Hausdorff Distance

In this section, we give the necessary prerequisites on the topic of measuring distances between metric spaces. In particular, we recall the definition of Gromov–Hausdorff distance between metric spaces X and Y, which measures how far X and Y are from being isometric. We further list some known results about it as they appear in [20, 24].

Definition 2.1.10. The **Hausdorff distance** between compact subsets A and B of a metric space (M,d) is

$$d_H(A,B) = \max\left\{\sup_{a\in A} d(a,B),\ \sup_{b\in B} d(A,b)\right\},$$

where $d(a, B) = \inf_{b\in B} d(a, b)$ and $d(A, b) = \inf_{a\in A} d(a, b)$. The **Gromov–Hausdorff distance** between two compact metric spaces (X, d_X) and (Y, d_Y) is

$$d_{GH}(X, Y) = \inf_{X \xhookrightarrow{f} Z \xhookleftarrow{g} Y} d_H^Z(f(X), g(Y)),$$

where the infimum ranges over metric spaces (Z, d_Z) and isometric embeddings $f\colon X \hookrightarrow Z$ and $g\colon Y \hookrightarrow Z$.

Gromov–Hausdorff distance is notoriously difficult to compute, which is why the alternative characterization by [27] via special properties of maps is especially useful.

Definition 2.1.11. Let (X, d_X) and (Y, d_Y) be two metric spaces, and $\varphi\colon X \to Y$ and $\psi\colon Y \to X$ two maps between them. The definitions of **distortion** of φ and the **codistortion** of the pair φ, ψ are, respectively,

$$\operatorname{dis}(\varphi) = \sup_{x,x'\in X} |d_X(x, x') - d_Y(\varphi(x), \varphi(x'))|$$
$$\operatorname{codis}(\varphi, \psi) = \sup_{x\in X,\ y\in Y} |d_X(x, \psi(y)) - d_Y(\varphi(x), y)|.$$

Theorem 2.1.12 ([27])**.** *For bounded metric spaces (X, d_X) and (Y, d_Y)*

$$d_{GH}(X, Y) = \frac{1}{2} \inf_{\substack{\varphi\colon X\to Y \\ \phi\colon Y\to X}} \max\{\operatorname{dis}(\varphi), \operatorname{dis}(\psi), \operatorname{codis}(\varphi, \psi)\},$$

where the infimum ranges over (not necessarily continuous) maps φ, ψ.

Even when the exact value of Gromov–Hausdorff distance cannot be computed, Theorem 2.1.12 gives a plethora of lower and upper limits for its value.

Corollary 2.1.13. *Let (X, d_X) and (Y, d_Y) be two metric spaces. Then*

$$d_{GH}(X, Y) \le \frac{1}{2} \inf_{\substack{\varphi\in S \\ \psi\in T}} \max\{\operatorname{dis}(\varphi), \operatorname{dis}(\psi), \operatorname{codis}(\varphi, \psi)\},$$

where S and T are subsets of all maps from X to Y and Y to X respectively. In particular,

$$d_{GH}(X, Y) \le \frac{1}{2} \max\{\operatorname{dis}(\varphi), \operatorname{dis}(\psi), \operatorname{codis}(\varphi, \psi)\}$$

for specific choices of $\varphi\colon X \to Y$ *and* $\psi\colon Y \to X$*. On the other hand,*

$$d_{GH}(X,Y) \geq \frac{1}{2} \inf_{\varphi\colon X \to Y} \operatorname{dis}(\varphi),$$
$$d_{GH}(X,Y) \geq \frac{1}{2} \inf_{\psi\colon Y \to X} \operatorname{dis}(\psi), \text{ and}$$
$$d_{GH}(X,Y) \geq \frac{1}{2} \inf_{\substack{\varphi\colon X \to Y \\ \phi\colon Y \to X}} \operatorname{codis}(\varphi,\psi).$$

This approach to estimating the value of Gromov–Hausdorff distance has been used in combination with the Borsuk–Ulam Theorem in [20] to improve the lower bounds and in some cases compute the exact value of Gromov–Hausdorff distance between m-dimensional spheres endowed with the geodesic distance. Some other useful properties of the Gromov–Hausdorff distance can be summarized as below.

Proposition 2.1.14 (Properties of the Gromov-Hausdorff distance)**.** *For bounded metric spaces* (X, d^X) *and* (Y, d^Y)

1. $d_{GH}(X,Y) < \infty$,
2. $d_{GH}(X,Y) \leq \frac{1}{2}\max\{\operatorname{Diam}(X), \operatorname{Diam}(Y)\}$,
3. $d_{GH}(X,Y) = \operatorname{Diam}(Y)$ *if* $X = \{x_0\}$.

2.2 Topological Methods in Data Science

In the research area of topological data analysis, one of the main assumptions made is that data is sampled from a nice lower-dimensional subspace in the ambient space. A lot of inherent information about the problem studied is therefore lost when taking a discrete sample. As a consequence, point clouds are augmented with topological structure, which is subsequently studied to hopefully uncover properties of the underlying subspace.

We describe one of the most popular such pipelines here. Some methods for augmentation with topological structure are presented in Section 2.2.1. Homology, the invariant of topological spaces used to analyze said augmentations, is described in Section 2.2.2, while its generalization,

persistent homology, used for analyzing nested families of augmentations, is presented in Section 2.2.3. The result of persistent homology is a persistence module which is considered to be a rich topological summary of a data set. In Section 2.2.4 we list some distances used to compare persistence modules, and we describe their algebraic structure in more detail in Section 2.2.5.

2.2.1 Shape Approximation

Throughout this section, $P = \{p_0, p_1, \ldots, p_k\}$ is a finite set. In practice, P is a point cloud and sits in an ambient space (such as, for example, $\mathbb{R}^d$), but the early definitions of this section can be stated more generally. We closely follow [28].

Definition 2.2.1. An **abstract simplicial complex** on P is a finite collection $K = \{K_i \subseteq P\}_i$ of subsets of P such that $\sigma \in K$ and $\tau \subseteq \sigma$ imply $\tau \in K$. The sets in K are called the **abstract simplices**. The **dimension** of an abstract simplex σ is $|\sigma| - 1$. Any non-empty subset $\tau \subseteq \sigma$ is a **face** of σ, and it is **proper** if $\sigma \neq \tau$. Similarly, σ is a **coface** of τ and is **proper** if $\sigma \neq \tau$. A **subcomplex** of K is an abstract simplicial complex L with $L \subseteq K$. For each $j \in \mathbb{N}$ the subcomplex $K^{(j)} = \{\sigma \in K \mid \dim(\sigma) \leq j\}$ consisting of all simplices in K of dimension at most j is called the **j-skeleton** of K. The 0-skeleton is also called the **vertex set**, and can be denoted as $\mathrm{Vert}(K)$. The **dimension** of an abstract simplicial complex is the maximum dimension of its abstract simplices. A **simplicial map** of K into L is a function $f\colon \mathrm{Vert}(K) \to \mathrm{Vert}(L)$ such that $\sigma \in K$ implies $f(\sigma) \in L$. Two abstract simplicial complexes K and L are **isomorphic** if there exists a bijection $f\colon \mathrm{Vert}(K) \to \mathrm{Vert}(L)$ such that $\alpha \in K$ if and only if $f(\alpha) \in L$, or, equivalently, when f and its inverse f^{-1} are simplicial maps.

Note that there exists a closely related notion of **geometric simplicial complex** with an additional requirement that P sits in an ambient space $\mathbb{R}^d$ and the points spanning any simplex in the complex must be affinely independent. It is easy to see this condition is not satisfied for a general abstract simplicial complex, since it is violated whenever the dimension of the abstract simplicial complex is bigger than the dimension of the ambient space. On the other hand, a geometric simplicial complex is always an abstract simplicial complex as well. Luckily, for any abstract simplicial complex K in ambient dimension d, one can always obtain a geometric simplicial complex G in a higher ambient dimension that is isomorphic to K. This result is called the Geometric Realization Theorem [28, pg. 64], and such a geometric simplicial complex G is called a **geometric realization** of K.

We give definitions of two methods of constructing a simplicial complex on a data cloud, namely the Čech and Vietoris–Rips complex. These are the only methods we discuss here, however, there are many others that are used frequently (such as the Delaunay complex [28, Chapter III.3], (weighted) alpha complex [28, Chapter III.4] and witness complex [29]).

Definition 2.2.2. The **Čech complex** on points $P = \{p_0, \ldots, p_n\} \subset \mathbb{R}^d$ at radius r is the simplicial complex

$$\check{\text{C}}\text{ech}(r) = \{S \subset P \mid \bigcap_{x \in S} B_x(r) \neq \emptyset\},$$

where $B_x(r)$ is the open ball of radius r centered at x.

Definition 2.2.3. The **Vietoris–Rips (VR) complex** on points $P = \{p_0, \ldots, p_n\} \subset \mathbb{R}^d$ at radius r is the simplicial complex

$$\text{VR}(r) = \{S \subset P \mid \max_{s,s' \in S} \|s - s'\| \leq 2r\}.$$

Note that the condition $\max_{s,s' \in S} \|s - s'\| \leq 2r$ is equivalent to saying that the open balls of radius r centered at points in S all pairwise intersect. A simple example of when the two con-

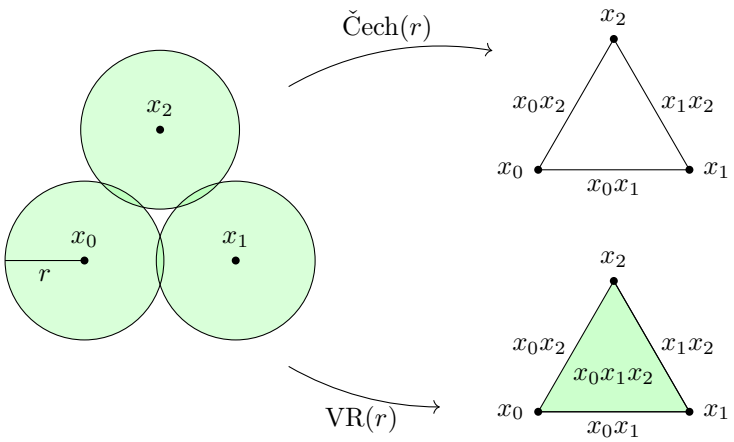


Figure 2.1: A simple example where the Čech and VR complex on the same dataset and with the same radius differ. Because the unit balls $B_x(r)$ do not have a triple intersection, the Čech complex Čech(r) does not contain a 2-simplex. However, since the balls intersect pairwise on $\{x_0, x_1, x_2\}$, the simplex they span is in VR(r).

structions differ is shown in Figure 2.1. Observe the following: the Vietoris-Rips complex $\mathrm{VR}(r)$ is determined by its 1-skeleton. A k-simplex σ is in $\mathrm{VR}(r)$ if and only if each of its edges $\epsilon \in \sigma^{(1)}$ is in $\mathrm{VR}(r)$. This makes computations easier, which is why Vietoris-Rips complexes are the most widely used construction for shape approximation. Čech complexes, however, come with nice theoretical guarantees.

Definition 2.2.4. A **homotopy** between continuous maps $f, g\colon X \to Y$, where X and Y are topological spaces, is another continuous map $H\colon X \times [0,1] \to Y$ such that

$$H(x,0) = f(x)$$
$$H(x,1) = g(x)$$

for all $x \in X$. If a homotopy between f and g exists, we write $f \simeq g$.

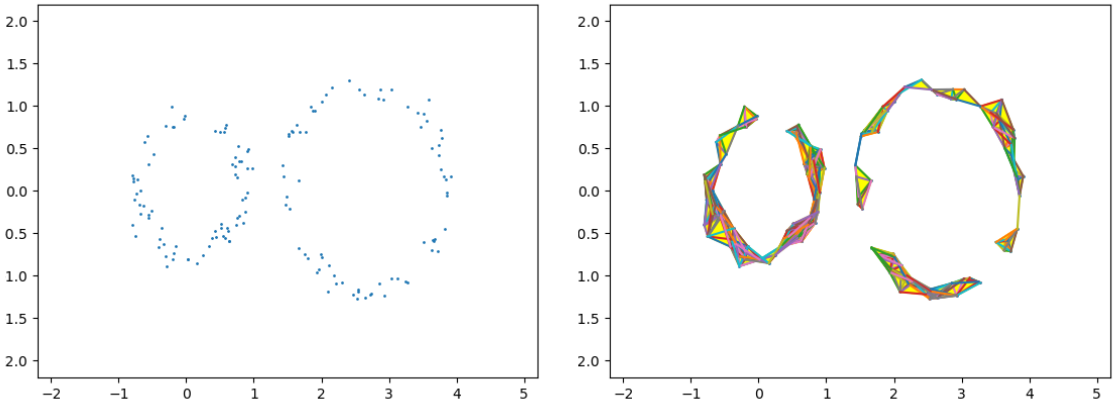
Two topological spaces X and Y are **homotopy equivalent** if there exist continuous maps $f\colon X \to Y$ and $g\colon Y \to X$ such that $g \circ f \simeq \mathrm{Id}_X$ and $f \circ g \simeq \mathrm{Id}_Y$. In that case, we write $X \simeq Y$ and refer to f and g as homotopy equivalences.

Note that in both cases $\simeq$ is an equivalence relation. As a consequence we often say that homotopy equivalent spaces are "of the same homotopy type".

Theorem 2.2.5. *The geometric realization of the Čech complex on P at radius r is homotopy equivalent to the union of open balls $\cup_{p\in P} B_p(r)$.*

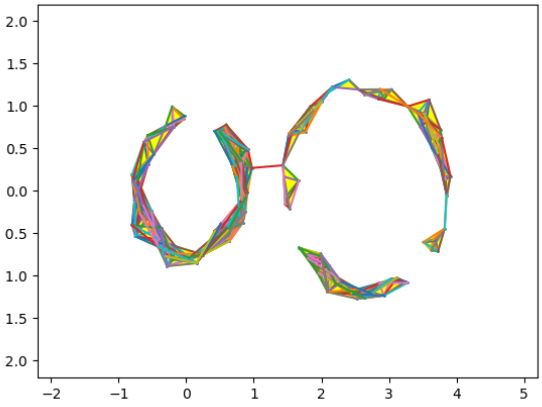
Remark 2.2.6. Theorem 2.2.5 is a corollary of the **Nerve Theorem**, several versions of which appear in [30, 31, 32]. It is especially strong if we assume that the data set P lies along a latent low-dimensional manifold embedded in ambient space, which is known as the **manifold hypothesis**. If the chosen radius r is sufficiently small, and the sample P is dense enough with respect to r (for specific conditions and more details on this line of research, see [33, 34]), then Theorem 2.2.5 posits that $\check{\mathrm{C}}\mathrm{ech}(r)$ on P is homotopy equivalent to the latent manifold.

Shape approximation using a simplicial complex requires the choice of radius. Intuitively, choosing smaller radius will lead to more local properties of the latent manifold being preserved, while a bigger radius favors global properties (using too large a radius, however, leads to loss of all relevant information). Unfortunately, properties on different scales can rarely be preserved in one single simplicial complex for one chosen radius, as for example in Figure 2.2.

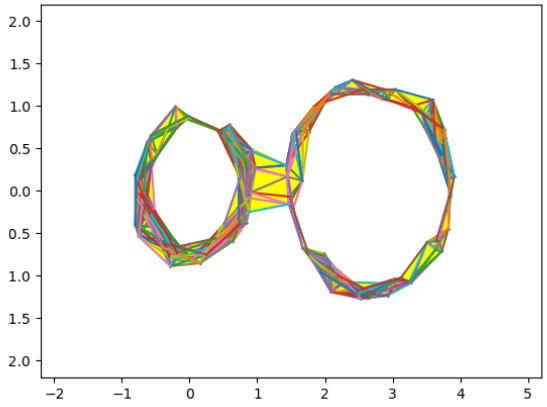


(a) Point cloud $P \subseteq \mathbb{R}^2$.

(b) VR(0.4) on P.



(c) VR(0.47) on P.



(d) VR(0.6) on P.

Figure 2.2: An example of a point cloud (a) and Vietoris-Rips complexes on it at different radii: $r = 0.4$ in (b), $r = 0.47$ in (c), and $r = 0.6$ in (d). We clearly see the point cloud is sampled from two disjoint circles, but the Vietoris-Rips complex is not able to capture this information at any one radius. This is especially well illustrated in (c), where a part of the bigger circle is already connected to the smaller one while not yet being fully connected with parts of itself. In addition, we would expect two cycles to be visible from the complex, but none have been formed yet. When they are formed, see (d), so many points are connected that the information about there being two connected components is lost.

Luckily, both Čech and VR complexes (use K to denote either) have the property that for radii $r_1 \leq r_2$ the complex $K(r_1)$ is contained in $K(r_2)$. This is very useful when we want to study the evolution of topological properties with respect to scale.

Definition 2.2.7 (Filtered simplicial complex)**.** A sequence $\{K_i\}_{i=0,\ldots,n}$ of simplicial complexes, such that

$$\emptyset = K_0 \subseteq K_1 \subseteq \ldots \subseteq K_n,$$

is called **filtered simplicial complex**.

Sometimes, as is the case for filtered Čech or VR simplicial complexes induced by an increasing sequence of radii $r_0 < r_1 < \ldots < r_n$, the first simplicial complex in the sequence need not be an empty set. If that is the case, we add it artificially. Further, we sometimes require that K_n be a saturated simplicial complex (meaning that any simplex spanned by a subset of vertices of K_n is in K_n).

When using filtered simplicial complexes to study topological properties at different scales, further justification for using VR complexes can be given.

Theorem 2.2.8 (Theorem 2.5 of [35])**.** *For a finite $S \subset \mathbb{R}^d$ and $r \geq 0$,*

$$\check{\text{C}}\text{ech}(r) \subseteq VR(r) \subseteq \check{\text{C}}\text{ech}\left(\sqrt{\frac{2d}{d+1}}r\right).$$

This fact can be leveraged to show that under certain assumptions about "goodness" of our sample P with respect to a measure of convexity defect, the VR and Čech complexes on P are homotopy equivalent [36], which means a version of Theorem 2.2.5 under stronger assumptions holds for VR complexes as well.

The main method used to study filtered simplicial complexes is **persistent homology**, which we present in Sections 2.2.2 and 2.2.3.

2.2.2 Homology

In this section we present homology, a method to study topological properties of spaces. As we are mainly interested in simplicial complexes, we define simplicial homology. Although most concepts of this section can be formulated over commutative rings, we choose to instead work with a fixed field $\mathbb{F}$ to bypass certain complications that would otherwise arise.

Definition 2.2.9. Let $p \in \mathbb{Z}$ be a dimension and K a simplicial complex. The free $\mathbb{F}$-module on the basis of p-simplices in K is called the **p-chain group** and denoted as $C_p = C_p(K)$. Each element in $C_p(K)$ is called a **p-chain**.

A p-chain is therefore a formal sum $\sum a_i\sigma_i$ of p-simplices in K with $a_i \in \mathbb{F}$. The addition is done component-wise, *i.e.* $\sum a_i\sigma_i + \sum b_i\sigma_i = \sum(a_i + b_i)\sigma_i$, and it inherits associativity and commutativity from $\mathbb{F}$. Further, the neutral element is $0 = \sum 0\sigma_i$, and an inverse of $c = \sum a_i\sigma_i$ is $-c = \sum(-a_i)\sigma_i$. Since $\mathbb{F}$ is a field, chain groups C_p are vector spaces.

Although chain groups can be defined for all integer values of p, they are trivial for $p < 0$ and $p > \dim(K)$. They are related between each other in the following way.

Definition 2.2.10. The **boundary** of a p-simplex $\sigma(x_0, x_1, \ldots, x_p)$ is the formal sum

$$\partial_p(\sigma) = \sum_{k=0}^{p}(-1)^k[x_0, \ldots, \hat{x_k}, \ldots, x_p],$$

where $[x_0, \ldots, \hat{x_k}, \ldots, x_p]$ is the co-dimension 1 face of σ spanned by $x_0, \ldots, x_{k-1}, x_{k+1}, \ldots, x_p$. The boundary generalizes to p-chains as

$$\partial_p(c) = \sum a_i\partial_p(\sigma_i),$$

where $c = \sum a_i\sigma_i$. This defines a map $\partial_p \colon C_p \to C_{p-1}$ called the **p-th boundary map**. By combining the chain groups and boundaries for all dimensions $p \in \mathbb{Z}$, we obtain the **chain complex**

$$\cdots \xrightarrow{\partial_{p+2}} C_{p+1} \xrightarrow{\partial_{p+1}} C_p \xrightarrow{\partial_p} \cdots \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0 \xrightarrow{\partial_0} 0.$$

By definition the boundary maps commute with addition in the chain groups and so are linear maps. We give elements in their image and their kernel a special name.

Definition 2.2.11. As the name suggests, a p-chain is a **p-boundary**, if it is a boundary of a $(p+1)$-chain. The set $B_p = B_p(K)$ of all p-boundaries is a subspace of C_p, called the **group of p-boundaries**. A **p-cycle** is a p-chain, whose boundary is the zero $(p-1)$-chain. The set $Z_p = Z_p(K)$ of all p-cycles is a subspace of C_p, called the **group of p-cycles**.

As we implied before, $B_p = \operatorname{im} \partial_{p+1}$ and $Z_p = \ker \partial_p$. What is even more interesting, the group of p-boundaries is a subgroup of the group of p-cycles.

Theorem 2.2.12 (Fundamental Lemma of Homology, [28, p. 95])**.** *For any $p \in \mathbb{Z}$ the composition of boundary maps $\partial_{p-1}\partial_p$ is the zero map.*

This enables us to form quotients and differ between cycles up to boundaries (see Figure 2.3), which naturally leads to the notion of homology.

Definition 2.2.13. The quotient vector space $H_p(K) = Z_p(K)/B_p(K)$ is the **p-th homology group** of simplicial complex K. The rank of H_p is the p-th Betti number $\beta(K)$.

Homology groups are invariants of topological spaces. Intuitively, $H_0(K)$ describes the connected structure of K, and its elements are formal sums of **connected components**. The higher dimensional homology groups describe the **holes** (for $p = 1$) and **voids** (for $p \geq 2$) appearing in K.

Example 2.2.14 (Homology of spheres)**.** Approximate an n-sphere $\mathbb{S}^n = \{x \in \mathbb{R}^{n+1} \mid \|x\|_2 = 1\}$ with the simplicial complex that is the boundary of an $(n+1)$-simplex. One can then compute the homology groups and obtain that

$$H_p(\mathbb{S}^0) \cong \begin{cases} \mathbb{F} \oplus \mathbb{F}, & p = 0, \\ 0, & \text{otherwise}, \end{cases} \quad \text{and} \quad H_p(\mathbb{S}^n) \cong \begin{cases} \mathbb{F}, & p = 0 \text{ or } p = n, \\ 0, & \text{otherwise}. \end{cases} \qquad \triangle$$

2.2.3 Persistent Homology

As mentioned in Section 2.2.1, persistent homology is a method used to study the evolution of topological properties of a filtered simplicial complex

$$\emptyset = K_0 \subseteq K_1 \subseteq \ldots \subseteq K_n.$$

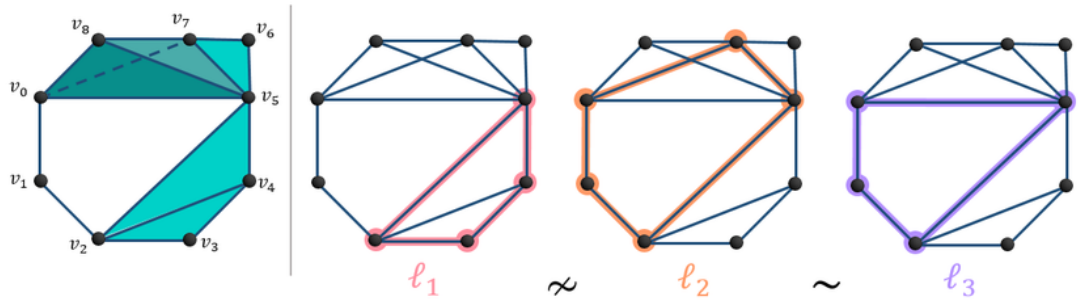


Figure 2.3: An example of a simplicial complex K on the left, and three elements of the group of its 1-cycles $Z_1(K)$ on th right: $\ell_1 = v_2v_3+v_3v_4+v_4v_5+v_2v_5$, $\ell_2 = v_0v_1+v_1v_2+v_2v_5+v_5v_7-v_0v_7$, and $\ell_3 = v_0v_1 + v_1v_2 + v_2v_5 - v_0v_5$. Notice that ℓ_1 is also a boundary of $v_2v_3v_4 + v_2v_4v_5$, and is contractible within K. This is not true for cycles ℓ_2 and ℓ_3, however one can be transformed into the other using a homotopy within K. Notice that these properties are respected by homotopy: $q_1(\ell_1) = 0$, while $q_1(\ell_2), q_1(\ell_3) \neq 0$. Moreover, since the difference $\ell_2-\ell_3 = v_0v_5+v_5v_7-v_0v_7$ is a boundary, $q_1(\ell_2) = q_1(\ell_3)$. This illustrates that H_1 homology counts "holes" in a simplicial complex. Source: [37].

The topological properties we speak of are the homology groups. We can apply H_p to each complex in the sequence for any dimension p we are interested in. The inclusions $K_i \hookrightarrow K_j$ for $i \leq j$ induce homomorphisms $H_p(K_i) \to H_p(K_j)$, which gives a sequence

$$0 = H_p(K_0) \to H_p(K_1) \to \ldots \to H_p(K_n).$$

This sequence is an example of a persistence module.

Remark 2.2.15. Persistent homology is more generally the study of the evolution of homology through a nested sequence of topological spaces, called a **filtration**. A filtered simplicial complex is a filtration where each of the topological spaces is a simplicial complex, but it is far from being the only useful or interesting example. Other examples include **sublevel set filtration** and **superlevel set filtration**, which are, given a function $f\colon X \to \mathbb{R}$ on a topological space X, the nested sequence of $\{f^{-1}((-\infty, t])\}_{t\in\mathbb{R}}$ and the nested sequence of $\{f^{-1}([t, \infty))\}_{t\in\mathbb{R}}$ respectively. Those are the standard filtrations used in Morse theory [38].

Definition 2.2.16 (General definition of a persistence module)**.** A **persistence module** is a covariant functor $V\colon P \to \mathbb{V}\mathrm{ect}(\mathbb{F})$, where $\mathbb{V}\mathrm{ect}(\mathbb{F})$ is the category of $\mathbb{F}$-vector spaces and P is a poset viewed as a category with objects $p \in P$ and the set of morphisms $\mathrm{Hom}(p, q)$ containing one element if and only if $p \leq q$. In other words, V assigns an $\mathbb{F}$-module V_p to each element $p \in P$

and a linear map $v_{p,q}\colon V_p \to V_q$, called an **inner morphism**, to any pair of indices $p, q \in P$ with $p \leq q$. Let V and W be two persistence modules indexed over the same poset P. A **(persistence) morphism** between V and W is a family $\Phi = \{\Phi_p\colon V_p \to W_p\}_{p\in P}$ of linear maps that commute with the inner morphisms, that is $\Phi_q \circ v_{p,q} = w_{p,q} \circ \Phi_p$ for all $p \leq q$. It is a **(persistence) isomorphism** if its constituent maps Φ_p are all isomorphisms.

Note that Definition 2.2.16 is given in full generality. However, throughout this work we only ever work with two types of persistence modules:

a) A **finitely indexed pointwise finite dimensional (p.f.d.) persistence module** is a persistence module $V\colon [\ell+1] \to \mathbb{V}\mathrm{ect}$, where $[\ell+1] = \{0, 1, \ldots, \ell\}$ is the totally ordered set with $\ell+1$ elements and $\mathbb{V}\mathrm{ect}$ is the category of finite dimensional vector spaces over $\mathbb{F}$. In practice, it is sensible to assume a persistence module is of this type, because the homology of a one-parameter filtration built on a finite real-world data set changes finitely many times when we vary the filtration parameter.

b) A **ladder persistence module** (V, W, Φ) consists of finitely indexed p.f.d. persistence modules $V, W\colon [\ell+1] \to \mathbb{V}\mathrm{ect}$ and a family of linear maps $\{\Phi_i\}_{i\in[l+1]}$ arranged in a commutative diagram

$$\begin{array}{ccccccccccc} V_0 & \xrightarrow{v_{0,1}} & V_1 & \xrightarrow{v_{1,2}} & V_2 & \longrightarrow & \cdots & \longrightarrow & V_{l-1} & \xrightarrow{v_{l-1,l}} & V_l \\ \downarrow{\scriptstyle \Phi_0} & & \downarrow{\scriptstyle \Phi_1} & & \downarrow & & \downarrow & & \downarrow{\scriptstyle \Phi_{l-1}} & & \downarrow{\scriptstyle \Phi_l} \\ W_0 & \xrightarrow[w_{0,1}]{} & W_1 & \xrightarrow[w_{1,2}]{} & W_2 & \longrightarrow & \cdots & \longrightarrow & W_{l-1} & \xrightarrow[w_{l-1,l}]{} & W_l. \end{array}$$

Note that this definition suffices in our setting, but it can be stated more generally [39] in the setting of **zig-zag persistence** [40], where the arrows of inner morphisms v and w can be reversed, as long as the direction is the same for $v_{i,i+1}$ and $w_{i,i+1}$ for all $i \in [l+1]$.

Notice that the notion of a morphism Φ between finitely indexed p.f.d. persistence modules V and W is equivalent to the notion of a ladder persistence module (V, W, Φ). Further, p.f.d. persistence modules are nice to work with because they are a direct sum of intuitive building blocks.

Definition 2.2.17. Let P be a poset and $J \subset P$ an interval within it, *i.e.* for all $j \leq k \in J$ any l satisfying $j \leq l \leq k$ is also in J. The **interval persistence module** $k_J\colon P \to \mathbb{V}\mathrm{ect}$ consists of vector spaces

$$(k_J)_p = \begin{cases} \mathbb{F}, & \text{if } p \in J, \\ 0, & \text{otherwise,} \end{cases}$$

combined with linear maps $(k_J)_p \to (k_J)_q$ that are identity whenever $p \leq q \in J$, and the zero map otherwise.

Theorem 2.2.18 (Structure Theorem [41])**.** *Every finitely indexed p.f.d. persistence module $V\colon P \to \mathbb{V}ect$ is isomorphic to a direct sum of finitely many interval persistence modules. In other words, there exist a finite multiset $Bar(V) = \{(a_i, b_i) \in P^2\}_i$ such that*

$$V \cong \bigoplus_{(a,b)\in Bar(V)} k_{[a,b]}.$$

This decomposition uniquely determines the persistence module, so we compress the information it carries as follows.

Definition 2.2.19. The multiset $\mathrm{Bar}(V)$ is the **barcode** of persistence module V. The multiset

$$\mathrm{Diag}(V) = \mathrm{Bar}(V) \cup \Delta,$$

where Δ is the multiset containing each point $(a, a) \in P^2$ with multiplicity ∞, is the **persistence diagram** of V.

2.2.4 Bottleneck and Interleaving Distance

To compare pairs of persistence modules, many notions of distance have been introduces on their set. Most commonly used, the bottleneck and the Wasserstein distances, are defined via persistence diagrams. We state the definition of the former, and refer the reader to [42] for information on the latter.

Definition 2.2.20. Let V, W be finitely indexed p.f.d. persistence modules and $\mathrm{Diag}(V)$ and $\mathrm{Diag}(W)$ their persistence diagrams. The **bottleneck distance** between V and W is

$$d_B(V, W) = \inf_{\gamma} \sup_{x} ||x - \gamma(x)||_\infty$$

where the supremum ranges over all $x \in \mathrm{Diag}(V)$, and the infimum ranges over all bijections $\gamma\colon \mathrm{Diag}(V) \to \mathrm{Diag}(W)$.

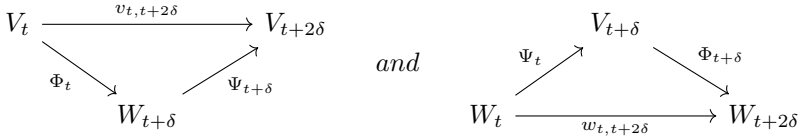
The bottleneck distance is a metric on the space of persistence diagrams (assuming as above they are obtained from finitely indexed p.f.d. persistence diagrams). However, one can also define an

extended pseudo-metric on the space of persistence modules that is algebraic in nature and does not require the computation or even existence of the interval decomposition. An integral role in its definition is played by δ-interleaving morphisms.

Definition 2.2.21. The **shift** of a persistence module V is a persistence module $V(\delta)$ defined as

$$V(\delta)_t = V_{t+\delta}$$
$$v(\delta)_{t_1,t_2} = v_{t_1+\delta,t_2+\delta}$$

for any $\delta \geq 0$ (where the persistence module $V(0)$ is simply V itself). Persistence modules V and W are δ-**interleaved** if there exist morphisms $\Phi\colon V \to W(\delta)$ and $\Psi\colon W \to V(\delta)$ such that the diagrams



commute. The pair (Φ, Ψ) is called a δ-**interleaving**.

Note that two persistence modules are 0-interleaved if and only if they are isomorphic. As a consequence, we often consider the parameter δ to measure how far from isomorphic two persistence modules can be.

Definition 2.2.22. The **interleaving distance** d_I is an extended pseudometric on the set of persistence modules defined as

$$d_I(V, W) = \inf \{\delta \mid V \text{ and } W \text{ are } \delta\text{-interleaved}\}.$$

Remark 2.2.23. Since interleaving distance does not require the existence of interval decomposition, it can be stated in a more general setting than the bottleneck distance, in which it is "only" an extended pseudometric. Indeed, notice that interval persistence modules $k_{[a,b]}$ and $k_{[a,b)}$ are δ-interleaved for all $\delta > 0$ but not isomorphic, which violates identity; and that there is no δ for which $k_{[a,\infty)}$ is δ-interleaved with an interval persistence module with bounded support, and it is thus possible for the interleaving distance not to be finite. However, on the set of finitely indexed p.f.d. persistence module, the interleaving distance is, in fact, a metric. Further, it agrees with the bottleneck distance on that same set.

Theorem 2.2.24 (Isometry Theorem [10]). *For any pair V, W of finitely indexed p.f.d. persistence modules*

$$d_B(V,W) = d_I(V,W).$$

In the rest of this work, we assume all persistence modules are finitely indexed and p.f.d. unless stated otherwise.

2.2.5 Barcode Basis

Here, we mostly follow the terminology of [11]. To begin with, define the following relations on the set of intervals:

$$\begin{aligned}
[i_1,j_1] \leq [i_2,j_2] &\iff i_1 < i_2 \text{ or } (i_1 = i_2 \text{ and } j_1 < j_2),\\
[i_1,j_1] \preceq [i_2,j_2] &\iff i_1 \leq i_2 \leq j_1 \leq j_2\\
[i_1,j_1] \subset [i_2,j_2] &\iff i_2 < i_1 \leq j_1 < j_2
\end{aligned}$$

The first, $\leq$, is simply the lexicographical order (total), while the second, $\preceq$, is not transitive and therefore not an order. The relation $\preceq$ is uniquely useful in persistence barcodes, since it encodes the restrictions of how a morphism of persistence modules can map (see Remark 2.2.28). It is often referred to as the **overlapping relation** [43]. Finally, the relation $I \subset J$ simply states that I is **strictly nested** in J. Note that when a pair of bars $I \leq J$ has a nonempty intersection and $I \not\preceq J$, it must be strictly nested as $J \subset I$.

Let V be a persistence module indexed over $[\ell+1] = \{0, 1, \ldots, \ell\}$ and let $n_i = \dim_{\mathbb{F}} V_i$. Select a **basis family**

$$\mathcal{B} = \{B_i \subset V_i \mid B_i \text{ is an ordered basis of } V_i \text{ for all } i \in [\ell+1]\}.$$

In these bases the inner morphisms $v_{i-1,i}\colon V_{i-1} \to V_i$ can be represented as $n_i \times n_{i-1}$ matrices A_i. Consequently, the module V is isomorphic to

$$\mathbb{F}^{n_0} \xrightarrow{A_1} \mathbb{F}^{n_1} \xrightarrow{A_2} \cdots \xrightarrow{A_{\ell-1}} \mathbb{F}^{n_{\ell-1}} \xrightarrow{A_\ell} \mathbb{F}^{n_\ell}.$$

Definition 2.2.25. Let $\omega\colon \oplus_{[a,b]} k_{[a,b]} \to V$ be an isomorphism between the interval decomposition of V and V, and α_i its component at index i. Then the basis family

$$\mathcal{B} = \{\operatorname{im}\omega_i \subset V_i \mid i \in [\ell+1]\}$$

is a **barcode basis** of V.

Intuitively, a barcode basis is a basis family $\mathcal{B}$ in which a bar corresponds to a sequence of basis vectors, each one mapping to the next with the inner morphisms. After fixing an order on $\mathrm{Bar}(V)$, this corresponds to the matrix representations A_i of inner morphisms being in **barcode form**: in row-echelon form with all pivots equaling 1 and all other entries being 0,

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Via this observation, we could give an alternative definition of the barcode basis as a basis family in which all matrices A_i are in barcode form with respect to the chosen order (as in [11]).

The choice of barcode basis for a persistence module is not unique, and we can change the barcode basis via the action of $G = \prod_{i=0}^{l} \mathrm{GL}(n_i; \mathbb{F})$. The action of an element $g = (g_0, g_1, \ldots, g_l)$ of G on a matrix sequence $A = \{A_i\}_i$ is given by

$$(gA)_i = g_i \cdot A_i \cdot g_{i-1}^{-1},$$

which corresponds to switching from basis $\mathcal{B} = \{B_i\}$ to $g\mathcal{B} = \{g_i\mathcal{B}_i\}$. By [11, Proposition 2.4.] any matrix sequence $A \in X$ can be put in a barcode form by the action of G. The changes of basis in G that keep the matrix sequence A unchanged are part of the **stabiliser** of A,

$$\mathrm{Stab}(A) = \left\{ g \in G \mid A_i = g_i \cdot A_i \cdot g_{i-1}^{-1} \text{ for all } 1 \le i \le l \right\}. \tag{2.2}$$

Introducing barcode bases enables us to write persistence morphisms as matrices.

Definition 2.2.26. Let $\{B_i\}_{i\in[l+1]}$ be a barcode basis of persistence module V corresponding to an isomorphism $\omega\colon \oplus_{[a,b]} k_{[a,b]} \to V$. For a bar J of multiplicity μ in $\mathrm{Bar}(V)$, a family

$$x_J^{(j)} = \{(\omega|_{k_J^{(j)}})_i \in B_i\}_{i\in J},$$

where $0 \le j \le \mu$, is called a **generator of bar** J. Note that there are μ distinct choices for a generator of J as implied by the index j. Whenever the explicit choice is not important, we use the notation x_J.

A collection $= \{b_i \in B_i\}_{i\in J}$ is a generator of bar $J = [\alpha, \beta]$ if and only if it satisfies

- $A_{i+1}b_i = b_{i+1}$ for $i \in [\alpha, \beta - 1]$,
- $A_{\beta+1}b_\beta = 0$ and
- $\langle A_\alpha b_{\alpha-1}, b_\alpha\rangle_\alpha = 0$ for all $b_{\alpha-1} \in B_{\alpha-1}$,

where the inner product $\langle\cdot,\cdot\rangle_i$ is induced by the basis B_i.

Proposition 2.2.27. *Choose barcode bases for persistence modules V and W. Then any morphism $\Phi\colon V \to W$ can be written as a single matrix*

$$M_\Phi = \begin{bmatrix}
X_{[0,0]}^{[0,0]} & X_{[0,0]}^{[0,1]} & \dots & X_{[0,0]}^{[0,l]} & 0 & \dots & 0 & \dots & 0 \\
 & X_{[0,1]}^{[0,1]} & \dots & X_{[0,1]}^{[0,l]} & X_{[0,1]}^{[1,1]} & \dots & X_{[0,1]}^{[1,\ell]} & \dots & 0 \\
 & & \ddots & \vdots & \vdots & & \vdots & & \vdots \\
 & & & X_{[0,l]}^{[0,l]} & 0 & \dots & X_{[0,\ell]}^{[1,\ell]} & \dots & X_{[0,\ell]}^{[\ell,\ell]} \\
 & & & & X_{[1,1]}^{[1,1]} & \dots & X_{[1,1]}^{[1,\ell]} & \dots & 0 \\
 & & & & & \ddots & \vdots & & \vdots \\
 & & & & & & X_{[1,\ell]}^{[1,\ell]} & \dots & X_{[1,\ell]}^{[\ell,\ell]} \\
 & & & & & & & \ddots & \vdots \\
 & & & & & & & & X_{[\ell,\ell]}^{[\ell,\ell]}
\end{bmatrix}, \tag{2.3}$$

where each sub-matrix $X_{[i_2,j_2]}^{[i_1,j_1]}$ encodes how Φ maps generators of bar $[i_1, j_1]$ in $Bar(V)$ to generators of bar $[i_2, j_2]$ in $Bar(W)$. The rows and columns are ordered with respect to the overlapping relation, $\preceq$, increasingly, using arbitrary order among generators of the same bar.

The proof of Proposition 2.2.27 is included in the proof of [11, Theorem 4.3.] as Step 1. Notice that a bar J with multiplicity μ_J has μ_J columns (or rows) associated to it, each belonging to one of the generators x_J.

Remark 2.2.28. It is easy to see that there cannot be a non-zero morphism $I_{J_1} \to I_{J_2}$ between interval persistence modules unless $J_2 \preceq J_1$ (this must hold if the commuting-squares requirement in the definition of a morphism of persistence modules is to be satisfied). This is reflected in the general matrix shape (2.3), where the only non-zero block matrices $X_{J_2}^{J_1}$ are associated with bars $J_2 \preceq J_1$.

2.3 Directed Spaces

Following [44, 19] we present directed spaces, which are topological spaces with additional directed structure. Let therefore X be a topological space and I the unit interval $[0, 1]$ with euclidean topology.

Definition 2.3.1. A continuous map $\gamma\colon I \to X$ is called a **path** on X, and $\gamma(0)$ and $\gamma(1)$ are its **source** and **target** respectively. The **reverse** path of γ is the path $\gamma^*\colon I \to X$ with $\gamma^*(t) = \gamma(1-t)$. Given two paths, γ, with source x and target x', and γ', with source x' and target x'', their **concatenation**, $\gamma \star \gamma'$ is a path defined as

$$\gamma \star \gamma'(t) = \begin{cases} \gamma(2t), & \text{if } 0 \leq t \leq \frac{1}{2} \\ \gamma'(2t-1), & \text{if } \frac{1}{2} \leq t \leq 1 \end{cases}$$

and it is a path with source x and target x''.

Note that this definition of a path differs from the one in Section 2.1.2, since the domain interval I in Definition 2.3.1 is fixed. As we justify later in this section, however, the two theories coincide nicely.

Definition 2.3.2. A **directed space** or **d-space** is a pair $(X, \vec{P}(X))$, where X is a topological space and $\vec{P}(X)$ is a collection of paths on X, called the **directed structure** on X, such that:

- **DS1** Every constant path $c_x\colon I \to \{x\}$ for $x \in X$ belongs to $\vec{P}(X)$.
- **DS2** The precomposition $h\gamma$ of a path γ in $\vec{P}(X)$ with any weakly increasing continuous $h\colon I \to I$ is in $\vec{P}(X)$ (**partial reparametrization**).

DS3 The concatenation $\gamma \star \gamma'$ of two paths γ and γ' in $\vec{P}(X)$ is also in $\vec{P}(X)$.

Whenever the directed structure does not need to be specified we denote the pair $(X, \vec{P}(X))$ as $\vec{X}$. The elements of $\vec{P}(X)$ are called **d-paths** on X. The subset of $\vec{P}(X)$ containing those d-paths γ with $\gamma(0) = x$ and $\gamma(1) = x'$ is denoted by $\vec{P}(x, x')$. Any subset $Y \subseteq X$ can inherit the topology and the directed structure from $\vec{X}$ by setting $\vec{P}(Y) = \{\gamma \in \vec{P}(X) \mid \gamma(I) \subseteq Y\}$. Such a d-space $(Y, \vec{P}(Y))$ is called a **d-subspace** of $\vec{X}$. Further, any d-space $\vec{X}$ has a **reverse d-space** $\vec{X}^* = (X, \vec{P}(X)^*)$ whose directed structure $\vec{P}(X)^*$ is given by the reverse paths of d-paths in $\vec{P}(X)$. Given two d-spaces $\vec{X}$ and $\vec{Y}$, a **directed map** or **d-map** is a continuous map $F\colon X \to Y$ such that, for every γ in $\vec{P}(X)$, the composition $F\gamma$ is in $\vec{P}(Y)$. We denote it by $\vec{F}\colon \vec{X} \to \vec{Y}$. The category of d-spaces with d-maps is denoted **dTop**.

Since partial reparametrizations are not necessarily surjective, Property **DS2** implies that $\vec{P}(X)$ is closed under taking subpaths. As we defined it, concatenation is not associative, but it is up to reparametrization.

Observe that a topological space X can be endowed with many directed structures. Two simple examples that can be defined for any X are the **discrete directed structure** $\vec{P}(X) = \{c_x\}_{x \in X}$, which consists of only the constant paths, and the **trivial directed structure** $\vec{P}(X) = X^I$, which consists of all paths on X. The d-spaces $(X, \{c_x\}_{x \in X})$ and (X, X^I) are called the **discrete d-space** and the **trivial d-space** respectively. Moreover, the set of all directed structures on a topological space X form a lattice [44].

Definition 2.3.3. Let $\mathbb{P}(X)$ be the set of all possible directed structures on a topological space X, so that $(X, \vec{P})$ is a d-space for every $\vec{P} \in \mathbb{P}(X)$. Endow it with a partial order

$$\vec{P} \leq_{\mathbb{P}} \vec{Q} \iff \vec{P} \subseteq \vec{Q}.$$

The order induces a lattice structure on $\mathbb{P}(X)$ by defining the **meet** of $\vec{P}$ and $\vec{Q}$ as $\vec{P} \wedge \vec{Q} := \vec{P} \cap \vec{Q}$, and their **join** as $\vec{P} \vee \vec{Q} := \overline{\vec{P} \cup \vec{Q}}$, where $\overline{\vec{P} \cup \vec{Q}}$ is the closure of $\vec{P} \cup \vec{Q}$ under finite concatenation and partial reparametrization. The lattice $(\mathbb{P}(X), \leq_{\mathbb{P}})$ is complete, and bounded by the trivial directed structure as **top element** $\top$ and the discrete directed structure as **bottom element** $\bot$.

Definition 2.3.4. For a subset $A \subset X^I$ of paths on a topological space X, the **directed structure generated by** A is the smallest directed structure containing A,

$$\vec{A} = \bigwedge \{\vec{P} \in \mathbb{P}(X) \mid A \subseteq \vec{P}\} = \bigcap \{\vec{P} \in \mathbb{P}(X) \mid A \subseteq \vec{P}\}.$$

The d-space $(X, \vec{A})$ is called the **d-space generated by** A.

An interesting class of examples of d-spaces arises from posets. A topological space with a partial order $(X, \leq_X)$ defines a d-space $(X, \vec{P}(X))$ where $\vec{P}(X)$ is the set of weakly increasing paths with respect to $\leq_X$, i.e., paths γ such that $\gamma(s) \leq_X \gamma(t)$ for every $s \leq t$. For example, the product order induces a directed structure on $\mathbb{R}^k$.

Example 2.3.5 (Directed (Hollow) Hypercubes)**.** Let $Q_n = \delta I^n \subset \mathbb{R}^n$ be the boundary of the n-cube for some $n \geq 1$. Denote its facets as

$$F^i_j = I^{i-1} \times \{j\} \times I^{n-i},$$

where $j = 0, 1$ and $i = 1, \ldots, n$, and endow each of them with the partial order $\leq^i_j$ inherited from $\mathbb{R}^n$. Since the corresponding orders agree on the intersections of facets, we can define a partial order $\leq_{Q_n}$ on Q_n so that $\leq_{Q_n}$ restricted to F^i_j is $\leq^i_j$ for any $i = 1, \ldots, n$ and $j = 0, 1$. Define the **directed (hollow)** n**-cube** $\vec{Q}_n$ to be the pospace $(Q_n, \vec{P}(Q_n))$ with the directed structure $\vec{P}(Q_n)$ containing all weakly increasing paths with respect to the partial order $\leq_{Q_n}$. △

2.3.1 Operations on Directed Spaces

The category **dTop** of d-spaces is complete and cocomplete [19, Proposition 4.5], which in particular means it is closed under finite products, coproducts, pullbacks and pushouts. Below, we specify what the underlying topological spaces and the distinguished paths are for a few such constructions. Therefore, let $(X, \vec{P}(X))$ and $(Y, \vec{P}(Y))$ be two d-spaces.

Definition 2.3.6. The **Cartesian product of d-spaces** $\vec{X}$ **and** $\vec{Y}$, denoted by $\vec{X} \times \vec{Y}$, is the topological space $X \times Y$ with the directed structure given by the product $\vec{P}(X \times Y) := \vec{P}(X) \times \vec{P}(Y)$ in **Set**.

The **disjoint union of d-spaces** $\vec{X}$ **and** $\vec{Y}$, denoted by $\vec{X} \sqcup \vec{Y}$, is the topological space $X \sqcup Y$ with the directed structure given by the coproduct $\vec{P}(X \sqcup Y) := \vec{P}(X) \sqcup \vec{P}(Y)$ in **Set**.

Let $\vec{X}$ be a directed space, $\sim$ an equivalence relation on the underlying topological space X, and $\pi \colon X \to X/_\sim$ the corresponding quotient map. Quotient space $X/_\sim$ can be endowed with a set of d-paths $\vec{P}(X/_\sim)$ inherited from $\vec{X}$ via the quotient map, namely

$$\vec{P}(X/_\sim) = \{\pi(\gamma) \mid \gamma \in \vec{P}(X)\}.$$

The resulting d-space is called the **quotient d-space of** $\vec{X}$ **under** $\sim$, and denoted as $\vec{X}/_\sim$.

2.3.2 Directed Spaces and Length Structures

Consider a d-space $\vec{X}$ where the underlying space X is a metric space (X, d) and its topology is induced by said metric. As seen in Definition 2.1.7, the metric induces a length function ℓ^d on paths in X, called the length by total variation. Further, Proposition 2.1.8 justifies restricting the path structure to paths from the unit interval to X. In accordance with this result, define $\mathcal{C}(\vec{P}(X))$ to be the set of all continuous functions $\gamma\colon [a, b] \to X$ for which a composition $\gamma \circ \varphi$ is in $\vec{P}(X)$ for some weakly increasing, continuous function $\varphi\colon [0, 1] \to [a, b]$ satisfying $\varphi(0) = a$ and $\varphi(1) = b$. Note that paths in $\vec{P}(X)$ and singleton paths are in $\mathcal{C}(\vec{P}(X))$, which is also closed under restriction, concatenation and linear reparametrization.

Corollary 2.3.7. *The pair $(\mathcal{C}(\vec{P}(X)), \ell^d)$ is a path structure on X.*

2.4 Gene Expression Data

Historically, studying function and disease in healthcare has relied exclusively on behavior and phenotype observation. Developments of new sequencing methods, however, have enabled researchers to enrich their studies with genotype information as well. The first methods, falling in the family of **bulk sequencing methods**, would obtain the genetic information from a tissue and average it out. With the advancement of sequencing procedures, however, it became possible to obtain gene expression data at the resolution of individual cells, providing detailed information on cellular heterogeneity. A family of such sequencing methods is called **single-cell RNA sequencing (scRNA-seq)**.

In this section we give the molecular biology prerequisites needed for understanding single-cell sequencing methods and the resulting data sets. We list the steps of the sequencing process and the technical artifacts each step might introduce to the data. We also briefly discuss the natural variance present in the data. Further, we detail the standard pipeline used to perform data analysis (and clustering more in particular) on single-cell RNA sequencing (scRNA-seq) data, and some drawback of the methods used. The general sources for the sequencing technologies are [45, 46], and [47, 48, 49, 50, 51] for the computational aspects.

2.4.1 Cellular Processes

Every organism is built from cells, which are also considered to be basic functional units. They are active all the time, and each cell has the same set of instructions (barring the very interesting phenomenon of mosaicism), called the deoxyribonucleic acid (DNA).

The DNA is a double-stranded molecule, where each of the two famously twisted strands is a sequence of smaller units, called nucleotide bases. There are four types of nucleotide bases in DNA, called cytosine, guanine, adenine and thymine. They can follow each other in whatever sequence, as long as the two strands satisfy the base pairing rule: adenine on one strand is always accompanied by thymine on the other (and vice versa), and cytosine is always paired with guanine. The DNA is a very long molecule, with the number of nucleotide pairs varying across different species (for example, the approximate number for human DNA is 3 billion) [52, page 12]. It can be split in subsequences called **genes**, each corresponding to one set of instructions.

However, DNA is not the only molecule carrying genetic information in a cell. In fact, there are many others including several types of **ribonucleic acid (RNA)**. In contrast to DNA, RNA is a single-stranded molecule. While its strand is composed of nucleotide bases in a similar way to DNA, a new base, called uracil, takes the place of thymine. Here, we focus on **messenger RNA (mRNA)** whose purpose is to carry the instructions for synthesizing protein from the cell nucleus to the ribosomes. It is a product of a process called **transcription** (see Figure 2.4), in which the two strands of DNA temporarily split at the location of the gene that is being copied (each gene that has been copied is called **expressed**). mRNA is then assembled as the base-pair-negative of one of the strands where uracil is again replaced by thymine. Once assembled, it travels to the ribosome where protein is synthesized in a process called **translation**.

The importance of proteins for cellular structure and function could not be overstated. They have many roles and are involved in nearly all cellular processes, including replicating and transcribing DNA, metabolism, controlling in-flow and out-flow of materials or information, ... This is why knowledge about which proteins are being produced by a cell at any given time provides incredible insight. Since their synthesis depends on mRNA molecules, this is mirrored in the amount of mRNA, and more particularly, the amount of times a specific gene has been expressed. This is exactly the type of information collected during single-cell RNA sequencing, with the resulting data being appropriately called **gene expression data**.

2.4.2 Single-Cell RNA Sequencing

In the section we give a rough outline of steps during the single-cell RNA sequencing to obtain a gene expression data set.

It all begins with a sample, which is either a tissue, an organism or a collection thereof. Individual cells must then be separated from the rest of the sample and isolated, which is usually done by

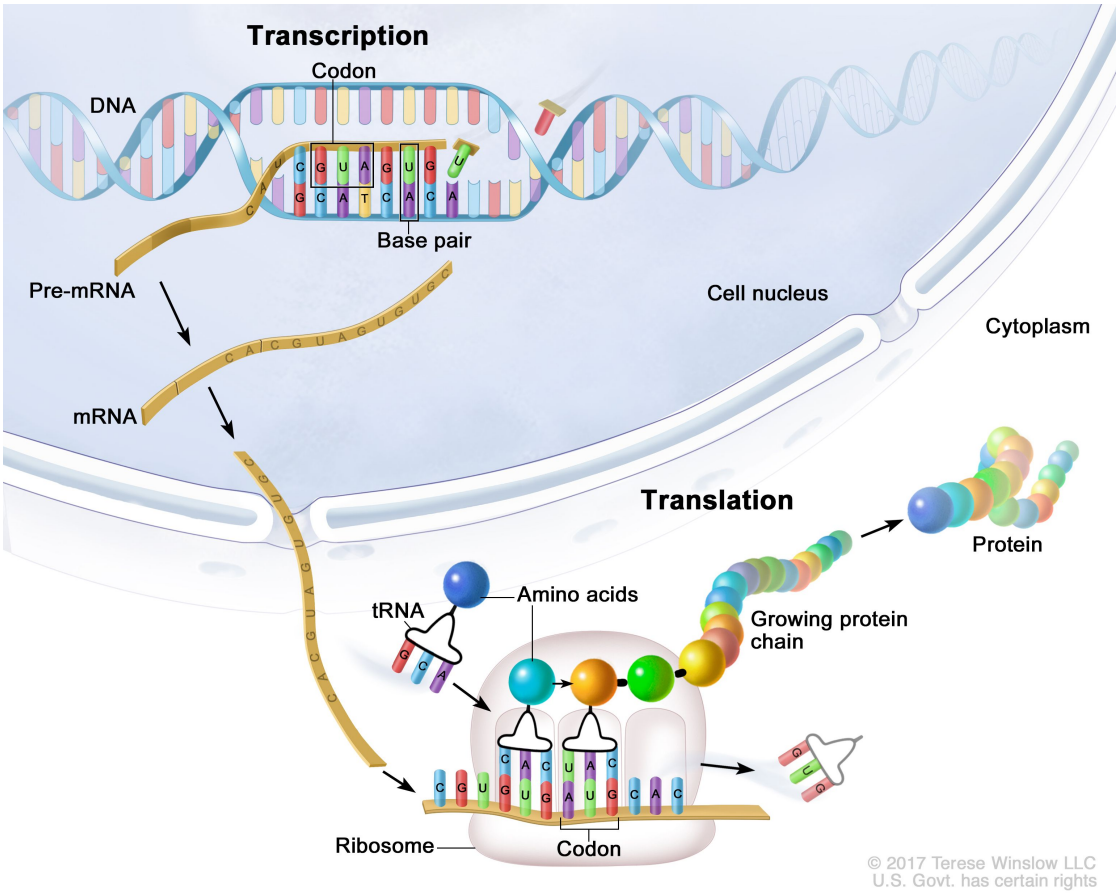


Figure 2.4: An illustration of the processes of transcription and translation. Source: [53]

immersing each cell into its own **droplet** of oil-based emulsion. This is a very intricate step, and it is often not entirely successful. As a result, some droplets might have "too little" genetic material (so called **empty droplets** [54]), while some others might contain genetic material belonging to other cells in addition to theirs. Such a droplet is referred to as a **doublet** [55]. Regardless, mRNA is captured from each droplet. The next step is called **reverse transcription**, in which mRNA is used as a template for constructing a more stable double-stranded copy DNA (cDNA) molecule. To be precise, a second strand is attached to each mRNA molecule according to the base-pair rules. This conversion also enables the subsequent use of DNA-based techniques. An example of such a technique is polymerase chain reaction, better known as PCR [56]. It is perhaps the most popular among many available methods [57] used in the **amplification** step.

Here, each cDNA molecule is copied many times so that its amount is sufficient to be detected in the subsequent sequencing. Next, a **sequencing library** is prepared. This usually means that each molecule gets elongated on both ends with different tagging sequences. These ensure that each molecule binds to the sequencing instrument correctly, that sequencing primer reacts with it, and that we are able to track the source cell of each molecule. During **sequencing**, the sequence of nucleotide bases in each mRNA molecule is read and transcribed, making a large data set. Gene expression data is then summarized during **quantification**, when a count of how many times a specific gene is expressed in each cell is computed. These counts are collected in a matrix, with an integer entry for each cell (a row in the matrix) and each gene (a column in the matrix).

2.4.3 Properties of Gene Expression Data

Here, we list properties or commonly made assumptions about the distribution of the data and geometry in the ambient space.

Distributional Assumptions

The most common distributions used to model its predecessor, **bulk RNA-seq data**, are sometimes used to model scRNA-seq data as well, namely Poisson [58] and nagative binomial distribution [59, 60, 61]. However, a strong presence of zeros and consequent bimodality in the scRNA-seq data often inspires the choice of mixture models instead. In particular, mixture models of Poisson or nagative binomial distribution with point mass at zero, called **zero-inflated Poisson** (ZIP) and **zero-inflated negative binomial** (ZINB) respectively, are popular choices. Interestingly, the debate among the scientific community about whether zero-inflated distributions should be used is not settled yet, with some arguing that the high presence of zeros is primarily a biological signal and not a technical artifact [62, 63] (see the description of dropout effect in 2.4.4).

Expression Range

If L is the largest entry in the count matrix M, then any expression count lies in range $[0, L]$. Based on the entry, we wish to answer the following: is a gene expressed in a specific cell, and if so, how strongly? In general, only the count of 0 signifies that the gene is not expressed, however a certain amount of false non-expression (which we discuss further in Section 2.4.4) and false expression is expected to be present in the data. Thus, entries that equal 0 or 1 are often

not trusted. Higher entries in general reliably signify expression, but the exact scale might be affected by a technical artifact or biological variance within the data (see Section 2.4.4).

There are many zeros in scRNA-seq data, and most counts are significantly smaller than L (approximately 95% of counts in Thirsty Fly data set from Section 6.1 are smaller than $0.01 \cdot L$, which is approximately 20). This, coupled with the fact that there are many dimensions (in which an unexpectedly large count can be observed), results in many outliers being present in scRNA-seq data sets.

Measures of Similarity and Effect of Scale

It can be difficult to find a meaningful measure of similarity between gene expression vectors. This can be seen even when comparing the expression levels of a single gene. There the count of 2 is much closer in most commonly used notions of distance to 0 than it is to 8, although we are comparing expression with non-expression in the first case versus two different levels of expression in the second (whether this difference in scale is significant can be gene dependent, which further complicates the choice of similarity measure). To mitigate the effect of scale, some sort of log-normalization is usually applied (for example, the default in Seurat package (R) [64] is $M(c,i) \mapsto \ln(1 + 10^4 \times \frac{M(c,i)}{\sum_i M(c,i)})$). An alternative to log-normalization is to use a form of square-root transform (for example Freeman-Tukey transform [65]). However, these methods do not remove the effect of scale completely (as they should not).

When comparing expression vectors, the problem of finding a meaningful measure of similarity gains an additional dimension that adds to its complexity. Namely, some notions of distance are heavily affected by scale while we might care about the pattern of expression more. When this is the case, scale invariant measures of similarity are preferable. An example of such a measure is **cosine similarity**, defined as

$$S_C(A,B) = \frac{A \cdot B}{\|A\|\|B\|} = \cos(\angle(A,B)), \tag{2.4}$$

for any pair of vectors A, B in the same ambient space $\mathbb{R}^n$, where $\|\star\|$ is the euclidean distance. To illustrate this, take expression vectors $A, B, C \in \mathbb{R}^5$ to be

$$\begin{aligned} A &= (0,2,0,3,25), \\ B &= (2,0,4,0,27), \\ C &= (0,1,0,1,3). \end{aligned}$$

The standard choice, the euclidean distance, identifies A and B as much closer (at ~ 6.08) than A and C (at ~ 22.11). Cosine similarity, however, identifies higher similarity in expression patterns between A and C (at ~ 0.045) than between A and B (at ~ 0.023).

Number of Relevant Genes and Curse of Dimensionality

Due to often not knowing a priori which genes are involved in processes one wishes to study, all the counts collected during single-cell RNA sequencing are included in the analysis. Thus, the dimension of ambient space is often in thousands, even though it is often assumed only a handful of genes are biologically relevant. As a consequence, the data suffers from the curse of dimensionality and an appropriate pipeline should be followed. To remove as many irrelevant genes, feature selection may be applied, but often further dimensionality reduction step is necessary (a common pipeline with both these steps is described in Section 2.4.5).

2.4.4 Biological and Technical Artifacts in Gene Expression Data

Due to the intricacy of the sequencing procedure, there is a lot of room for introducing technical artifacts to the data. Further, natural variance is present in scRNA-seq data sets. We discuss specific examples of such phenomena here.

Batch Effect

Sequencing experiments are often long lasting and laborious, which is why they are commonly split into batches. Although one tries to keep the environment in which each batch is analyzed unchanged, some slight differences are usually present and they can affect the result. This is known as the batch effect [66].

The goal of batch effect correction methods is to minimize these differences and simultaneously preserve as much biological information as possible. Most methods begin by identifying biological signals (usually cell populations) that are shared across different batches, and comparing these signals to estimate batch effect. This insight can then be used to align batches in a correction step. Often these steps are applied many times, obtaining a more refined alignment with each iteration. Some commonly used batch effect correction methods are mutual nearest neighbors (MNN) [67], Seurat alignment [68, 69] and ComBat [70].

mRNA Levels

The amount of mRNA and the specific genes expressed are ever changing within a cell. This depends among other things on cell function, stimuli from the environment, current part of cell's life cycle, and cell health. This introduces two phenomena that might need addressing, depending on the specific goal of the analysis. Firstly, unhealthy cells (often meaning cancerous or dying) have been reported to have inflated ribosomal and mitochondrial gene counts [71, 72]. Such cells often need to be removed from analysis, as their function is severely affected. Secondly, the number of mRNA molecules obtained from each cell varies greatly. This might be due to the presence of empty droplets or doublets, which extremely high or low total gene expression counts are usually attributed to, and the corresponding entries in the data set can be removed accordingly. However, some variance in total counts is to be expected, either due to natural variance (for example, sensitivity varies across different tissues and mRNA degrades non-uniformly [73, 74]) or technical effects (such as batch effect). Unfortunately, this might overshadow the biological heterogeneity we wish to study. To address it, SCTransform [60] or a similar stabilizing method may be applied.

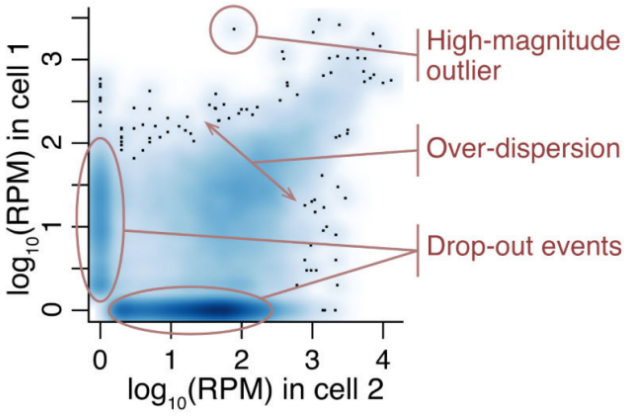
Dropout

This is a technical artifact introduced at the stage of mRNA capture, reverse transcription or amplification. In any of these steps, the signal of a single molecule or a group of molecules may be lost, resulting in false negative expression of some genes, referred to as dropout [51, 75, 76]. The presence of dropout-events can be detected by analyzing cell-to-cell relationships (see Figure 2.5), or, similarly, by gene-to-gene relationships.

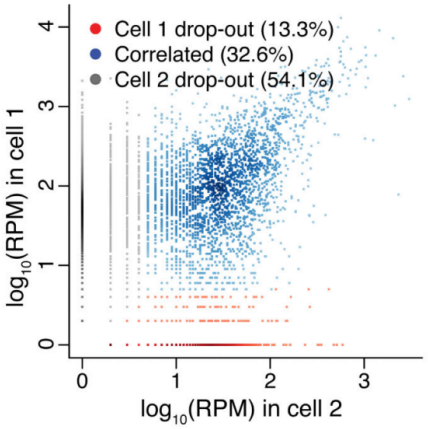
There is a plethora of methods for addressing the dropout effect [77, 78, 79, 80, 81, 82]. For a thorough overview of them, consult [83]. Note, however, that high presence of zero counts is partially due to true non-expression. These methods therefore need to distinguish non-biological from biological zeros based on observed counts, which they (mostly) attempt by probabilistic arguments. On the other hand, this separation of biological and non-biological zeros is sometimes described as an unfeasible task, and addressing dropout is warned against [62, 63].

2.4.5 Standard pipeline for Clustering Analysis of Gene Expression Data

Here, we give a rough outline of commonly used pipeline for clustering analysis on scRNA-seq data sets, following [47, 48, 49].



(a) A smoothed scatter plot illustration of cell-to-cell variability within gene expression data. The artifacts that can be recognized from these plots are high-magnitude outliers, over-dispersion in the anti-diagonal direction, and drop-out events.



(b) Actual cell vs. cell plot as in Figure 2.5a for scRNA-seq data set from the study of mouse embryonic fibroblast stem cell [84].

Figure 2.5: Dropout effect in cell vs. cell plots (with a point per each gene) from scRNA-seq data sets. The assumption is that the chosen cells are of the same type and therefore gene expressions should be correlated. However, it may happen that the areas close to the axes are highly populated, meaning many genes are highly expressed in one cell while not expressed in the other, hinting at the presence of dropout events. Note that RPM stands for "reads per million" and it signifies that additional normalization with respect to mRNA levels across cells was applied before $\log_{10}$-normalization. Source of figures: [76].

Pre-processing

During pre-processing, many steps are taken to address and hopefully mitigate the variance and other artifacts present in the data, and prepare it for further analysis. For example, the cells and genes are filtered to remove genes that are not expressed often enough on our dataset and remove unhealthy cells or droplets. Secondly, normalization or other scaling techniques can be applied to dampen the effect of scale and different mRNA levels across cells. Some such techniques are RPM (reads per million), SCTransform and log-transformation. If the experiment from which the data was obtained was split into batches, the data can be aligned using batch effect correction methods [70, 69, 67]. Presence of dropout can be addressed in pre-processing too [77, 78, 79, 80, 81, 82].

Feature Selection

As mentioned, the set of genes whose expression is analyzed is far larger than the set of relevant genes. This abundance of information can be costly due to the curse of dimensionality. Further, the smaller the percentage of relevant genes, the less they will contribute to the analysis. However, which genes are relevant is rarely known a priori. In the step of feature selection, some assumptions are made about the statistical summaries of gene expression profiles of relevant genes, and genes satisfying those assumptions are selected for further analysis.

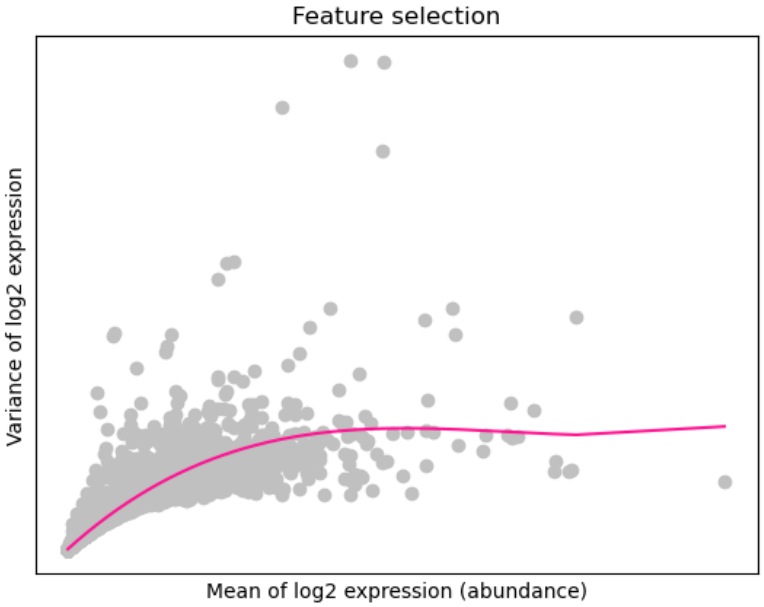


Figure 2.6: Variance vs. mean plot for $\log_2$-normalized Thirsty Fly data set (introduced in Section 6.1), where each point in the plot corresponds to a gene in the data set. The trend is a polynomial in degree 3 fit to the data using mean-squared error.

For example, it is often assumed that the majority of the variability in the expression of a specific gene is due to technical noise, and that it is larger for genes with high means [47]. (Note that log- or some other type of normalization is often applied in advance, and different measures of variability can be chosen, such as variance and squared coefficient of variation.) When that is the case, a trend is fit to the plot of variability vs. mean with a point for each gene, see Figure 2.6. This trend signifies the expected variability of gene's expression given its mean (also called the **technical component**), and the difference to actual variability is called the

biological component. The latter is commonly considered to measure relevance, and the data is projected to genes with high biological components.

Dimensionality Reduction

Although this step is optional, there are many reasons for applying dimensionality reduction at this point. Firstly, the latent dimension of the point cloud obtained after feature selection is often assumed to still be much smaller then the ambient one. The dimension might also still be too high to successfully apply certain clustering methods. In addition, most dimensionality reduction methods combine correlated signals, thus eliminating redundancy in data. Most commonly, principal component analysis (PCA) is used to project down to $d \in [10, 50]$ dimensions, where the exact dimension is sometimes determined using a Jackstraw function [85]. Other popular methods include UMAP [86] and t-SNE [87], with UMAP being especially prevalent in data visualization.

Clustering

Next, the point cloud is clustered, the goal of which is usually to group cells based on their cell type. Most popular clustering methods for the analysis of scRNA-seq data are based on community detection on k-nearest neighbors (kNN) graphs, which try to obtain a partition of the graph into subgraphs, or communities, on which some measure of edge density or connectedness is maximized. This can be done following an agglomerative process of merging communities if the gain in the chosen measure is positive. An example of such method is the Louvain algorithm [88], in which the chosen measure is modularity – the difference between the actual number of edges within a community and the expected one. Another approach, followed by the Walktrap algorithm [89], is to simulate random walks on the graph and use the results to merge communities, following the reasoning that random walks tend to linger in highly connected regions.

Louvain and Walktrap algorithms are perhaps most commonly used, but of course, other methods of community detection and, more generally, clustering can be applied. However, one should be aware of potential distributional assumptions made by individual methods [90], since not all can be expected to perform well on zero-inflated, highly variable and high-dimensional data with many outliers, such as scRNA-seq data.

Marker Detection or Differential Expression Analysis

To assign biological meaning to the obtained clusters one can study statistical properties (such as mean, median, maximum, minimum, ...) of expression patterns on the level of clusters, and compare the results between clusters or between each cluster and its complement. This is called **differential expression analysis** and its goal is to identify genes that drive cluster separation. These genes are called **markers**. Commonly used heuristics for differential expression analysis differ greatly, which is why we do not detail them here. Instead, we direct the reader to [91] for a comparison of some popular methods, and to Section 6.2.4 for the description of the method used in this work.

Chapter 3

Ladder Decomposition for Morphisms of Persistence Modules

In the spirit of the structure theorem giving a barcode for one-parameter p.f.d. persistence modules, Jacquard et al. [11] identify assumptions under which a morphism viewed as a ladder persistence module decomposes into a direct sum of elementary ladder persistence modules:

- $\mathbf{I}_J^+$ consists of an interval persistence module k_J on the source side that is mapped to 0 on the target side.
- $\mathbf{I}_K^-$ consists of 0 on the source side and an interval persistence module k_K on the target side.
- $\mathbf{R}_K^J$ consist of an interval persistence module k_J on the source side and an interval persistence module k_K on the target side with a morphism that in each degree $i \in J$ sends the basis vector of $(k_J)_i$ to the basis vector of $(k_K)_i$ if $i \in K$, and to 0 otherwise.

A ladder decomposition is then an isomorphism of $\Phi\colon V \to W$ to a direct sum of elementary ladder persistence modules, and the main result of [11] is that one exists whenever the barcodes $\mathrm{Bar}(V)$ and $\mathrm{Bar}(W)$ do not admit strictly nested bars.

In this chapter we focus on a special family of morphisms of persistence modules, namely the interleaving morphisms. They come in pairs (Φ, Ψ), with $\Phi\colon V \to W(\delta)$ and $\Psi\colon W \to V(\delta)$,

where the persistence module $V(\delta)$ is obtained from V via shifts in the indexing parameter, i.e. $V(\delta)_t = V_{t+\delta}$. The pair must further satisfy that their composition (both instances) is exactly the family of inner morphisms of the corresponding persistence module. Because of this property, their existence implies the persistence modules in the domain and codomain are algebraically related, and the smaller the shifting parameter, the stronger this relation is. The smallest δ for which a δ-interleaving pair between V and W exists is denoted by d_I and called their interleaving distance [92] (it can be defined in more general settings, for example in any category with flow [93]). As a consequence of various stability results [94, 10] interleaving morphisms arise between persistence modules obtained from closely related inputs to the persistent homology pipeline. They also arise between restrictions of multi-parameter persistence modules to close-enough parallel lines in the parameter space [95]. As such, interleaving morphisms are interesting from the point of view of both applications and theoretical results.

In most of this chapter we are only interested in one of the morphisms in the interleaving pair. For this reason we introduce the notion of a δ-invertible morphism - a morphism $\Phi\colon V \to W$ for which there exists another morphism $\Psi : W \to V(2\delta)$ so that both of their compositions are just the inner morphisms in the corresponding persistence module. It is easy to see a δ-invertible morphism $\Phi\colon V \to W$ is equivalent to a morphism in a δ-interleaving pair between V and $W(-\delta)$. Special properties of δ-invertible morphisms make them somewhat easier to work with than the general morphisms, which we leverage to obtain their ladder decompositions under looser assumptions. To elaborate, we define a constant $\Xi(V)$ called the nestedness of persistence module V, as the minimal distance between endpoints (either of birth-points or of death-points) of strictly nested bars. The main result in this chapter is the following.

Theorem 3.2.8. *For a δ-invertible morphism $\Phi\colon V \to W$ with $\delta < \frac{1}{2}\min(\Xi(V), \Xi(W))$ there exist parameters $r_{J_1}^{J_2}, d_J^+, d_K^- \in \mathbb{N}$ such that*

$$(V, W, \Phi) \cong \bigoplus_{J_1 \preceq J_2} \left(\mathbf{R}_{J_1}^{J_2}\right)^{r_{J_1}^{J_2}} \oplus \bigoplus_{J} \left(\mathbf{I}_J^+\right)^{d_J^+} \oplus \bigoplus_{K} \left(\mathbf{I}_K^-\right)^{d_K^-}.$$

In persistent homology pipelines, long bars are the signals and short bars are often associated with noise in the input. Therefore it is potentially useful to truncate and discard the bars shorter than some threshold. Here we explore how this process interacts with the ladder decomposition theorem above. For this reason, we introduce another parameter q to be the length of the longest bar we wish to disregard. We consider restrictions $V_{\geq q}$ and $W_{\geq q}$ of modules V and W to the

features which persist for at least q, which are given by projection maps $pr^V_{\geq q}$ and $pr^W_{\geq q}$ and inclusion maps $i^V_{\geq q}$ and $i^W_{\geq q}$ respectively.

Theorem 3.3.4. *For any $q \geq 0$ a δ-invertible morphism $\Phi\colon V \to W$ induces the following $(\delta+\frac{q}{2})$-invertible morphisms:*

1. *Morphism $pr^W_{\geq q} \circ \Phi\colon V \to W_{\geq q}$. If $\delta < \frac{1}{2}\min(\Xi(V), \Xi(W_{\geq q})) - \frac{q}{2}$ holds, it decomposes as a ladder persistence module.*
2. *Morphism $\Phi \circ i^V_{\geq q}\colon V_{\geq q} \to W$. If $\delta < \frac{1}{2}\min(\Xi(V_{\geq q}), \Xi(W)) - \frac{q}{2}$ holds, it decomposes as a ladder persistence module.*
3. *Morphism $pr^W_{\geq q} \circ \Phi \circ i^V_{\geq q}\colon V_{\geq q} \to W_{\geq q}$. If $\delta < \frac{1}{2}\min(\Xi(V_{\geq q}), \Xi(W_{\geq q})) - \frac{q}{2}$ holds, it decomposes as a ladder persistence module.*

Via the correspondence between δ-invertible morphisms and morphisms appearing in δ-interleaving pairs we obtain ladder decompositions of both morphisms in an interleaving pair. Comparing them we find they are as compatible as the shifting parameter allows.

Theorem 3.2.21. *Let (Φ, Ψ) be a δ-interleaving pair between modules V and W with $\delta < \frac{1}{2}\min(\Xi(V), \Xi(W))$. For any pair of bars $J_V \in Bar(V)$ and $J_W \in Bar(W)$ satisfying $|J_V|, |J_W| \geq 2\delta$, and for any $\mu \in \mathbb{N}$ the following statements are equivalent*

- $(\mathbf{R}^{J_V}_{J_W(\delta)})^\mu$ *appears in the ladder decomposition of Φ,*
- $(\mathbf{R}^{J_W}_{J_V(\delta)})^\mu$ *appears in the ladder decomposition of Ψ.*

As observed already in [11], whenever a ladder decomposition can be obtained, it induces a partial matching on the barcodes of the persistence modules involved. A partial matching can be defined on a general multi-set as a partial bijection. The first instance of a morphism-induced partial matching appeared in [10] in order to prove the Isometry Theorem, stating that the interleaving distance on the one-parameter persistence modules agrees with the bottleneck distance [94] on persistence barcodes. This construction, however, is not linear with respect to direct sums of ladder persistence modules. Addressing this (and some other grievances) the notion of basis-independent partial matchings has been introduced [13], which the ladder decomposition induced partial matchings are examples of. We analyse their properties when the morphism is a part of an interleaving and show that their cost is limited above by the interleaving parameter.

Corollary 3.4.2 (of Theorem 3.2.8)**.** *Let $\Phi\colon V \to W(\delta)$ be one of two morphisms making a δ-interleaving pair for $\delta < \frac{1}{2}\min(\Xi(V), \Xi(W))$, and χ_Φ the partial matching induced by the ladder decomposition of Φ. Its cost is at most δ. If further $\delta = d_I(V, W)$, then the induced matching realizes the bottleneck distance.*

Further, the matchings induced by ladder decompositions of the pair of morphisms making an interleaving are compatible for all bars of sufficient length.

Corollary 3.4.3 (of Theorem 3.2.21)**.** *Let $\chi_\Phi\colon Bar(V) \nrightarrow Bar(W)$ and $\chi_\Psi\colon Bar(W) \nrightarrow Bar(V)$ be the partial matchings induced by morphisms (Φ, Ψ) forming a δ-interleaving pair where $\delta < \frac{1}{2}\min(\Xi(V), \Xi(W))$. For any pair of bars $J_V \in Bar(V)$ and $J_W \in Bar(W)$ satisfying $|J_V| \geq 2\delta$ and $|J_W| \geq 2\delta$, and any $\mu \in \mathbb{N}$,*

$$((J_V, J_W), \mu) \in \chi_\Phi \iff ((J_W, J_V), \mu) \in \chi_\Psi.$$

Preliminary material originally present in [1] can be found in Sections 2.2.3 to 2.2.5. We begin this chapter in Section 3.1, where we specify how to view morphisms as ladder persistence modules and state the ladder decomposition theorem of [11]. Section 3.2 is dedicated to interleavings and δ-invertible morphisms. We quote their definition and state some basic results in the language of barcode bases in Section 3.2.1. We define the nestedness constant and state the main theorem about the ladder decomposition of δ-invertible morphisms in Section 3.2.2. It also contains a plethora of technical lemmata used to prove the main theorem. Section 3.2.3 compares the ladder decompositions of both morphisms making an interleaving pair. The generalisation of the theory introduced in Section 3.2 to the case when we discard short bars is presented in Section 3.3. It includes the definition of a q-splitting of a persistence module and the generalisation of the main theorem – Theorem 3.3.6. Lastly, we state results regarding the ladder decomposition induced partial matchings in Section 3.4, where they are also compared with other notions of induced partial matchings.

3.1 Ladder Decomposition of a Persistence Morphism

Let $\Phi\colon V \to W$ be a morphism of persistence modules between V and W, which are both indexed over $[\ell + 1] = \{0, 1, \ldots, \ell\}$. Fix barcode bases $\mathcal{B}_V$ and $\mathcal{B}_W$ and let M_Φ be the single-matrix representation of Φ in these bases, which exists according to Proposition 2.2.27. Further,

let $M_\Phi(x_K, x_J)$ be the entry of the matrix M_Φ in the row belonging to x_K and column belonging to x_J, where x_K and x_J are generators of bars K and J respectively. (Recall that there are μ distinct generators of a bar with multiplicity μ.) The value of $M_\Phi(x_K, x_J)$ is to be understood as follows: the image of a basis vector in the generator x_J with Φ can be expressed for each component $\Phi_i \colon V_i \to W_i$ as

$$\Phi_i((x_J)_i) = \sum_{\substack{x_K \\ K \ni i}} M_\Phi(x_K, x_J) \cdot (x_K)_i.$$

In light of Remark 2.2.28, the bar generators x_K with non-zero coefficient $M_\Phi(x_K, x_J)$ correspond to bars with $K \preceq J$.

Definition 3.1.1. The **support** of the image of the bar generator x_J with morphism Φ is the set

$$\operatorname{supp} \Phi(x_J) = \{x_K \mid M_\Phi(x_K, x_J) \neq 0\}.$$

Lemma 3.1.2. *If $x_K \in \operatorname{supp} \Phi(x_J)$ then $K \preceq J$.*

Proof. This is a simple reiteration of the observation in Remark 2.2.28 using the new notation. □

Introduce the following simple and intuitive components as building blocks in the decomposition of ladder modules: $\mathbf{R}^{[i_2,j_2]}_{[i_1,j_1]}, \mathbf{I}^+_{[i_1,j_1]}$ and $\mathbf{I}^-_{[i_1,j_1]}$ for $[i_1, j_1] \preceq [i_2, j_2]$.

$\mathbf{R}^{[i_2,j_2]}_{[i_1,j_1]}$:

$$\begin{array}{ccccccccccccc}
\cdots & \longrightarrow & 0 & \xrightarrow{0} & \mathbb{F}_{i_2} & \xrightarrow{\mathrm{id}} & \cdots & \xrightarrow{\mathrm{id}} & \mathbb{F}_{j_2} & \xrightarrow{0} & 0 & \longrightarrow & \cdots \\
 & & \downarrow{\scriptstyle 0} & & \downarrow{\scriptstyle \mathrm{id}} & & & & \downarrow{\scriptstyle \mathrm{id}} & & \downarrow{\scriptstyle 0} & & \\
\cdots & \longrightarrow & 0 & \xrightarrow[0]{} & \mathbb{F}_{i_1} & \xrightarrow[\mathrm{id}]{} & \cdots & \xrightarrow[\mathrm{id}]{} & \mathbb{F}_{j_1} & \xrightarrow[0]{} & 0 & \longrightarrow & \cdots
\end{array}$$

$\mathbf{I}^+_{[i_1,j_1]}$:

$$\begin{array}{ccccccccccc}
\cdots & \longrightarrow & 0 & \xrightarrow{0} & \mathbb{F}_{i_1} & \xrightarrow{\mathrm{id}} & \cdots & \xrightarrow{\mathrm{id}} & \mathbb{F}_{j_1} & \xrightarrow{0} & 0 & \longrightarrow & \cdots \\
 & & \downarrow{\scriptstyle 0} & & \downarrow & & & & \downarrow{\scriptstyle 0} & & & & \\
 & & \cdots & \longrightarrow & 0 & \longrightarrow & \cdots & \longrightarrow & 0 & \longrightarrow & \cdots & &
\end{array}$$

$\mathbf{I}^-_{[i_1,j_1]}$:

$$\begin{array}{ccccccccccc}
 & & \cdots & \longrightarrow & 0 & \longrightarrow & \cdots & \longrightarrow & 0 & \longrightarrow & \cdots & & \\
 & & \downarrow{\scriptstyle 0} & & \downarrow & & & & \downarrow{\scriptstyle 0} & & & & \\
\cdots & \longrightarrow & 0 & \xrightarrow[0]{} & \mathbb{F}_{i_1} & \xrightarrow[\mathrm{id}]{} & \cdots & \xrightarrow[\mathrm{id}]{} & \mathbb{F}_{j_1} & \xrightarrow[0]{} & 0 & \longrightarrow & \cdots
\end{array}$$

Theorem 3.1.3 (Theorem 4.3. of [11]). *Let (V, W, Φ) be a ladder persistence module where neither V nor W admit a pair of strictly nested bars. Then there are integers $r_{J_1}^{J_2}, d_J^+, d_K^- \in \mathbb{N}$ for which*

$$(V, W, \Phi) \cong \bigoplus_{J_1 \preceq J_2} \left(\mathbf{R}_{J_1}^{J_2}\right)^{r_{J_1}^{J_2}} \oplus \bigoplus_{J} \left(\mathbf{I}_J^+\right)^{d_J^+} \oplus \bigoplus_{K} \left(\mathbf{I}_K^-\right)^{d_K^-}. \tag{3.1}$$

The right side of Equation (3.1) is called the **ladder decomposition** of morphism Φ. The isomorphism is given by a change of barcode bases of the domain and codomain. The matrix representation of Φ in the bases in which it decomposes as a ladder persistence module is in partial matching form.

Definition 3.1.4. A matrix is in **partial matching form** if there is at most one 1 in each row and each column, with all the other entries being 0.

We will often refer to the partial matching form as the matching form for short. Since the ladder decomposition is unique up to an automorphism, the matching form is unique up to (compositions of) permutations of the order of columns (or rows) belonging to a collection of indistinguishable bars.

Jacquard et al. show with examples that if either $\mathrm{Bar}(V)$ or $\mathrm{Bar}(W)$ contain a pair of strictly nested bars then such a decomposition need not exist. The focus of our work is to refine this and show that, under additional hypotheses on Φ, a decomposition will exist even when there are nested bars.

3.2 Ladder Decompositions and Interleavings

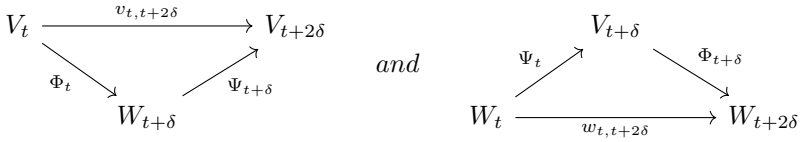
In this section the theory of ladder decompositions of morphisms in the case of interleavings is developed further. Interleavings come in pairs of morphisms, which we (with slight abuse of notation) call δ-invertible morphisms and are interesting in their own right. Theorem 3.2.8 relaxes the assumptions of the Ladder Decomposition Theorem of [11] for δ-invertible morphisms, while Corollary 3.2.20 summarises the relation between ladder decompositions of the two morphisms making an interleaving pair.

3.2.1 Interleavings and δ-Invertible Morphisms of Persistence Modules

Recall from Section 2.2.4 that a δ-**shift** of a persistence module V is a persistence module $V(\delta)$ with

$$V(\delta)_t = V_{t+\delta}$$
$$v(\delta)_{t_1,t_2} = v_{t_1+\delta,t_2+\delta},$$

and that two persistence modules V and W are δ-**interleaved** if there exists a pair of morphisms $\Phi\colon V \to W(\delta)$ and $\Psi\colon W \to V(\delta)$ such that the diagrams



commute. The pair (Φ, Ψ) is a δ-**interleaving**

Example 3.2.1. Let us introduce an interleaving which we will use as a running example. Let V and W be persistence modules

$$V\colon \quad 0 \longrightarrow \mathbb{R} \xrightarrow{\binom{1}{0}} \mathbb{R}^2 \xrightarrow{\mathrm{Id}} \mathbb{R}^2 \xrightarrow{\mathrm{Id}} \mathbb{R}^2 \xrightarrow{\left(\begin{smallmatrix}1&0\\0&1\\0&0\end{smallmatrix}\right)} \mathbb{R}^3 \xrightarrow{(0\,1\,0)} \mathbb{R} \xrightarrow{\mathrm{Id}} \mathbb{R} \xrightarrow{\mathrm{Id}} \mathbb{R} \longrightarrow 0,$$

$$W\colon \quad 0 \longrightarrow 0 \longrightarrow \mathbb{R}^2 \xrightarrow{\mathrm{Id}} \mathbb{R}^2 \xrightarrow{\mathrm{Id}} \mathbb{R}^2 \xrightarrow{\mathrm{Id}} \mathbb{R}^2 \xrightarrow{\mathrm{Id}} \mathbb{R}^2 \xrightarrow{(0\,1)} \mathbb{R}^1 \longrightarrow 0 \longrightarrow 0,$$

with barcode bases given by the unit vectors of the vector spaces $\mathbb{R}^n$. Their barcodes are

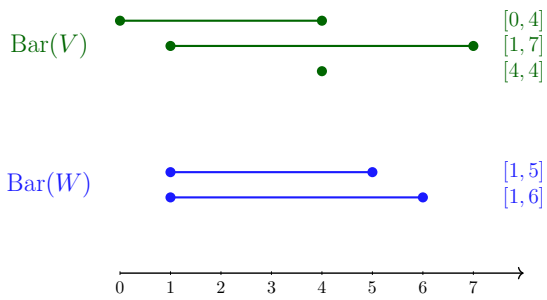


Figure 3.1: The barcodes of modules V and W in Example 3.2.1.

$$\mathrm{Bar}(V) = \{[0,4],[1,7],[4,4]\} \qquad \text{and} \qquad \mathrm{Bar}(W) = \{[1,5],[1,6]\},$$

where the bars have been written in the lexicographical order. Bar generators associated with our choice of barcode bases are

$$\begin{aligned} x^V_{[0,4]} &= \{\vec{e_1}^{(0)}, \vec{e_1}^{(1)}, \vec{e_1}^{(2)}, \vec{e_1}^{(3)}, \vec{e_1}^{(4)}\}, \\ x^V_{[1,7]} &= \{\vec{e_2}^{(1)}, \vec{e_2}^{(2)}, \vec{e_2}^{(3)}, \vec{e_2}^{(4)}, \vec{e_1}^{(5)}, \vec{e_1}^{(6)}, \vec{e_1}^{(7)}\}, \\ x^V_{[4,4]} &= \{\vec{e_3}^{(4)}\}, \end{aligned}$$

$$\begin{aligned} x^W_{[1,5]} &= \{\vec{f_1}^{(1)}, \vec{f_1}^{(2)}, \vec{f_1}^{(3)}, \vec{f_1}^{(4)}, \vec{f_1}^{(5)}\}, \\ x^W_{[1,6]} &= \{\vec{f_2}^{(1)}, \vec{f_2}^{(2)}, \vec{f_2}^{(3)}, \vec{f_2}^{(4)}, \vec{f_2}^{(5)}, \vec{f_1}^{(6)}\}, \end{aligned}$$

where $\vec{e}^{(i)}_j$ and $\vec{f}^{(i)}_j$ denote the j-th unit vector in V_i and W_i respectively. Define morphisms $\Phi\colon V \to W(1)$ and $\Psi : W \to V(1)$ with components

$$\Phi_0 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \quad \Phi_1, \Phi_2, \Phi_3 = \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}, \quad \Phi_4 = \begin{bmatrix} 2 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad \Phi_5 = \begin{bmatrix} 1 \end{bmatrix}, \quad \Phi_6, \Phi_7 = 0,$$

$$\Psi_0, \Psi_7 = 0, \quad \Psi_1, \Psi_2 = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ 0 & 1 \end{bmatrix}, \quad \Psi_3 = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \Psi_4, \Psi_5 = \begin{bmatrix} 0 & 1 \end{bmatrix}, \quad \Phi_6 = \begin{bmatrix} 1 \end{bmatrix}.$$

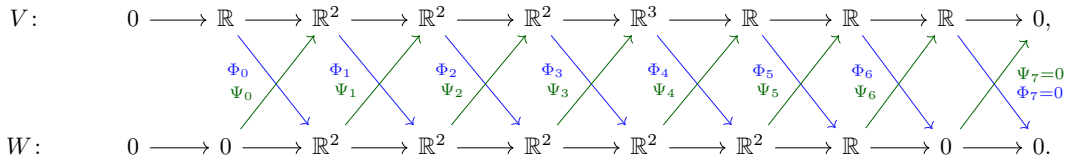
To see that these define two morphisms of persistence modules, one simply checks that

$$w_{i+1,i+2} \circ \Phi_i = \Phi_{i+1} \circ v_{i,i+1} \qquad \text{and} \qquad v_{i+1,i+2} \circ \Psi_i = \Psi_{i+1} \circ w_{i,i+1}$$

for $i = -1, 0, \ldots, 7$. Their respective single-matrix representations from Proposition 2.2.27 are

$$M_\Phi = \begin{bmatrix} 2 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \text{and} \qquad M_\Psi = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ 0 & 1 \\ 0 & 0 \end{bmatrix},$$

and they make a 1-interleaving pair



We can verify that they make a 1-interleaving pair by showing that $\Phi_{i+1} \circ \Psi_i = w_{i,i+2}$ and $\Psi_{i+1} \circ \Phi_i = v_{i,i+2}$ for all $i \in \mathbb{Z}$ and on all basis vectors of the chosen barcode basis for W and V respectively. To show the first, consider only basis vectors in bar generator $x^W_{[1,6]}$ and see that

$$\begin{aligned}
\Phi_{i+1} \circ \Psi_i\big((x^W_{[1,6]})_i\big) &= \begin{cases} \Phi_{i+1}\big((x^V_{[1,7]})_{i+1} - \frac{1}{2}(x^V_{[0,4]})_{i+1}\big), & \text{if } 1 \le i \le 3, \\ \Phi_{i+1}\big((x^V_{[1,7]})_{i+1}\big), & \text{if } 3 < i \le 6, \end{cases} \\
&= \begin{cases} (x^W_{[1,5]})_{i+2} + (x^W_{[1,6]})_{i+2} - \frac{1}{2}(2(x^W_{[1,5]})_{i+2}), & \text{if } 1 \le i \le 3, \\ (x^W_{[1,6]})_{i+2}, & \text{if } 3 < i \le 6, \end{cases} \\
&= (x^W_{[1,6]})_{i+2}.
\end{aligned}$$

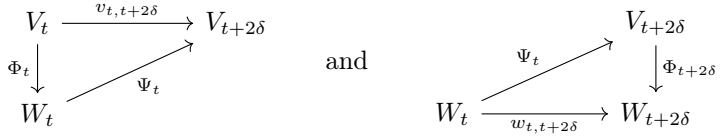
The same holds for $x^W_{[1,5]}$ following similar computation. To prove $\Psi_{i+1} \circ \Phi_i = v_{i,i+2}$, let us compute how the composition maps basis vectors in bar generator $x^V_{[4,4]}$:

$$\begin{aligned}
\Psi_5 \circ \Phi_4\big((x^V_{[4,4]})_4\big) &= \Psi_5\big((x^W_{[1,5]})_5\big) \\
&= 0 \\
&= v_{4,6}\big((x^V_{[4,4]})_4\big).
\end{aligned}$$

Again, similar computation can be done to show it holds also for basis vectors in bar generators $x^V_{[0,4]}$ and $x^V_{[1,7]}$. △

It is often useful to decouple the definition of an interleaving morphism by singling one of the morphisms out and implying the existence of the other without committing to a choice of it. This motivates the definition of a **δ-invertible morphism**.

Definition 3.2.2. A morphism $\Phi : V \to W$ is **δ-invertible** if there exists another morphism $\Psi : W \to V(2\delta)$ such that the diagrams



commute. Morphism Ψ is called a δ-**inverse** of Φ.

As before, parameter δ measures how close a morphism is to being an isomorphism, as a 0-invertible morphism is simply an isomorphism.

Remark 3.2.3 (Correspondence between δ-interleavings and δ-invertible morphisms)**.** Let us make the relationship between the notions of a δ-invertible morphism and a δ-interleaving pair explicit. A morphism $\Phi\colon V \to W$ is δ-invertible if and only if, when regarded as a morphism $\Phi'\colon V \to W'(\delta)$ with $W' = W(-\delta)$, it is half of a δ-interleaving. In fact, any δ-inverse of Φ gives (after the necessary shifts) a morphism making a δ-interleaving pair with Φ'.

From here on, we will work with δ-invertible morphisms and obtain results that hold for morphisms in a δ-interleaving pair via the correspondence in Remark 3.2.3.

Example 3.2.4. Take the 1-interleaving pair (Φ, Ψ) from Example 3.2.1. The induced 1-invertible morphism $\Phi^{(1)}$ maps from V to $W' = W(1)$. Since we replace W with its shift, the barcode is now

$$\mathrm{Bar}(W') = \{[0,4],[0,5]\}$$

and the bar generators in our choice of barcode basis are

$$x^{W'}_{[0,4]} = \{\vec{f}_1{}^{(0)}, \vec{f}_1{}^{(1)}, \vec{f}_1{}^{(2)}, \vec{f}_1{}^{(3)}, \vec{f}_1{}^{(4)}\},$$
$$x^{W'}_{[0,5]} = \{\vec{f}_2{}^{(0)}, \vec{f}_2{}^{(1)}, \vec{f}_2{}^{(2)}, \vec{f}_2{}^{(3)}, \vec{f}_2{}^{(4)}, \vec{f}_1{}^{(5)}\}.$$

Notice, however, that the matrix representations of the components Φ_i are equal to the matrix representations $\Phi_i^{(1)}$, and so are the single matrix representations

$$M_\Phi = M_{\Phi^{(1)}}.$$

The same holds for the 1-inverse $\Psi^{(1)}$ of $\Phi^{(1)}$. △

The Shift Operator

As has become clear in this section, shifting in degree will be common throughout this work. To avoid confusion, we explain some shift-related notation here.

Definition 3.2.5. Let $\delta \in \mathbb{N}_0$ be a parameter.

- For any interval $I = [i_1, i_2]$, define the δ**-shift of** I as $I(\delta) = [i_1 - \delta, i_2 - \delta]$.
- For any bar generator x_I, define the δ**-shift of bar generator** x_I as

$$x_I(\delta) = x_{I(\delta)} = \{(x_I)_{t+\delta}\}_{t \in I(\delta)}.$$

- For any persistence module V with inner morphisms $v_{i,j} \colon V_i \to V_j$, define the **inner** δ**-morphism** as $[\delta]_V \colon V \to V(\delta)$ and $([\delta]_V)_t = v_{t,t+\delta}$.
- For any morphism $\Phi \colon V \to W$ of persistence module, define the δ**-shift of** Φ as $\Phi(\delta) \colon V(\delta) \to W(\delta)$ and $\Phi(\delta)_t = \Phi_{t+\delta}$.

Using this notation, we can rephrase the requirements for morphisms $\Phi \colon V \to W(\delta)$ and $\Psi \colon W \to V(\delta)$ to be a δ-interleaving as

$$\Psi(\delta) \circ \Phi = [2\delta]_V \qquad \text{and} \qquad \Phi(\delta) \circ \Psi = [2\delta]_W.$$

Similarly, a morphism $\Phi \colon V \to W$ is δ-invertible when there exists a morphism $\Psi \colon W \to V(2\delta)$ so that

$$\Psi \circ \Phi = [2\delta]_V \qquad \text{and} \qquad \Phi(2\delta) \circ \Psi = [2\delta]_W.$$

3.2.2 Nestedness Condition for Ladder Decomposition of a δ-Invertible Morphisms

Since δ-invertible morphisms are special examples of morphisms of persistence modules, Theorem 3.1.3 applies to them. However, the assumption that neither $\mathrm{Bar}(V)$ nor $\mathrm{Bar}(W)$ admit strictly nested bars restricts the use of the theorem to a small family of morphisms. To see this clearly, observe Figure 3.2. Luckily, the special properties of δ-invertible morphism allow us to loosen the requirements of Theorem 3.1.3. In order to do that, we analyse nested bars in barcodes of persistence modules.

Definition 3.2.6. For a persistence module M define a constant

$$\Xi(M) = \min_{[a,b] \subset [c,d] \in \mathrm{Bar}(M)} \min\{|a - c|,\ |b - d|\}$$

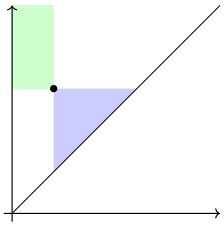


Figure 3.2: Observe a point in a persistence diagram that corresponds to a bar J. Any bar K that is strictly nested with J corresponds to a point in one of the shaded regions of the diagram: if $J \subset K$ it is in the green, and if $K \subset J$ it is in the blue region. This illustrates that many diagrams have strictly nested points.

and call it the **nestedness of persistence module** M. Note that the minimum loops over all pairs of strictly nested bars in $\mathrm{Bar}(M)$ and is defined to equal ∞ when such bars do not exist. It therefore takes values in $(0, \infty]$.

Example 3.2.7. Let us compute nestedness for the instructive example in Figure 3.3. The barcode consists of bars

$$I = [0, 8], \qquad J = [1, 5], \qquad K = [1, 8], \qquad L = [3, 5].$$

The minimum in the definition loops over pairs $J \subset I$, $L \subset K$, and $L \subset I$. The minimum

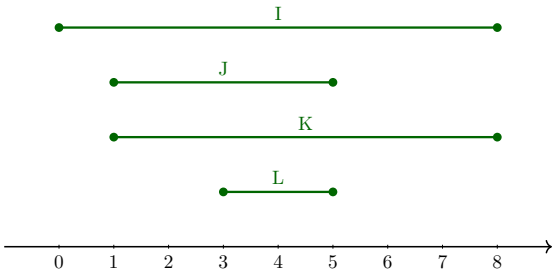
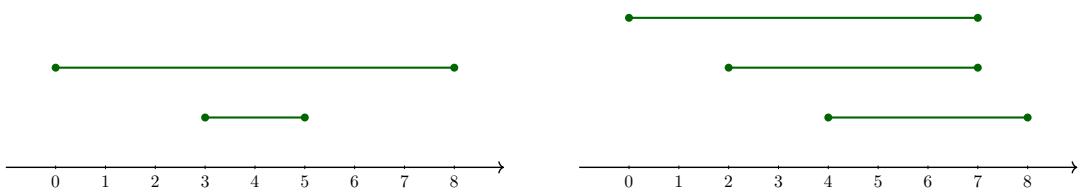


Figure 3.3: Example of a barcode with nestedness 1.

distance between endpoints for each pair respectively is 1, 2 and 3. Perhaps not intuitively the nestedness is defined as the smallest of these values, 1. Two further examples of barcodes with different nestedness are shown in Figure 3.4. △

The main result of this chapter is that, as long as the nestedness is not "too small", we can still obtain the ladder decomposition for a δ-invertible morphism.



(a) Barcode with nestedness 3. (b) Barcode with nestedness ∞.

Figure 3.4: Further examples of barcodes with different nestedness.

Theorem 3.2.8. *For a δ-invertible morphism $\Phi\colon V \to W$ with $\delta < \frac{1}{2}\min(\Xi(V), \Xi(W))$ there exist parameters $r_{J_1}^{J_2}, d_J^+, d_K^- \in \mathbb{N}$ such that*

$$(V, W, \Phi) \cong \bigoplus_{J_1 \preceq J_2} \left(\mathbf{R}_{J_1}^{J_2}\right)^{r_{J_1}^{J_2}} \oplus \bigoplus_{J} \left(\mathbf{I}_J^+\right)^{d_J^+} \oplus \bigoplus_{K} \left(\mathbf{I}_K^-\right)^{d_K^-}.$$

Remark 3.2.9. The restriction of Theorem 3.2.8 to the case when barcodes contain no nested bars, this is when $\min(\Xi(V), \Xi(W)) = \infty$, states the same as the restriction of Theorem 3.1.3 to δ-invertible morphisms.

Remark 3.2.10 (Sufficient but not necessary condition)**.** To see a trivial example of when a ladder decomposition exists for a δ-invertible morphism but Theorem 3.2.8 does not guarantee it, pick your favorite example of a persistence module M with nested bars, i.e. $\Xi(M) < \infty$, and observe the identity morphism $\mathrm{Id}\colon M \to M$. It is a δ-invertible morphism for all $\delta \in [0, \infty)$, and it does not satisfy the assumptions of Theorem 3.2.8 for any $\delta \geq \frac{1}{2}\Xi(M)$. However, it is obvious that it admits a ladder decomposition even for those choices of δ.

For a slightly less trivial example, let the morphism be given by its ladder decomposition

$$\mathbf{R}_{[0,2]}^{[0,2+m]} \oplus \mathbf{I}_{[1,1]}^+$$

for some $m \geq 2$. This is a δ-invertible morphism for any $\delta \geq \frac{m}{2}$, and its ladder decomposition obviously exists. However, the nestedness $\Xi(\{[0, 2+m], [1, 1]\})$ is 1, and not even the smallest $\delta = \frac{m}{2}$ satisfies the assumption $\delta < \frac{1}{2}\min(\Xi(V), \Xi(W)) = \frac{1}{2}$.

The proof of Theorem 3.2.8 follows a similar approach as the proof of Theorem 3.1.3 in [11]. By

Proposition 2.2.27, morphism Φ can be represented as a single block matrix

$$M_\Phi = \begin{bmatrix} X_{[0,0]}^{[0,0]} & X_{[0,0]}^{[0,1]} & \dots & X_{[0,0]}^{[0,l]} & 0 & \dots & 0 & \dots & 0 \\ & X_{[0,1]}^{[0,1]} & \dots & X_{[0,1]}^{[0,l]} & X_{[0,1]}^{[1,1]} & \dots & X_{[0,1]}^{[1,l]} & \dots & 0 \\ & & \ddots & \vdots & \vdots & & \vdots & & \vdots \\ & & & X_{[0,l]}^{[0,l]} & 0 & \dots & X_{[0,l]}^{[1,l]} & \dots & X_{[0,l]}^{[l,l]} \\ & & & & X_{[1,1]}^{[1,1]} & \dots & X_{[1,1]}^{[1,l]} & \dots & X_{[1,1]}^{[l,l]} \\ & & & & & \ddots & \vdots & & \vdots \\ & & & & & & X_{[1,l]}^{[1,l]} & \dots & 0 \\ & & & & & & & \ddots & \vdots \\ & & & & & & & & X_{[l,l]}^{[l,l]} \end{bmatrix},$$

This matrix will be inductively reduced to matching form. During the reduction we are allowed to use only the matrix operations whose result is the same morphism written in another barcode basis. These operations correspond to the actions of the stabilisers $\mathrm{Stab}(\{v_{t,t+1}\}_t)$ and $\mathrm{Stab}(\{w_{t,t+1}\}_t)$ (as in Equation (2.2)) on the barcode bases of the domain and codomain respectively. As a consequence of the properties of the stabilisers, namely that their elements commute with the inner morphisms, the admissible matrix operations (as deduced in [11]) are the following:

AO1 Any invertible operation between columns corresponding to the same bar J, or between rows corresponding to the same bar J.

AO2 Modifying C_J using C_K whenever $K \preceq J$.

AO3 Modifying R_J using R_K whenever $J \preceq K$.

Let us introduce a few technical lemmas, which will be used in the proof.

Lemma 3.2.11. *Let $\Phi\colon V \to W$ be a morphism of persistence modules and J and K two bars in $Bar(V)$. If $\operatorname{supp}\Phi(x_J)$ and $\operatorname{supp}\Phi(x_K)$ have a non-empty intersection, then $J \cap K \neq \emptyset$. Similarly, if generators of bars J and K in $Bar(W)$ both lie in $\operatorname{supp}\Phi(x_L)$ for some bar $L \in Bar(V)$, then $J \cap K \neq \emptyset$.*

Proof. Let us begin with the first statement. Without loss of generality, assume $J = [c,d] \leq K = [a,b]$. By Lemma 3.1.2 a bar $[i,j]$ that is in the support of both $\Phi(x_{[c,d]})$ and $\Phi(x_{[a,b]})$ must

satisfy

$$\begin{aligned} &i \leq c, \\ &c \leq j \leq d, \qquad (3.2) \\ &i \leq a, \text{ and} \\ &a \leq j \leq b. \qquad (3.3) \end{aligned}$$

Since such a bar $[i, j]$ exists by assumption, combining inequalities (3.2) and (3.3) gives $j \in [a, b] \cap [c, d]$ and so the intersection $J \cap K$ is not empty.

For the second statement, set $J = [c, d]$ and $K = [a, b]$. Then the bar $L = [i, j]$ in the support of whose image x_J and x_K lie, must satisfy

$$a, c \leq i \quad \text{and} \quad i \leq b, d$$

by Lemma 3.1.2. It follows that $i \in J \cap K \neq \emptyset$. □

Lemma 3.2.12. *Let $\Phi\colon V \to W$ be a morphism between persistence modules V and W, and let $[c, d] \subset [a, b]$ be a pair of nested bars in $Bar(V)$. Then any bar $[i, j] \in Bar(W)$ whose generator is contained in both $\operatorname{supp}\Phi(x_{[a,b]})$ and $\operatorname{supp}\Phi(x_{[c,d]})$ must satisfy*

$$i \leq a \qquad \text{and} \qquad c \leq j \leq d. \qquad (3.4)$$

As a consequence, the length of any such bar $[i, j]$ must be at least $c - a$.

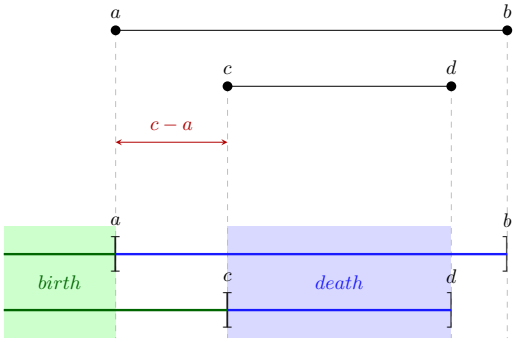
Similarly, any bar $[i, j] \in Bar(V)$ for which $\operatorname{supp}\Phi(x_{[i,j]})$ contains generators $x_{[a,b]}$ and $x_{[c,d]}$ of nested bars $[c, d] \subset [a, b] \in Bar(W)$ must satisfy

$$c \leq i \leq d \qquad \text{and} \qquad b \leq j. \qquad (3.5)$$

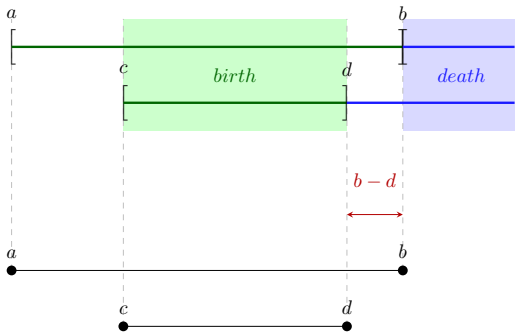
As a consequence, the length of any such bar $[i, j]$ must be at least $b - d$.

Proof. Both statements are a simple consequence of Lemma 3.1.2. For the bars in (3.4) it states that

$$i \leq a \leq j \leq b \qquad \text{and} \qquad i \leq c \leq j \leq d$$



(a) The restrictions for a bar in the support of both $\Phi(x_{[a,b]})$ and $\Phi(x_{[c,d]})$, where $[c,d] \subset [a,b]$ are strictly nested bars. The restrictions for birth point are marked in green, while the restrictions for death point are marked in blue, with the second line from the bottom showing the restrictions induced by bar $[a,b]$ and the bottom one those induced by $[c,d]$. The interval in which both are satisfied is marked with a square in the respective colour.



(b) The restrictions for a bar with both $x_{[a,b]}$ and $x_{[c,d]}$ in the support of its image, where $[c,d] \subset [a,b]$ are strictly nested bars. The restriction for birth point are marked in green, while the restrictions for death point are marked in blue, with the top line showing the restrictions induced by bar $[a,b]$ and the second line from the top showing those induced by $[c,d]$. The interval in which both are satisfied is marked with a square in the respective colour.

Figure 3.5: The implications of Lemma 3.1.2 for a pair of nested bars $[c,d] \subset [a,b]$ in $\mathrm{Bar}(V)$ (case a) or $\mathrm{Bar}(W)$ (case b).

(observe Figure 3.5a). Since the bars are nested, these requirements can be summarised as

$$i \leq a \qquad \text{and} \qquad c \leq j \leq d.$$

It follows readily that $j - i \geq c - a$. Statement (3.5) can be proved in a similar way (observe Figure 3.5b). □

Lemma 3.2.13. *Let $\Phi\colon V \to W$ be a δ-invertible morphism and $\Psi : W \to V(2\delta)$ its δ-inverse. For any bar $[a,b] \in Bar(V)$ of length at least 2δ with generator $x_{[a,b]}$ there exists a bar $[i,j] \in Bar(W)$ with generator $x_{[i,j]}$ such that*

$$x_{[i,j]} \in \operatorname{supp} \Phi(x_{[a,b]}), \tag{3.6}$$

$$x_{[a,b]}(2\delta) \in \operatorname{supp} \Psi(x_{[i,j]}). \tag{3.7}$$

Similarly, for any bar $[i,j] \in Bar(W)$ of length at least 2δ with generator $x_{[i,j]}$ there exists a bar $[a,b] \in Bar(V)$ with generator $x_{[a,b]}$ such that (3.6) and (3.7) hold. It is easy to see that

if $x_{[i,j]}$ is a generator that satisfies (3.6) and (3.7) for $x_{[a,b]}$, then $x_{[a,b]}$ is a generator that satisfies it for $x_{[i,j]}$.

The endpoints of bars $[a,b] \in Bar(V)$ and $[i,j] \in Bar(W)$ with bar generators for which (3.6) and (3.7) hold, must satisfy

$$a - 2\delta \leq i \leq a \leq i + 2\delta \qquad \text{and} \qquad b - 2\delta \leq j \leq b \leq j + 2\delta.$$

Proof. The containment statements are a simple consequence of the fact that compositions $\Phi \circ \Psi$ and $\Psi \circ \Phi$ map generators belonging to bars of length at least 2δ to themselves. The rest is a consequence of applying Lemma 3.1.2 to (3.6) and (3.7) and combining the obtained inequalities. □

Remark 3.2.14. Given a δ-interleaving pair (Φ, Ψ) between V and W, Lemma 3.2.13 implies that for any generator $x_{[a,b]}$ of a bar $[a,b] \in \mathrm{Bar}(V)$ with $b - a \geq 2\delta$ there exists a generator $x_{[i,j]}$ of bar $[i,j] \in \mathrm{Bar}(W)$ such that

$$x_{[i,j]}(\delta) \in \operatorname{supp} \Phi(x_{[a,b]}), \tag{3.8}$$
$$x_{[a,b]}(\delta) \in \operatorname{supp} \Psi(x_{[i,j]}). \tag{3.9}$$

Further, bar $[i,j]$ must satisfy

$$|i - a| \leq \delta \qquad \text{and} \qquad |j - b| \leq \delta.$$

Similar conditions hold for any generator $x_{[i,j]}$ of a bar with length at least 2δ.

Lemma 3.2.15. *Given $\Phi \colon V \to W$ let M_Φ be the matrix of a δ-invertible morphism $\Phi \colon V \to W$ written in barcode bases $\mathcal{B}_V$ and $\mathcal{B}_W$. Let $X_{[i,j]}^{[c,d]}$ be a sub-matrix of M_Φ (containing the rows and columns corresponding to generators of bars $[i,j]$ and $[c,d]$ respectively) such that all the sub-matrices to the left and below it have already been reduced to matching form with operations **AO1**, **AO2** and **AO3**. Then $X_{[i,j]}^{[c,d]}$ can also be reduced using these operations.*

Proof. Let A denote the sub-matrix containing all the sub-matrices appearing to the left and downward of $X_{[i,j]}^{[c,d]}$ (observe Figure 3.6). Since all other sub-matrices in A have already been reduced there is at most one 1 in each row and column of A outside of $X_{[i,j]}^{[c,d]}$. A non-zero entry of $X_{[i,j]}^{[c,d]}$ in row R and column C falls in (at least) one of the following categories:

1. The entries of row R and column C in A, that are not in $X_{[i,j]}^{[c,d]}$, are zero.
2. There is a 1 in the row R to the left of $X_{[i,j]}^{[c,d]}$.
3. There is a 1 in the column C bellow $X_{[i,j]}^{[c,d]}$.

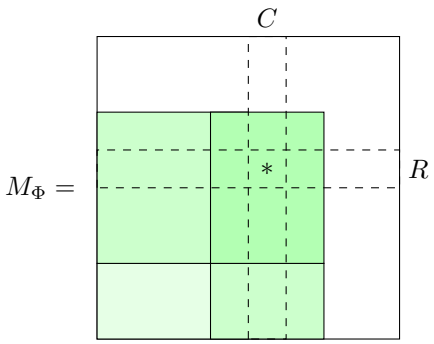


Figure 3.6: Areas in green denote the sub-matrix A containing all sub-matrices to the left and below sub-matrix $X_{[i,j]}^{[c,d]}$, marked with the darkest shade. The non-zero entry $M_\Phi(R, C)$ is denoted with $*$.

When reducing $X_{[i,j]}^{[c,d]}$, start with entries of type 2 and 3. We will justify that we can set them to zero using matrix operations of type **AO2** and **AO3**. After this is done, the non-zero entries will all be of type 1. The remaining steps in the reduction can be performed using operations of type **AO1**.

To show we can set an entry of type 2 to zero, assume 1 in row R lies in a column belonging to a bar $[a, b]$. Let us prove, that $[a, b] \preceq [c, d]$ and we can use **AO2** to reduce $M_\Phi(R, C)$. We already know that $[a, b] \leq [c, d]$, and by Lemma 3.2.11 the bars $[a, b]$ and $[c, d]$ have a non-empty intersection. It remains to be proven that $[c, d] \not\subset [a, b]$. For that purpose, assume $[c, d] \subset [a, b]$ and observe Figure 3.7a. By Lemma 3.2.12 the bar $[i, j]$, which is in the support of both $\Phi(x_{[c,d]})$ and $\Phi(x_{[a,b]})$, must satisfy

$$i \leq a \qquad \text{and} \qquad c \leq j \leq d, \tag{3.10}$$

and be of length at least $c - a \geq \Xi(V) > 2\delta$. By Lemma 3.2.13 there exists a bar $[k, l] \in \mathrm{Bar}(V)$ for which

$$x_{[i,j]} \in \operatorname{supp} \Phi(x_{[k,l]}) \qquad \text{and} \qquad x_{[k,l]}(2\delta) \in \operatorname{supp} \Psi(x_{[i,j]}), \tag{3.11}$$

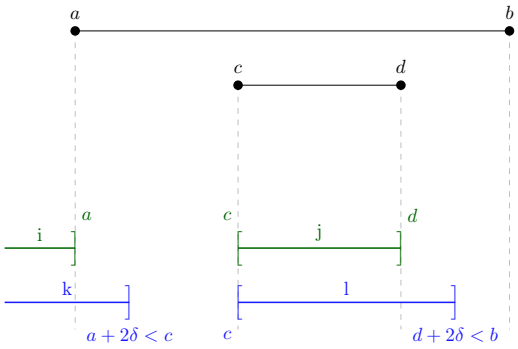
$$i \leq k \leq i + 2\delta \qquad \text{and} \qquad j \leq l \leq j + 2\delta, \tag{3.12}$$

where $\Psi : W \to V(2\delta)$ is an arbitrary choice of a δ-inverse of Φ. The combined inequalities (3.10) and (3.12) give us restrictions on $[k, l]$:

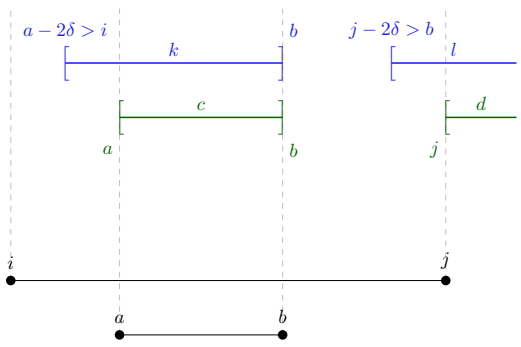
$$k \leq a + 2\delta \qquad \text{and} \qquad c \leq l \leq d + 2\delta.$$

Because $\Xi(V) > 2\delta$ we further have $l \leq d + 2\delta < b$. Now notice that k cannot be larger than a: if it is, then $[k, l] \subset [a, b]$ and $k - a \leq 2\delta$, which violates the assumption that $\delta < \frac{1}{2}\Xi(V)$. Similarly, l cannot be larger than d, since $d < l \leq d + 2\delta$ implies $[c, d] \subset [k, l]$, which violates the same assumption. As a consequence of these observations, the bar $[k, l]$ cannot be equal to $[a, b]$ or $[c, d]$.

To summarise, there is a bar $[k, l] \neq [a, b], [c, d]$ satisfying (3.11), which appears before $[a, b]$ and $[c, d]$ in the order $\leq$. Because $x_{[i,j]} \in \operatorname{supp} \Phi(x_{[k,l]})$, the entry in row R and column belonging to $[k, l]$ must be non-zero. This means there are two non-zero entries in row R of sub-matrix A to the left of $X^{[c,d]}_{[i,j]}$, which cannot be true, since all sub-matrices in A except for $X^{[c,d]}_{[i,j]}$ are reduced. Since we obtained a contradiction, bars $[c, d]$ and $[a, b]$ are not strictly nested. We can use operations of type **AO2** to reduce the entry $M_\Phi(R, C)$.



(a) The restrictions for endpoints of any bar $[i, j]$ whose generator is in the support of $\Phi(x_{[a,b]})$ and $\Phi(x_{[c,d]})$ for a δ-invertible morphism Φ are shown in green. In blue are the restrictions for the endpoints of a bar $[k, l]$ satisfying (3.11) and (3.12), which exists by Lemma 3.2.13.



(b) The restrictions for endpoints of any bar $[c, d]$ for which the support $\Phi(x_{[c,d]})$ contains generators $x_{[a,b]}$ and $x_{[i,j]}$ for a δ-invertible morphism Φ are shown in green. In blue are the restrictions for the endpoints of a bar $[k, l]$ satisfying (3.14) and (3.15), which exists by Lemma 3.2.13.

Figure 3.7: Restrictions for endpoints of bars used in the proof of Lemma 3.2.15.

The fact that the entries of the third type can be set to zero using operations of type **AO3** can be proven in a similar way. Assume 1 in column C lies in a row belonging to a bar $[a, b] \in \operatorname{Bar}(W)$.

We know that $[i,j] \leq [a,b]$, and by Lemma 3.2.11 the bars $[a,b]$ and $[i,j]$ have a non-empty intersection. As before, assume $[a,b] \subset [i,j]$ and observe Figure 3.7b. By Lemma 3.2.12 the bar $[c,d] \in \mathrm{Bar}(V)$, for which $\operatorname{supp}\Phi(x_{[c,d]})$ contains both $x_{[i,j]}$ and $x_{[a,b]}$, must satisfy

$$a \leq c \leq b \qquad \text{and} \qquad j \leq d, \tag{3.13}$$

and be of length at least $j - b \geq \Xi(V) > 2\delta$. By Lemma 3.2.13 there exists a bar $[k,l] \in \mathrm{Bar}(W)$ for which

$$x_{[k,l]} \in \operatorname{supp}\Phi(x_{[c,d]}) \qquad \text{and} \qquad x_{[c,d]}(2\delta) \in \operatorname{supp}\Psi(x_{[k,l]}), \tag{3.14}$$

$$c - 2\delta \leq k \leq c \qquad \text{and} \qquad d - 2\delta \leq l \leq d. \tag{3.15}$$

Notice that l cannot be smaller than j: if it is, the combined restrictions (3.13) and (3.15) give us $j - 2\delta \leq l < j$ and $a - 2\delta \leq k \leq b$. As a consequence $[k,l] \subset [i,j]$ for $|j-l| < 2\delta$, which violates the assumption that $\delta < \frac{1}{2}\Xi(V)$. Similarly, k cannot be smaller than a, since $a - 2\delta \leq k < a$ implies $[a,b] \subset [k,l]$, which violates the same assumption.

To summarise, there is a bar $[k,l] \neq [a,b], [i,j]$ satisfying (3.14), which appears after $[a,b]$ and $[i,j]$ in the order $\leq$. Because $x_{[k,l]} \in \operatorname{supp}\Phi(x_{[c,d]})$, the entry in column C and row belonging to $[k,l]$ must be non-zero. This means there are two non-zero entries in column C of sub-matrix A below $X_{[i,j]}^{[c,d]}$, which cannot be, since all sub-matrices in A except for $X_{[i,j]}^{[c,d]}$ are reduced. Since we obtained a contradiction, bars $[i,j]$ and $[a,b]$ are not strictly nested. We can therefore use the row belonging to bar $[a,b]$ in the reduction of $M_\Phi(R,C)$. □

We are now ready to conclude the proof of Theorem 3.2.8.

Proof of Theorem 3.2.8. Let us reduce the matrix M_Φ to matching form by admissible operations **AO1**, **AO2** and **AO3**. The reduction is done on sub-matrices inductively, processing the columns from left to right, starting at the lowest non-zero sub-matrix in each column and continuing upwards. Choosing this order, the assumptions of Lemma 3.2.15 are satisfied at each step, including for the first sub-matrix in the order. As a consequence the fact that the whole matrix can be reduced using admissible operations **AO1**, **AO2** and **AO3** follows readily. □

As is the case for the general morphism, the matching form of M_Φ is unique up to barcode basis changes acting among different bar generators of the same bar.

Example 3.2.16. The inequality in the assumption $\delta < \frac{1}{2}\min(\Xi(V), \Xi(W))$ of Theorem 3.2.8 is strict. Here, we provide an example of a δ-invertible $\Phi\colon V \to W$ for $\delta = \frac{1}{2}\min(\Xi(V), \Xi(W))$ that does not admit a ladder decomposition. Let V and W be persistence modules with barcodes $\mathrm{Bar}(V) = \{[0,5],[2,3]\}$ and $\mathrm{Bar}(W) = \{[0,3]\}$, and barcode bases given by generators

$$\begin{aligned} x^V_{[0,5]} &= \{\vec{e_1}^{(0)}, \vec{e_1}^{(1)}, \vec{e_1}^{(2)}, \vec{e_1}^{(3)}, \vec{e_1}^{(4)}, \vec{e_1}^{(5)}\}, \\ x^V_{[2,3]} &= \{\vec{e_2}^{(2)}, \vec{e_2}^{(3)}\}, \\ x^W_{[0,3]} &= \{\vec{f_1}^{(0)}, \vec{f_1}^{(1)}, \vec{f_1}^{(2)}, \vec{f_1}^{(3)}\}. \end{aligned}$$

Define the 1-invertible morphism Φ and its 1-inverse $\Psi\colon W \to V(2)$ by giving their matrix representations

$$M_\Phi = \begin{bmatrix} 1 & 1 \end{bmatrix} \qquad \text{and} \qquad M_\Psi = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Notice that $\Xi(V) = 2$ and $\Xi(W) = \infty$, and so $\delta = 1$ satisfies $2\delta = \min(\Xi(V), \Xi(W))$. Further, since the two bars constituting the barcode of V are nested, the only allowed operation on the columns of M_Φ is scaling, which is not enough to reduce M_Φ to matching form. △

Example 3.2.17. Consider again the 1-invertible morphism $\Phi^{(1)}\colon V \to W'$ from Example 3.2.4. Remember, the barcodes of the modules are

$$\mathrm{Bar}(V) = \{[0,4],[1,7],[4,4]\} \qquad \text{and} \qquad \mathrm{Bar}(W') = \{[0,4],[0,5]\},$$

where the bars have been written in the lexicographical order, and the single-matrix representation of $\Phi^{(1)}$ in the barcode bases chosen in Example 3.2.1 and Example 3.2.4 is

$$M_{\Phi^{(1)}} = \begin{bmatrix} 2 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

Note that the nestedness of modules V and W' are 3 and ∞ respectively, and since $\delta < \frac{1}{2}\min(3,\infty)$ holds for the shifting parameter $\delta = 1$, morphism $\Phi^{(1)}$ satisfies the assumptions of Theorem 3.2.8. Let us reduce the matrix $M_{\Phi^{(1)}}$ to matching form while keeping track of bases changes. We begin with the entry $M_{\Phi^{(1)}}(1,1) = 2$, which is reduced by an admissible operation

of type **AO1**,

$$x^V_{[0,4]} \mapsto \frac{1}{2}x^V_{[0,4]} = \{\frac{1}{2}(x^V_{[0,4]})_i\}_{i\in[0,4]}.$$

Moving on to the second column, leave the entry $M_{\Phi^{(1)}}(2,2) = 1$ unchanged and eliminate $M_{\Phi^{(1)}}(1,2) = 1$ by subtracting the second row from the first, which is the admissible operation of type **AO3** since $[0,4] \preceq [0,5]$. This corresponds to the basis change

$$x^{W'}_{[0,5]} \mapsto x^{W'}_{[0,5]} + x^{W'}_{[0,4]} = \{(x^{W'}_{[0,5]})_i + (x^{W'}_{[0,4]})_i\}_{i\in[0,4]} \cup \{(x^{W'}_{[0,5]})_5\}.$$

The updated matrix $M_{\Phi^{(1)}}$ is now

$$M_{\Phi^{(1)}} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

To eliminate $M_{\Phi^{(1)}}(1,3)$ and finish the reduction process, perform the basis change

$$x^V_{[4,4]} \mapsto x^V_{[4,4]} - x^V_{[0,4]} = \{(x^V_{[4,4]})_4 - (x^V_{[0,4]})_4\},$$

which corresponds to the admissible operation of type **AO2** subtracting first column from the last. The ladder decomposition of $(V, W', \Phi^{(1)})$ is therefore

$$\mathbf{R}^{[0,4]}_{[0,4]} \oplus \mathbf{R}^{[1,7]}_{[0,5]} \oplus \mathbf{I}^{+}_{[4,4]}$$

and is obtained in barcode bases in which the bar generators are

$$\begin{aligned}
x^V_{[0,4]} &= \{\frac{1}{2}\vec{e_1}^{(0)}, \frac{1}{2}\vec{e_1}^{(1)}, \frac{1}{2}\vec{e_1}^{(2)}, \frac{1}{2}\vec{e_1}^{(3)}, \frac{1}{2}\vec{e_1}^{(4)}\}, \\
x^V_{[1,7]} &= \{\vec{e_2}^{(1)}, \vec{e_2}^{(2)}, \vec{e_2}^{(3)}, \vec{e_2}^{(4)}, \vec{e_1}^{(5)}, \vec{e_1}^{(6)}, \vec{e_1}^{(7)}\}, \\
x^V_{[4,4]} &= \{\vec{e_3}^{(4)} - \frac{1}{2}\vec{e_1}^{(4)}\},
\end{aligned}$$

$$\begin{aligned}
x^{W'}_{[0,4]} &= \{\vec{f_1}^{(0)}, \vec{f_1}^{(1)}, \vec{f_1}^{(2)}, \vec{f_1}^{(3)}, \vec{f_1}^{(4)}\}, \\
x^{W'}_{[0,5]} &= \{\vec{f_2}^{(0)} + \vec{f_1}^{(0)}, \vec{f_2}^{(1)} + \vec{f_1}^{(1)}, \vec{f_2}^{(2)} + \vec{f_1}^{(2)}, \vec{f_2}^{(3)} + \vec{f_1}^{(3)}, \vec{f_2}^{(4)} + \vec{f_1}^{(4)}, \vec{f_1}^{(5)}\}.
\end{aligned}$$ △

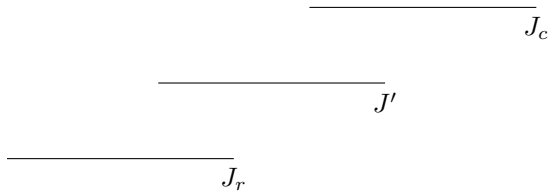


Figure 3.8: Suppose we compose a morphism mapping $x_{J'}$ to x_{J_r} with a morphism mapping x_{J_c} to $x_{J'}$. Since the bars J_c and J_r do not intersect, the composition of these morphisms would not map between x_{J_c} and x_{J_r}, which means the matrix representation of composition is not simply the product of matrix representations of morphisms.

3.2.3 Ladder Decompositions of an Interleaving Pair

Let (Φ, Ψ) be a δ-interleaving pair between modules V and W. Representing the composition $\Phi(\delta) \circ \Psi$ or $\Psi(\delta) \circ \Phi$ in a single matrix in a chosen barcode basis gives

$$\mathrm{Id}_{\geq 2\delta}(x_{J_1}, x_{J_2}) = \begin{cases} 1, & \text{if } |J_1| \geq 2\delta \text{ and } x_{J_1}(2\delta) = x_{J_2}, \\ 0, & \text{otherwise,} \end{cases}$$

which follows from the definition of a δ-interleaving pair. In the rest of this section we analyse these properties further to obtain results relating ladder decompositions of Φ and Ψ.

Remark 3.2.18. When working with single matrix representations of morphisms of persistence modules as defined in Proposition 2.2.27, we cannot rely on the intuition developed for matrices of linear maps between vector spaces. An example of such discrepancy is the fact that the matrix representation of the composition $\Phi \circ \Psi$ is in general not equal to the matrix product of M_Φ and M_Ψ. However, it can be obtained from the matrix product as follows

$$M_{\Psi\circ\Phi}(x_{J_r}, x_{J_c}) = \begin{cases} (M_\Psi \cdot M_\Phi)(x_{J_r}, x_{J_c}), & \text{if } J_r \preceq J_c, \\ 0, & \text{otherwise.} \end{cases}$$

To clarify, the entry $M_{\Psi\circ\Phi}(x_{J_r}, x_{J_c})$ might be zero if J_c and J_r have an empty intersection (see Figure 3.8).

Let i be the index of a row (or a column) and denote by x_i the bar generator corresponding to row (or column) i. Denote the corresponding bar by $J_i = [i_1, i_2]$.

Lemma 3.2.19. *Let M_Φ and M_Ψ be the matrix representations of morphisms $\Phi\colon V \to W(\delta)$*

and $\Psi : W \to V(\delta)$ *making an interleaving pair in the barcode bases in which* Φ *decomposes as in Theorem 3.2.8. For any non-zero entry* $M_\Phi(r,c) = 1$*, the following hold:*

1. *If* $|J_c| \geq 2\delta$ *or* $|J_r| \geq 2\delta$ *then* $M_\Psi(c,r) = 1$.
2. *If* $|J_c| \geq 2\delta$ *and there exists* $z \neq c$ *for which* $M_\Psi(z,r) \neq 0$ *then*
   (a) $J_z \preceq J_c$,
   (b) $z_2 < c_1 + 2\delta$.
3. *If* $|J_r| \geq 2\delta$ *and there exists* $z \neq r$ *for which* $M_\Psi(c,z) \neq 0$ *then*
   (a) $J_r \preceq J_z$,
   (b) $z_1 > r_2 - 2\delta$.

Proof. Suppose $|J_c| \geq 2\delta$ and observe the matrix representation of the composition $M_{\Psi\circ\Phi} = \mathrm{Id}_{\geq 2\delta}$. Since the entry $M_{\Psi\circ\Phi}(c,c) = 1$ is non-zero, Remark 3.2.18 suggests that $M_{\Psi\circ\Phi}(c,c) = (M_\Psi \cdot M_\Phi)(c,c)$. From the matrix equation

$$M_\Psi \cdot M_\Phi \quad = \quad \begin{array}{c} \\ c \\ z \end{array} \overset{r}{\begin{pmatrix} & & \\ * & a & * \\ & & \\ * & b & * \\ & & \end{pmatrix}} \cdot \overset{c}{\begin{pmatrix} & 0 & \\ & & \\ 0 & 1 & 0 \\ & 0 & \end{pmatrix}} \begin{array}{c} \\ \\ \\ r \\ \\ \end{array}$$

we can deduce that $1 = (M_\Psi \cdot M_\Phi)(c,c) = 1 \cdot a = M_\Psi(c,r)$. With a similar procedure we can obtain the same result for when $|J_r| \geq 2\delta$ which proves the first property.

Continue with the assumption that $|J_c| \geq 2\delta$ and there is a z such that $M_\Psi(z,r) \neq 0$. Now the entry $M_{\Psi\circ\Phi}(z,c)$ is zero, and since $(M_\Psi \cdot M_\Phi)(z,c) = M_\Psi(z,r)$ is not zero, it must be that the bars $J_z(2\delta)$ and J_c do not intersect. In other words, $z_2 < c_1 + 2\delta$ (property 2b). Now assume $J_z \not\preceq J_c$, which means we are in one of the following cases:

- $z_2 > c_2$,
- $z_1 > c_1$,
- $z_2 < c_1$.

The first case cannot happen since $c_2 < z_2 < c_1 + 2\delta$ and $|J_c| \geq 2\delta$ cannot hold simultaneously. In the second case, the chain of inequalities $c_1 < z_1 \leq z_2 < c_1 + 2\delta \leq c_2$ gives us $J_z \subset J_c$ and $|z_1 - c_1| < 2\delta$, which is a contradiction with the assumption that $\delta < \frac{1}{2}\min(\Xi(V), \Xi(W))$. Lastly, assume $z_2 < c_1$. Since $|J_c| \geq 2\delta$, the bar J_r must be the one from Remark 3.2.14. In particular,

$$|c_i - r_i| \leq \delta \text{ for } i = 1, 2.$$

Since $c_1 - \delta \leq r_1$, a bar J_z with the bar generator in $\operatorname{supp} \Psi(x_r)$ must satisfy $z_2 \geq r_1 + \delta \geq c_1$, which cannot be satisfied in the third case. We arrive to contradictions in all three cases, hence $J_z \preceq J_c$ holds.

Properties 3a and 3b can be obtained similarly by observing the entries of $M_{\Phi\circ\Psi} = \mathrm{Id}_{\geq 2\delta}$ and comparing them to

$$M_\Phi \cdot M_\Psi \quad = \quad \begin{array}{c} \\ r \end{array} \begin{array}{c} \begin{array}{c} c \end{array} \\ \begin{pmatrix} & \vdots & \\ \cdots & 0 & \cdots \\ 0 & 1 & 0 \\ \cdots & 0 & \cdots \\ & \vdots & \end{pmatrix} \end{array} \cdot \begin{array}{c} \begin{array}{cc} z & r \end{array} \\ \begin{pmatrix} * & * \\ a & b \\ * & * \end{pmatrix} \end{array} \begin{array}{l} \\ c \end{array} .$$

□

Corollary 3.2.20. *Let (Φ, Ψ) be a δ-interleaving pair between persistence modules V and W for $\delta < \frac{1}{2}\min(\Xi(V), \Xi(W))$. Let the matrix representation M_Φ of Φ, written in barcode bases B_V^Φ and B_W^Φ of V and W respectively, be in matching form. There exists a pair of barcode bases B_V^Ψ and B_W^Ψ for V and W respectively, in which the matrix representation M_Ψ of Ψ is in matching form and*

$$M_\Phi(r, c) = M_\Psi(c, r)$$

whenever $|J_c| \geq 2\delta$ and $|J_r| \geq 2\delta$.

Proof. Write M_Ψ in barcode bases B_V^Φ and B_W^Φ and perform the reduction process using admissible operations **AO1**, **AO2** and **AO3**. Assume we reduced everything to the left of and below $M_\Psi(c, r)$, and that $M_\Psi(c, r) \neq 0$ (as we can skip it if it is zero). Further, since operations of type **AO1** can be used to scale it, assume that $M_\Psi(c, r) = 1$.

First, consider the case when $M_\Phi(r,c) = 1$. Since $|J_c| \geq 2\delta$ and $|J_r| \geq 2\delta$, the combined properties 2a and 3a from Lemma 3.2.19 guarantee that the row c to the left of $M_\Psi(c,r)$ and the column r below $M_\Psi(c,r)$ were zero at the start of the reduction. This means that the steps of the reduction before encountering $M_\Psi(c,r)$ did not modify them and they are both still zero. This also means we can leave the entry $M_\Psi(c,r)$ to be one and move to the next step of the reduction.

On the other hand, when $M_\Phi(r,c) = 0$ another entry in column c of M_Φ must be equal to 1 because $|J_c| \geq 2\delta$. Say this entry is $M_\Phi(r',c)$. By property 1 of Lemma 3.2.19 this means $M_\Psi(c,r') = 1$. Now we have two non-zero entries in row c of M_Ψ, namely $M_\Psi(c,r')$ and $M_\Psi(c,r)$. By property 3a of Lemma 3.2.19, $J_{r'} \preceq J_r$ and we can reduce the entry $M_\Psi(c,r)$ by subtracting column r' from column r, which is an admissible operation of type **AO2**. □

Theorem 3.2.21. *Let (Φ,Ψ) be a δ-interleaving pair between modules V and W with $\delta < \frac{1}{2}\min(\Xi(V),\Xi(W))$. For any pair of bars $J_V \in Bar(V)$ and $J_W \in Bar(W)$ satisfying $|J_V|, |J_W| \geq 2\delta$, and for any $\mu \in \mathbb{N}$ the following statements are equivalent*

- $(\mathbf{R}^{J_V}_{J_W(\delta)})^\mu$ *appears in the ladder decomposition of* Φ,
- $(\mathbf{R}^{J_W}_{J_V(\delta)})^\mu$ *appears in the ladder decomposition of* Ψ.

Proof. For each appearance of $\mathbf{R}^{J_V}_{J_W(\delta)}$ in the ladder decomposition of Φ we have a unique entry $M_\Phi(R,C) = 1$ in the reduced matrix in the matching form, where R is a row index belonging to the bar $J_W(\delta)$ and C a column index belonging to the bar J_V. By Corollary 3.2.20, the entry $M_\Psi(C,R)$ in the matching form of Ψ also equals 1. It corresponds to an appearance of $\mathbf{R}^{J_W}_{J_V(\delta)}$ in the ladder decomposition of Ψ. □

Example 3.2.22. Continue Examples 3.2.1 and 3.2.17 by considering the morphism Ψ, which forms an interleaving with Φ. First, remember the ladder decomposition

$$\mathbf{R}^{[0,4]}_{[0,4]} \oplus \mathbf{R}^{[1,7]}_{[0,5]} \oplus \mathbf{I}^{+}_{[4,4]} \qquad \triangle$$

that we obtained for the 1-invertible morphism $\Phi^{(1)}$ in Example 3.2.17. The ladder decomposition of Φ is then

$$\mathbf{R}^{[0,4]}_{[1,5]} \oplus \mathbf{R}^{[1,7]}_{[1,6]} \oplus \mathbf{I}^{+}_{[4,4]}.$$

To obtain a ladder decomposition for Ψ as well, begin by writing the matrices Ψ_i in the (shifted equivalents of) barcode bases obtained in Example 3.2.17:

$$\Psi_0, \Psi_7 = 0, \qquad \Psi_1, \Psi_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \Psi_3 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \qquad \Psi_4, \Psi_5 = \begin{bmatrix} 0 & 1 \end{bmatrix}, \qquad \Psi_6 = \begin{bmatrix} 1 \end{bmatrix}.$$

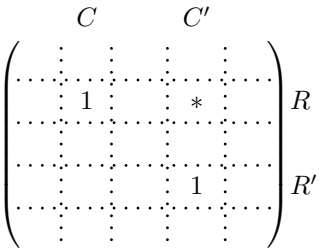
Since the single matrix representation of Ψ

$$M_\Psi = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{3.16}$$

is already in matching form, further reduction is not necessary. The ladder decomposition of Ψ is

$$\mathbf{R}_{[0,4]}^{[1,5]} \oplus \mathbf{R}_{[1,7]}^{[1,6]} \oplus \mathbf{I}_{[4,4]}^{-}.$$

Remark 3.2.23. The ladder decompositions of Φ and Ψ are in most cases obtained in different pairs of barcode bases $(\mathcal{B}_V^\Phi, \mathcal{B}_W^\Phi)$ and $(\mathcal{B}_V^\Psi, \mathcal{B}_W^\Psi)$ respectfully, which is not illustrated in Example 3.2.22. This happens whenever M_Ψ written in bases $(\mathcal{B}_V^\Phi, \mathcal{B}_W^\Phi)$ contains a non-zero entry $M_\Psi(R, C')$ as



or when a similar entry can be found in M_Φ written in bases $(\mathcal{B}_V^\Psi, \mathcal{B}_W^\Psi)$.

3.3 q-Coarse Ladder Decomposition

As deduced in Section 3.2.2, the range of parameters δ for which δ-invertible morphisms between persistence modules V and W will decompose as in Theorem 3.2.8 is controlled by the nestedness of V and W. More precisely, small nestedness imposes a harsher limit on the parameter δ in Theorem 3.2.8. In this section we focus on persistence modules with small nestedness which is

achieved in (relatively) short bars, as illustrated in Figure 3.9. We explore to which extent short nested bars can be ignored and how this weakens our results from Sections 3.2.2 and 3.2.3.

Figure 3.9: Example of a barcode of a persistence module with small nestedness. The minimum from the definition of nestedness is achieved in the lower two bars, which are significantly shorter than the top-most bar and can in some cases be attributed to noise.

Definition 3.3.1. Let V be a persistence module and $q \in \mathbb{R}_{\geq 0}$. An isomorphism $V \cong V_{\geq q} \oplus V_{<q}$, where the pair $(V_{\geq q}, V_{<q})$ of persistence modules satisfies

- any bar $J \in \mathrm{Bar}(V_{\geq q})$ is of length at least q,
- any bar $J \in \mathrm{Bar}(V_{<q})$ is of length smaller than q,

is called a q-**splitting** of V.

A q-splitting induces epimorphisms $pr^V_{\geq q}, pr^V_{<q}$ and monomorphisms $i^V_{\geq q}, i^V_{<q}$, which map as follows:

$$V_{\geq q} \underset{pr^V_{\geq q}}{\overset{i^V_{\geq q}}{\hookleftarrows}} V \underset{i^V_{<q}}{\overset{pr^V_{<q}}{\twoheadleftarrow}} V_{<q}.$$

Whenever we wish to discard shorter bars, let us say shorter than q, we will project the module V onto the part $V_{\geq q}$ of its splitting. We refer to $V_{\geq q}$ as the q-**coarse part** of V. Such a splitting exists for any persistence module and any parameter $q \geq 0$, and is especially convenient since it allows us to define barcode bases $\mathcal{B}_{V_{\geq q}}$ and $\mathcal{B}_{V_{<q}}$ for $V_{\geq q}$ and $V_{<q}$ separately. The barcode basis $\mathcal{B}_V$ these induce on V is then defined simply as

$$\mathcal{B}_V = i^V_{\geq q}(\mathcal{B}_{V_{\geq q}}) \cup i^V_{<q}(\mathcal{B}_{V_{<q}}).$$

Remember that by Definition 3.2.5 the notation $[\delta]_V$ is used to denote the collection of inner morphisms $\{v_{i,i+\delta}\}_i$ of module V. For the sake of brevity let us omit the subscript denoting the module and write only $[\delta]$ whenever the module can be discerned from the context.

Lemma 3.3.2. *For any parameter q and a persistence module V the following properties hold:*

1. $\Xi(V_{\geq q}) = \min_{\substack{[c,d]\subset[a,b]\in Bar(M) \\ d-c\geq q}} \min\{|a-c|, |b-d|\}$.
2. $d_B(Bar(V), Bar(V_{\geq q})) < \frac{q}{2}$.
3. *V and $V_{\geq q}$ are $\frac{q}{2}$-interleaved with interleaving morphisms*

$$\varphi^V_{\geq q}\colon V \to V_{\geq q}(\frac{q}{2}) \qquad \tilde{\varphi}^V_{\geq q}\colon V_{\geq q} \to V(\frac{q}{2})$$
$$\varphi^V_{\geq q} = [\frac{q}{2}] \circ pr^V_{\geq q} \qquad \tilde{\varphi}^V_{\geq q} = i^V_{\geq q}(\frac{q}{2}) \circ [\frac{q}{2}].$$

4. *Epimorphism $pr^V_{\geq q}$ and monomorphism $i^V_{\geq q}$ are $\frac{q}{2}$-invertible.*

Proof. Property 1 is rather obvious since the barcode $\mathrm{Bar}(V_{\geq q})$ can be obtained from $\mathrm{Bar}(V)$ by discarding bars shorter than q. To prove Property 2 observe that the the bottleneck distance will be achieved in a matching where all bars in $\mathrm{Bar}(V)$ of length at least q are matched with their copies in $\mathrm{Bar}(V_{\geq q})$ and the rest are left unmatched. The first contribute nothing to the cost, while the second are of length $< q$, and not matching them contributes less than $\frac{q}{2}$ to the cost. As a consequence of Property 2 and the algebraic stability theorem [10], modules V and $V_{\geq q}$ are $\frac{q}{2}$-interleaved. To prove that a possible choice for $\frac{q}{2}$-interleaving morphisms are $\varphi^V_{\geq q}$ and $\tilde{\varphi}^V_{\geq q}$, see that

$$\tilde{\varphi}^V_{\geq q}(\frac{q}{2}) \circ \varphi^V_{\geq q} = i^V_{\geq q}(q) \circ [q] \circ pr^V_{\geq q} = [q] \circ i^V_{\geq q} \circ pr^V_{\geq q},$$

where we use the morphism property $i^V_{\geq q}(q) \circ [q] = [q] \circ i^V_{\geq q}$ in the last step. Further, since $V_{<q} \subseteq \ker([q])$ we obtain the desired

$$\tilde{\varphi}^V_{\geq q}(\frac{q}{2}) \circ \varphi^V_{\geq q} = [q].$$

The proof that $\varphi^V_{\geq q}(\frac{q}{2}) \circ \tilde{\varphi}^V_{\geq q} = [q]$ also follows from similar considerations. These equalities together finish the proof of Property 3. To prove that $pr^V_{\geq q}$ is $\frac{q}{2}$-invertible, we simply need to restate Property 3 as

$$(i^V_{\geq q}(q) \circ [q]) \circ pr^V_{\geq q} = [q]$$
$$pr^V_{\geq q}(q) \circ (i^V_{\geq q}(q) \circ [q]) = pr^V_{\geq q}(q) \circ ([g] \circ i^V_{\geq q}) = [q]$$

and see that $i^V_{\geq q}(q) \circ [q]$ is its $\frac{q}{2}$-inverse. Similarly, $i^V_{\geq q}$ is $\frac{q}{2}$-invertible with $pr^V_{\geq q}(q) \circ [q]$ as its $\frac{q}{2}$-inverse. □

3.3.1 q-Coarse Ladder Decomposition of a δ-invertible Morphism

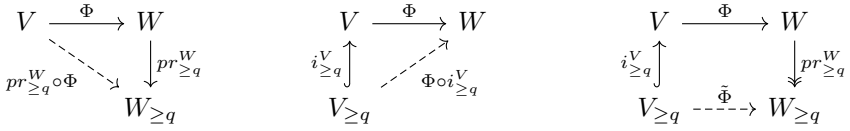
In this section we state the weaker versions of our results from Section 3.2 that hold for δ-invertible morphisms.

Lemma 3.3.3. *Let $q \geq 0$. A δ-invertible morphism $\Phi\colon V \to W$ with a δ-inverse Ψ induces the following maps onto q-coarse parts of V and W:*

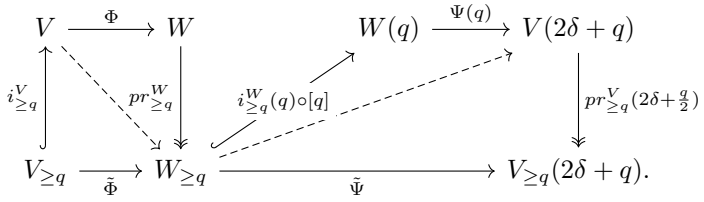
1. *A $(\delta+\frac{q}{2})$-invertible morphism $pr^W_{\geq q}\circ\Phi\colon V \to W_{\geq q}$ with a $(\delta+\frac{q}{2})$-inverse $\Psi(q)\circ i^W_{\geq q}(q)\circ[q]$.*
2. *A $(\delta+\frac{q}{2})$-invertible morphism $\Phi \circ i^V_{\geq q}\colon V_{\geq q} \to W$ with a $(\delta+\frac{q}{2})$-inverse $pr^V_{\geq q}(q+2\delta)\circ [q](2\delta)\circ\Psi$.*
3. *A $(\delta+\frac{q}{2})$-invertible morphism $\tilde{\Phi}\colon V_{\geq q} \to W_{\geq q}$ with a $(\delta+\frac{q}{2})$-inverse $\tilde{\Psi}$ where*

$$\tilde{\Phi} = pr^W_{\geq q} \circ \Phi \circ i^V_{\geq q},$$
$$\tilde{\Psi} = pr^V_{\geq q}(2\delta+q) \circ \Psi(q) \circ (i^W_{\geq q}(q) \circ [q]).$$

Lemma 3.3.3 gives rise to the following commutative diagrams:



Proof. The first two statements are a simple consequence of the fact that a composition of a δ-invertible morphism with a $\frac{q}{2}$-invertible morphism is a $(\delta+\frac{q}{2})$-invertible morphism. To show the third, draw the composition $\tilde{\Psi}\circ\tilde{\Phi}$ in a diagram as



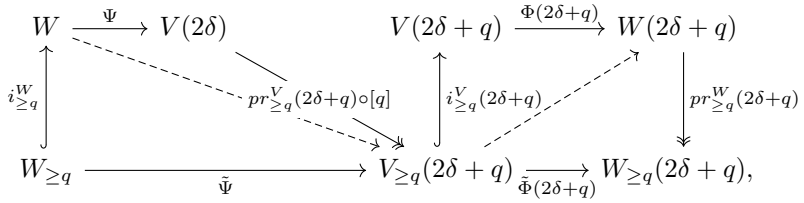
where the dashed arrows denote compositions $pr^V_{\geq q} \circ \Phi$ of a δ- and $\frac{q}{2}$-invertible morphism, and $\Psi(q) \circ i^W_{\geq q}(q) \circ [q]$ of their δ- and $\frac{q}{2}$-inverses. Consequently, the composition

$$\Psi(q) \circ i^W_{\geq q}(q) \circ [q] \circ pr^V_{\geq q} \circ \Phi$$

is simply the inner morphism $[2\delta + q]_V$. Further,

$$\tilde{\Psi} \circ \tilde{\Phi} = pr^V_{\geq q}(2\delta + q) \circ [2\delta + q]_V \circ i^V_{\geq q}$$

is simply the inner morphism $[2\delta + q]_{V_{\geq q}}$. The proof that $\tilde{\Phi}(2\delta + q) \circ \tilde{\Psi} = [2\delta + q]_{W_{\geq q}}$ follows in a similar way. Draw the composition $\tilde{\Phi}(2\delta + q) \circ \tilde{\Psi}$ in a diagram as



where we use the fact that morphisms commute with the inner morphisms $[q]$ to equivalently write

$$\tilde{\Psi} = pr^V_{\geq q}(2\delta + q) \circ \Psi(q) \circ (i^W_{\geq q}(q) \circ [q]) \qquad \text{as} \qquad (pr^V_{\geq q}(2\delta + q) \circ [q]) \circ \Psi \circ i^W_{\geq q}.$$

The dashed arrows again denote compositions $(\Phi \circ i^V_{\geq q})(2\delta + q)$ of a δ- and $\frac{q}{2}$-invertible morphism (on the right), and $pr^V_{\geq q}(2\delta + q) \circ [q] \circ \Psi$ of their δ- and $\frac{q}{2}$-inverses (on the left). The composition of the dashed arrows is therefore the family of inner morphisms $[2\delta + q]_W$ and the composition

$$pr^W_{\geq q}(2\delta + q) \circ [2\delta + q]_W \circ i^W_{\geq q}$$

is also a family of inner morphisms, $[2\delta + q]_{W_{\geq q}}$. □

The following theorem lists the conditions that must be satisfied so that the induced morphisms from Lemma 3.3.3 decompose as ladder persistence modules.

Theorem 3.3.4. *For any $q \geq 0$ a δ-invertible morphism $\Phi \colon V \to W$ induces the following $(\delta + \frac{q}{2})$-invertible morphisms:*

1. *Morphism* $pr^W_{\geq q} \circ \Phi \colon V \to W_{\geq q}$. *If* $\delta < \frac{1}{2}\min(\Xi(V), \Xi(W_{\geq q})) - \frac{q}{2}$ *holds, it decomposes as a ladder persistence module.*
2. *Morphism* $\Phi \circ i^V_{\geq q} \colon V_{\geq q} \to W$. *If* $\delta < \frac{1}{2}\min(\Xi(V_{\geq q}), \Xi(W)) - \frac{q}{2}$ *holds, it decomposes as a ladder persistence module.*
3. *Morphism* $pr^W_{\geq q} \circ \Phi \circ i^V_{\geq q} \colon V_{\geq q} \to W_{\geq q}$. *If* $\delta < \frac{1}{2}\min(\Xi(V_{\geq q}), \Xi(W_{\geq q})) - \frac{q}{2}$ *holds, it decomposes as a ladder persistence module.*

Furthermore, the barcode bases in which these ladder decompositions are obtained can be extended to barcode bases of persistence modules V *and* W.

Proof. All three statements of this proposition are a direct consequence of Theorem 3.2.8. Let us provide details only for the proof of the last one, since we follow the same approach in all cases.

The morphism $\tilde{\Phi} = pr^W_{\geq q} \circ \Phi \circ i^V_{\geq q}$ is a $(\delta + \frac{q}{2})$-invertible morphism between $V_{\geq q}$ and $W_{\geq q}$ by Lemma 3.3.3. Since

$$\delta + \frac{q}{2} < \frac{1}{2}\min(\Xi(V_{\geq q}), \Xi(W_{\geq q}))$$

holds by our assumption, we can apply Theorem 3.2.8 to $\tilde{\Phi}$. This means there is a pair of barcode bases $\mathcal{B}_{V_{\geq q}}$ and $\mathcal{B}_{W_{\geq q}}$ in which Φ' decomposes as a ladder persistence module. As noted before, the fact that $V_{\geq q}$ is the q-coarse part of a q-splitting of V means that $\mathcal{B}_{V_{\geq q}}$ can be supplemented with any barcode basis $\mathcal{B}_{V_{<q}}$ to form a barcode basis for V. By extending both $\mathcal{B}_{V_{\geq q}}$ and $\mathcal{B}_{W_{\geq q}}$ we obtain barcode bases $\mathcal{B}_V$ and $\mathcal{B}_W$ claimed to exist by the theorem. □

3.3.2 q-Coarse Ladder Decompositions of a δ-Interleaving Pair

As before, we can leverage the correspondence between δ-invertible morphisms and δ-interleavings to obtain a statement similar to Theorem 3.3.4 that holds for a single morphism in a δ-interleaving pair. We state the analogues of the two theoretical results of Section 3.3.1 here, omitting all the proofs, since they are simple exercises in applying the correspondence from Remark 3.2.3. Comparing the q-coarse ladder decompositions of the two morphisms making a δ-interleaving we again show that there is a nice correspondence between them for all bars of sufficient length.

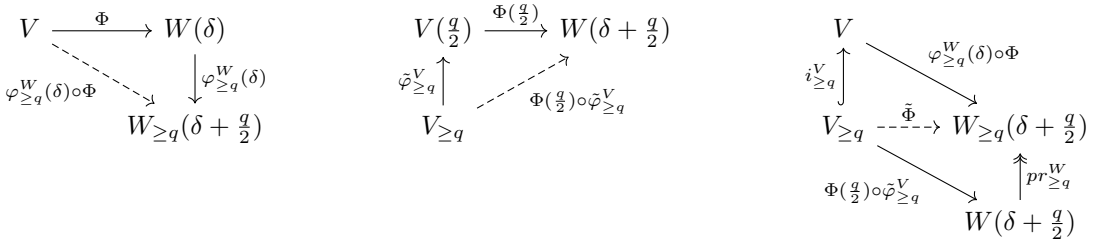
Lemma 3.3.5 (Analogue of Lemma 3.3.3)**.** *Let* $q \geq 0$. *A* δ*-interleaving pair* (Φ, Ψ) *between modules* V *and* W *induces the following maps onto their* q*-coarse parts:*

1. *Morphisms* $\varphi^W_{\geq q}(\delta) \circ \Phi \colon V \to W_{\geq q}(\delta + \frac{q}{2})$ *and* $\Psi(\frac{q}{2}) \circ \tilde{\varphi}^W_{\geq q} \colon W_{\geq q} \to V(\delta + \frac{q}{2})$ *making a* $(\delta + \frac{q}{2})$*-interleaving pair.*
2. *Morphisms* $\Phi(\frac{q}{2}) \circ \tilde{\varphi}^V_{\geq q} \colon V_{\geq q} \to W(\delta + \frac{q}{2})$ *and* $\varphi^V_{\geq q}(\delta) \circ \Psi \colon W \to V_{\geq q}(\delta + \frac{q}{2})$ *making a* $(\delta + \frac{q}{2})$*-interleaving pair.*
3. *Morphisms* $\tilde{\Phi} \colon V_{\geq q} \to W_{\geq q}(\delta + \frac{q}{2})$ *and* $\tilde{\Psi} : W_{\geq q} \to V_{\geq q}(\delta + \frac{q}{2})$ *where*

$$\tilde{\Phi} = \varphi^W_{\geq q}(\delta) \circ \Phi \circ i^V_{\geq q},$$
$$\tilde{\Psi} = \varphi^V_{\geq q}(\delta) \circ \Psi \circ i^W_{\geq q},$$

which make a $(\delta + \frac{q}{2})$*-interleaving pair.*

Lemma 3.3.5 gives rise to the following commutative diagrams:



Theorem 3.3.6 (Analogue of Theorem 3.3.4)**.** *For any* $q \geq 0$*, a morphism* $\Phi \colon V \to W(\delta)$ *which is part of a* δ*-interleaving pair induces the following morphisms:*

1. *Morphism* $\varphi^W_{\geq q}(\delta) \circ \Phi$*. If* $\delta < \frac{1}{2}\min(\Xi(V), \Xi(W_{\geq q})) - \frac{q}{2}$ *holds, it decomposes as a ladder persistence module.*
2. *Morphism* $\Phi(\frac{q}{2}) \circ \tilde{\varphi}^V_{\geq q}$*. If* $\delta < \frac{1}{2}\min(\Xi(V_{\geq q}), \Xi(W)) - \frac{q}{2}$ *holds, it decomposes as a ladder persistence module.*
3. *Morphism* $\tilde{\Phi}$ *from Lemma 3.3.5. If* $\delta < \frac{1}{2}\min(\Xi(V_{\geq q}), \Xi(W_{\geq q})) - \frac{q}{2}$ *holds, it decomposes as a ladder persistence module.*

All these decompositions are obtained in partial barcode bases that can be extended to barcode bases of modules V *and* W*.*

Notice how Lemma 3.3.5 and Theorem 3.3.6 fit together. Given a δ-interleaving pair (Φ, Ψ), morphism Φ satisfies the assumptions of case (1) of Theorem 3.3.6 for some q if and only if the

morphism Ψ satisfies the assumptions of case (2) of Theorem 3.3.6 for the same parameter q. The induced morphisms each of them gives make a $(\delta + \frac{q}{2})$-interleaving pair by Lemma 3.3.5(1). By switching the roles of Φ and Ψ we can see the reverse also holds giving us the $(\delta + \frac{q}{2})$-interleaving pair from Lemma 3.3.5(2). Further, morphisms Φ and Ψ satisfy the assumptions of case (3) simultaneously, giving us the $(\delta + \frac{q}{2})$-interleaving pair from Lemma 3.3.5(3). This means Theorem 3.3.6 assures that whenever one of the morphisms in the pair can be decomposed, then so can the other. More importantly, we can compare them.

Let (Φ', Ψ') be any of the induced $(\delta + \frac{q}{2})$-interleaving pairs from Lemma 3.3.5 for which the ladder decomposition can be obtained by Theorem 3.3.6. By applying Corollary 3.2.20 and Theorem 3.2.21 to (Φ', Ψ') we obtain the following result.

Corollary 3.3.7. *Let the matrix representation $M_{\Phi'}$ of Φ', written in barcode bases B_V^{Φ} and B_W^{Φ} of V and W respectively, be in matching form. There exists a pair of barcode bases B_V^{Ψ} and B_W^{Ψ} for V and W respectively, in which the matrix representation $M_{\Psi'}$ of Ψ' is in matching form and*

$$M_{\Phi'}(r, c) = M_{\Psi'}(c, r)$$

whenever $|J_c| \geq 2\delta + q$ and $|J_r| \geq 2\delta + q$. For such two bars and any $\mu \in \mathbb{N}$, the following statements are equivalent

- $(\mathbf{R}_{J_r}^{J_c})^{\mu}$ *appears in the ladder decomposition of* Φ',
- $(\mathbf{R}_{J_c}^{J_r})^{\mu}$ *appears in the ladder decomposition of* Ψ'.

Since the barcode bases in which we write the matrix $M_{\Phi'}$ can be expanded to barcode bases of the whole persistence modules in the domain and codomain of Φ, we can think of $M_{\Phi'}$ as a sub-matrix of M_{Φ}.

3.4 Induced Partial Matchings

Here we give a brief introduction to multisets, which barcodes are examples of. A **multiset** S is a set where each element $s \in S$ has a non-zero multiplicity $\mu(s) \in \mathbb{Z}_{>0}$. An isomorphism of multisets is a bijection of the underlying sets that preserves the multiplicities. A **sub-multiset** is a subset $T \subseteq S$ in which the multiplicity of each element is not bigger than its multiplicity

in S. A morphism of multisets S_1 and S_2 consists of sub-multisets $T_1 \subseteq S_1$ and $T_2 \subseteq S_2$ and an isomophism $\chi : T_1 \to T_2$. We call it a **partial matching** between S_1 and S_2 and denote it by

$$\chi \colon S_1 \nrightarrow S_2.$$

We often write and define partial matchings as multisets of pairs $(t_1, \chi(t_1)) \in T_1 \times T_2$. They appear in the definitions of various notions of distances on the space of barcodes, such as the bottleneck [94] and the p-Wasserstein distance [42]. More precisely, each of these distances $d(B_1, B_2)$ is the smallest cost of a partial matching between B_1 and B_2, where the associated cost is different for each of the distances. In the case of the bottleneck distance, which is relevant for the use in this chapter, the **cost of a partial matching** $\chi \colon S_1 \nrightarrow S_2$ is

$$\max\Big\{ \max_{(I,J)\in\chi} \{\max(|i_1 - j_1|, |i_2 - j_2|)\}, \max_{\substack{J\in B_1\cup B_2 \\ J\notin\chi}} \frac{1}{2}(j_2 - j_1)\Big\},$$

where $I = [i_1, i_2]$ and $J = [j_1, j_2]$.

Given a morphism between one-parameter persistence modules, one might ask whether it induces a partial matching on the level of barcodes. This question was central in the proof of the **algebraic stability theorem** [10], when a **BL induced matching** (BL stands for "Bauer and Lesnick") was introduced. In this section we look at an alternative construction of a morphism induced partial matching given by the ladder decompositions. We compare them to the BL-induced matching and show that they are an example of **basis-independent induced matchings** [13]. We conclude the chapter with the matchings induced by the ladder decompositions of the coarser versions of the δ-invertible morphisms.

3.4.1 Ladder Decomposition Induced Partial Matching

In [11], Jacquard et al. observe that an alternative definition of an induced partial matching can be retrieved from the ladder decomposition of the morphism in question.

Corollary 3.4.1 (of Theorem 3.1.3)**.** *The ladder decomposition of a morphism $\Phi \colon V \to W$ induces a matching of barcodes $Bar(V)$ and $Bar(W)$ defined as*

$$\chi_\Phi \colon Bar(V) \nrightarrow Bar(W)$$
$$\chi_\Phi = \{((J_1, J_2), r_{J_1}^{J_2}) \mid \mathbf{R}_{J_1}^{J_2} \text{ appears in ladder decomposition of } \Phi\}.$$

Note that χ_Φ is a multiset in which each pair (J_1, J_2) appears with the multiplicity $r_{J_1}^{J_2}$.

The uniqueness of the ladder decomposition implies that the induced matching is also unique. Theorem 3.2.8 assures the existence of a ladder decomposition for a wider range of δ-invertible morphisms, which induce partial matchings in a similar way. By the correspondence from Remark 3.2.3 the same holds for morphisms that are part of a δ-interleaving pair. The following results describe the properties of the matchings they induce.

Corollary 3.4.2 (of Theorem 3.2.8)**.** *Let $\Phi\colon V \to W(\delta)$ be one of two morphisms making a δ-interleaving pair for $\delta < \frac{1}{2}\min(\Xi(V), \Xi(W))$, and χ_Φ the partial matching induced by the ladder decomposition of Φ. Its cost is at most δ. If further $\delta = d_I(V, W)$, then the induced matching realizes the bottleneck distance.*

Proof. The decomposition

$$(V, W, \Phi) \cong \bigoplus_{[i_1,j_1] \preceq [i_2,j_2]} \left(\mathbf{R}_{[i_1,j_1]}^{[i_2,j_2]}\right)^{r_{[i_1,j_1]}^{[i_2,j_2]}} \oplus \bigoplus_{i \le j} \left(\mathbf{I}^+[i_1, j_1]\right)^{d_{ij}^+} \oplus \bigoplus_{i \le j} \left(\mathbf{I}^-[i_1, j_1]\right)^{d_{ij}^-}$$

is obtained by finding a pair of barcode bases $(\mathcal{B}_V, \mathcal{B}_W)$ in which the matrix representation of Φ is in matching form. Remember, each appearance of $\mathbf{R}_{J_r}^{J_c}$ in the decomposition corresponds to a non-zero entry in the matrix M_Φ in a row r belonging to J_r and column c belonging to column J_c. Similarly, each appearance of $\mathbf{I}_J^+$ corresponds to an empty column in M_Φ belonging to bar J, and each appearance of $\mathbf{I}_J^-$ corresponds to an empty row in M_Φ belonging to bar J.

First, let us prove that the matched bars contribute a cost smaller than δ. If the bars $[i_2, j_2] \in \mathrm{Bar}(V)$ and $[i_1, j_1] \in \mathrm{Bar}(W)$ are matched, a generator of bar $x_{[i_1,i_1]}$ is in the support of the image of the generator $x_{[i_2,j_2]}$ with the δ-invertible morphism Φ. By Lemma 3.2.13 this implies that $|i_2 - i_1| \le \delta$ and $|j_2 - j_1| \le \delta$. Therefore, the cost of matching these bars is smaller or equal to δ.

All there is left to prove is that the cost the bars that are left unmatched contribute is less than δ as well. Assume $\mathbf{I}^+[i, j]$ for $[i, j] \in \mathrm{Bar}(V)$ appears as a summand in the decomposition. Since the $\operatorname{supp}\Phi(x_{[i,j]})$ is empty, Lemma 3.2.13 implies that $|j - i| < 2\delta$ and the cost of not matching it is smaller than δ. In a similar manner one can show that if $\mathbf{I}^-[i, j]$ appears as a summand in the decomposition, then the cost of not matching $[i, j] \in \mathrm{Bar}(W)$ is smaller than δ. □

Corollary 3.4.3 (of Theorem 3.2.21)**.** *Let* $\chi_\Phi \colon Bar(V) \nrightarrow Bar(W)$ *and* $\chi_\Psi \colon Bar(W) \nrightarrow Bar(V)$ *be the partial matchings induced by morphisms* (Φ, Ψ) *forming a* δ*-interleaving pair where* $\delta < \frac{1}{2}\min(\Xi(V), \Xi(W))$*. For any pair of bars* $J_V \in Bar(V)$ *and* $J_W \in Bar(W)$ *satisfying* $|J_V| \geq 2\delta$ *and* $|J_W| \geq 2\delta$*, and any* $\mu \in \mathbb{N}$*,*

$$((J_V, J_W), \mu) \in \chi_\Phi \iff ((J_W, J_V), \mu) \in \chi_\Psi.$$

Proof. By Theorem 3.2.21, $M_\Phi(r, c) = 1$ implies $M_\Phi(c, r) = 1$ for an index c corresponding to the bar J_V and r corresponding to bar J_W. Consequently the multiplicity $r_{J_W}^{J_V}$ of $\mathbf{R}_{J_W}^{J_V}$ in the ladder decomposition of Φ is smaller or equal to the multiplicity $r_{J_V}^{J_W}$ of $\mathbf{R}_{J_W}^{J_V}$ in the ladder decomposition of Ψ. By using the same arguments with the roles of Φ and Ψ reversed, we obtain that $r_{J_V}^{J_W} \leq r_{J_W}^{J_V}$. Considering the definition of the ladder decomposition induced partial matching, this concludes the proof. □

Example 3.4.4. Return to the δ-interleaving pair (Φ, Ψ) from Examples 3.2.1, 3.2.17 and 3.2.22. The ladder decompositions we obtained are

$$(V, W(\delta), \Phi) \cong \mathbf{R}_{[1,5]}^{[0,4]} \oplus \mathbf{R}_{[1,6]}^{[1,7]} \oplus \mathbf{I}_{[4,4]}^{+} \qquad \text{and} \qquad (W, V(\delta), \Psi) \cong \mathbf{R}_{[0,4]}^{[1,5]} \oplus \mathbf{R}_{[1,7]}^{[1,6]} \oplus \mathbf{I}_{[4,4]}^{-}$$

respectively. The matchings they induce are therefore

$$\chi_\Phi = \{([0,4],[1,5]),\ ([1,7],[1,6])\}$$
$$\chi_\Psi = \{([1,5],[0,4]),\ ([1,6],[1,7])\}.$$

They happen to be the opposite matchings, which is not always the case (they can differ on bars shorter than 2δ). It is easy to see that they are of cost 1, which agrees with Corollary 3.4.2. △

3.4.2 Comparisson with the Bauer-Lesnick Induced Matchings

To our knowledge the first notion of a partial matching induced by a morphism of persistence modules was introduced by Bauer and Lesnick in [10, 43]. Requiring a choice of an order on bars with the same endpoints, the construction follows three steps:

1. A (general) morphism $\Phi \colon V \to W$ is split into a surjection onto its image and inclusion into the codomain as follows:

$$V \xrightarrow{q_\Phi} \operatorname{im} \Phi \xhookrightarrow{i_\Phi} W.$$

The matchings $\chi^{BL}_{q_\Phi}$ and $\chi^{BL}_{i_\Phi}$ are defined separately and later combined into a single matching χ^{BL}_{Φ} as

$$\chi^{BL}_{\Phi} = \{(J_V, J_W) \mid \exists J_{\text{im}} \in \text{Bar}(\text{im}\Phi) \text{ s.t. } (J_V, J_{\text{im}}) \in \chi^{BL}_{q_\Phi} \text{ and } (J_{\text{im}}, J_W) \in \chi^{BL}_{i_\Phi}\}.$$

2. The matching of the injection i_Φ is constructed for each family $\langle \cdot, d\rangle$ of bars with the second endpoint d individually. First, both $\langle \cdot, d\rangle_{\text{Bar}(\text{im}\Phi)}$ and $\langle \cdot, d\rangle_{\text{Bar}(W)}$ are ordered by the length decreasingly, combining it with the chosen order on bars of the same length. Then the n-th bar in $\langle \cdot, d\rangle_{\text{Bar}(\text{im}\Phi)}$ gets matched with the n-th bar in $\langle \cdot, d\rangle_{\text{Bar}(W)}$. If the cardinalities differ, the residual bars are left unmatched. The matchings of families $\langle \cdot, d\rangle$ for all possible endpoints d are combined into a matching $\chi^{BL}_{i_\Phi}$.
3. The matching of the surjection q_Φ is constructed for each family $\langle b, \cdot\rangle$ of bars with the first endpoint b individually. As before, families $\langle b, \cdot\rangle_{\text{Bar}(\text{im}\Phi)}$ and $\langle b, \cdot\rangle_{\text{Bar}(W)}$ are ordered as before and bars get matched based on their position in the order. The matchings of families $\langle b, \cdot\rangle$ for all possible endpoints b are combined into a matching $\chi^{BL}_{q_\Phi}$.

As noted by the authors, the construction is determined by the barcodes $\text{Bar}(V)$, $\text{Bar}(W)$ and $\text{Bar}(\text{im}\Phi)$. This means that the only way the morphism influences the construction is not through its image, but through the barcode $\text{Bar}(\text{im}\Phi)$. More explicitly, as long as the barcodes of the image of two parallel morphisms are the same, the induced matchings will coincide. Let us illustrate this with an example.

Example 3.4.5. Let V and W be the following persistence modules:

$$V: \quad 0 \longrightarrow \mathbb{F}^2 \xrightarrow{\text{Id}} \mathbb{F}^2 \xrightarrow{\text{Id}} \mathbb{F}^2 \xrightarrow{(1\ 0)} \mathbb{F} \longrightarrow 0,$$

$$W: \quad 0 \longrightarrow \mathbb{F} \xrightarrow{\binom{1}{0}} \mathbb{F}^2 \xrightarrow{\text{Id}} \mathbb{F}^2 \xrightarrow{\text{Id}} \mathbb{F}^2 \longrightarrow 0.$$

Notice that we assume a specific choice of barcode bases in which we have written the transition maps. The bottleneck and interleaving distances between these modules are both 1 and there is an obvious choice for a 1-invertible morphism, namely

$$\begin{array}{ccccccccccccc}
V: & & 0 & \longrightarrow & \mathbb{F}^2 & \xrightarrow{\mathrm{Id}} & \mathbb{F}^2 & \xrightarrow{\mathrm{Id}} & \mathbb{F}^2 & \xrightarrow{(1\ 0)} & \mathbb{F} & \longrightarrow & 0 \\
\downarrow{\scriptstyle \Phi} & & \downarrow{\scriptstyle 0} & & \downarrow{\scriptstyle \mathrm{Id}} & & \downarrow{\scriptstyle \mathrm{Id}} & & \downarrow{\scriptstyle \mathrm{Id}} & & \downarrow{\scriptstyle 0} & & \\
W(\delta=1): & & 0 & \longrightarrow & \mathbb{F} & \xrightarrow[\binom{1}{0}]{} & \mathbb{F}^2 & \xrightarrow[\mathrm{Id}]{} & \mathbb{F}^2 & \xrightarrow[\mathrm{Id}]{} & \mathbb{F}^2 & \longrightarrow & 0.
\end{array}$$

The 1-invertible morphism making an interleaving pair with Φ is defined to be the identity when possible, which also determines the other components through the commuting squares. However, this is not the only choice of an interleaving pair. Alternatively, we could define a morphism Ψ as

$$\begin{array}{ccccccccccccc}
V: & & 0 & \longrightarrow & \mathbb{F}^2 & \xrightarrow{\mathrm{Id}} & \mathbb{F}^2 & \xrightarrow{\mathrm{Id}} & \mathbb{F}^2 & \xrightarrow{(1\ 0)} & \mathbb{F} & \longrightarrow & 0 \\
\downarrow{\scriptstyle \Psi} & & \downarrow{\scriptstyle 0} & & \downarrow{\scriptstyle \left(\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right)} & & \downarrow{\scriptstyle \left(\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right)} & & \downarrow{\scriptstyle \left(\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right)} & & \downarrow{\scriptstyle 0} & & \\
W(\delta=1): & & 0 & \longrightarrow & \mathbb{F} & \xrightarrow[\binom{1}{0}]{} & \mathbb{F}^2 & \xrightarrow[\mathrm{Id}]{} & \mathbb{F}^2 & \xrightarrow[\mathrm{Id}]{} & \mathbb{F}^2 & \longrightarrow & 0,
\end{array}$$

and the morphism making its interleaving pair as the exchange matrix whenever possible, which again defines the other components through commuting squares. No matter which definition we choose, the barcode of the image is $\{[0,2],[0,2]\}$ in both cases. The BL-induced matchings of the two morphisms, computed as

$$\chi^{BL}_{i_\Phi} = \left\{ \begin{array}{ccc} \langle \cdot, 2\rangle_{\mathrm{Bar(im)}} & & \langle \cdot, 2\rangle_{\mathrm{Bar}(W(\delta=1))} \\ [0,2] & \mapsto & [-1,2] \\ [0,2] & \mapsto & [0,2] \end{array} \right\} = \left\{ \begin{array}{ccc} \langle \cdot, 2\rangle_{\mathrm{Bar(im)}} & & \langle \cdot, 3\rangle_{\mathrm{Bar}(W)} \\ [0,2] & \mapsto & [0,3] \\ [0,2] & \mapsto & [1,3] \end{array} \right\} = \chi^{BL}_{i_\Psi},$$

$$\chi^{BL}_{i_\Phi} = \left\{ \begin{array}{ccc} \langle 0, \cdot\rangle_{\mathrm{Bar(im)}} & & \langle 0, \cdot\rangle_{\mathrm{Bar}(V)} \\ [0,2] & \mapsto & [0,3] \\ [0,2] & \mapsto & [0,2] \end{array} \right\} = \chi^{BL}_{i_\Psi},$$

$$\chi^{BL}_{\Phi} = \{([0,3],[0,3]),([0,2],[1,3])\} = \chi^{BL}_{\Psi},$$

are therefore the same and do not respect the mapping of the morphism fully.

This is not true for the matchings induced by the ladder decompositions of the interleavings.

Notice that in the chosen barcode bases Φ and Ψ are already in matching form, namely

$$\Phi = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \text{and} \qquad \Psi = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

The matchings induced by their ladder decompositions are

$$\chi_\Phi = \{([0,3],[0,3]),([0,2],[1,3])\} \text{ and}$$
$$\chi_\Psi = \{([0,3],[1,3]),([0,2],[0,3])\},$$

which are clearly different. Despite a δ-interleaving being used in this example, the difference in the two definitions of induced matchings can be observed for a general morphism of persistence modules. △

3.4.3 Basis-Independent Partial Matchings

As in this chapter, Gonzalez Diaz and Soriano Trigueros in [13] adopt the view of morphisms as ladder persistence modules. They define a different notion of a partial matching, called **basis-independent partial matching**, which is independent of the choice of order on the barcode (hence basis-independent).

Definition 3.4.6. A **basis-independent partial matching** between persistence modules V and W, indexed over posets P_V and P_W respectively, is a function

$$\mathcal{M}_V^W : \overline{P_V} \times \overline{P_W} \to \mathbb{Z}_{\geq 0},$$

where $\overline{P_V}$ and $\overline{P_W}$ are the sets of intervals in P_V and P_W respectively. Further, it must satisfy

$$\sum_{1\leq d\leq n} \sum_{1\leq c\leq d} \mathcal{M}_V^W(a,b,c,d) \leq \mu^V([a,b]) \text{ and}$$
$$\sum_{1\leq b\leq n} \sum_{1\leq a\leq b} \mathcal{M}_V^W(a,b,c,d) \leq \mu^W([c,d]),$$

where $\mu^V([a,b])$ is the multiplicity of bar $[a,b]$ in $\mathrm{Bar}(V)$ and $\mu^W([c,d])$ is the multiplicity of bar $[c,d]$ in $\mathrm{Bar}(W)$.

The ladder decomposition induced partial matchings of [11], and therefore the ones we study in this chapter, are examples of basis-independent partial matchings. To see this, define $\mathcal{M}_V^W$ for a

morphisms Φ as

$$\mathcal{M}_V^W(a,b,c,d) = r_{[c,d]}^{[a,b]},$$

where $r_{[c,d]}^{[a,b]}$ is the multiplicity of $R_{[c,d]}^{[a,b]}$ appearing in the ladder decomposition of Φ. Consequently, $\mathcal{M}_V^W(a,b,c,d)$ is the multiplicity of the pair $([a,b],[c,d])$ in the ladder decomposition induced partial matching χ_Φ. It is rather obvious that the sum $\sum_J r_{[c,d]}^J$ is not bigger than the multiplicity of $[c,d]$ in $\mathrm{Bar}(W)$ and the sum $\sum_J r_J^{[a,b]}$ is not bigger than the multiplicity of $[a,b]$ in $\mathrm{Bar}(V)$.

3.4.4 q-Coarse Induced Partial Matchings

Whenever the interleaving parameter δ is too big to apply Corollary 3.4.2, we might still leverage the results of Section 3.3 to define partial matchings of potentially higher cost. The following result is obtained by combining Theorem 3.3.6 and Corollaries 3.3.7, 3.4.1 and 3.4.3.

Corollary 3.4.7. *Let (Φ,Ψ) be a δ-interleaving pair between modules V and W, and suppose there exists a parameter q such that*

$$\delta < \frac{1}{2}\min(\Xi(V_{\geq q}),\Xi(W_{\geq q})) - \frac{q}{2}.$$

Then the $(\delta+\frac{q}{2})$-interleaving pair (Φ',Ψ') it induces by Theorem 3.3.6 (1), (2) or (3) further induces a partial matching

$$\chi_{\Phi'} : \mathit{Bar}(V) \nrightarrow \mathit{Bar}(W)$$
$$\chi_{\Phi'} = \{((J_1,J_2), r_{J_1}^{J_2}) \mid \mathbf{R}_{J_1}^{J_2} \text{ appears in ladder decomposition of } \Phi'\}$$

which leaves bars in $\mathit{Bar}(V_{<q})$ and $\mathit{Bar}(W_{<q})$ unmatched. It is of cost smaller or equal to $\delta+\frac{q}{2}$. By Corollary 3.3.7 for any pair of bars $J_V \in \mathit{Bar}(V)$ and $J_W \in \mathit{Bar}(W)$ with $|J_V| \geq 2\delta+q$ and $|J_W| \geq 2\delta+q$

$$((J_V,J_W),\mu) \in \chi_{\Phi'} \iff ((J_W,J_V),\mu) \in \chi_{\Psi'},$$

where μ is the multiplicity of this pairing in both $\chi_{\Phi'}$ and $\chi_{\Psi'}$.

Chapter 4

Initialization Strategy for Deep Neural Networks with ReLU Activation

Neural networks are essentially families of functions, characterized by parameters known as weights and biases. They are large and flexible families, which enables them to approximate solutions to many problems. By continuously changing the parameters based on the performance of the current function on data, increasingly accurate approximations can be found. Not surprisingly, the success of this optimization process, called training, depends heavily on how suitable the initial values of the parameters are. The heuristic we follow in setting them is called an initialization, and studies of different initialization strategies (tailored to specific tasks or network architectures) is an integral part of machine learning research. As a result, a plethora of methods have been developed and made available. In practice, neural networks are usually initialized randomly, with only mean and variance of the layer outputs controlled to prevent them from getting too large or too small for successful training. Due to the fact that the probability of a unit in the network being inactive (meaning the activation is zero for all data) is highly dependent on the geometry of the data, some advocate for the use of empirical approaches [96] which learn the "best" initialization for the task at hand.

For networks with piecewise linear activation, such as ReLU, linear regions play an important theoretic role in their understanding. Most prior studies into linear regions of ReLU networks focus on their maximal or expected number [97, 98, 99, 100, 101] which are interesting due to relative ease of computation and their relation to network expressivity. While their number can grow exponentially with the number of layers [98], it is usually proportional to the total number of neurons [101]. Since networks with higher number and more even spread of linear regions are believed to be able to approximate a richer class of functions [97, 98], it may be beneficial to maximize their number.

In this work, we devise our own initialization strategy with that exact aim, and monitor how the number of regions changes during training. Our experiments reveal that whether or not there are benefits to maximizing the number of linear regions at initialization can vary significantly based on the variance control heuristic and training configuration (where the choice of network optimizer is of particular importance). In addition, they shine a light on a few phenomena which are not yet well-understood within the machine learning community.

The chapter is organized as follows. In Section 4.1, we give the necessary prior knowledge about the use of neural networks on the specific network we focus on in this work, namely a deep neural network with ReLU activation. The details of our initialization strategy are given in Section 4.2, while a report on the experiments comparing it to other strategies follows in Section 4.3. Our observations are summarized in Section 4.4.

4.1 Neural Network Preliminaries

In this work we focus entirely on deep neural networks with fully connected layers and ReLU activation. We provide preliminary details about them here, and also write about the pre-deployment stages of their use, namely initialization, training and testing, as it relates to our subsequent work.

4.1.1 Network Architecture

Let us represent a **deep neural network** by the diagram

$$f\colon \overbrace{\underbrace{\mathbb{R}^{n_0} \xrightarrow{f^{(1)}} \mathbb{R}^{n_1}}_{=:f^{[1]}} \xrightarrow{g^{(1)}} \mathbb{R}^{n_2} \xrightarrow{f^{(2)}} \dots \xrightarrow{g^{(h-2)}} \mathbb{R}^{n_{h-2}} \xrightarrow{f^{(h-1)}} \mathbb{R}^{n_{h-1}}}^{f^{[h-1]}} \xrightarrow{g^{(h-1)}} \mathbb{R}^{n_{h-1}} \xrightarrow{f^{(h)}} \mathbb{R}^{n_h} \xrightarrow{g^{(h)}} [0,1]^{n_h}.$$

Each $f^{(\ell)}$ denotes an affine function $f^{(\ell)} = W^{(\ell)} \cdot x + b^{(\ell)}$, where $W^{(\ell)} = (w_{i,j}^{(\ell)}) \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ is a suitable **weight matrix** and $b^{(\ell)} \in \mathbb{R}^{n_\ell}$ is a **bias**. The composition $g^{(\ell)} \circ f^{(\ell)}$ is called a **layer**, and $g^{(\ell)}$ is an **activation function**. In our setting, $g^{(\ell)}$ for $\ell < h$ is **rectified linear unit (ReLU)** activation function, which is just coordinate-wise maximum $x_i \mapsto \max(x_i, 0)$. The last activation function, $g^{(n_h)} \colon \mathbb{R}^{n_h} \to [0,1]^{n_h}$, is a **LogSoftmax function**, defined coordinate-wise as

$$g^{(n_h)}(z)_i = \ln\left(\frac{e^{z_i}}{\sum_{j=1}^{n_h} e^{z_j}}\right)$$

for $z = (z_1, \ldots, z_{n_h}) \in \mathbb{R}^{n_h}$. It transforms the output of the last affine function to a vector of log probabilities for n_h possible outcomes. In some contexts, it is necessary to look at each coordinate of $\mathbb{R}^{n_\ell}$ separately. In that case, each composition $\pi_i \circ g^{(\ell)} \circ f^{(\ell)}$, where $\pi_i \colon \mathbb{R}^{n_\ell} \to \mathbb{R}^{n_\ell}$ is the projection onto the i-th coordinate in the codomain, is called a **unit** of layer ℓ. We denote the partial compositions by $f^{[\ell]} \coloneqq f^{(\ell)} \circ g^{(\ell-1)} \circ f^{(\ell-1)} \circ \cdots \circ g^{(1)} \circ f^{(1)}$ as indicated above, so that $f = g^{(h)} \circ f^{[h]}$.

4.1.2 Pipeline

A neural network is first initialized with some values for weights and biases of every layer. These values are then optimized based on the objective given by a loss function, and lastly tested on previously unseen data. We explain the necessary details of each of these steps here.

Let us postpone discussing the topic of initialization, and assume the network parameters take random values for now. Each network is trained for a specific task, and here, we focus on **classification**. This means that the network is supposed to assign each point in the data set to one of n_h pre-defined classes. We require a labeled data set X, which means that for each point $x \in X$ in the data it is also known which class it belongs to, as given by a map $\mathcal{L} \colon X \to [n_h]$ into the set of labels.

It is crucial to exclude a portion of this data set from training, as the generalization capacity of the trained network needs to be assessed on previously unseen data. Thus, X is split into two parts: the **training set** X_{train} and **testing set** X_{test}. A typical recommendation is to allocate $80 - 90\%$ of the data to X_{train} and $10 - 20\%$ to X_{test}. The network is then trained exclusively on the training set in an iterative process involving the following steps.

- The layers of the neural network are applied to the data in what is called a **forward pass** to obtain a prediction. For point x, the prediction $y = f(x)$ is a vector of log probabilities. Namely, its j-th coordinate, y_j, is the natural logarithm of predicted probability that x belongs to class $j \in [n_h]$, and thus $\sum_{j\in[n_h]} e^{y_j} = 1$.
- Denote by $\alpha(x)$ the one-hot vector encoding of the label $\mathcal{L}(x)$, that is $\alpha(x)_j = 1$ if $\mathcal{L}(x) = j$ and 0 otherwise. In the next step, the predictions are compared to the target values $\alpha(x)$ via a **loss function**. The most common loss function in classification tasks (and also the one we use later on) is the **(multiclass) cross-entropy loss**. It is defined as

$$c(\alpha(x), y) = -\sum_{j=1}^{n_h} \alpha(x)_j \log(y_j). \tag{4.1}$$

  To compute it for a batch $B \subseteq X_{\text{train}}$, it is custom to take the average,

$$c(B) = \frac{1}{|B|} \sum_{x\in B} c(\alpha(x), f(x)).$$

- Notice that, implicitly, the cost function depends on all weights and biases in the network. Partial derivatives of the loss function with respect to all of the parameters are used to compute their updates. Very generally, the update at step t of a parameter p is computed following

$$p^{(t)} = p^{(t-1)} + \eta \cdot \varphi(\nabla_p c^{(t)}, \nabla_p c^{(t-1)}, \ldots, \nabla_p c^{(0)}),$$

  where η is a parameter called the **learning rate**, and controls the step size for the update, and φ is the method-dependent function which depends on partial derivatives of the loss function with respect to the parameter being updated at t and all the previous steps. Note that the loss function is not differentiable when using ReLU activation. It is, however, piecewise differentiable and most methods choose the appropriate derivative at each step in an efficient way. Due to a dependence of the derivatives in the earlier layers on the derivatives in the later layers, the updates are computed in the "backwards" direction. This step is accordingly called **backpropagation**. The specific approach to the computation of parameter updates depends on the choice of the optimizer. In our work, we use **stochastic gradient descent (SGD)** [17, Chap. 8.1.3] and **adaptive moment estimation (Adam)** [18] with the default parameters of the Pytorch implementation.

Training is rarely performed on the whole training data set simultaneously due to high memory demands, especially on large data sets. Instead, the set X_{train} is usually partitioned into **batches** $B_1, \ldots, B_b$. This approach can also accelerate computation and introduce randomness, which can be beneficial in helping the model escape local minima. Similarly, a single pass of all the batches through the network, called an **epoch**, might not be sufficient for the model to learn the intricacies of the task. It is beneficial to iterate over the data set multiple times, allowing the model to refine the parameters with every epoch. The choice of batch size (and therefore the number of batches b) and the number of epochs should be given whenever describing a training process.

A particular issue that can arise during training are **vanishing** or **exploding gradients**. This is when the partial derivatives become very small or very large, respectively, as they are back-propagated through the network. The first leads to slow updates and consequently stalls network training, while the second can cause the training to be unstable and to diverge. Among the approaches for preventing these issues, it is recommended to use ReLU activation, optimizers with adaptive learning rate (such as Adam) which can amplify small and prevent excessive updates by choosing the learning rate accordingly, and to choose an appropriate strategy for initialization of network parameters.

The choice of initialization heuristic is known to have significant impact on subsequent training of the network in question, with a good initialization leading to faster convergence and better accuracy. Most commonly used initialization techniques, for example He [102], Xavier [103], and LeCun [104] initializations (which is the default PyTorch initialization for fully connected layers), sample network parameters randomly from uniform or normal distribution with mean and variance controlled per layer to prevent the gradients from vanishing or exploding. To be more precise, the guiding principle of all these techniques is that one can avoid vanishing and exploding gradients by scaling the weights of a layer so that its input and output have the same variance. When the input data set is normalized, the variance of the output of every layer should thus be close to 1. This is a principle we also adopt in the design of our custom initialization method. In addition to the latter, we use LeCun initialization, in which the entries of $W^{(\ell)}$ and $b^{(\ell)}$ are drawn randomly from the uniform distribution $\mathcal{U}(-\frac{1}{\sqrt{n_{\ell-1}}}, \frac{1}{\sqrt{n_{\ell-1}}})$.

Finally, let us address how the success of the training is assessed. Typically, several quantitative metrics are monitored throughout the training process, and their final values are compared against the same metrics applied to the test dataset, X_{test}. The selection of metrics varies based

on the specific task the network is trained for, and the most common metrics for classification tasks are loss and accuracy. Let y again denote the prediction of the network on input x, and let $y_{i(x)} = \max(y)$ be the largest coordinate of the prediction, implying that x belongs to class $i(x)$. Then, the **accuracy** of the network on a subset $S \subseteq X$ is defined as

$$\operatorname{acc}(S) = \frac{|\{x \in S \mid i(x) = \mathcal{L}(x)\}|}{|S|}.$$

In other words, it is the percentage of correct predictions the network makes on the given set of inputs. To monitor accuracy and loss during the training process, we plot their values every s steps in the training (where a step here means a pass of one batch). We postpone giving further details until Section 4.3.

4.1.3 Activation Regions

The notion of activation regions is central to our work in this chapter. We provide the necessary prior knowledge related to them here.

An **activation pattern** is a binary vector $s = (s_i^{(\ell)})_{\ell=1,\dots,h, i=1,\dots,n_\ell} \in \{\pm 1\}^{[n_1;n_2;\dots;n_h]}$, where $[n_1; n_2; \dots; n_h] = \{(\ell, i) \mid \ell = 1, \dots, h, i = 1, \dots, n_\ell\}$. They can be used to describe the network's "train of thought": the activation pattern at a point $x \in \mathbb{R}^{n_0}$ is

$$s(x) \coloneqq \left(s_i^{(\ell)}(x)\right)_{\substack{\ell=1,\dots,h,\\ i=1,\dots,n_\ell}} \quad \text{where} \quad s_i^{(\ell)}(x) \coloneqq \begin{cases} +1 & \text{if } f_i^{[\ell]}(x) > 0, \\ -1 & \text{if } f_i^{[\ell]}(x) \leq 0. \end{cases}$$

For ReLU, if $s_i^{(\ell)} = -1$ the activation nulls the result of $f_i^{(\ell)}$, and if $s_i^{(\ell)} = 1$ the result passes through unchanged. The activation pattern at x therefore collects the network's "decision" at each unit. The set of points for which $f_i^{[\ell]} = 0$ is accordingly called a **decision boundary**.

Each activation pattern $s \in \{0, 1\}^{[n_1;n_2;\dots;n_h]}$ has an associated **activation region**

$$A(s) \coloneqq \overline{\left\{x \in \mathbb{R}^{n_0} \mid s = s(x)\right\}},$$

where $\overline{(\cdot)}$ denotes the euclidean closure. Thus, each point $x \in \mathbb{R}^{n_0}$ belongs to an activation region $A(x) \coloneqq A(s(x))$. If $x \in \operatorname{Int}(A(x))$, we say x is **generic**. For each layer, the Jacobian of $f^{[\ell]}$ is well-defined and constant in the interior of the maximal activation regions. We denote it by $J_{f^{[\ell]}}(s)$.

Remark 4.1.1. On specific subsets of the input space $\mathbb{R}^{n_0}$, the network f behaves as a linear function. Each of these (maximal) subsets is referred to as a **linear region**. This concept is often mistaken for the notion of an activation region, likely because they coincide under the assumption of genericity for the types of networks where these concepts are typically analyzed, namely fully connected deep neural networks with all activation functions being piecewise linear. Example 4.1.2 illustrates that these notions can differ in special cases. In our case, there is an additional reason for distinction due to the use of (non-linear) LogSoftmax as the last activation function. However, we rarely consider the units of the last layer and shamelessly use the two notions interchangeably.

Example 4.1.2. Define a network $\mathbb{R} \xrightarrow{f^{(1)}} \mathbb{R}^2 \xrightarrow{g^{(1)}} \mathbb{R}^2 \xrightarrow{f^{(2)}} \mathbb{R}$ where $f^{(1)}(x) = (x, -x)$, $f^{(2)}(x, y) = x - y$ and $g^{(1)}$ is ReLU activation. Such a network maps any input $x \in \mathbb{R}$ to

$$\begin{aligned} f^{(2)} \circ g^{(1)} \circ f^{(1)}(x) &= f^{(2)} \circ g^{(1)}(x, -x) \\ &= f^{(2)}\big(\max(x, 0), \max(-x, 0)\big) \\ &= \max(x, 0) - \max(-x, 0) \\ &= x. \end{aligned}$$

Thus, the whole $\mathbb{R}$ is one linear region, while there are two activation regions belonging to activation patterns $(1, -1)$ and $(-1, 1)$. △

The following lemma (which appears originally in [105], but we restate it here in our notation) gives assurance that these activation regions are "nice". To be precise, they are either empty or full-dimensional closed convex polyhedra, and their set covers the entire input space $\mathbb{R}^{n_0}$.

Lemma 4.1.3. *Fix an activation pattern $s = s(x_0)$ with $x_0 \in int(A(s))$. Any activation index $(\ell, i) \in [n_1; n_2; \ldots; n_h]$ defines a closed affine half-space*

$$\begin{aligned} F_s(\ell, i) &\coloneqq \Big\{ y \in \mathbb{R}^{n_0} \mid s_{\ell,i} \cdot \big[(J_{f^{[\ell]}}(x_0))_i \cdot (y - x_0) + f_i^{[\ell]}(x_0) \big] \geq 0 \Big\} \\ &= \Big\{ y \in \mathbb{R}^{n_0} \mid s_{\ell,i} \cdot (J_{f^{[\ell]}}(x_0))_i \cdot y \geq s_{\ell,i} \cdot \big[(J_{f^{[\ell]}}(x_0))_i \cdot x_0 - f_i^{[\ell]}(x_0) \big] \Big\}. \end{aligned}$$

Their intersection for all units in layer ℓ is a convex polyhedron

$$\begin{aligned} F_s(\ell) &\coloneqq \bigcap_{i=1}^{n_\ell} F_s(\ell, i) \\ &\coloneqq \left\{ y \in \mathbb{R}^{n_0} \mid D_s^\ell J_{f^{[\ell]}}(x_0) \cdot y \geq D_s^\ell \big[J_{f^{[\ell]}}(x_0) \cdot x_0 - f^{[\ell]}(x_0) \big] \right\} \end{aligned}$$

where D_s^ℓ is a diagonal matrix with entries $(D_s^\ell)_{ii} = s_{\ell,i}$. Further, the activation region $A(s)$ is the convex polyhedron $\bigcap_{\ell=1}^{h} \bigcap_{i=1}^{n_\ell} F_s(\ell, i) = \bigcap_{\ell=1}^{h} F_s(\ell)$.

Proof. By induction on the layer depth $\ell \in [h]$, we show that $y \in \bigcap_{j=1}^{\ell} \bigcap_{i=1}^{n_j} F_s(j, i)$ holds if and only if $s_{j,i}(y) = s_{j,i}$ for all $j \in [\ell]$ and $i \in [n_j]$.

Begin with the first layer. For $i \in [n_1]$, we have that $(J_{f^{[1]}})_i = (J_{f^{(1)}})_i$ is simply $W_{i:}^{(1)}$, the i-th row of $W^{(1)}$. Further, $f_i^{(1)}(x_0)$ can be written as $W_{i:}^{(1)} x_0 + b_i^{(1)}$. Then for any $y \in \mathbb{R}^{n_0}$

$$\begin{aligned} f_i^{(1)}(y) &= W_{i:}^{(1)}(y) + b_i^{(1)} \\ &= W_{i:}^{(1)} \cdot y + f_i^{(1)}(x_0) - W_{i:}^{(1)} \cdot x_0 \\ &= (J_{f^{[1]}})_i \cdot y + f_i^{[1]}(x_0) - (J_{f^{[1]}})_i \cdot x_0 \end{aligned}$$

and $s_{1,i}(y) = s_{1,i}$ holds if and only if

$$s_{1,i} \cdot \big[(J_{f^{[1]}})_i \cdot y + f_i^{[1]}(x_0) - (J_{f^{[1]}})_i \cdot x_0 \big] \geq 0.$$

Now, assume $y \in \mathbb{R}^{n_0}$ lies in $F_s(j)$ for all layers $j \in [\ell - 1]$. Then

$$(g^{(\ell-1)} \circ f^{[\ell-1]})_i = \max(0, s_{\ell-1,i}) \cdot f_i^{[\ell-1]}$$

for all $i \in [n_{\ell-1}]$, which is a fixed linear function on $\bigcap_{j=1}^{\ell-1} F_s(j)$. As a consequence, the composition $f_i^{[\ell]} = (f^{(\ell)} \circ g^{(\ell-1)} \circ f^{[\ell-1]})_i$ is also linear on $\bigcap_{j=1}^{\ell-1} F_s(j)$, where its Taylor expansion around x_0 gives

$$\begin{aligned} f_i^{[\ell]}(y) &= (J_{f^{[\ell]}}(x_0))_i \cdot (y - x_0) + f_i^{[\ell]}(x_0) \\ &= (J_{f^{[\ell]}}(x_0))_i \cdot y - (J_{f^{[\ell]}}(x_0))_i \cdot x_0 + f_i^{[\ell]}(x_0). \end{aligned}$$

The activation patterns $s_{\ell,i}(y)$ and $s_{\ell,i}$ are equal if and only if

$$s_{\ell,i} \cdot \left[(J_{f^{[\ell]}}(x_0))_i \cdot y - (J_{f^{[\ell]}}(x_0))_i \cdot x_0 + f_i^{[\ell]}(x_0) \right] \geq 0,$$

which concludes the proof. □

Remark 4.1.4. Note that the activation regions remain unchanged under scaling of weights and biases. This is true when scaling is **isotropic** and **anisotropic**, first meaning that the scaling factor is the same for each unit within a layer, and the second that it is not. To see this, suppose we scale the unit i in layer ℓ by a factor $a > 0$, which means $(a \cdot f_i^{(\ell)}) \circ g^{(\ell)} \circ f^{[\ell-1]} = a \cdot f_i^{[\ell]}$. The half-space $F_s(\ell, i)$ we obtain after scaling is

$$\begin{aligned}
&\left\{ y \in \mathbb{R}^{n_0} \mid s_{\ell,i} \cdot \left[a \cdot (J_{f^{[\ell]}}(x_0))_i \cdot (y - x_0) + a \cdot f_i^{[\ell]}(x_0) \right] \geq 0 \right\} \\
=&\left\{ y \in \mathbb{R}^{n_0} \mid a \cdot s_{\ell,i} \left[(J_{f^{[\ell]}}(x_0))_i \cdot (y - x_0) + \cdot f_i^{[\ell]}(x_0) \right] \geq 0 \right\} \\
=&\left\{ y \in \mathbb{R}^{n_0} \mid s_{\ell,i} \cdot \left[(J_{f^{[\ell]}}(x_0))_i \cdot (y - x_0) + \cdot f_i^{[\ell]}(x_0) \right] \geq 0 \right\},
\end{aligned}$$

which is the same as before scaling. This also implies that the activation region $A(s)$, which is the intersection $\bigcap_{\ell=1}^{h} \bigcap_{i=1}^{n_\ell} F_s(\ell, i)$, is unchanged.

4.2 Initialization Strategy

The aim of our work was to develop and test an initialization strategy which optimizes the number and distribution of activation regions. We chose the parameters for each unit of each layer (in an increasing order) separately, aiming to split multi-labeled regions at each step. We decide which region to split based on a custom-defined measure, called the **cross-entropy region cost** c_{reg}. Given a region R in which a subset $S_R = X_{\text{train}} \cap R$ of the training data lies, its cost is defined as

$$c_{\text{reg}}(R) := \sum_{x \in S_R} c(\alpha(x), v_{\%}), \tag{4.2}$$

where $v_{\%} \in \mathbb{R}^{n_h}$ is a vector with the percentage of points in S_R labeled with i as the i-th coordinate. Notice that when all points in S_R belong to the same class, $c_{\text{reg}}(R) = 0$.

Our strategy has been implemented in `Python` (using the machine learning library `PyTorch`). For pseudo-code of the main method, `reinitialize_network`, and the methods it relies on, see Algorithms 1 to 6. Its steps are roughly the following.

1. Initialize the network following the LeCun initialization heuristic, *i.e.* draw entries of $W^{(\ell)}$ and $b^{(\ell)}$ randomly from $\mathcal{U}(-\frac{1}{\sqrt{n_{\ell-1}}}, \frac{1}{\sqrt{n_{\ell-1}}})$.
2. Loop through the layers $\ell = 1, \ldots, h$ and the units $u = 1, \ldots, n_\ell$, and reinitialize. Throughout the reinitialization, collect the information about which point belongs to which region (within table $\mathcal{R}$), and what is each region's cost (within list $\mathcal{C}$). These are initialized with values signaling the existence of one region containing all training data and of appropriate cost. Further, we keep track of the images of the training set X_{train} with $g^{(\ell)} \circ f^{[\ell]}$ and denote it by $X^{(\ell)}_{\text{train}}$. The specific steps taken during the iteration depend on which stage the algorithm is in.
   - **Stage 1:** First, it is determined which region R to split with the activation of the unit u, which can be identified easily from $\mathcal{C}$. The set of points $S_R = X_{\text{train}} \cap R$ that lie within R is read from $\mathcal{R}$. Let $w^{(\ell)}_{u:}$ be the u-th row in the weight matrix $W^{(\ell)}$, which contains all the entries contributing to the value of unit u. Further, let $b^{(\ell)}_u$ be the u-th coordinate of the bias for that layer. Then, the unit u maps an input x to

     $$\max(w^{(\ell)}_{u:} \cdot x + b^{(\ell)}_u, 0) = \max(w^{(\ell)}_{u:} \cdot x, -b^{(\ell)}_u) + b^{(\ell)}_u,$$

     where $\cdot$ denotes the scalar product. Notice that if we wish to split the region R in half, it is enough to change the bias $b^{(\ell)}_u$. Indeed, one can consider the set $P = \{w^{(\ell)}_{u:} \cdot x \mid x \in S_R\}$ and choose such $b^{(\ell)}_u$ that $\lceil \frac{|P|}{2} \rceil$ of the points in P are bigger than $-b^{(\ell)}_u$ and $\lfloor \frac{|P|}{2} \rfloor$ are smaller. Thus, a new value for the bias is set, and $\mathcal{R}$ and $\mathcal{C}$ are updated. If u is the last unit in its layer, an additional step in which mean and variance of the weights are controlled is applied. Here we use two different scaling methods, which are detailed in Section 4.2.1. The choice between the two is controlled with the boolean parameter *keepVariance*. Lastly, the output of the previous layer, $X^{(\ell-1)}_{\text{train}}$ (or the training data set X_{train} if $\ell = 1$), is passed through layer ℓ and $X^{(\ell)}_{\text{train}}$ is computed.
   - **Stage 2:** Once the cost of all regions is 0 (meaning each region contains points belonging to the same class), no further network parameters are changed. Nevertheless, values of $\mathcal{R}$ and $\mathcal{C}$ are updated.

Algorithm 1 (`reinitialize_network` method). Custom network initialization.

Input: *network*, X (data), Y (labels)
Flags: *keepVariance* (controls which scaling is used, see Section 4.2.1)
Output: $\mathcal{C}$, list containing the cost of each region

initialize $\mathcal{R}$, $\mathcal{C}$
if *keepVariance* **then** $\Sigma = (\sigma_1, \ldots, \sigma_h) \leftarrow$ `layerwise_deviation`(*network*, X)
    ▷ $\sigma_\ell \in \mathbb{R}^{n_\ell}$ is a coordinate-wise standard deviation of $g^{(\ell)} \circ f^{[\ell]}(X)$
else $\Sigma \leftarrow$ ([ ] for each *layer* in *network*)
for *layer* **in** *network* **do**
    X, $\mathcal{R}$, $\mathcal{C} \leftarrow$ `reinitialize_relu_layer`(*layer*, X, Y, $\mathcal{R}$, $\mathcal{C}$, σ_{layer}, *keepVariance*)
return $\mathcal{C}$

Algorithm 2 (`reinitialize_relu_layer` method). Custom layer initialization.

Input: *layer*, X (data), Y (labels), $\mathcal{R}$ (table containing the activation pattern and region of each data point), $\mathcal{C}$ (list containing the cost of each region), $\hat{\sigma}$ (target standard deviation)
Flags: *keepVariance* (controls which scaling is used, see Section 4.2.1)
Output: X, $\mathcal{R}$, $\mathcal{C}$ after being passed through *layer*

$c \leftarrow$ highest c_{reg} cost of regions in $\mathcal{R}$ ▷ for definition of c_{reg}, see Equation (4.2)
stage $\leftarrow$ determine the stage
for $u = 0, 1, \ldots, n_{layer}$ **do** ▷ Loop through units in layer
    if *stage* $= 1$ **then**
        $w_{u:}^{(layer)}, b_u^{(layer)} \leftarrow$ parameters of unit u in *layer*
        $X_R \leftarrow$ points in region R, which has the highest c_{reg} cost in $\mathcal{R}$
        $b_u^{(layer)} \leftarrow$ `bias_update`$(X_R, w_{u:}^{(layer)})$
        update $\mathcal{R}$, $\mathcal{C}$
        if $c_{\text{reg}}(R) = 0$ **for** $\forall R \in \mathcal{R}$ **then**
            stage $\leftarrow 2$
    else if *stage* $= 2$ **then**
        update $\mathcal{R}$, $\mathcal{C}$
$Xtemp \leftarrow$ pass X through the reinitialized *layer*
if *keepVariance* **then**
    `reset_layer_deviation`(*layer*, $Xtemp$, $\hat{\sigma}$) ▷ Scale parameters to control variance
else
    `fix_layer_deviation`(*layer*, $Xtemp$) ▷ Scale parameters to control variance
$X \leftarrow$ pass X through *layer*
return X, $\mathcal{R}$, $\mathcal{C}$

Algorithm 3 (`bias_update` method). For a layer with given weight, compute the value for the corresponding bias so that the most costly region is split in half.

Input: X_R (data belonging to region R), w (weight of the layer)
Output: b, bias so that activation in associated unit splits X_R in half

1: $N \leftarrow \lfloor \frac{|X_R|}{2} \rfloor$
2: $WX_R \leftarrow [w \cdot x \text{ for } x \in X_R]$
3: $WX_R \leftarrow$ order WX_R increasingly
4: $a_N,\ a_{N+1} \leftarrow WX_R[N],\ WX_R[N+1]$
5: **return** $\frac{a_N + a_{N+1}}{2}$

4.2.1 Adjusting Layer Variance

After changing the parameters of the layer, we need to perform additional steps to ensure the variance is controlled and thus the gradients do not vanish or explode during optimization. To do this, we call one of the methods `fix_layer_deviation` and `reset_layer_deviation` whose steps are detailed in Algorithms 4 and 5 respectively. Both essentially scale the parameters of the layer, which does not change the activation regions, per Remark 4.1.4. Which of the two methods we use is controlled by the boolean parameter $keepVariance$ in `reinitialize_network` and `reinitialize_relu_layer` (Algorithms 1 and 2 respectively): `reset_layer_deviation` is used when `True`, and `fix_layer_deviation` when `False`.

The method `fix_layer_deviation` applies isotropic scaling to the parameters of each layer which sets the variance of its output to 1. To be precise, all weights and biases are scaled by a factor of $\frac{1}{\sigma}$, where σ is the standard deviation of the norms of the output of the layer. As we noted above, our choice of this scaling method follows a guiding principle that, on normalized data sets, exploding and vanishing gradients can be avoided by making sure the variance of the output of each layer is approximately 1 [102, 103, 104].

Due to the unexpected performance of `fix_layer_deviation` in our experiments (see Sections 4.3.1 and 4.3.2), we implement another scaling method which resets the variance of layer output to what it was prior to us changing the parameters of the layer. This is implemented within `reset_layer_deviation` method, which therefore needs to be provided with the desired standard deviation $\hat{\sigma}$ for each coordinate of the output (this can be computed with `layerwise_deviation` method in Algorithm 6). To determine the scaling factors, coordinate-wise standard deviation of the layer as is, σ, is computed. The weights and biases of unit u are then scaled by $\frac{\hat{\sigma}[u]}{\sigma[u]}$. Contrary to `fix_layer_deviation`, this scaling is anisotropic.

Algorithm 4 (`fix_layer_deviation` method). Scaling of layer parameters so that the layer output has variance 1.

Input: *layer*, X (data), $\hat{\sigma}$ (target standard deviation)

$\sigma \leftarrow std(X)$ ▷ *std* stands for coordinate-wise standard deviation
replace zero entries in σ and $\hat{\sigma}$ with 1
$scalars \leftarrow \left[\hat{\sigma}[u]/\sigma[u] \text{ for unit } u \text{ in } layer\right]$
for $u = 1, \ldots, n_{layer}$ **do**
    $w_{u:}^{(layer)} \leftarrow \frac{\hat{\sigma}[u]}{\sigma[u]} \cdot w_{u:}^{(layer)}$
$b^{(layer)} \leftarrow \frac{\hat{\sigma}}{\sigma} \cdot b^{(layer)}$ ▷ coordinate-wise operations

Algorithm 5 (`reset_layer_deviation` method). Scaling of layer parameters so that the layer output has the same variance as before the reinitialization.

Input: *layer*, X (data)

$\sigma \leftarrow std(||input|| \text{ for } input \text{ in } X)$ ▷ *std* stands for standard deviation of a list
$w^{(layer)} \leftarrow \frac{1}{\sigma} \cdot w^{(layer)}$
$b^{(layer)} \leftarrow \frac{1}{\sigma} \cdot b^{(layer)}$

Algorithm 6 (`layerwise_deviation` method). Computing the standard deviation of layer output.

Input: *network*, X (data)
Output: $\Sigma = (\sigma^{(1)}, \ldots, \sigma^{(h)})$ where $\sigma^{(\ell)} \in \mathbb{R}^{n_\ell}$ is a coordinate-wise st. deviation of $g^{(\ell)} \circ f^{[\ell]}(X)$

$\Sigma \leftarrow [\,]$
for *layer* **in** *network* **do**
    $X \leftarrow$ pass X through *layer*
    append $std(X) \in \mathbb{R}^{n_{layer}}$ to Σ ▷ *std* stands for coordinate-wise standard deviation
return Σ

4.2.2 Computing Region Membership

For any input $x \in \mathbb{R}^{n_0}$ it can be determined which activation region it belongs to by computing its activation pattern. An example of such method, `determine_region_membership`, is given in Algorithm 7.

In practice, however, the activation patterns of both regions and input are continuously computed and updated during the reinitialization process. At the beginning of iteration step belonging to unit u in layer ℓ, the only entries of the activation pattern that are known are $s_{k,j}$ for $(k,j) < (\ell,u)$ in the lexicographical order. Once the bias is reset, the only activation regions R whose sets $S_R = X_{\text{train}} \cap R$ change are the ones that are intersected by the new decision boundary. Two new regions replace each such region, and the membership is recomputed for points in S_R. Note that the entries to the activation patterns of these points do not change for $(k,j) < (\ell,u)$, only the entry for (ℓ,u) gets appended. For regions that are not split by the new decision boundary, similarly, only the entry for (ℓ,u) is computed and appended.

Algorithm 7 (`determine_region_membership` method). Determines which region a data point belongs to.

Input: *layers* (list of network's layers), x (data point), $\mathcal{R}$ (table containing the activation pattern and region of each data point)
Output: $R \in \mathcal{R}$ that x belongs to, as given by its activation pattern

$s \leftarrow [\,]$ ▷ initialize activation pattern
for *layer* **in** *layers* **do**
  $z \leftarrow [w_{u:}^{(layer)} \cdot x + b_u^{(layer)}$ for $u = 1, \ldots, n_{layer}]$
  $s^{(layer)} \leftarrow [\text{sgn}(z_u)$ for $z_u \in z]$
  append $s^{(layer)}$ to s
  $x \leftarrow$ pass x through *layer*
return s

4.2.3 Complexity Analysis

To address the question of scalability of our initialization strategy to the practical neural network setting, let us comment on its computational complexity.

Let D denote the number of data points, L the number of layers, and n the maximal number of units in a layer of our network. In such a setting, one forward pass contributes at most $\mathcal{O}(D \cdot L \cdot n^2)$ operations. To reinitialize all units in our network, two forward passes are needed: one in order to split the regions, and one to update both the region table $\mathcal{R}$ and the cost vector $\mathcal{C}$. The

complexity of the subsequent step – adjusting the layer variance – depends on the chosen method. In the worst case scenario, when `reset_layer_deviation` is used, this step involves computing two coordinate-wise standard deviations of D points in $\mathbb{R}^n$ per layer, along with an additional forward pass, again contributing an order of $\mathcal{O}(D \cdot L \cdot n^2)$ operations. Finally, an extra forward pass is required to keep track of the images of the data points with each layer.

In total, this results in five forward passes, yielding an overall computational complexity of $\mathcal{O}(D \cdot L \cdot n^2)$. Note, however, that we performed complexity analysis on the worst case scenario. In practice, the initialization strategy almost always terminates prematurely because it runs out of regions to split. Further, the number of data points, D, to use in the reinitialization should be chosen to be significantly smaller than the size of the dataset. Lastly, the number of units in a layer need not be constant across the network. As a result, with carefully chosen parameters, the computational complexity of our initialization can be significantly lower than that of a single training epoch.

4.3 Experiments

Our strategy was implemented in `Python` (using the machine learning library `PyTorch` for the construction and training of neural networks) and is available on Github [106]. We provide a detailed description of the experiments and their results in this section.

Data Set. We chose the `MNIST` data set [16] for the experiments, as it is arguably the most well-studied classification data set in the machine learning community. It consists of 70000 images of hand-written digits from 0 to 9. Each image is represented by a 28×28 grid of grayscale pixels, *i.e.* each pixel is an integer in $[0, 255]$. It is used in classification tasks, where neural networks are trained to predict which digit each image portrays.

In order to feed an image into a fully connected layer, it is first flattened into a 784-dimensional vector by concatenating its rows. In addition, the data set is normalized. In line with the standard practice, we randomly allocate 60000 images into the training, and 10000 images into the test data set.

Network Architecture. The networks used in all experiments have identical architecture:

$$f\colon \mathbb{R}^{784} \longrightarrow \mathbb{R}^{10} \longrightarrow \mathbb{R}^{10} \longrightarrow [0,1]^{10},$$

where the first two layers have ReLU, and the last one has LogSoftmax activation. The architecture was chosen to be as small as possible so that the network still reliably achieves reasonable accuracy after a small number of training steps.

Initialization Strategies. To test whether maximizing the number of linear regions is beneficial for the success of training, we compare our initialization with the LeCun initialization. Recall, that the first step of our method is to use LeCun initialization and then readjust the bias with `reinitialize_relu_layer`. To ease the computational strain, we randomly choose 3000 images per class from the X_{train} data set to pass to the reinitialization method. The sets of experiments we ran differ in the scaling method that was used for variance control. In experiments from Sections 4.3.1 and 4.3.2, `fix_layer_deviation` method was used. To see whether potential improvements in training following our initialization are actually due to scaling, we also include networks that are first initialized following the LeCun strategy and then scaled with `fix_layer_deviation`. This is not necessary in Section 4.3.3 where `reset_layer_deviation` was used instead.

Training. The networks have been trained for 24 runs and 24 epochs, with the training data split in batches of size 100. As detailed in Section 4.1, cross entropy loss was used in either Adam (Sections 4.3.1 and 4.3.3) or SGD (Section 4.3.2) optimization. During training, the accuracy and loss are recorded 10 times per epoch, while the number and costs of the regions are recorded 4 times per epoch.

4.3.1 Adam Optimization and `fix_layer_deviation`

The results of this set of experiments are illustrated in Figure 4.1, with the plots in blue belonging to the networks initialized via LeCun, the ones in orange belonging to networks that are additionally scaled with `fix_layer_deviation`, and, lastly, the plots in green belonging to the networks initialized with our initialization strategy and scaled with `fix_layer_deviation`. All the networks were trained with Adam optimizer. We observe the following:

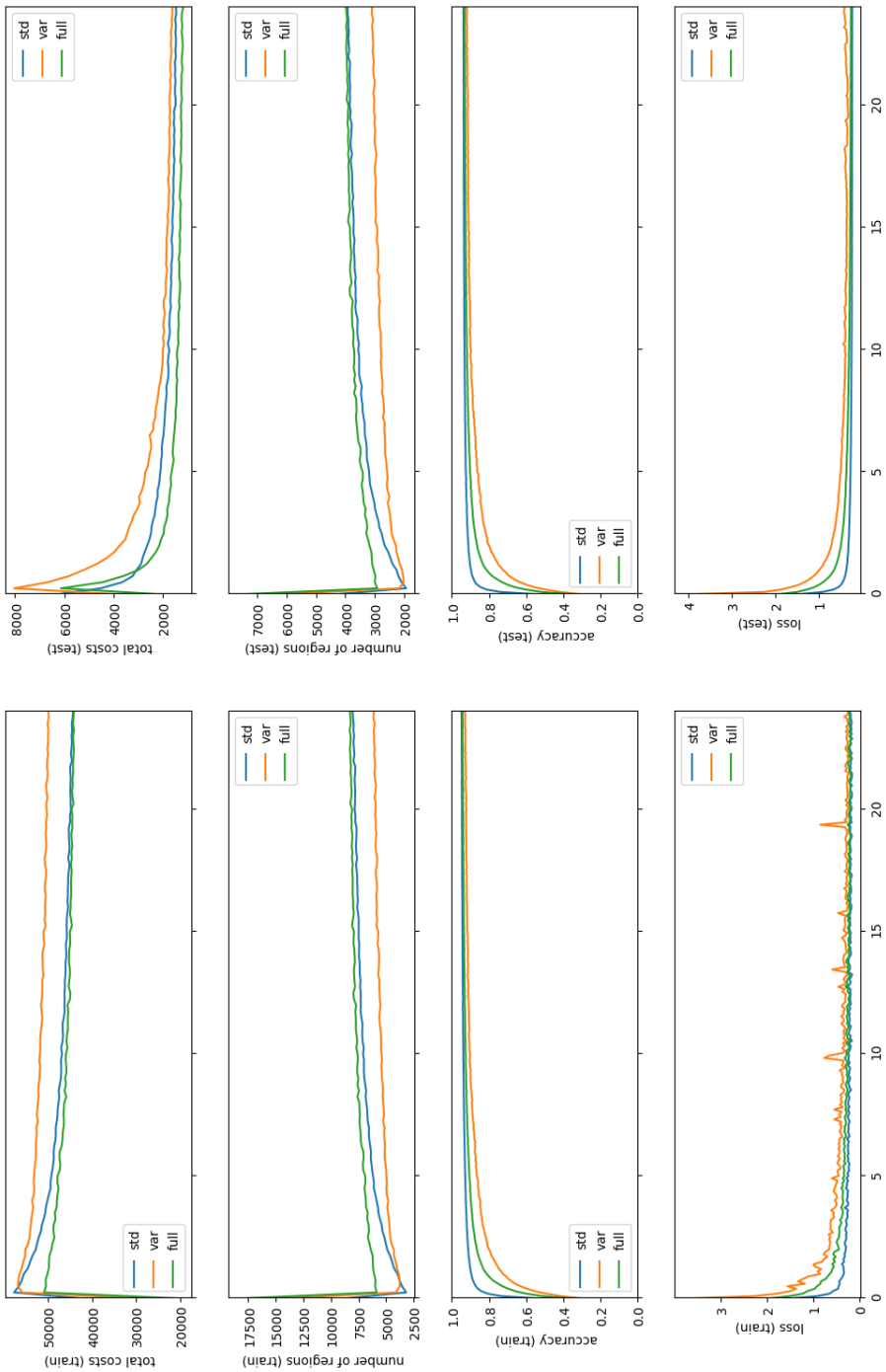


Figure 4.1: Plots illustrating the results of the experiments on neural networks trained with Adam optimizer and with `fix_layer_deviation` scaling. The results of networks initialized following our strategy are tagged with `'full'`, the ones with LeCun initialization with `'std'`, and the ones with LeCun initialization and additional scaling to control the variance with `'var'`. They are shown in green, blue, and orange, respectively, as denoted by the legend.

- For all initialization strategies, there is an immediate drop in the number of (populated) regions at the beginning of training. Their number then gradually, but continuously increases throughout all subsequent stages of training.
- The networks that are first initialized via LeCun and then scaled, consistently reach lower accuracy than the other networks. We can therefore conclude that the scaling with `fix_layer_variance` does not influence the performance of our initialization strategy in the positive. It might, in fact, hinder it. Nevertheless, based on the results of this experiment we can argue that maximizing the number of regions does have an added benefit when the weights are not scaled optimally at initialization.
- All strategies lead to comparable performance after many epochs. Since all networks follow identical accuracy vs. epoch curves for the training and test data sets, we argue that their capacity for generalization is comparable as well.
- The total costs for the training data set are much higher than the ones for the test data set. This is due to the fact that we did not take the size of the data sets into account, and the regions for the training data set simply contain more points. In addition, notice that the plots for the number of regions for networks with scaling (in green and orange) seem to differ by a constant in both training and test setting. This hints that the surplus of regions obtained by our strategy is maintained throughout training. It is uncertain, whether that would hold if we followed a more appropriate scaling heuristic, thus reaching an even higher number of regions and potentially surpassing LeCun initialization with regards to accuracy.

4.3.2 SGD Optimization and `fix_layer_deviation`

The results of this set of experiments are illustrated in Figure 4.2, with the plots in blue belonging to the networks initialized via LeCun, the ones in orange belonging to the networks that are additionally scaled with `fix_layer_deviation`, and, lastly, the plots in green belonging to the networks initialized with our initialization strategy and scaled with `fix_layer_deviation`. All the networks were trained with SGD optimizer. We observe the following:

- The drop in the number of regions in the early stages of training in Section 4.3.1 can be observed here as well. It is more smooth and gradual, especially for the networks initialized via LeCun. While an increase in the number of regions in subsequent training could be observed when Adam was used, this is not generally true here. These discrepancies suggest that the optimization steps have different consequences for the linear regions

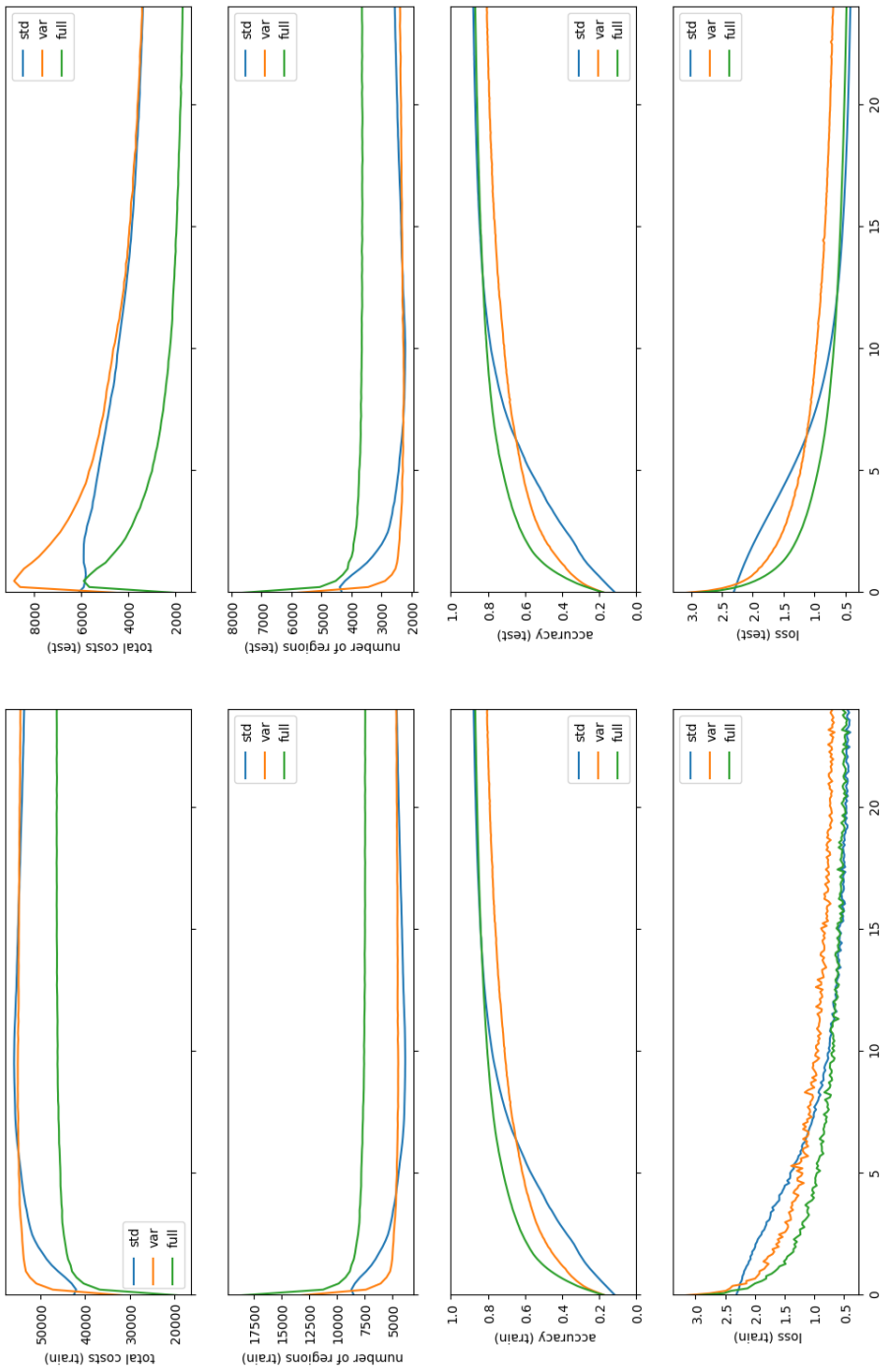


Figure 4.2: Plots illustrating the results of the experiments on neural networks trained with SGD optimizer and with `fix_layer_deviation` scaling. The results of networks initialized following our strategy are tagged with `'full'`, the ones with LeCun initialization with `'std'`, and the ones with LeCun initialization and additional scaling to control the variance with `'var'`. They are shown in green, blue, and orange, respectively, as denoted by the legend.

depending on which optimizer is used. In particular, it seems the way linear regions are treated by SGD depends on the variance of the output at initialization, since the plots for the networks with LeCun initialization follow different curves than the ones scaled with `fix_layer_deviation`.

- In these experiments, scaling the layers with `fix_layer_deviation` seems to have an initial benefit, with both sets of networks using this scaling reaching higher accuracy in the first 5 (for only scaled networks) and 10 (for reinitialized networks) epochs than the networks initialized via LeCun. Maximizing the number of linear regions at initialization again leads to faster convergence and better final accuracy when compared with networks whose parameters are only scaled. However, the networks with LeCun initialization reach a comparable accuracy to the reinitialized networks in the later stages of the training. This poses a question whether our initialization only fixes the damage done by choosing a subpar scaling method.
- Since all networks follow identical accuracy vs. epoch curves for the training and test data sets, we argue that their capacity for generalization is comparable.
- The total costs for the training data set are much higher than the ones for the test data set, which is partially a consequence of the difference in data set sizes. However, the curves the costs follow are significantly different for the two data sets, with the ones for test data set decreasing consistently after the first epoch. We do not know what to attribute this observation to.

4.3.3 Adam Optimization and `reset_layer_deviation`

The results of this set of experiments are illustrated in Figure 4.3, with the plots in blue belonging to the networks initialized via LeCun and the plots in orange belonging to the networks initialized with our initialization strategy and scaled with `reset_layer_deviation`. All the networks were trained with Adam optimizer. We observe the following:

- Similar as in Section 4.3.1, there is an immediate drop in the number of linear regions at the beginning of training for both sets of networks.
- The accuracies the networks reach during training is independent of whether we maximize the number of linear regions at initialization. This shows that this additional step brings no benefit when the variance is set to a fitting value.

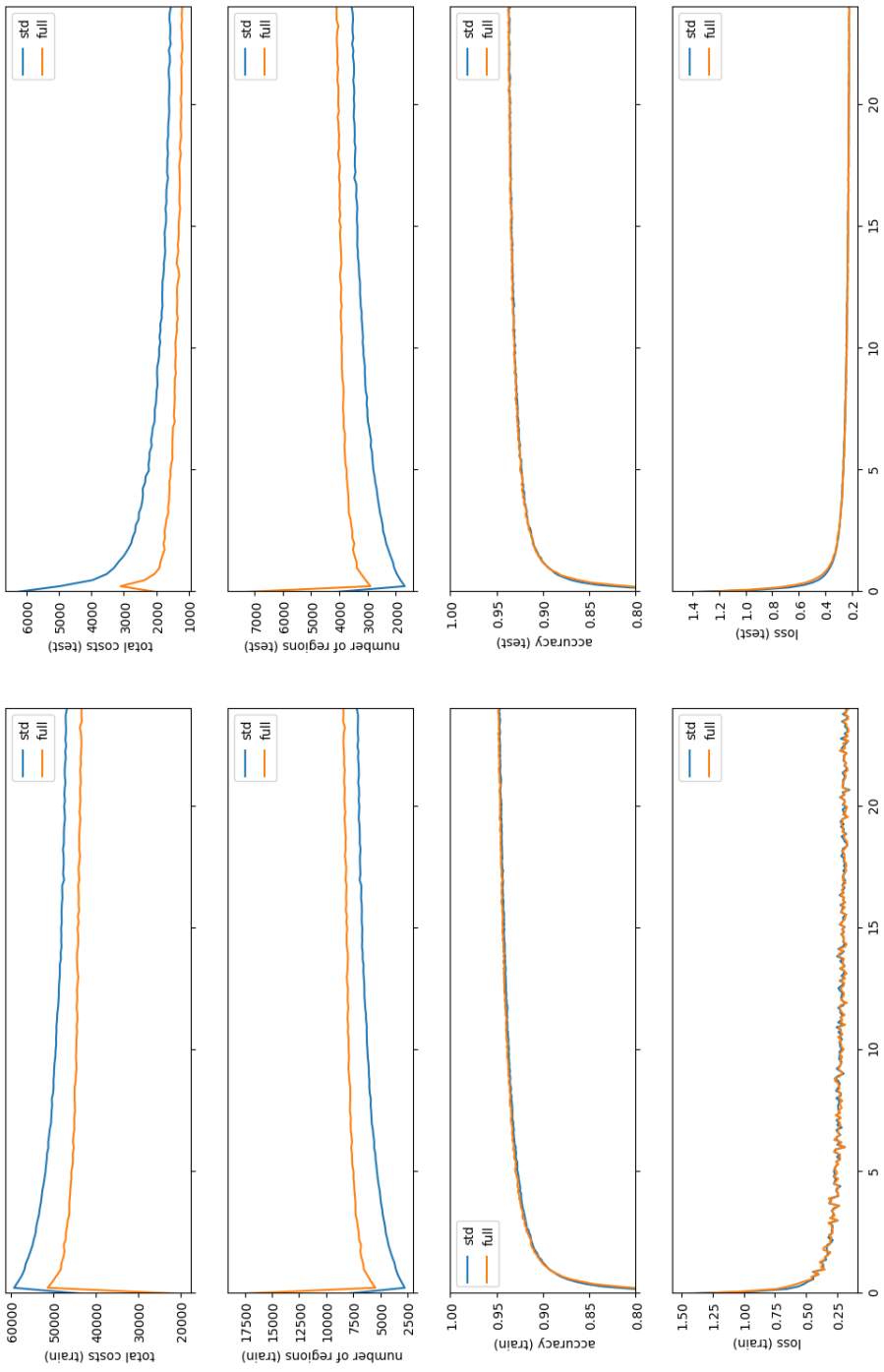


Figure 4.3: Plots illustrating the results of the experiments on neural networks trained with Adam optimizer and with `reset_layer_deviation` scaling. The results of networks initialized following our strategy are tagged with `'full'`, and the ones for LeCun initialization with `'std'`. They are shown in orange and blue, respectively, as denoted by the legend.

4.4 Conclusion

The experiments we ran to determine whether maximizing the number of linear regions at initialization is beneficial, are inconclusive and depend highly on the optimization method used. All experiments seem to agree that such an additional step at initialization does not lead to improvement in the final accuracy reached, at least not when the initial variance is set to a sensible value. In fact, the effects of variance control at initialization seem to be more important for the success of training than maximizing the number of linear regions. Additionally, our empirical observations shine a light on three phenomenons which are not well understood and deserve further theoretical study.

- How different optimizers "treat" linear regions throughout training is not the same. This agrees with the observations in [107], where the study involved not only the cost and number of linear regions, but also their more geometric properties, such as the radii of their inspheres, directions of the decision boundaries and the angles at which they meet. They showed that different optimization techniques introduce linear regions with different properties even when they reach similar final accuracies. Surprisingly, our empirical tests hint that in the case of SGD optimizer, the evolution of linear regions additionally depends on the variance of the layers at initialization.
- The number of linear regions experiences a significant drop in the early stages of training. While it does begin increasing later on when using Adam optimizer, it does not increase for all networks when using SGD. This behavior has been observed for Adam before in [2, 101] (see Figure 4.4), but the optimization aims that drive it are not yet fully understood. The speculation of [101, 108] is that neural networks learn global patterns first, for which fewer and larger linear regions are sufficient. In subsequent stages the focus switches to memorization and the regions are fragmented to fit to the individual data points. This interpretation, however, is challenged by our experiments with SGD optimizer.
- We do not yet fully understand how adjusting the variance at initialization affects the training. Several heuristics for variance control have been suggested with the primary focus of avoiding exploding or vanishing gradients, which make training difficult or even halt it. Our experiments suggest that the choice of the heuristic might not only influence the speed of convergence, but also determine what final accuracy the network is able to reach (this seems to be the case especially when using SGD). Further, our experiments seem to contradict the common principle that a good initialisation should ensure the variance of each layer's output is 1.

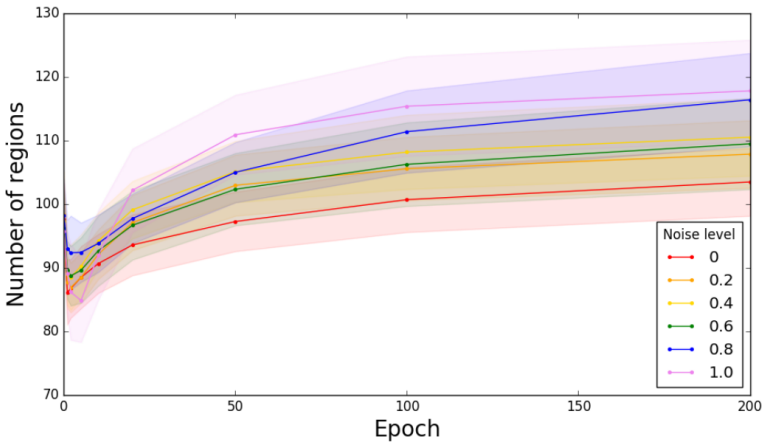


Figure 4.4: The evolution of the number of linear regions (computed along lines through input space) during training of neural networks with fixed architecture on MNIST data set at several noise levels [2].

Chapter 5

Gromov–Hausdorff Distance for Directed Metric Spaces

The phenomena studied in certain domains are not reversible. This can be reflected in having privileged directions in the ambient space. Perhaps the most clear examples of non-reversible phenomena are those involving time, where any trajectory going forwards in time cannot be reversed. When modeling such phenomena one should take directionality into account.

Directed algebraic topology [109, 19] provides a natural theoretical framework for such applications. Its conception was driven by two key motivations: considerations of non-reversibility in homotopy theory inherently gives rise to it, while directed spaces were also recognized as the appropriate model for the study of concurrent processes. The interest in directed structures has risen in recent years, primarily due to the use of networks in mathematical modeling and, in particular, machine learning. To conduct further analysis on the directed space models, it is paramount that one is able to measure distances within them and compare them between each other. Both tasks ask for notions of distance, either on a specific example or on the entire set of directed spaces. While new distances sensitive to direction have been introduced in the context of networks, we are not aware of any in the general setting of directed spaces. With the work presented in this chapter we aim to fill this gap.

Our inspiration for the definition of a distance between directed spaces is the Gromov–Hausdorff distance. It is a metric on the isometry classes of compact metric spaces, hence it measures how close two compact metric spaces are to being isometric. We use two of its equivalent definitions: the original one via isometries between spaces, and an alternative one via distortion of maps. We generalize both notions to the setting of directed spaces equipped with a metric, calling the first the **directed Gromov–Hausdorff** distance, and the second the **distortion distance**. Note that the original metric on the space does not necessarily take direction into account. Thus, we define a novel notion of distance, called the **zigzag distance**, which is induced by the underlying metric and the directed structure on a d-space. Our definitions of distances between d-spaces are phrased with respect to the zigzag distance. Curiously, we show that, contrary to the undirected analogues, these two notions are not equivalent. Directed Gromov–Hausdorff distance between directed spaces whose metric structure is given by the zigzag distance coincides with the original Gromov–Hausdorff distance between their underlying (undirected) topological spaces with the same metric structure. The distortion distance, however, is stronger and better captures the differences in directed structures.

Throughout this chapter $\vec{X}$ is a d-space where the underlying topological space is a metric space (X, d), and the topology on X is induced by the metric. Moreover, $\vec{X}$ is always assumed to be rectifiable. For the necessary preliminary details, see Sections 2.1 and 2.3. We begin this chapter in Section 5.1 by defining the zigzag distance. Directed spaces on which it is a metic can then be compared with the analogue of the classical Gromov–Hausdorff distance, the directed Gromov–Hausdorff distance, which we define in Section 5.2, where we also explore some of its properties. In Section 5.3, we introduce a directed analogue for an alternative formulation of the Gromov–Hausdorff distance: the distortion distance. We continue by listing its properties as well, and comparing it to the directed Gromov–Hausdorff distance. Lastly, Sections 5.4 and 5.5 are dedicated to some interesting examples of directed spaces: the directed flat torus and the directed weighted graph, respectively. We define them to be the directed analogues of flat torus and weighted directed graph, and detail how they can be equipped with a directed metric structure via the zigzag distance.

Definition 5.0.1. A **rectifiable d-space** is a d-space $(X, \vec{P}(X))$, where X is a metric space and every d-path in $\vec{P}(X)$ is rectifiable.

Recall that a path $\gamma\colon I \to X$ is **rectifiable** if its length by total variation,

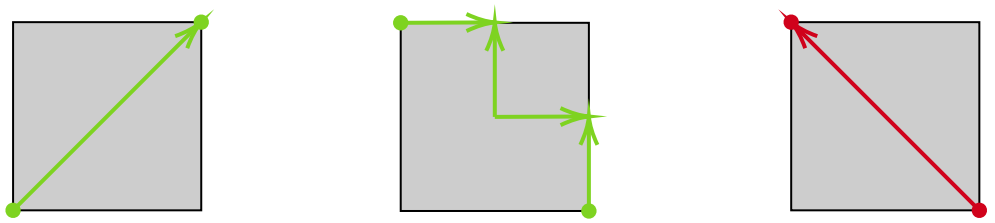
$$\ell^d(\gamma) = \sup \sum_{i=1}^{N} d(\gamma(t_{i-1}), \gamma(t_i))$$

where the supremum ranges over all finite sequences $0 \leq t_0 < t_1 < \cdots < t_N \leq 1$, is finite. Since X is a metric space, a path is rectifiable if and only if there exists a homeomorphism $h\colon I \to I$ such that γh is 1-Lipschitz. Observe that all constant paths, finite concatenations of rectifiable paths, and partial reparametrizations of rectifiable paths are rectifiable. This guarantees that a rectifiable d-space is well-defined.

5.1 Zigzag Distance

The notion of direction in d-spaces is encoded in its set of distinguished paths. Thus, if we wish to define a metric on a d-space that takes direction into account, it is sensible to define it via d-paths. However, most d-spaces are not d-path connected, meaning that for some $x, x' \in X$ the set $\vec{P}(x, x')$ of d-paths with source x and target x' is empty.

Definition 5.1.1. A **zigzag path** between $x, x' \in \vec{X}$ is a sequence $(\gamma_i)_{i=1}^m$ of d-paths such that $\gamma_i \in \vec{P}(p_{i-1}, p_i) \cup \vec{P}(p_i, p_{i-1})$, where $p_0 = x$ and $p_m = x'$. Denote the set of all zigzag paths between x and x' as $\vec{P}_{zz}(x, x')$, and with $\vec{P}_{zz}(X)$ the set of all zigzag paths on $\vec{X}$. A d-space $\vec{X}$ is **zigzag connected** if $\vec{P}_{zz}(x, x') \neq \emptyset$, for any pair $x, x' \in X$.



(a) A d-path. (b) A zigzag path. (c) *Not* a zigzag path.

Figure 5.1: Let $\vec{I}^2$ be the d-space where $\vec{P}(I^2)$ is given by the product order on I^2. The single green arrow (left) is a d-path from $(0, 0)$ to $(1, 1)$. The multiple green arrows (middle) give a zigzag path between $(1, 0)$ and $(0, 1)$, whereas the red arrow (right) is *not* a zigzag path.

A zigzag connected d-space $\vec{X}$ is also path connected because $\vec{P}(Z)(X) \subset C([0, 1], X)$. To see that the converse is in general not true, consider a path connected space with $\vec{P}(X)$ containing only the constant paths.

Since X is a metric space we can define the length of a rectifiable zigzag path, which in turn induces a distance function on X.

Definition 5.1.2. The length of a zigzag path $\gamma = (\gamma_i)_{i=1}^m$ is the sum of lengths of constituent paths, namely

$$\ell_{zz}(\gamma) = \sum_{i=1}^{m} \ell^d(\gamma_i),$$

where ℓ^d is the length by total variation induced by d on X. The **zigzag distance** induced by d on $\vec{X}$ is defined as

$$d_{zz}(x, x') = \inf_{\gamma \in \vec{P}_{zz}(x,x')} \ell_{zz}(\gamma).$$

The following result is a analogue of (2.1) in Proposition 2.1.9 in our setting.

Lemma 5.1.3. *For a d-space $\vec{X}$, the underlying space of which is a metric space (X, d), and every x, x' in X,*

$$d_{zz}(x, x') \geq d(x, x').$$

Proof. If there are no zigzag paths between x and x', then $d_{zz}(x, x') = \infty$ and the statement holds. Assume, therefore, that $\vec{P}_{zz}(x, x') \neq \emptyset$, and there exists a path $\gamma = (\gamma_i)_{i=1}^m \in \vec{P}_{zz}(x, x')$ with $\ell_{zz}(\gamma) < d(x, x')$. Then,

$$d(x, x') > \ell_{zz}(\gamma) = \sum_{i=1}^{m} l(\gamma_i) \geq \sum_{i=1}^{m} d(\gamma_i(0), \gamma_i(1)),$$

which violates the triangle inequality of the metric d. Consequently, $\ell_{zz}(\gamma) \geq d(x, x')$ holds for any path $\gamma \in \vec{P}_{zz}(x, x')$, and taking the infumum of ℓ_{zz} over all such paths gives

$$d_{zz}(x, x') \geq d(x, x').$$ □

Proposition 5.1.4. *The zigzag distance d_{zz} on $\vec{X}$ is an extended metric, and it is a metric when $\vec{X}$ is zigzag connected.*

Proof. We need to show d_{zz} satisfies identity, positivity, symmetry and triangle inequality. The first two follow from Lemma 5.1.3, since d is a metric. Now, observe that that there is a path-preserving bijection between the sets $\vec{P}_{zz}(x,x')$ and $\vec{P}_{zz}(x',x)$: for each $\gamma \in \vec{P}_{zz}(x,x')$, its reverse $\gamma^\star$ is in $\vec{P}_{zz}(x',x)$. As a consequence, $d_{zz}(x,x') = d_{zz}(x',x)$, implying that d_{zz} is symmetric. Lastly, we note that, for every x', $\vec{P}_{zz}(x,x'')$ contains all the paths from x to x'' passing through x'. Thus,

$$\begin{aligned} d_{zz}(x,x'') &= \inf_{\gamma\in\vec{P}_{zz}(x,x'')} \ell_{zz}(\gamma) \\ &\leq \inf_{\gamma_1\in\vec{P}_{zz}(x,x'),\gamma_2\in\vec{P}_{zz}(x',x'')} (\ell_{zz}(\gamma_1)+\ell_{zz}(\gamma_2)) \\ &= d_{zz}(x,x') + d_{zz}(x',x''), \end{aligned}$$

showing the triangle inequality for d_{zz}.

Note that $d_{zz}(x,x') = \infty$ if and only if $\vec{P}_{zz}(x,x')$ is empty, justifying the second part of the statement. □

Definition 5.1.5. The pair $(\vec{X}, d_{zz})$, where $\vec{X}$ is a zigzag connected d-space, is a **directed metric space**.

Example 5.1.6. The assumption that the underlying space is a metric space is necessary, since a zigzag distance induced from a more general distance function on the space may violate certain requirements of an extended metric. To illustrate this, let X be a topological space and $\ell\colon X^I \to \mathbb{R}_{\geq 0}$ some length function on paths in X. Further, let $S \subseteq X$ be a closed subspace and C_A the set of all constant paths in $A \subseteq X$.

Given distinct points $a, b \in S$, define for each $n \in \mathbb{N}$ a directed structure $\vec{P}(X)^{(n)} = \langle C_X \cup \{f_n\}\rangle$ where $f_n : [0,1] \to X$ is a continuous map such that $f_n(0) = a$, $f_n(1) = b$, and $\ell(f_n) = \frac{1}{n}$. For any n, the pair $(X, \vec{P}(X)^{(n)})$ is a directed space with a directed subspace (S, C_S).

Consider the disjoint union $\bigsqcup_{n\in\mathbb{N}}(X, \vec{P}(X)^{(n)})$ and define an equivalence relation $\sim$ on it in the following way. Given points $x \in (X, \vec{P}(X)^{(n)})$ and $x' \in (X, \vec{P}(X)^{(m)})$ for some $m, n \in \mathbb{N}$, $x \sim x'$ if and only if $x, x' \in S$ and $x = x'$. Denote the quotient d-space

$$\bigsqcup_{n\in\mathbb{N}} (X, \vec{P}(X)^{(n)})/_\sim$$

by $(Y, \vec{P}(Y))$ and equip it with the zigzag distance d_{zz}. Note that $\vec{P}(Y) = C_Y \cup \langle \{f_n\}_{n\in\mathbb{N}} \rangle$, and that $d_{zz}(a,b) = \inf_{n\in\mathbb{N}} \frac{1}{n} = 0$, despite the assumption of $a \neq b$ in the construction. Therefore, d_{zz} is not an extended metric on $(Y, \vec{P}(Y))$. △

Example 5.1.7 (Non-Equivalent Topology)**.** It is perhaps not surprising that the topology induced by the zigzag distance in general is not equivalent to the topology of the underlying metric space. To illustrate this, take (X, d) to be the Euclidean plane. Let the set of d-paths $\vec{P}(X)$ in d-space $(X, \vec{P}(X))$ be generated by

$$\{\gamma_x \colon I \to X,\ \gamma_x(t) = t \cdot x \mid x \in X\}.$$

In the zigzag metric d_{zz} induced by these choices, the shortest path between any two points x, x' in X with $\frac{x}{||x||} \neq \frac{x'}{||x'||}$ is given by following a straight line from x to the origin and continuing out in a straight line to x'. Since it suggests that the space is centralized, it is known by many names, including post office, French Metro, British Rail, and SNCF metric.

Fix a radius $R > 0$ and take $x \in X$ for which $d(0, x) > R$. Notice that $B_{zz}(x, R) \subset B(x, r)$ for any $r \geq R$, but there exists no $r' > 0$ so that

$$B(x, r') \subseteq B_{zz}(x, R),$$

where B is a ball in the Euclidean metric d, and B_{zz} a ball in the zigzag metric d_{zz}. △

5.2 Directed Gromov–Hausdorff Distance

In this section we study the Gromov–Hausdorff distance on directed spaces. Unless stated otherwise, directed metric spaces are endowed with the zigzag metric induced from the underlying metric space.

Definition 5.2.1. Let $(\vec{X}, d_{zz}^X)$ and $(\vec{Y}, d_{zz}^Y)$ be two directed metric spaces. A d-map $\vec{F} \colon \vec{X} \to \vec{Y}$ is called a **d-isometry** if

$$d_{zz}^X(x, x') = d_{zz}^Y(\vec{F}(x), \vec{F}(x'))$$

holds for any x, x' in X. D-spaces $\vec{X}$ and $\vec{Y}$ are **d-isometric** if there exists a bijective d-isometry $\vec{F} \colon \vec{X} \to \vec{Y}$ whose inverse $\vec{F}^{-1} \colon Y \to X$ (as a function $F^{-1} \colon Y \to X$) is a d-map.

Definition 5.2.2. Let $\vec{X}$ and $\vec{Y}$ be d-subspaces of the directed metric space $(\vec{Z}, d_{zz})$. The **directed Hausdorff distance** of $\vec{X}$ and $\vec{Y}$ in $\vec{Z}$ is defined by

$$\vec{d}_H(\vec{X}, \vec{Y}) = d_H((X, d_{zz}^X), (Y, d_{zz}^Y)).$$

The **directed Gromov–Hausdorff distance** between two directed metric spaces $\vec{X}$ and $\vec{Y}$ is defined as

$$\vec{d}_{GH}(\vec{X}, \vec{Y}) = \inf_{\vec{F}, \vec{G}} \vec{d}_H(\vec{F}(\vec{X}), \vec{G}(\vec{Y})),$$

where $\vec{F}\colon \vec{X} \to \vec{Z}$ and $\vec{G}\colon \vec{Y} \to \vec{Z}$ are directed isometries into some directed metric space $(\vec{Z}, d_{zz})$.

Interestingly, insisting on isometries between directed spaces in the definition of the Gromov–Hausdorff distance being d-maps is not restrictive. In fact, the Gromov–Hausdorff distance depends only on the zigzag metric structure induced by the d-structure.

Theorem 5.2.3. *Let $(\vec{X}, d_{zz}^X)$ and $(\vec{Y}, d_{zz}^Y)$ be directed metric spaces. Then*

$$d_{GH}((X, d_{zz}^X), (Y, d_{zz}^Y)) = \vec{d}_{GH}((\vec{X}, d_{zz}^X), (\vec{Y}, d_{zz}^Y)).$$

Proof. Observe that each d-isometry from $(\vec{X}, d_{zz}^X)$ or $(\vec{Y}, d_{zz}^Y)$ to $(\vec{Z}, d_{zz}^Z)$ is in particular an isometry from (X, d_{zz}^X) or (Y, d_{zz}^Y), respectively, to (Z, d_{zz}^Z). Thus, $d_{GH}(X, Y) \leq \vec{d}_{GH}(\vec{X}, \vec{Y})$.

Secondly, let us prove that $d_{GH}(X, Y) \geq \vec{d}_{GH}(\vec{X}, \vec{Y})$. For every $\delta > 0$, there is a metric space (Z^δ, d^δ) and isometries $F^\delta\colon X \to Z^\delta$ and $G^\delta\colon Y \to Z^\delta$ so that

$$d_H^\delta(F^\delta(X), G^\delta(Y)) \leq d_{GH}(X, Y) + \delta,$$

as illustrated in Figure 5.2. Fix an $\epsilon > 0$ and consider $[0, \epsilon]$ with the Euclidean metric. Define a d-space $\vec{Z}_\epsilon^\delta = (Z^\delta \times [0, \epsilon], \vec{P})$ where the set $\vec{P}$ of d-paths is generated by

$$\begin{aligned} &\{t \mapsto (F \circ \gamma(t), 0) \mid \gamma \in \vec{P}(X)\} \\ &\bigcup \{t \mapsto (G \circ \gamma(t), \epsilon) \mid \gamma \in \vec{P}(Y)\} \\ &\bigcup \{t \mapsto (z, \epsilon \cdot t) \mid z \in Z^{(\delta)}\}, \end{aligned}$$

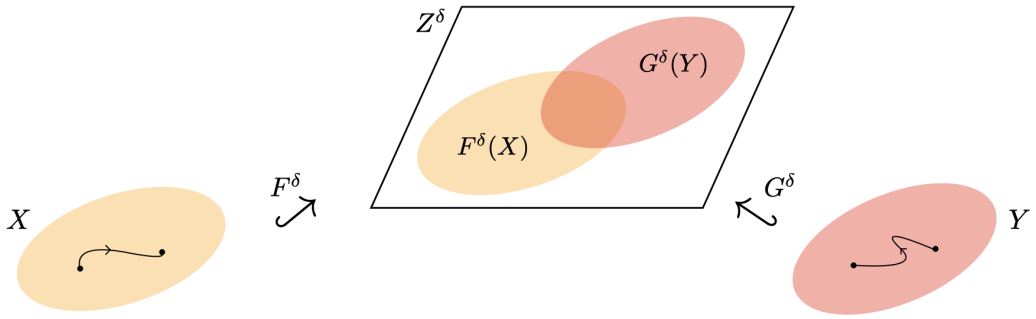


Figure 5.2: Isometric embeddings of X and Y into a metric space Z^δ in which their Hausdorff distance is δ-close to their (undirected) Gromov–Hausdorff distance.

see Figure 5.3. If we endow $\vec{Z}_\epsilon^\delta$ with the zigzag metric induced by the sup-metric on $Z^\delta \times [0,\epsilon]$, denoted by $d_{zz}^{\delta,\epsilon}$, then embeddings $\iota_X \colon \vec{X} \to \vec{Z}_\epsilon^\delta$ mapping $x \mapsto (F(x),0)$, and $\iota_Y \colon \vec{Y} \to \vec{Z}_\epsilon^\delta$ mapping $y \mapsto (G(y),\epsilon)$ are d-isometries. Furthermore, notice that

$$d_{zz}^{\delta,\epsilon}(\iota_X(x),\iota_Y(Y)) = d_H^\delta(F^\delta(x),G^\delta(Y)) + \epsilon, \quad \text{and}$$
$$d_{zz}^{\delta,\epsilon}(\iota_X(X),\iota_Y(y)) = d_H^\delta(F^\delta(X),G^\delta(y)) + \epsilon$$

for any $x \in X, y \in Y$. As a consequence,

$$\vec{d}_H^{\delta,\epsilon}(\iota_X(\vec{X}),\iota_Y(\vec{Y})) = d_H^\delta(F^\delta(X),G^\delta(Y)) + \epsilon \leq d_{GH}(X,Y) + \epsilon + \delta,$$

where $\vec{d}_H^{\delta,\epsilon}$ is the directed Hausdorff distance in $(Z_\epsilon^\delta, d_{zz}^{\delta,\epsilon})$, and d_H^δ is the Hausdorff distance in (Z^δ, d^δ). Since this holds for every $\epsilon > 0$ and every $\delta > 0$, $\vec{d}_{GH}(\vec{X},\vec{Y}) \leq d_{GH}(X,Y)$. □

Recall that the Gromov-Hausdorff distance is a metric on the set of isometry classes of compact metric spaces. If we therefore consider directed spaces up to isometry only, Theorem 5.2.3 immediately implies the following.

Corollary 5.2.4. *The directed Gromov-Hausdorff distance is a metric on the space of compact directed metric spaces $(\vec{X}, d_{zz}^X)$ up to isometry.*

Corollary 5.2.5. *For any zigzag connected d-space, $\vec{d}_{GH}(\vec{X},\vec{X}^*) = 0$.*

Note that Corollary 5.2.4 does not require compactness of the underlying metric space (X, d^X),

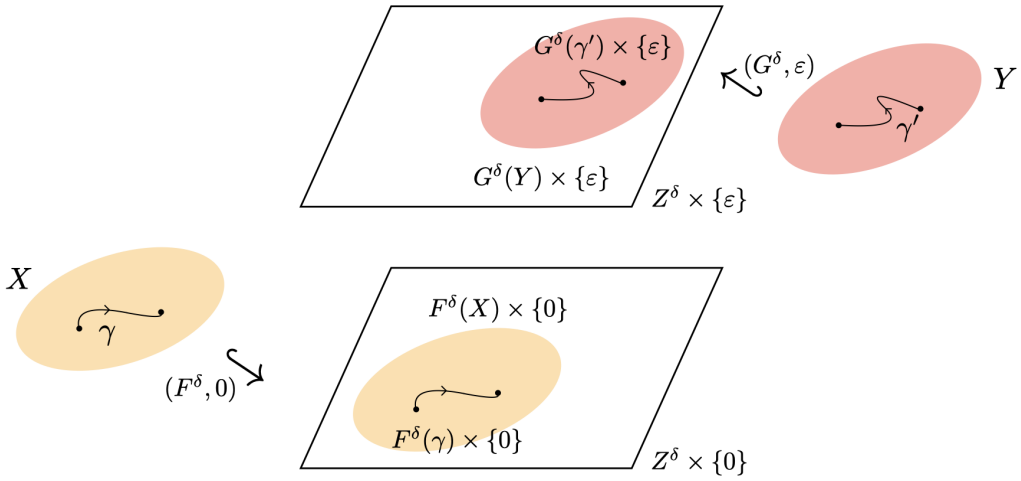


Figure 5.3: Using the isometric embeddings $F^\delta \colon X \to Z^\delta$ and $G^\delta \colon Y \to Z^\delta$ from Figure 5.2, a new space $Z^\delta_\epsilon = Z^\delta \times [0, \epsilon]$ is constructed, with X embedded into layer 0 via $(F^\delta, 0)$, and Y embedded into layer ϵ via (G^δ, ϵ). A directed structure for Z^δ_ϵ is then chosen so that these embeddings are d-maps and Z^δ_ϵ is zigzag connected.

but of the zigzag metric space $(\vec{X}, d^X_{zz})$. This is a much stricter assumption, given that d_{zz}, in general, is an extended metric (see Proposition 5.1.4). Corollary 5.2.5 follows by observing that $\vec{X}$ and $\vec{X}^*$ induce the same zigzag distance d^X_{zz} on X.

The following example illustrates that despite the result stated in Corollary 5.2.4, the directed Gromov–Hausdorff distance is not a metric on the space of compact directed metric spaces up to d-isometry, as it can be zero on spaces that are not d-isometric.

Example 5.2.6. Let $X = [-1, 1]$ and define the maps $\gamma_1, \gamma_2 \colon I \to X$ as $\gamma_1(t) = t$ and $\gamma_2(t) = -t$. Endow X with the path space $\vec{P}(X) = \langle \gamma_1, \gamma_2 \rangle$ and the Euclidean metric. Denote the resulting d-space as $\vec{X}$ and its reverse space as $\vec{X}^*$.

By Corollary 5.2.5, $\vec{d}_{GH}(\vec{X}, \vec{X}^*) = 0$, where both spaces are equipped with the zigzag metric.

For any d-map $\vec{F}\colon \vec{X} \to \vec{X}^*$, the compositions $\vec{F} \circ \gamma_1$ and $\vec{F} \circ \gamma_2$ must be d-paths in $\vec{P}(X)^*$. Further, since $\vec{F} \circ \gamma_1(0) = \vec{F}(0) = \vec{F} \circ \gamma_2(0)$, one of the following holds:

- $\vec{F} \circ \gamma_1(I), \vec{F} \circ \gamma_2(I) \subseteq \gamma_1^*(I) = [0,1]$,
- $\vec{F} \circ \gamma_1(I), \vec{F} \circ \gamma_2(I) \subseteq \gamma_2^*(I) = [-1,0]$.

As a consequence, there is no d-isometry between $\vec{X}$ and $\vec{X}^*$. △

Other consequences of Theorem 5.2.3 and Proposition 2.1.14 are the following.

Corollary 5.2.7. *For compact directed metric spaces* $(\vec{X}, d_{zz}^X)$ *and* $(\vec{Y}, d_{zz}^Y)$

1. $d_{GH}^{\rightarrow}(\vec{X}, \vec{Y}) \leq \frac{1}{2}\max\{\mathrm{Diam}(\vec{X}), \mathrm{Diam}(\vec{Y})\}$,
2. $d_{GH}^{\rightarrow}(\vec{X}, \vec{Y}) = \frac{1}{2}\mathrm{Diam}(\vec{Y})$, *if* $X = \{x_0\}$.

Considering that the metric of the underlying metric space and the zigzag distance induced by said metric and a specified set of d-paths are not equivalent, it comes as no surprise that the Gromov–Hausdorff distance between (X, d^X) and (X, d_{zz}^X) can be non-zero. Regardless, we illustrate this fact with an example.

Example 5.2.8. Take $X = I^2$ with the euclidean metric d^X inherited from $\mathbb{R}^2$, and denote by $\vec{X} = (X, \vec{P}(X))$ the d-space induced by the product order on I^2. We show that

$$d_{GH}\left((X, d^X), (X, d_{zz}^X)\right) = 1 - \frac{\sqrt{2}}{2}.$$

As stated in Corollary 2.1.13, $\frac{1}{2}\inf_\varphi \mathrm{dis}(\varphi) \leq d_{GH}\big((X, d^X), (X, d_{zz}^X)\big)$ with the infimum running over all maps $\varphi\colon (X, d_{zz}^X) \to (X, d^X)$. By taking the anti-diagonal points, we can further show

$$\begin{aligned}\inf_\varphi \mathrm{dis}(\varphi) &\geq \inf_\varphi |d_{zz}^X((1,0),(0,1)) - d^X(\varphi(1,0), \varphi(0,1))| \\ &= \inf_\varphi |2 - d^X(\varphi(1,0), \varphi(0,1))| \\ &\geq 2 - \sqrt{2},\end{aligned}$$

where we use $\mathrm{diam}(X, d^X) = \sqrt{2}$ in the last step. Further, distortion $2-\sqrt{2}$ is attained for $\varphi = \mathrm{Id}$. Indeed, observe that

$$\mathrm{dis}(\mathrm{Id}) = \sup_{x,x' \in X} |d_{zz}^X(x,x') - d^X(x,x')|$$

is attained at a pair of points x, x' that are incomparable in the product order as they are the only ones for which $d_{zz} \neq d^X$. On such points, $d_{zz}(x, x') = |x_1 - x_1'| + |x_2 - x_2'|$ and computing the distortion corresponds to maximization of $|\ell_1(x, x') - \ell_2(x, x')|$, where ℓ_1 is the Manhattan and ℓ_2 the euclidean metric. This maximum is attained by anti-diagonal points, which implies that $\inf_\varphi \operatorname{dis}(\varphi) = \operatorname{dis}(\mathrm{Id}) = 2 - \sqrt{2}$. Thus, we have shown that $1 - \frac{\sqrt{2}}{2}$ is a lower bound for d_{GH} between these d-spaces.

Following similar arguments, we can show that $\operatorname{dis}(\psi) = \operatorname{codis}(\varphi, \psi) = 2 - \sqrt{2}$ where both maps $(X, d_{zz}^X) \xrightarrow{\varphi} (X, d^X)$ and $(X, d_{zz}^X) \xleftarrow{\psi} (X, d^X)$ are the identity. Thus,

$$\max\big\{ \operatorname{dis}(\varphi), \operatorname{dis}(\psi), \operatorname{codis}(\varphi, \psi)\big\} = 2 - \sqrt{2},$$

proving that $d_{GH}\left((X, d^X), (X, d_{zz}^X)\right) = 1 - \frac{\sqrt{2}}{2}$. △

5.3 Distortion Distance

Theorem 2.1.12 connects the Gromov–Hausdorff distance with distortion and codistortion, two properties of maps between spaces. It is often useful, as it provides bounds for otherwise notoriously difficult-to-compute Gromov–Hausdorff distance [110]. Since distortion and codistortion are also defined for d-maps, we explore whether a similar connection exists for d-spaces.

Definition 5.3.1. The **distortion distance** between directed metric spaces $(\vec{X}, d_{zz}^X)$ and $(\vec{Y}, d_{zz}^Y)$ is defined as

$$d_{\mathrm{dis}}(\vec{X}, \vec{Y}) = \frac{1}{2} \inf_{\vec{F}, \vec{G}} \max\{\operatorname{dis}(\vec{F}), \operatorname{dis}(\vec{G}), \operatorname{codis}(\vec{F}, \vec{G})\},$$

where $\vec{F}\colon \vec{X} \to \vec{Y}$ and $\vec{G}\colon \vec{Y} \to \vec{X}$ are two d-maps.

Unlike the directed Gromov–Hausdorff distance, distortion distance is in general not equal to the classic notion of Gromov–Hausdorff distance on directed metric spaces. We illustrate this fact by continuing Example 5.2.6. In Lemma 5.3.3 we further elaborate how the notions of distortion distance and directed Gromov–Hausdorff distance are related.

Example 5.3.2. Consider $\vec{X}$ and $\vec{X}^*$ as in Example 5.2.6. We observed there that the image of every d-map $\vec{F}\colon \vec{X} \to \vec{X}^*$ is either included in $[0, 1]$ or in $[-1, 0]$. As a consequence,

$$d_{zz}^X(-1, 1) - d_{zz}^X(\vec{F}(-1), \vec{F}(1)) \geq 2 - 1,$$

and $\operatorname{dis}(\vec{F}) \geq 1$. By symmetry we also have that $\operatorname{dis}(\vec{G}) \geq 1$ for any d-map $\vec{G}\colon \vec{X}^* \to \vec{X}$. This means that

$$\max\{\operatorname{dis}(\vec{F}), \operatorname{dis}(\vec{G}), \operatorname{codis}(\vec{F}, \vec{G})\} \geq 1 \tag{5.1}$$

for any pair of d-maps $\vec{F}\colon \vec{X} \to \vec{X}^*$ and $\vec{G}\colon \vec{X}^* \to \vec{X}$. This already implies that $d_{\mathrm{dis}}(\vec{X}, \vec{X}^*) > d_{GH}(\vec{X}, \vec{X}^*) = 0$ (see Example 5.2.6). However, for this example, we can do better and show that $d_{\mathrm{dis}}(\vec{X}, \vec{X}^*) = \frac{1}{2}$. Let us define the d-maps $\vec{F}\colon \vec{X} \to \vec{X}^*$ and $\vec{G}\colon \vec{X}^* \to \vec{X}$ as follows:

$$\vec{F}(t) = \begin{cases} -1, & \text{for } t \in [-1, 0], \\ t - 1, & \text{for } t \in [0, 1], \end{cases} \qquad \vec{G}(t) = \begin{cases} t + 1, & \text{for } t \in [-1, 0], \\ 1, & \text{for } t \in [0, 1]. \end{cases}$$

To compute the distortion of $\vec{F}$, first notice that $d^X_{zz}(x, y) - d^X_{zz}(\vec{F}(x), \vec{F}(y))$ is at most 1 when $x, y \in [-1, 0]$ and it is 0 when $x, y \in [0, 1]$. Now assume $x \in [-1, 0]$ and $y \in [0, 1]$. Then,

$$d^X_{zz}(x, y) - d^X_{zz}(\vec{F}(x), \vec{F}(y)) = y - x - |-1 - (y - 1)| = -x,$$

which is maximized at 1 when $x = -1$. Thus $\operatorname{dis}(\vec{F}) = 1$, and we can follow similar steps to show $\operatorname{dis}(\vec{G}) = 1$ as well. Next, let us compute the codistortion of $\vec{F}$ and $\vec{G}$. By following the definitions of $\vec{F}$ and $\vec{G}$, we see that

$$|d^X_{zz}(x, \vec{G}(y)) - d^X_{zz}(\vec{F}(x), y)| = \begin{cases} |x|, & \text{if } x, y \in [-1, 0], \\ y, & \text{if } x, y \in [0, 1], \\ |x + y|, & \text{if } x \in [-1, 0] \text{ and } y \in [0, 1], \\ 0, & \text{if } x \in [0, 1] \text{ and } y \in [-1, 0]. \end{cases}$$

Because the maximum of these values is 1, $\operatorname{codis}(\vec{F}, \vec{G}) = 1$. Lastly, since (5.1) holds for any pair of d-maps $\vec{F}, \vec{G}$, and we found specific pair for which the maximum equals 1,

$$\frac{1}{2} \inf_{\vec{F}, \vec{G}} \max\{\operatorname{dis}(\vec{F}), \operatorname{dis}(\vec{G}), \operatorname{codis}(\vec{F}, \vec{G})\} = \frac{1}{2}. \qquad \triangle$$

Lemma 5.3.3. *For any metric d-spaces $\vec{X}$ and $\vec{Y}$ with zigzag metric,*

$$d_{\mathrm{dis}}(\vec{X}, \vec{Y}) \geq d_{GH}(\vec{X}, \vec{Y}).$$

Proof. Recall Theorem 2.1.12. Because the family of pairs $(\vec{F}\colon \vec{X} \to \vec{Y}, \vec{G}\colon \vec{Y} \to \vec{X})$ of d-maps is included in the set of all (not necessarily directed) maps between the two spaces, the lemma follows. □

Since the two notions are different, we cannot automatically claim that d_{dis} is a metric on zigzag connected directed metric spaces. In fact, it is not: whenever there is no d-map from one of the d-spaces to the other, distortion distance equals ∞. It is clearly non-negative, and it satisfies symmetry and identity. Below, we show it satisfies triangle inequality as well, and is thus an extended pseudometric on the set of directed metric spaces.

Lemma 5.3.4. *Let $\vec{X}$, $\vec{Y}$, and $\vec{Z}$ be directed metric spaces. Then,*

$$d_{\mathrm{dis}}(\vec{X}, \vec{Z}) \leq d_{\mathrm{dis}}(\vec{X}, \vec{Y}) + d_{\mathrm{dis}}(\vec{Y}, \vec{Z}).$$

Proof. Consider the compositions $\vec{X} \xrightarrow{\vec{F}_1} \vec{Y} \xrightarrow{\vec{G}_1} \vec{Z}$ and $\vec{X} \xleftarrow{\vec{F}_2} \vec{Y} \xleftarrow{\vec{G}_2} \vec{Z}$. It is rather obvious that

$$\begin{aligned} \mathrm{dis}(\vec{G}_1 \circ \vec{F}_1) &\leq \mathrm{dis}(\vec{G}_1) + \mathrm{dis}(\vec{F}_1), \text{ and} \\ \mathrm{dis}(\vec{F}_2 \circ \vec{G}_2) &\leq \mathrm{dis}(\vec{F}_2) + \mathrm{dis}(\vec{G}_2). \end{aligned}$$

Similarly,

$$\mathrm{codis}(\vec{G}_1 \circ \vec{F}_1, \vec{F}_2 \circ \vec{G}_2) \leq \mathrm{codis}(\vec{G}_1, \vec{G}_2) + \mathrm{codis}(\vec{F}_1, \vec{F}_2),$$

which can be shown by expanding the right sight as

$$\begin{aligned} &\mathrm{codis}(\vec{G}_1 \circ \vec{F}_1, \vec{F}_2 \circ \vec{G}_2) = \sup_{x \in X, z \in Z} \left| d\left(x, \vec{F}_2 \circ \vec{G}_2(z)\right) - d\left(z, \vec{G}_1 \circ \vec{F}_1(x)\right) \right| \\ &= \sup_{x \in X, z \in Z} \left| d\left(x, \vec{F}_2 \circ \vec{G}_2(z)\right) - d\left(\vec{G}_2(z), \vec{F}_1(x)\right) + d\left(\vec{G}_2(z), \vec{F}_1(x)\right) - d\left(z, \vec{G}_1 \circ \vec{F}_1(x)\right) \right| \\ &\leq \sup_{x \in X, z \in Z} \left[\left| d\left(x, \vec{F}_2 \circ \vec{G}_2(z)\right) - d\left(\vec{G}_2(z), \vec{F}_1(x)\right) \right| + \left| d\left(\vec{G}_2(z), \vec{F}_1(x)\right) - d\left(z, \vec{G}_1 \circ \vec{F}_1(x)\right) \right| \right] \\ &\leq \sup_{x \in X, y \in Y} \left| d\left(x, \vec{F}_2(y)\right) - d\left(y, \vec{F}_1(x)\right) \right| + \sup_{y \in Y, z \in Z} \left| d\left(y, \vec{G}_2(z)\right) - d\left(z, \vec{G}_1(y)\right) \right|. \end{aligned}$$

As such compositions $\vec{G}_1 \circ \vec{F}_1$ and $\vec{F}_2 \circ \vec{G}_2$ are contained in the set of maps $\vec{X} \to \vec{Z}$ and $\vec{Z} \to \vec{X}$

respectively,

$$d_{\mathrm{dis}}(\vec{X},\vec{Z}) \leq \frac{1}{2} \inf_{\substack{\vec{G}_1\circ\vec{F}_1,\\ \vec{F}_2\circ\vec{G}_2}} \max\{\mathrm{dis}(\vec{G}_1\circ\vec{F}_1), \mathrm{dis}(\vec{F}_2\circ\vec{G}_2), \mathrm{codis}(\vec{G}_1\circ\vec{F}_1, \vec{F}_2\circ\vec{G}_2)\},$$

which is, by the properties shown above, smaller or equal to

$$\begin{aligned}&\frac{1}{2}\inf_{\vec{F}_1,\vec{F}_2} \max\{\mathrm{dis}(\vec{F}_1), \mathrm{dis}(\vec{F}_2), \mathrm{codis}(\vec{F}_1,\vec{F}_2)\} + \frac{1}{2}\inf_{\vec{G}_1,\vec{G}_2} \max\{\mathrm{dis}(\vec{G}_1), \mathrm{dis}(\vec{G}_2), \mathrm{codis}(\vec{G}_1,\vec{G}_2)\}\\ &= d_{\mathrm{dis}}(\vec{X},\vec{Y}) + d_{\mathrm{dis}}(\vec{Y},\vec{Z}).\end{aligned}$$

□

Distortion distance may, however, be an extended *metric* on the set of directed metric spaces up to d-isometry. We have not yet proved or disproved identity, namely that $d_{\mathrm{dis}}(\vec{X},\vec{Y}) = 0$ if and only if $\vec{X}$ and $\vec{Y}$ are d-isometric. We can, however, show the that spaces are d-isometric if an only if $d_{\mathrm{dis}}(\vec{X},\vec{Y}) = 0$ **and** the infimum in the distortion distance is attained.

Theorem 5.3.5. *D-maps $\vec{F}\colon \vec{X} \to \vec{Y}$ and $\vec{G}\colon \vec{Y} \to \vec{X}$, for which* $\mathrm{dis}(\vec{F})$, $\mathrm{dis}(\vec{G})$ *and* $\mathrm{codis}(\vec{F},\vec{G})$ *are all zero, exist if and only if spaces $\vec{X}$ and $\vec{Y}$ are d-isometric.*

Proof. Suppose such maps exist and prove the forward implication. It is straightforward to see that $\mathrm{dis}(\vec{F}) = \mathrm{dis}(\vec{G}) = 0$ means that $\vec{F}$ and $\vec{G}$ are directed isometries, but it also implies they are injective. Let $x, x' \in X$ be such that $\vec{F}(x) = \vec{F}(x')$. Then

$$d^X_{zz}(x,x') - d^Y_{zz}\big(\vec{F}(x),\vec{F}(x')\big) = d^X_{zz}(x,x'),$$

and because $\mathrm{dis}(\vec{F}) = 0$, it must be 0. Since d^X_{zz} is a metric, $x = x'$. A similar argument shows $\vec{G}$ is injective.

They are also surjective, which we show for $\vec{F}$. Suppose there is $y \in Y$ such that $y \notin \vec{F}(\vec{X})$. This means that for any $x \in X$, $d^X_{zz}\big(x,\vec{G}(y)\big) = d^Y_{zz}\big(\vec{F}(x),y\big) > 0$, where the equality follows from $\mathrm{codis}(\vec{F},\vec{G}) = 0$. However, we can choose $x = \vec{G}(y) \in X$, which gives

$$d^X_{zz}\big(x,\vec{G}(y)\big) = d^Y_{zz}\big(\vec{F}(x),y\big) = 0.$$

By contradiction, $Y = \vec{F}(X)$. What is more, $\vec{F}\circ\vec{G} = \mathrm{Id}_Y$, since for any $y \in Y$,

$$d^Y_{zz}\big(\vec{F}\circ\vec{G}(y),y\big) = d^X_{zz}\big(\vec{G}(y),\vec{G}(y)\big) = 0.$$

Similarly, $\vec{G} \circ \vec{F} = \mathrm{Id}_X$, which shows $\vec{F}$ and $\vec{G}$ are inverses of each other.

The backwards implication follows easily. Let $\vec{F}\colon \vec{X} \to \vec{Y}$ be a bijective d-isometry and $\vec{G}\colon \vec{Y} \to \vec{X}$ its inverse. Since both maps are injective, $\mathrm{dis}(\vec{F}) = \mathrm{dis}(\vec{G}) = 0$, and because $\vec{F}$ is a d-isometry,

$$d_X\big(x, \vec{G}(y)\big) = d_Y\big(\vec{F}(x), \vec{F} \circ \vec{G}(y)\big) = d_Y\big(\vec{F}(x), y\big)$$

for all $x \in X$ and $y \in Y$. Consequently, $\mathrm{codis}(\vec{F}, \vec{G}) = 0$. □

Note that if identity holds, Theorem 5.3.5 implies that zero distortion distance is always attained.

5.4 Directed Flat Torus

Consider the d-space $\vec{\mathbb{R}}^2 = (\mathbb{R}^2, \vec{P}(\mathbb{R}^2))$ induced by the product order on $\mathbb{R}^2$, and a relation $\sim$ such that $(x, y) \sim (x+1, y) \sim (x, y+1)$. Denote by $\vec{T}$ the quotient d-space $\vec{\mathbb{R}}^2/_\sim$, and call it the **directed flat torus**.

In the zigzag metric, a ball on the directed flat torus with a small radius (Figure 5.4a) looks like a combination of balls in the Manhattan and euclidean distance. When we increase the radius sufficiently, the ball starts overlapping with itself, which can be seen in Figure 5.4b. Interestingly, in this example the topology given by the zigzag metric is equivalent to the topology given by the metric inherited from the Euclidean space via the quotient. This is because

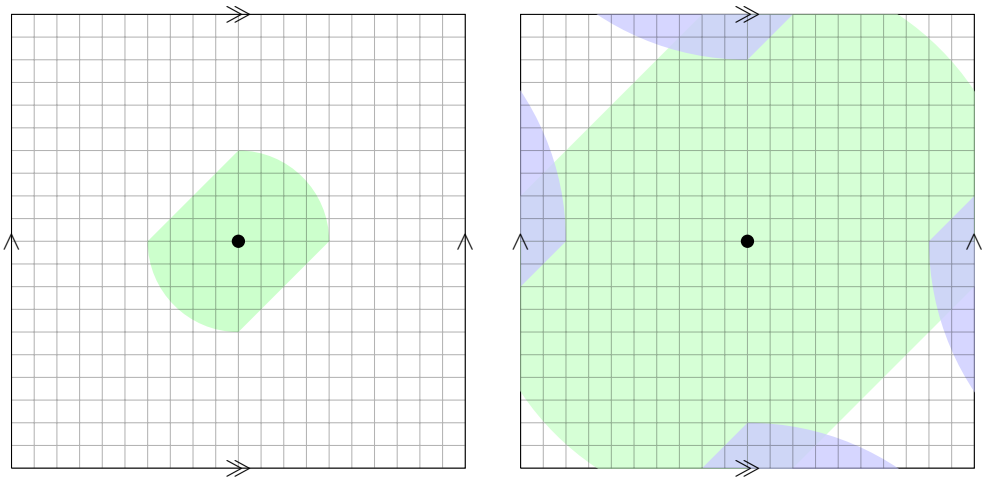
$$B\Big(\frac{r\sqrt{2}}{2}\Big) \subset B_{zz}(r) \subset B(r)$$

for any $r > 0$, where B denotes balls in the metric inherited by $\mathbb{R}^2$, and B_{zz} denotes balls in the zigzag metric.

Proposition 5.4.1. *The diameter of the directed flat torus in the zigzag metric is* $\sqrt{3} - 1$.

Proof. In order to find the diameter, let us compute the infimum of radii for which the ball covers the whole torus. Notice that this is the same as computing the radius for which three of the self-intersections of the boundary of the ball (as illustrated in Figure 5.5) coincide.

First, compute the lengths of segments in green and red from Figure 5.6a. The length of the green segment can be computed by observing the triangle it makes with the point A lying at



(a) Ball with a small radius. (b) Ball with a big radius.

Figure 5.4: Balls in the zigzag metric on a directed flat torus.

distance 1 below the center of the square, as illustrated in Figure 5.6b. It is a right triangle, with the right angle at vertex B. The edge AC is of length R and AB is of length $\frac{\sqrt{2}}{2}$, so the length of BC solves the equation

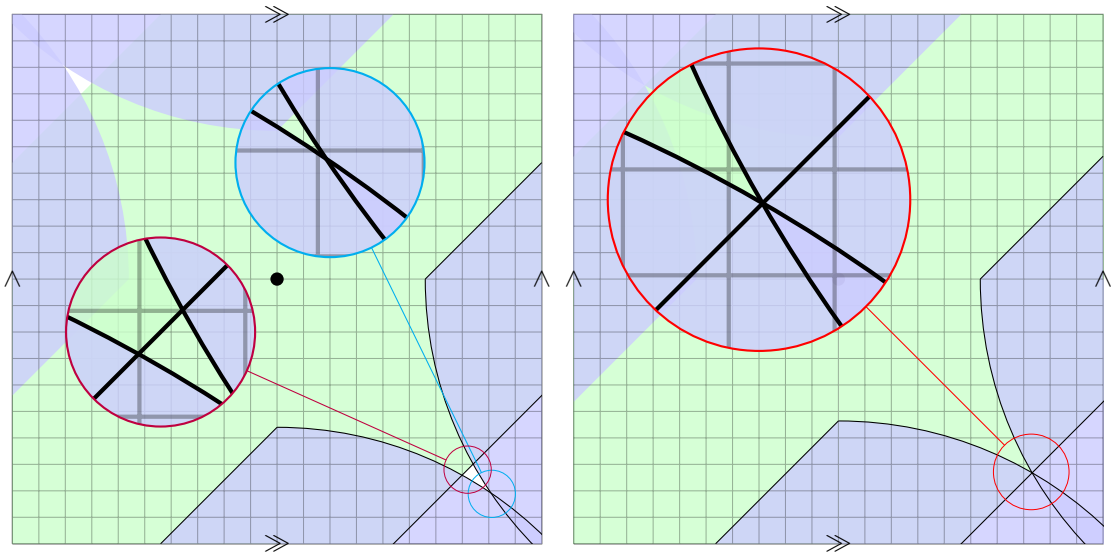
$$|BC|^2 + \frac{1}{2} = R^2. \tag{5.2}$$

Secondly, as already denoted in Figure 5.6c, the length of the red segment is $\frac{\sqrt{2R^2}}{2}$.

Finally, whenever the intersections coincide, these segments connect and form a line between the center of the square and one of its corners. In other words, $|BC| = \frac{\sqrt{2}}{2} - \frac{\sqrt{2R^2}}{2} = \frac{\sqrt{2}}{2}(1-R)$. By plugging it in equation (5.2) we obtain $\frac{1}{2}(1-R)^2 + \frac{1}{2} = R^2$. This further simplifies to equation $R^2 + 2R - 2 = 0$, the only positive solution of which is $\operatorname{diam}(\vec{T}) = \sqrt{3} - 1$. □

5.5 Directed Weighted Graphs as D-spaces

An interesting potential application of our theory developed in previous sections is to compare directed weighted graphs, which are used to model networks such as transportation, social, or even increasingly popular neural networks. Here, we discuss how they can be represented as directed metric spaces.



(a) Self-intersections of the border of the ball with a radius that is slightly smaller than $\operatorname{diam}(\vec{T})$.

(b) Triple self-intersection of the border of the ball with radius $\operatorname{diam}(\vec{T})$.

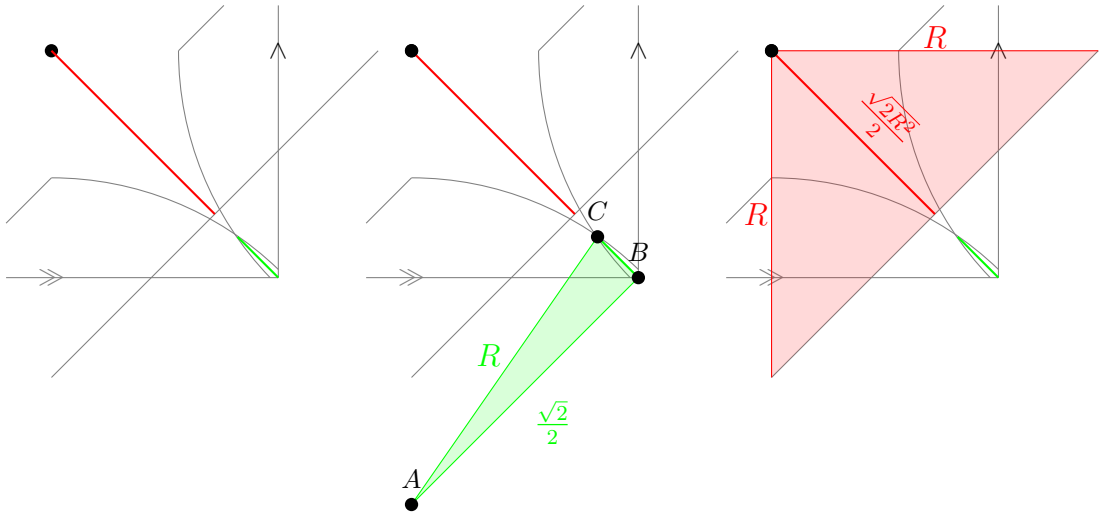
Figure 5.5: Close-up on self intersections of the border of the sphere at radii slightly smaller than and equal to $\operatorname{diam}(\vec{T})$.

First, let us recall some basic definitions and results of graph theory, as stated in [111]. A **weighted graph** $G = (V, E, W)$ consists of a finite set V of **vertices**, a set $E \subseteq V \times V$ of **edges**, and a function $W\colon E \to \mathbb{R}_{>0}$ assigning a **weight** to each edge. A **walk** on G is a sequence of edges $(e_1, \dots, e_n)$ with $e_i = (v_i, u_i)$ such that $u_i = v_{i+1}$ for any pair of consecutive edges. The set of vertices can be equipped with a metric d defined as

$$d(v, u) = \inf \sum_{e_i} W(e_i), \tag{5.3}$$

where the infimum runs over all walks on G with starting vertex $v_1 = v$ and last vertex $u_n = u$. This makes (V, d) a finite metric space. Interestingly, every finite metric space can be realized by a weighted graph as defined here (when the weight function is positive).

Let $G = (V, E, W, \alpha, \beta)$ be a directed weighted graph, where V, E and W are as before, and $\alpha, \beta\colon E \to V$ are functions assigning the **source**, $\alpha(e)$, and the **target** $\beta(e)$, to each



(a) The proof of Proposition 5.4.1 is based on computing the lengths of the segments in red and green.

(b) The green triangle is used in computation of the length of green segment.

(c) The red triangle is used in computation of the length of red segment.

Figure 5.6: Sketches accompanying the proof of Proposition 5.4.1.

edge $e \in E$. The direction of edges is thus induced by the functions α and β. Notice that the weights can be assumed to be non-negative, as a negative weight can be replaced by its absolute value with additional reversal of direction on the edge in question.

We modify the construction of CW-complexes to endow G with both a directed and a metric structure.

Definition 5.5.1. A **graph complex** of G, denoted X_G, is a topological space constructed as follows:

- $X_G^0 = V$,
- X_G^1 is formed by attaching one 1-cell for each edge $e \in E$: let $\vec{I}(e)$ be the directed interval $[0, W(e)]$ with the direction inherited from partial order $\leq$, and glue it onto x_G^0 via map $\varphi_e \colon \partial\vec{I}(e) \to X_G^0$ with $\varphi_e(0) = \alpha(e)$ and $\varphi_e(W(e)) = \beta(e)$.

Directed Structure on a Graph Complex Observe that a graph complex X_G is an example of a 1-dimensional CW-complex. Its construction via gluing maps makes it a quotient space and the directed structure on it is inherited from its 1-cells. To aid intuition, however, we work out the exact conditions for a path $\gamma\colon I \to X_G$ to be directed.

For a path $\gamma\colon I \to X_G$, let $\gamma^{-1}(V)$ be the set of all parameters which γ maps to the vertex set. Take one representative of each connected component of $\gamma^{-1}(V)$ and order them increasingly, constructing a sequence $0 \leq t_0 < t_1 < \ldots < t_n \leq 1$. Note that this is indeed a finite sequence, but it may be empty. Further, there may be infinitely many such sequences for γ, but denote any of them as $T(\gamma)$. Define the set $\mathbb{I}\big(T(\gamma)\big)$ as

$$\{(t_i, t_{i+1}) \mid i = 0, \ldots, n-1\} \cup \{(0, t_0) \text{ if } t_0 \neq 0\} \cup \{(t_n, 1) \text{ if } t_n \neq 1\}$$

if the sequence is not empty, and as $\{(0,1)\}$ if it is. Then the image $\gamma(J)$ of any interval J in $\mathbb{I}\big(T(\gamma)\big)$ is contained entirely within a 1-cell (with boundary).

Definition 5.5.2. If γ is a constant path with $\gamma(I) \subset X_G^0$, or if $\gamma|_J$ is weakly increasing in the order of its corresponding 1-cell for each $J \in \mathbb{I}\big(T(\gamma)\big)$, then γ is called a **directed path** or **d-path**.

Remark 5.5.3. It is not immediately clear that Definition 5.5.2 is well-defined. However, suppose $T(\gamma) = \{t_i\}_{i=1}^n$ and $S(\gamma) = \{s_i\}_{i=1}^n$ are two sequences of vertex parameters for γ that differ only at i, *i.e.* $t_i < s_i$ and $s_j = t_j$ for $j \neq i$. Then $\gamma|_{[t_i, s_i]} \equiv \gamma(t_i)$. Thus, γ is a d-path with respect to one sequence if and only if it is a d-path with respect to any other sequence.

This new definition of a d-path is potentially confusing, but the following result justifies it.

Proposition 5.5.4. *For a directed graph $G = (V, E, \alpha, \beta)$, the pair $(X_G, \vec{P}(X_G))$, where X_G is the graph complex of G and $\vec{P}(X_G)$ is the set of d-paths on it, is a d-space.*

Proof. If $\vec{P}(X_G)$, the set of d-paths, is a directed structure, it must contain all constant paths, and it must be closed under partial reparametrizations and concatenations.

All constant paths are weakly increasing in the order of any cell they map to, and are consequently all d-paths. If γ is a d-path and $h\colon I \to I$ is continuous and weakly increasing, then for each $T(h\gamma)$

and each $(a,b) \in \mathbb{I}\big(T(h\gamma)\big)$, its image $\big(h(a),h(b)\big)$ with h is in $\mathbb{I}\big(T(\gamma)\big)$, and the corresponding 1-cell in X_G is the same for both intervals, say e. Since $\gamma|_{\left(h(a),h(b)\right)}$ is weakly increasing with respect to $\leq_e$, then so is $h\gamma|_{(a,b)}$. Thus, $h\gamma$ is a d-path.

Let $\gamma \star \delta$ be a concatenation of d-paths γ and δ with respective sequences $T(\gamma) = \{t_i\}_{i=0,\dots,n}$ and $T(\delta) = \{s_i\}_{i=0,\dots,m}$. Without loss of generality, assume $t_n \neq 1$ and $s_0 \neq 0$. Then the analogous sequence $T(\gamma \star \delta)$ for the concatenation is

$$0 \leq \frac{t_0}{2} < \ldots < \frac{t_n}{2} < \frac{s_0+1}{2} < \ldots < \frac{s_m+1}{2} \leq 1.$$

A similar argument as before can be used to show that $\gamma \star \delta|_J$ is weakly increasing in the order of its corresponding cell for any $J \in \mathbb{I}\big(T(\gamma \star \delta)\big)$, with additional justification needed only for $J = (\frac{t_n}{2}, \frac{s_0+1}{2})$. However, since $\gamma \star \delta$ is weakly increasing on a $J - \{\frac{1}{2}\}$, it is weakly increasing on J as well. Thus, a concatenation of d-paths is a d-path. □

Definition 5.5.5. D-space $\vec{X_G} = (X_G, \vec{P}(X_G))$, where G is a directed graph, is called a **directed graph complex** or **d-graph complex** of G.

Note that, since the construction of d-graph complexes is heavily inspired by CW-complexes, there is a natural choice for the definition of a d-map on them.

Definition 5.5.6. A continuous map $F\colon \vec{X}_G \to \vec{X}_H$ between d-graph complexes is a **d-map** if it is cellular, *i.e.* $F(X^i_G) \subseteq X^i_H$ for $i = 0, 1$, and the path $\gamma\colon I \to \vec{X}_H$ defined by $\gamma(t) = F \circ \varphi_e(W(e) \cdot t)$ is directed for each edge e.

Metric Structure on a Graph Complex The adopted definition also makes it possible to give a natural notion of distance between points in X_G. First, let us define a length function

$$\ell_G\colon (X_G)^I \to [0,\infty]$$
$$\ell_G(\gamma) = \inf_{0=s_0<s_1<\ldots<s_m=1} \Big(\sum_{i=1}^{m} d^{e_i}\big(\gamma(s_{i-1}), \gamma(s_i)\big)\Big),$$

where the infimum is over all finite sequences $\{s_i\}$ such that $V \cap \gamma(I) \subseteq \{\gamma(s_i)\}$, e_i is the cell containing $\gamma\big((s_{i-1}, s_i)\big)$, and d^{e_i} is the metric on e_i that is inherited from the euclidean metric

on $[0, W(e_i)]$. Note, that on d-paths, ℓ_G is simply

$$\begin{aligned}\ell_G(\gamma) &= \sum_{(a,b)\in\mathbb{I}(\gamma)} d(\gamma(a),\gamma(b)) \\ &= d(\gamma(0),\gamma(t_0)) + d(\gamma(t_n),\gamma(1)) + \sum_{i=1}^{n} W\Big(\big(\gamma(t_{i-1}),\gamma(t_i)\big)\Big),\end{aligned}$$

where d denotes the metric on the correct 1-cell in each case. Whenever the d-path γ connects vertices, *i.e.* when $\gamma(\{0,1\}) \subseteq X_G^0$, this further simplifies to $\sum_{i=1}^{n} W\Big(\big(\gamma(t_{i-1}),\gamma(t_i)\big)\Big)$.

From now on, assume all paths on X_G are rectifiable with respect to ℓ_G.

Theorem 5.5.7. *The ℓ_G-induced distance on X_G, $d^{\ell_G}(x,y) = \inf \ell_G(\gamma)$, where infimum runs over all paths $\gamma \in X_G^I$ with $\gamma(0) = x$ and $\gamma(1) = y$, is an extended metric on X_G.*

Proof. This is a rather easy consequence of the construction. As a sum of metrics is non-negative, so is an infimum of such sums. Since the graph G is finite, the infimum is also always obtained, which further gives that $d^{\ell_G}(x,y) = 0$ if and only if $x = y$. It is symmetric, as the reverse of each path in $(X_G)^I$ is also in $(X_G)^I$. Lastly, let γ be a path between x and y in x_G that minimizes ℓ_G, and δ be a path between y and z in x_G that minimizes ℓ_G. Their concatenation, $\gamma \star \delta$ is then a path between x and z, and its length is $\ell_G(\gamma \star \delta) = d^{\ell_G}(x,y) + d^{\ell_G}(y,z)$. It follows readily that $d^{\ell_G}(x,z) \leq d^{\ell_G}(x,y) + d^{\ell_G}(x,z)$. □

Corollary 5.5.8. *If G is connected, then so is X_G and d^{ℓ_G} is a metric on it.*

Remark 5.5.9. When restricted to the vertices, the metric d^{ℓ_G} agrees with the usual notion of the weight-induced metric on a graph as defined in (5.3).

Remark 5.5.10. The graph complex X_G with topology induced by d^{ℓ_G} is a **metric graph** [112, 113], a commonly studied object in discrete geometry.

If G is connected, the zigzag distance d_{zz} induced on it by d-paths equals d^{ℓ_G} because we allow reversal of direction in zigzag paths. For simplicity sake, denote both by simply d^G. This immediately implies that $(\vec{X_G}, d^G)$ is a directed metric space.

Chapter 6

Classification of Gene Expression Data

With the advancement of modern sequencing technologies it has become possible to study how genetic information flows within individual cells. Although the fundamental genetic code in the form of DNA is generally the same across all cells in an organism, cells clearly differ in function, specializing to perform specific tasks. Properties of a cell – such as its function and overall health – can be inferred by examining how many local RNA transcriptions of each gene are present in it. While we now have the ability to obtain this kind of data, drawing meaningful conclusions requires understanding the role of each gene. Some genes have been extensively studied, and the processes they influence and the traits they contribute to are well-understood. However, for many genes, their exact roles remain unknown. Sometimes, these functions can be uncovered by observing other biological aspects of a cell. More often, though, we only have access to data that shows which genes are active simultaneously. The challenge then is to learn the semantics that allows us to translate patterns of gene expression into biological function.

Clustering is a fundamental technique in exploratory data analysis, providing a way to group similar data points in an unsupervised manner. It is particularly important in the field of bioinformatics, where it is an integral part of standard pipelines for analysis of gene expression data sets. These data sets often exhibit high variability and dimensionality, significant noise, feature

redundancy, and are often reused in other studies due to the high cost and duration of sequencing experiments. Additionally, researchers conducting such analyses often come from diverse backgrounds in biology and related fields rather than mathematics or computer science. Consequently, they may lack the specialized knowledge required to select the appropriate clustering method and fine-tune its parameters for the specific problem at hand. These factors complicate the clustering process, making it crucial to identify methods that not only perform well but also maintain robustness across different transformations, metrics, and parameter settings.

In this chapter, we explore a particularly complex gene expression data set. Along with exhibiting the aforementioned challenges, its small size further intensifies these difficulties. Our goal is to identify substructures present in the data set via clustering analysis. Since previous research following standard clustering pipelines for gene expression data has not been successful on this data set, we consider alternative approaches. Particular attention is given to a recently introduced clustering method inspired by the study of persistent homology. We illustrate that this method, called k-cluster [22], is especially robust and intuitive to use, and as a consequence has incredible potential for use within the genomic research community.

The dataset under investigation is introduced in Section 6.1. We evaluate four distinct clustering methods, each representing a different algorithmic paradigm. All methods, including the one for determining which genes drive cluster separation, are described in Section 6.2. The results of our analysis are detailed in Sections 6.3 and 6.4 and summarized in Section 6.5. The code implementing custom methods for the experiments and the resulting visualizations are publicly available [114].

6.1 Thirsty Fly Data Set and Classification Tasks

The dataset of Thirsty Flies was obtained as part of a study of the cellular changes in the brain of *Drosophila melanogaster* (fruit fly) triggered by water deprivation [21]. The authors of the study were observing water-seeking behavior and obtained gene expression data via single-cell sequencing in four settings: when the flies were sated, when they have been deprived of water for 6 hours, when they have been deprived of water for 12 hours, and 45 minutes after they have been rehydrated after having been deprived of water for 12 hours. In each of the settings they used 24 flies, 12 male and 12 female, from which they obtained gene expression information of approximately $600,000$ brain cells. The union of these data was split into seven classes of brain cells based on the prior knowledge of which gene codes for which type of a brain cell (see Figure 6.1):

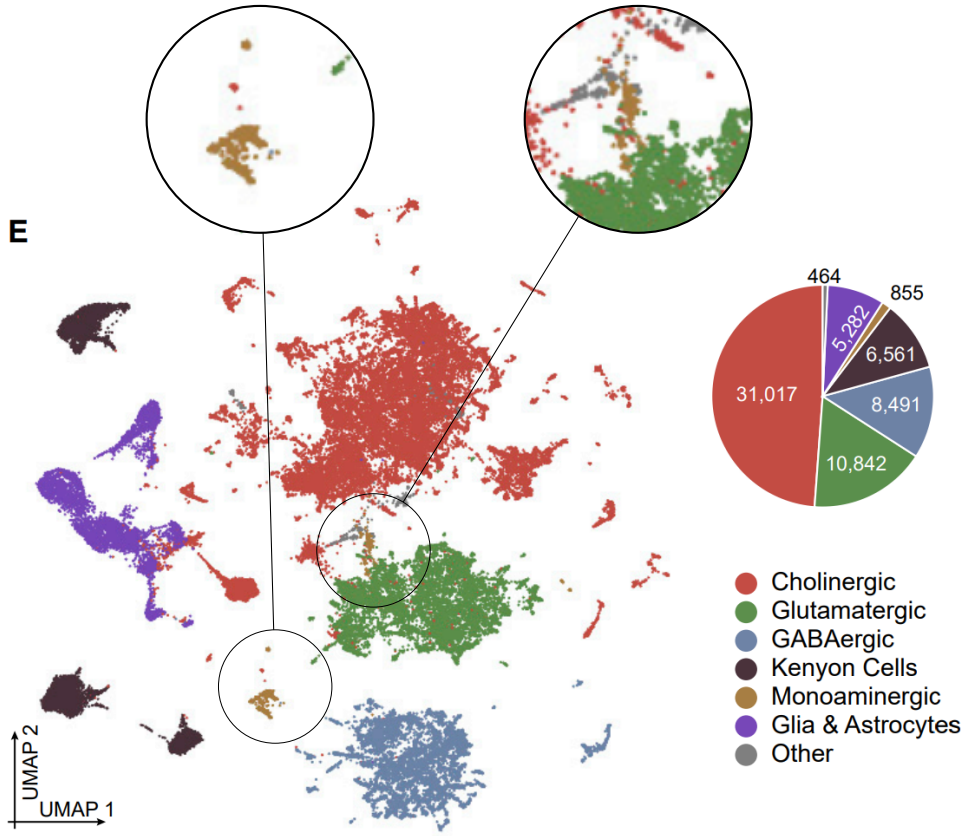


Figure 6.1: Clustering of drosophila brain cells into 7 clusters, plotted with UMAP [21]. Monoaminergic neurons are drawn in yellow and shown in magnified detail.

1. **Cholinergic neurons**. The division of neurons into subtypes is usually driven by which molecule, called a **neurotransmitter**, is used to regulate passing the signals to another cell via synapse. The main neurotransmitter of cholinergic neurons is acetylcholine.
2. **Glutamatergic neurons**. The neurotransmitter of these neurons is glutamate. It is the main **excitatory** neurotransmitter of our nervous systems – it promotes passing the signals along as opposed to **inhibitory** neurotransmitters, which prevent them from being passed any further.
3. **GABAergic neurons** use gamma-aminobutyric acid (GABA) neurotransmitter, which acts as an inhibitor and balances the excitatory function of glutamate. Thus, good co-

operation of glutamatergic and GABAergic neurons is vital for proper neurologic function.

4. **Kenyon cells**j are neurons characterized by their location rather than their neurotransmitter. Together with glial cells, they comprise dense neuronal networks, called **mushroom bodies**, present in the brains of arthropods. They are responsible for learning and memory based on odor information.

5. **Glia & astrocytes**. Glial cells are non-neuronal cells that provide physical and chemical support to neurons and maintain the stable conditions in their environment. They are located in the central and peripheral nervous system. A particular subtype of glial cells are called astrocytes.

6. **Monoaminergic neurons.** Monoamine is a compound that contains one amino group connected to an aromatic ring by a two-carbon chain, and acts as a neurotransmitter in monoaminergic neurons. In mammals, the most common monoamine neurotransmitters are dopamine, epinephrine, norepinephrine and serotonin. In insects (such as drosophila), however, epinephrine and norepinephrine (known also as adrenaline and noradrenaline) are synthesized only in trace amounts. Instead, monoamines tyramine and octopamine are much more prevalent, and they replace epinephrine and norepinephrine in their role as adrenergic transmittors. They have many roles including in regulating behavior, sensory processing, locomotion and metabolism [115].

7. Cells that do not belong to any of the above listed classes, were allocated to a common class "Other".

(We used [116, Ch. 2 and 13] as the general source for the information about different neurotransmitters and neurons.) Five of the labeled classes, namely the cholinergic, glutamatergic, GABAergic neurons, Kenyon cells and glia & astrocytes, have been clustered further. However, when trying to divide monoaminergic neurons into four subclasses, corresponding to the four main monoamine neurotransmitters in drosophila (dopamine, serotonin, tyramine and octopamine), the same clustering pipeline struggled to produce coherent clusters, likely due to the comparatively small number of monoaminergic neurons. In this work, we focus on the class of the monoaminergic neurons only. We wish to understand several levels of its heterogeneity, beginning with classifying the neurons into the four mentioned subtypes.

The data set is stored as a 855×9999 matrix M, where each row is a gene expression vector of a monoaminergic neuron. The entry $M(i, j)$ is an integer signifying the strength of expression of j-th gene in i-th neuron. To avoid issues in subsequent analysis, some entries were removed from

the Thisrty Fly data set. Firstly, genes that are never expressed on monoaminergic neurons, *i.e.* whose columns are zero, were removed. Next, groups of rows with the same expression pattern were identified. One row per each group was chosen randomly and kept, while the others were discarded. As a result, the matrix M was reduced to 765 rows and 9859 genes.

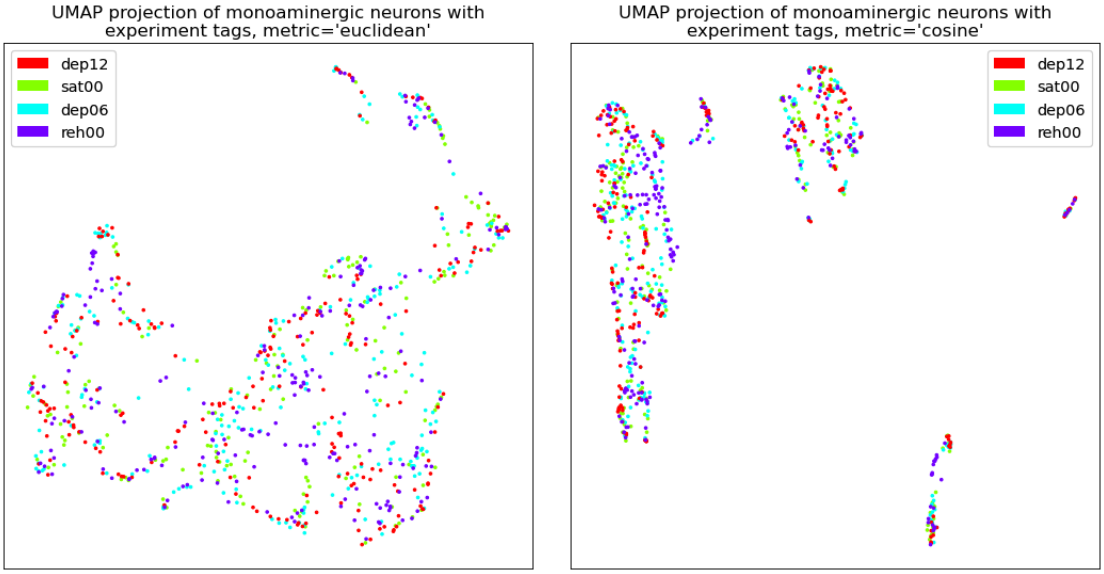
The genes, *i.e.* columns, are not ordered in any particular way, whereas the order of neurons is as follows:

- first 153 rows (rows 1 to 153) belong to neurons of sated flies (tagged with `'sat00'`),
- next 204 rows (rows 154 to 357) belong to neurons of flies that have been dehydrated for 6 hours (tagged with `'dep06'`),
- next 202 rows (rows 358 to 559) belong to neurons of flies that have been dehydrated for 12 hours (tagged with `'dep12'`),
- last 206 rows (rows 560 to 765) belong to neurons of flies that have first been dehydrated for 12 hours and then rehydrated (tagged with `'reh00'`).

A visualization of the point cloud with the points colored based on the experiment membership is shown in Figure 6.2. It is important to note that the monoaminergic neurons were confirmed not to be involved in the response to water-deprivation by the differential expression analysis carried out in [21]. The experiment related variance in the data is therefore not expected to be significant, and nothing is done to mitigate its effects. This separation of M based on the setting in which the data were obtained is in general ignored.

The clustering analysis on Thirsty Fly data set can be split into two tasks, as follows.

1. **Classification of Monoaminergic Neurons.** We expect to find four subtypes of monoaminergic neurons in our data, each consisting of neurons specialized for one of the four most common monoaminergic neurotransmitters in drosophila: dopamine, serotonin, octopamine and tyramine. Note that this data set is not labeled and so the clustering is unsupervised. However, we are given some prior knowledge about which genes code for which subtype. We present these relations together with the results of several clustering methods and pipelines in Section 6.3.
2. **Classification of Dopaminergic Neurons.** After successful classification of monoaminergic neurons, we wish to identify further subtypes of dopaminergic neurons. Due to the fact that we have an even smaller data set, no prior knowledge on which genes code



(a) UMAP projection of the point cloud in euclidean metric.

(b) UMAP projection of the point cloud in cosine distance.

Figure 6.2: UMAP projections of gene expression profiles of monoaminergic neurons in (a) euclidean metric and (b) cosine distance. The colors of the points correspond to which experiment the data was obtained from as listed in the legend.

for which subtype, and do not even know how many subtypes we should expect, this classification is significantly more challenging. As the data set used for this task relies on which neurons were classified as dopaminergic in the first task, we postpone its definition and description to Section 6.4, where we also present our results.

Note that as this data set was already used in [21], pre-processing was performed before we obtained the data set. Unless stated otherwise, we skipped the pre-processing step of the standard pipeline and performed the later steps only.

6.2 Methods

The methods used in our exploratory analysis of the Thirsty Fly data set are presented here. In particular, k-cluster, the relatively new, persistence homology-inspired clustering method is

described in Section 6.2.1, the more standard and popular clustering methods used to compare k-cluster to are presented in Section 6.2.2, while the method for detecting genes that drive cluster separation is detailed in Section 6.2.4.

6.2.1 Persistence-inspired Clustering Method: k-cluster

Recently, a statistical study of distance-based persistent homology [117] showed that it exhibits universal behavior in the following sense: Take "persistence" values of noisy features in the persistence diagrams arising from random point clouds. Their distribution, as we increase the cardinality of underlying point clouds, is independent of the method used to generate the point cloud. To obtain this result, the additive measure of persistence, $death - birth$, that is most commonly used in TDA, must be replaced with the multiplicative value $death/birth$. However, the birth times of all 0-dimensional features of distance-based persistent homology are 0, and as a consequence their multiplicative persistence values are not well-defined. To mitigate that, a novel filtration called k-cluster [22] has been proposed. It deviates from the standard distance-based filtrations in the degree 0 only, and as 0-homology conveys connectivity information, k-cluster filtration induces a novel topologically-informed clustering method, which we also call k-cluster. We describe them both in this section, closely following the original paper [22].

k-cluster Filtration

Although k-cluster filtration can be generalized to simplicial complexes of arbitrary dimension, all the necessary information for 0-dimensional homology is contained in its 1-skeleton or, in other words, an undirected weighted graph. In geometric settings (such as ours), an undirected weighted graph $G = (V, E, W)$ consists of three parts: a set of vertices V which lie in a metric space, a set of edges $E \subseteq V \times V$ between the vertices, and a weight function $W \colon E \to \mathbb{R}_{\geq 0}$ which assigns a weight $d(v_1, v_2)$ to each edge $e = (v_1, v_2)$, where d is the metric in our metric space.

One way to obtain a filtration on an undirected weighted graph is to first define a filtration function. That is a function $\tau \colon (V \cup E) \to \mathbb{R}_{\geq 0}$ for which $\tau(e) \geq \max(\tau(v_1), \tau(v_2))$ holds for all $e = (v_1, v_2) \in E$. The choice of τ then induces a filtration $\{\mathcal{G}_t\}_{t \geq 0}$ of the graph G, where

$$\mathcal{G}_t = \{\sigma \in V \cup E \mid \tau(\sigma) \leq t\}.$$

The standard is to set $\tau(v) = 0$ on all vertices and $\tau(e) = W(e)$ on all edges (this is the case for both Čech and Vietoris-Rips filtered simplicial complexes). However, that choice induces a

filtration where all connected components are born at time 0, which means the chosen measure of persistence, *death/birth*, is not well-defined on those components. This can be avoided by choosing a different filtration function. To obtain k-cluster filtration, define $N_t(v)$ for each vertex $v \in V$ and a value $t > 0$ to be the number of vertices in the connected component of $\mathcal{G}_t$ that contains v, where $\{\mathcal{G}_t\}_{t\geq 0}$ is the filtration induced by the standard filtration function. For a fixed $k \geq 1$, the k-cluster filtration function can then be defined as

$$\tau_k(v) = \inf\{t \mid N_t(v) \geq k\}$$
$$\tau_k((v_1, v_2)) = \max(\tau_k(v_1), \tau_k(v_2), W(v_1, v_2)).$$

Denote the filtration that τ_k induces on G by $\{\mathcal{G}_t^{(k)}\}_{t\geq 0}$ and call it the k-cluster filtration. (Note that $\mathcal{G}_t \equiv \mathcal{G}_t^{(1)}$.)

As the name suggests, clusters of size k play a central role in k-cluster filtration. Let us illustrate this fact with the following observations. No point is a component on its own – it is born only once it is a part of a cluster containing at least k points. Further, each birth time b corresponds to merging two components $C_1, C_2 \in \mathcal{G}_{b-\epsilon}$ of the standard filtration that contain less than k points each, and more than k points combined. On the other hand, each death time corresponds to an edge that merges two components, each of which contains more than k points. The choice of k therefore decides the scale at which meaningful substructure begins to appear.

k-cluster Clustering

Once the k-cluster filtration is computed, persistent homology can be applied to it. The resulting persistence module is p.f.d., so it can be summarized in a persistence barcode or a persistence diagram. The topological mode analysis tool (ToMATo) [118] can then be modified to obtain clusters corresponding to most persistent features from the persistence diagram in 0^{th} dimension. The resulting clustering method is called **k-cluster**, and we outline its algorithm here.

k-cluster takes the following input: the point cloud X, the metric of the ambient space (given in terms of the distance matrix for the point cloud), a parameter $k \geq 1$, and a number N of desired clusters. The values of k-cluster filtration function on the full graph are computed, followed by persistent homology computation. A threshold α for multiplicative values is determined so that N points in the persistence diagram have a multiplicative value higher than α. Next, an agglomerative process is started on the point cloud, using a union-find data structure to keep track of components. This means each point is in its own separate component at initiation, and

components are updated during an iteration over the edges in an increasing order given their filtration value. Say the iteration is at an edge $e = (u, v)$ and u and v belong to different components C_u and C_v at this stage (as otherwise adding the edge does not affect the components). If the fraction

$$\frac{\tau_k(e)}{\min\{\tau_k(w) \mid w \in C_u \cup C_v\}}$$

is smaller than the threshold α, components C_u and C_v are merged. If not, the iteration moves on to the next edge. Once all edges have been iterated through, the resulting components are the obtained clusters.

Note that k-cluster filtration and the resulting persistence diagram can inform the choice of the number of clusters N. Namely, the gaps in the (decreasingly) ordered multiplicative values of points in the persistence diagram hint at the number of clusters inherently present in the data. If, say, the gap between n^{th} and $(n+1)^{\text{st}}$ multiplicative value is especially large, it suggests the choice of $N = n$. In order to use this insight, persistence diagram associated with k-cluster filtration needs to be computed prior to running k-cluster.

6.2.2 Clustering Methods Used for Comparison

The methods we chose for comparison with k-cluster are k-means, spectral clustering [119, 120] and the Louvain algorithm. We chose the first two due to their popularity in the wider data science community, while the Louvain algorithm is chosen as a representative of community detection algorithms that are widely used for clustering analysis on gene expression data sets specifically. In this section, we provide an overview of the functioning of these methods.

k-means. We use the implementation of k-means that is available in the `Scikit-learn` library in Python [121]. We leave all parameters of the method at their default values except for the number of expected clusters, `n_clusters`, and the matrix containing the coordinates of each point in our point cloud as rows.

Let $X \subseteq \mathbb{R}^m$ be the point cloud we wish to cluster into n clusters. The central role in k-means clustering is played by **centroids**, n points in $\mathbb{R}^m$ denoted by $c_1, \ldots, c_n$. They can be initialized in many ways, and different initializations might bring gains in (speed of) convergence. However, for the purpose of this explanation we may assume they are initialized randomly. With that, an

iterative process begins: each point is assigned to cluster i, if its closest centroid in euclidean metric is c_i. New positions of the centroids are then assigned, with c_i being set to the mean of all points currently in cluster i. This process is repeated until the centroids converge, *i.e.* until the difference in position of each centroid changes by less than a threshold ϵ between consecutive steps. Note that ϵ is set to 0.0001 by default in the implementation we use, and it has an additional parameter `max_iter` (set to 300 by default) used to stop the iterative process if convergence is not reached.

The biggest advantage of k-means is its computing efficiency but it has many disadvantages [122]. It may struggle to converge to the global optimum, is sensitive to outliers and centroid initialization, and requires the number of clusters to be provided by the user. Additionally, it assumes that true cluster membership can be determined using the distance of a point to the cluster centroid, implying the cluster is roughly spherical in shape. This renders it inappropriate for use when that is not the case.

Spectral clustering. We use the implementation of spectral clustering that is available in the `Scikit-learn` library in Python [123]. Out of many parameters that can be tuned, we leave many at their default values with the exception of four:

- `n_clusters`: the number of clusters the data should be grouped into.
- `n_neighbors`: the number of neighbors to include in the construction of affinity matrix.
- `affinity`: the parameter instructing how to define the affinity matrix. Our choice, `'precomputed_nearest_neighbors'` allows us to pass a distance matrix from which the affinity matrix is constructed. As a consequence, we also pass
- `matrix`: distance matrix to use when constructing the affinity matrix.

The steps taken by spectral clustering are roughly the following. The geometry of the data set is given by a distance matrix DM (passed as parameter `matrix`), which is used to define an affinity matrix A. To be precise, for each point x_i in our dataset, and each of its k nearest neighbors x_j (where k is passed as parameter `n_neighbors`) set

$$A_{ij} = A_{ji} = \exp(-DM_{ij}^2),$$

and set all other entries in A to zero. Let D be a diagonal matrix with D_{ii} being the sum of i-th row in A. The data set is then projected to the subspace of the ambient space, spanned

by the largest N eigenvectors of the Laplacian $L = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ where N is the number of clusters we wish to obtain, passed to the method as `n_clusters`. The resulting point cloud is first normalized, and then clustered using a different clustering method. The default choice in the `scikit-learn` implementation is k-means, which is also the method we chose.

During our analysis in Section 6.3.4, the only parameters we vary are `n_clusters`, `n_neighbors` and `matrix`.

According to [122] there are many advantages of using spectral clustering. It requires only a few basic parameters, is suitable for use on high-dimensional data sets of arbitrary shape, and is not sensitive to outliers, while the main drawback is its time complexity and the requirement of passing the expected number of clusters.

Louvain method. We use the implementation of Louvain community detection method that is available in the `CDlib` library in Python [124]. Out of the possible parameters that can be passed to the method, we use two:

- G: a weighted graph built on the point cloud on which the communities should be detected.
- `resolution`: the parameter controlling the number and size of obtained communities. High values result in higher number of smaller communities, lower values result in smaller number of bigger communities. Setting the resolution value to 1 instructs the use of the standard Louvain method.

Let us postpone addressing the way we construct a graph on the point cloud, and first list the steps Louvain method takes to obtain communities, as described in [125]. It is agglomerative in nature, and it merges communities by considering gains in modularity

$$Q = \frac{1}{2m}\sum_{i,j}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\delta(c_i, c_j),$$

where A_{ij} is the weight of the edge between vertices i and j, $k_i = \sum_j A_{ij}$ is the sum of the weights of edges between i and its neighboring vertices, c_i is the community to which i belongs, $\delta(c_i, c_j)$ is 1 if the communities $c_i = c_j$ and 0 otherwise, and $m = \frac{1}{2}\sum_{i,j} A_{ij}$. Note that the implementation we use actually uses a modified version of modularity with a resolution parameter, which

allows the user to control the size and number of obtained communities. For more details, please refer to [126].

At the initial stage, there is one community per vertex in the graph. The vertices of the graph are iterated over many times, and for each vertex i all its neighbors j are considered. It is evaluated if removing i from its community and placing it in the community of j would increase modularity. If this gain in modularity is maximized for neighbor j of i and it is positive, i is indeed moved to the community of j. When the communities do not change anymore, the first phase is concluded. In the second phase, a new weighted graph is constructed on the communities obtained in the first phase. The process is then repeated over and over until maximum of modularity is attained.

As mentioned above, a weighted graph must be constructed on the point cloud in order to use the Louvain method. Since the weight of the graph must be higher for closer points (so that gain in modularity is related to merging communities of closer points), one cannot set the distance matrix between points as the weight of the edge connecting them. Thus, a transformation is applied to the distance matrix to obtain the weight matrix. The chosen transformations vary depending on the application, which is why we specify the chosen weight matrix whenever we use the Louvain method. Once the weight matrix is chosen, the graph is constructed using the `from_numpy_array` method from `networkx` library in Python [127].

The advantages of the Louvain method can at the same time be drawbacks, depending on its application. The fact that a graph needs to be built on the data points can be a benefit, as it provides great flexibility. Depending on the choice of the weight matrix, the method can encompass any geometric aspect of the data set and can be steered to group points based on user defined criteria. In addition, it does not take the number of clusters as a parameter, and makes no assumption on the shape and distribution of clusters. However, the dependence on the construction of a weighted graph and a rather mysterious parameter `resolution` that controls the number of clusters obtained, make it difficult to apply successfully as an unskilled user. A lot of thought needs to be put into choosing the weights, and many resolution parameters tested. As a consequence, it is rarely utilized to its full potential and most of its implementations offer limited tuning options (for example in libraries for genomic data analysis such as Seurat [128, 129, 64] and Bioconductor [47, 48, 130].

6.2.3 Evaluation of Clustering Results

In supervised clustering, when the input data is labeled based on which class it belongs to, the outcome of clustering can be assessed using various metrics based on these labels. Although

our classification tasks are unsupervised, prior knowledge about marker genes helps us identify a high-confidence subset of input data for the task of classifying monoaminergic neurons (as described in detail in Section 6.3). This subset can then be used to evaluate the performance of chosen clustering schemes, using a metric called **entropy** [131, Section 16.3].

Remark 6.2.1. The main reason why we chose entropy as opposed to other, more popular evaluation metrics (for example accuracy) is that the number of clusters obtained might not match the number of expected classes. Even when that is the case, many metrics would require us to manually determine the assignment to which class is "correct" for each cluster in order to compute the metric. This, however, is not true for entropy.

Definition 6.2.2. Let S be a finite set and $\Omega = \{\omega_1, \omega_2, \ldots, \omega_k\}$ its partition. Further, let $c\colon S \to [n]$ be a function which classifies elements of S into n classes. The **entropy of a cluster** ω_j with respect to classification c is the sum

$$H(\omega_j) = -\sum_{i \in [n]} P(\omega_j; i) \log_2 P(\omega_j; i),$$

where $P(\omega_j; i)$ is the proportion of points in ω_j classified into class i, *i.e.*

$$P(\omega_j; i) = \frac{|\{x \in \omega_j \mid c(x) = i\}|}{|\omega_j|}.$$

The **total entropy of the partition** Ω with respect to classification c is the sum

$$H(\Omega) = \sum_{i=1}^{k} H(\omega_i) \frac{|\omega_i|}{|S|}.$$

When we use entropy in practice to evaluate the results of a clustering method, we set S to be the above described high-confidence subset, c the function prescribing a class as informed by known marker genes, and Ω the partition given by the clustering. Observe that entropy is minimized when clusters are unanimously classified: when all of the proportions $P(\omega_j; i)$ for $\omega_j \in \Omega$ and $i \in [n]$ are either 0 or 1. Low values of entropy therefore indicate that each cluster in the obtained clustering is homogeneous with respect to the cell subtype.

6.2.4 Marker Detection

In order to gain insight into which genes drive cluster separation of a given clustering, a plethora of statistical measures are commonly computed for each cluster. The summaries we chose to

include are presented here.

Let g be the number of genes included in the analysis, and N the number of clusters obtained. Given a vector $T = \{\tau_1, \ldots, \tau_g\}$ of thresholds, we compute for each gene $i \in [g]$ and each cluster C the percentage of points in C whose expression of gene i is higher than the threshold τ_i. Denote this percentage as $M_T(C, i)$, *i.e.*

$$M_T(C, i) = \frac{|\{c \in C \mid M(c, i) \geq \tau_i\}|}{|C|},$$

where $M(c, i)$ is the expression of gene i in neuron c belonging to cluster C. Though the threshold vector T can be completely arbitrary, here are some intuitive choices:

- the mean: $\tau_i = \frac{1}{n} \sum_{j=1}^{n} M(j, i)$,
- the median: $\tau_i = p_{0.5}(\{M(j, i)\}_{j=1,\ldots,n})$,
- ℓ^{th} percentile: $\tau_i = p_\ell(\{M(j, i)\}_{j=1,\ldots,n})$,
- all ones: $\tau_i = 1$,

where n is the number of points in the data set and p_ℓ is the ℓ^{th} percentile of a vector (*i.e.* the percent of vector entries below p_ℓ is ℓ).

We summarize the results of these computations within tables (see Tables 6.1 to 6.7), where we write $M_T(C, i)$ in the row belonging to the gene i and the column belonging to cluster C. However, not all genes provide interesting insights. We choose a lower threshold β_{low} and an upper threshold β_{high}, and identify the genes for which at least one of the values $M_T(C, i)$ is smaller than β_{low} and at least one bigger than β_{high}. This means these genes are very highly expressed on some clusters, and almost not expressed at all on some others. We specify the chosen threshold vector and thresholds $\beta_{\text{low}}, \beta_{\text{high}}$ whenever displaying a table of these values.

6.3 Classification of Monoaminergic Neurons

In this section we detail several pipelines for clustering Thirsty Fly data set into four classes, and their results. Note, that although the expected number of classes is known, we cluster the data set into a bigger number of clusters in most pipelines. This is due to the fact that the classes are expected to be of different size, they might naturally consist of several subclasses, and inherent variability in the data might lead to class fragmentation.

As mentioned, some genes (or rather a combination of them) are known to code for a specific neurotransmitter [21, 132].

- High expression of genes `ple` and `DAT` suggests that the neuron in specialized for **dopamine**.
- High expression of genes `SerT` and `Trh` suggests that the neuron in specialized for **serotonin**.
- High expression of genes `Tdc2` and `Tbh` suggests that the neuron in specialized for **octopamine**.
- High expression of gene `Tdc2` and low expression of gene `Tbh` suggests that the neuron in specialized for **tyramine**.

These six listed genes together with another gene, `Fer2`, that identifies a large subtype of dopaminergic neurons, are considered to be important for the classification task at hand. They are referred to as **important genes** in this chapter, and their set is

$$\mathcal{I} = \{\texttt{ple}, \texttt{DAT}, \texttt{SerT}, \texttt{Trh}, \texttt{Tdc2}, \texttt{Tbh}, \texttt{Fer2}\}.$$

Of course, having such prior knowledge is incredibly useful. Firstly, a feature selection step may be replaced by projection to dimensions corresponding to $\mathcal{I}$. This 855×7 matrix is denoted as iM, where i stands for "important". Secondly, the results of different clustering methods can be evaluated using information about which genes code for which subtype. To do this, let $\mu = [\mu_1, \ldots, \mu_7]$ be the vector of mean expression for each gene in $\mathcal{I}$ in the order they appear in as the columns of iM. Then a list of **class representatives** can be comprised for each expected class based on how a neurons expression pattern compares with μ. For example, neurons whose expression of `ple` and `DAT` is higher than the corresponding mean while the expression of all other genes is 0 are identified as dopaminergic class representatives. Luckily, no neuron is identified as a representative for more than one class. The total number of class representatives is 214, with 97 belonging to dopamine, 55 to serotonin, 31 to octopamine, and 31 to tyramine class. For a visualization of the point cloud projected onto important genes with annotated class representatives, see Figure 6.3. Note that by choosing different criteria for the expression patterns, different list of class representatives could be chosen. We work with the one described above since the criteria is strict enough so that we can trust the class representatives fully. However, as can be seen in Figure 6.3, there are large regions of the point cloud in which

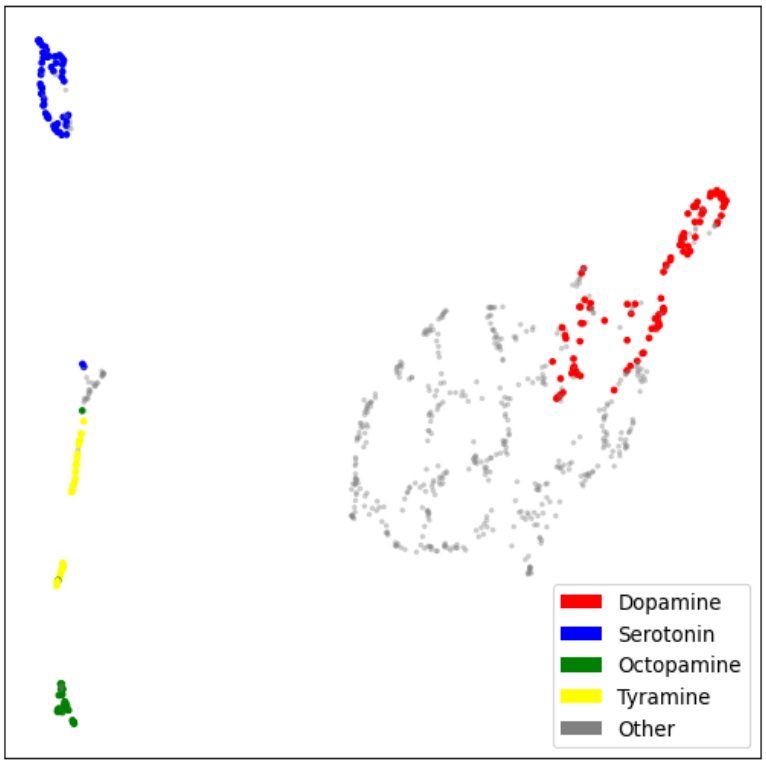


Figure 6.3: UMAP projection of iM. The colored points belong to cells, which were identified as class representatives, with each color corresponding to one of the four classes as denoted by the legend.

no representatives were identified. To guide intuition, we construct a similar plot in addition to Figure 6.3, where the points are colored based on their binary expression vectors (in which all positive entries are replaced by 1), see Figure 6.4.

In Python, we test several popular clustering methods on iM, namely `SpectralClustering` [123] and k-means [121] from `Scikit-learn` library, and Louvain algorithm [88, 133, 124] for community detection from `CDlib` library. We also test k-cluster, described in detail in Section 6.2.1, an implementation of which was provided to us by its authors [22]. Further, each of these methods is tested in three scenarios: when no transformation is applied to iM, once with euclidean distance and once with cosine similarity as a measure of similarity, and when iM is log-transformed. Their performance is evaluated on class representatives and the best perform-

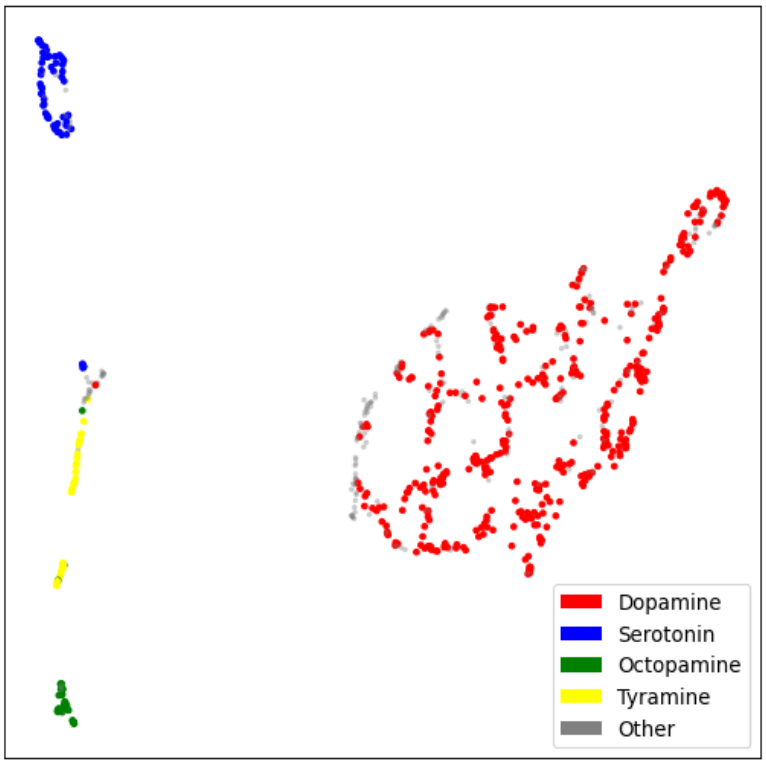


Figure 6.4: UMAP projection of iM. The color of each point is decided based on the its binary expression vector (in which all positive entries are replaced by 1). Thus, for example, cells with positive expression of `ple` and `DAT` and zero expression of all other genes are colored in red (as per the legend).

ing clustering pipelines are chosen for further analysis, where the entire list of genes is searched for possible marker genes not yet included in the set $\mathcal{I}$ of important genes.

6.3.1 Clustering on iM with euclidean distance

The clustering results of all tested methods are visualized in Figure 6.5. Let us detail the chosen parameters and comment on the performance of each method.

- **k-cluster:** First, we decide on the value of parameter k. Running k-cluster for a few different values, it seems 8, which is approximately 10% of the database size, is a good choice. Next, we choose N, the number of clusters the method should return. To do

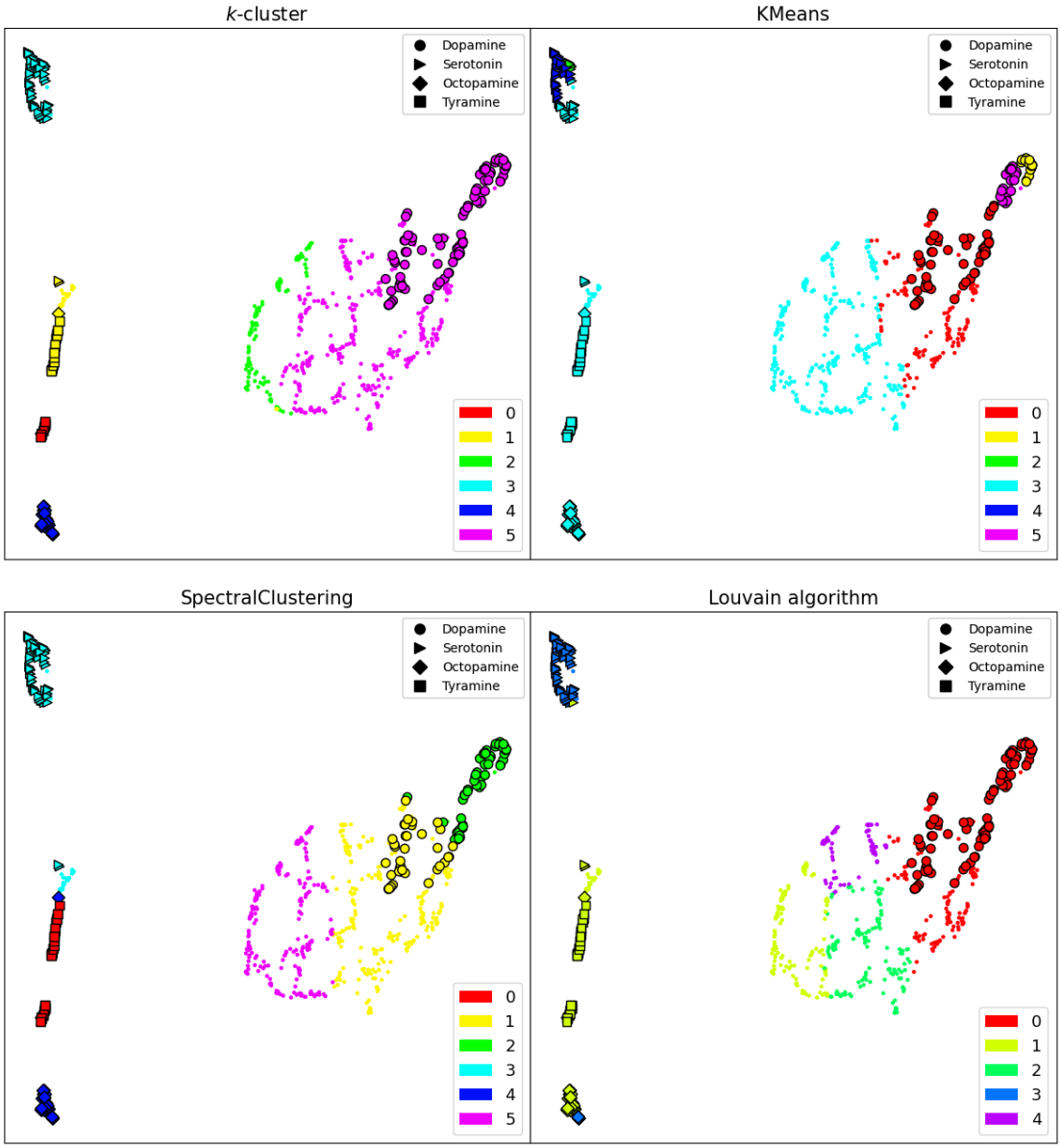


Figure 6.5: A comparison of clustering results on (not normalized) iM with respect to euclidean distance for four chosen methods: k-cluster, k-means, spectral clustering and Louvain algorithm (in the order left to right, top to bottom). The different colors denote clusters given by each method, while the added annotations denote representatives of different classes identified a priori.

this, we compute the persistence diagram of 8-cluster filtration on iM and order the list of multiplicative values of its points to obtain

$$\infty,\ 3.808,\ 2.390,\ 2.236,\ 2.041,\ 2.012,\ 1.528,\ 1.500,\ 1.430,\ \ldots$$

Notice that there is a significant gap between the sixth and the seventh value, 2.012 and 1.528 respectively, which can also be noticed in Figure 6.6. Thus, 6 is chosen as the number of clusters. Observing Figure 6.5 it seems the method performs well on the class representatives, with only one cluster (cluster 1 according to the legend) containing representatives of more than one class. Note that cluster 1 contains the points that are closest to the origin (see Figure 6.14), and so this might be due to the fact that, in euclidean distance, points with different expression patterns are much closer at smaller scales.

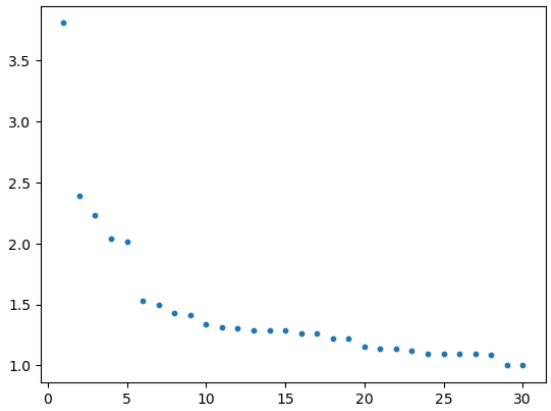


Figure 6.6: Plot of the ordered list of multiplicative values associated to points in the persistence diagram of 8-cluster filtration on iM with respect to euclidean distance. We plot each multiplicative value against its position in the ordered list, skipping multiplicative value ∞, which would be first in the order. First five points (and ∞) are significantly higher than the rest, suggesting iM has 6 inherently present components with respect to euclidean distance.

- **k-means:** The only parameters of the method are the matrix, iM, and the number of clusters we wished to obtain, $n = 6$. Notice that k-means is not as successful as k-cluster. It fragments the representatives of dopaminergic and serotoninergic neurons into many classes, while not separating representatives of tyraminergic and octopaminergic (and even some serotoninergic) neurons at all. Although n was chosen for k-cluster to perform

well, this choice is not the reason why k-means struggles. In fact, similar behavior can be observed for other choices of n. The reason is most likely that the assumptions k-means makes about the data simply do not hold for iM, namely the clusters are not of equal sizes and densities, and the data within a cluster is not normally distributed around the cluster center. Secondly, k-means relies on computation of cluster centroids, which is sensitive to outliers, and there are plenty of those in gene expression data sets.

- **Spectral clustering:** We ran spectral clustering with the following parameters:

$$\texttt{n\_clusters} = 6,$$
$$\texttt{n\_neighbors} = 8,$$
$$\texttt{matrix} = DM,$$

where DM is the distance matrix between cells, computed based on their gene expression vectors in iM in the euclidean metric. It performs extremely well on the class representatives, as none of the identified clusters contains representatives of multiple classes.

- **Louvain community detection:** This method was perhaps the most challenging to use. As described in Section 6.2.2, a graph with appropriate edge weights, which are higher for shorter edges and lower for longer ones, must first be built on our point cloud. Thus, we need to transform the distance between points into a weight, for which there are many methods. Here, we choose to define the weight of edge (i, j) between cells i and j as

$$W(i, j) = C - \log_{10}(1 + d(i, j)),$$

where $d(i, j)$ is the euclidean distance between cells i and j, and C is the smallest constant so that all weights are non-negative. Comparing the results for different methods of weight assignment is out of scope for this text, but would be interesting to consider. Next, we construct a graph as

```
G = networkx.from_numpy_array(W)
```

where W is the matrix of weights. We then call the `algorithms.louvain` method from `CDlib` library on G with parameter `resolution` $= 1.01$ and visualize the results. We see that clusters 1 and 3, similarly as for k-means, contain representatives of three and

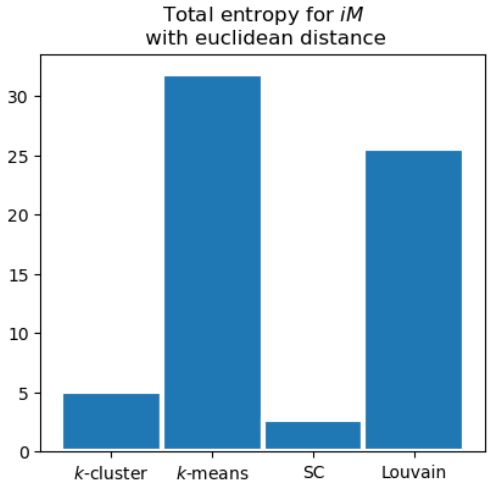


Figure 6.7: The total entropy for each of the tested clustering methods, computed on the set of representatives. As mentioned in Section 6.2.3, low values indicate the obtained clusters are homogeneous with respect to the cell subtype.

two classes, respectively. Similar mixed clusters were observed for other values of the resolution parameter as well (we tested 8 values in the range $[0.9, 1.025]$).

Our qualitative observations are confirmed by total entropy of each clustering (see Figure 6.7):

	k-cluster	k-means	SC	Louvain
total entropy	5.12	31.93	2.70	25.60

Low total entropy for spectral clustering and k-cluster suggests the clusters obtained with these two methods are finer than the subtype classes on representatives. In contrast, the presence of mixed clusters is reflected in higher total entropy for k-means and Louvain community detection.

6.3.2 Clustering on iM with cosine similarity

The clustering results of all tested methods are visualized in Figure 6.8. Again, we provide implementation details and comment on the results.

- **k-cluster:** We follow the argument from before, and set the of parameter k to 8. The number of clusters, N, is determined by studying the persistence diagram of 8-cluster

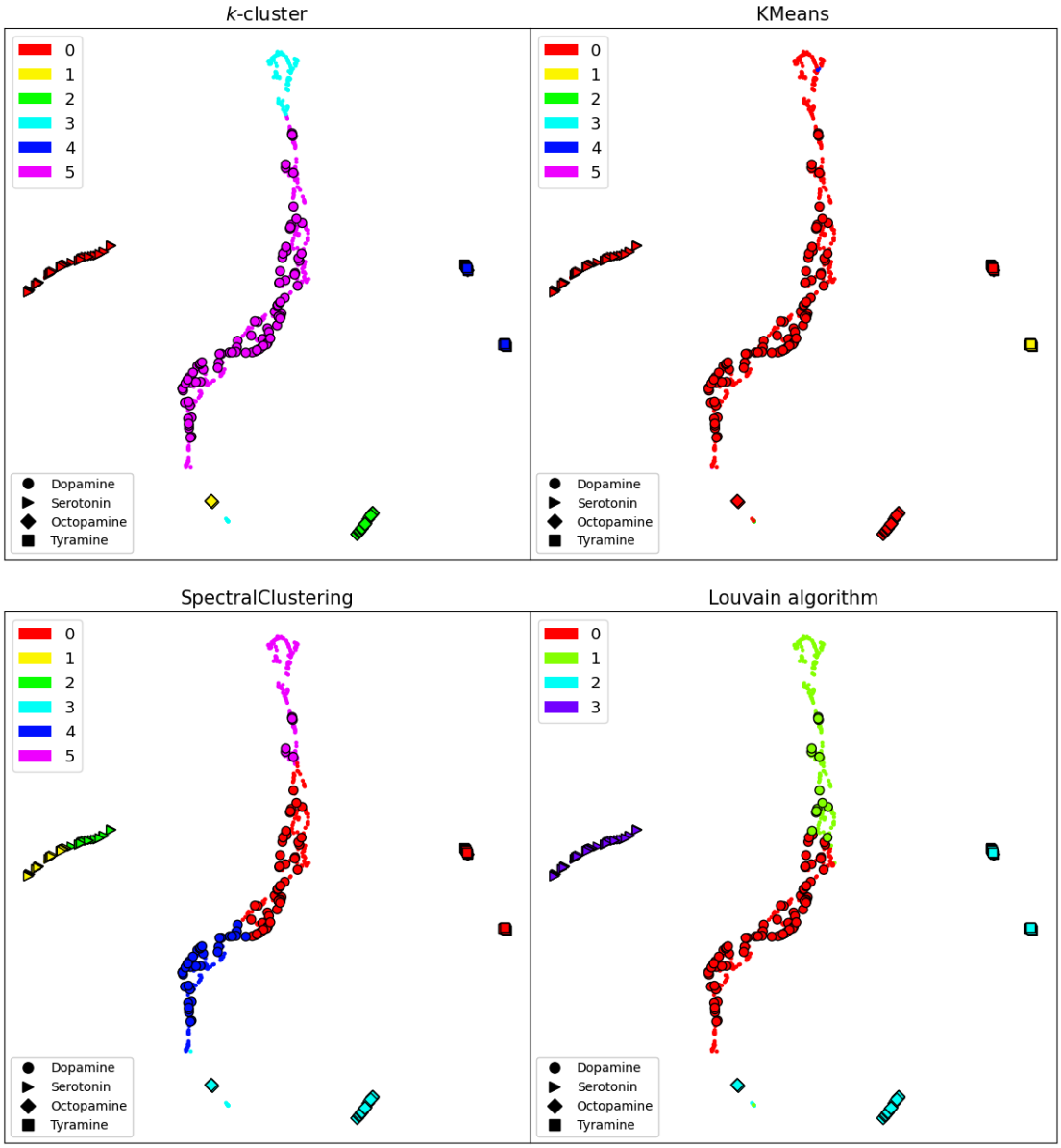


Figure 6.8: A comparison of clustering results on (not normalized) iM with respect to cosine similarity for four chosen methods: k-cluster, k-means, spectral clustering and Louvain algorithm (in the order left to right, top to bottom). Different colors denote clusters given by each method, while the added annotations denote representatives of different classes identified a priori.

filtration on iM with respect to cosine similarity, see Figure 6.9. We again notice a gap between 6^{th} and 7^{th} value, and so 6 is chosen as the number of clusters. Observing Figure 6.8 we can again argue the method performs well on the class representatives, with only cluster 4 containing representatives of more than one class.

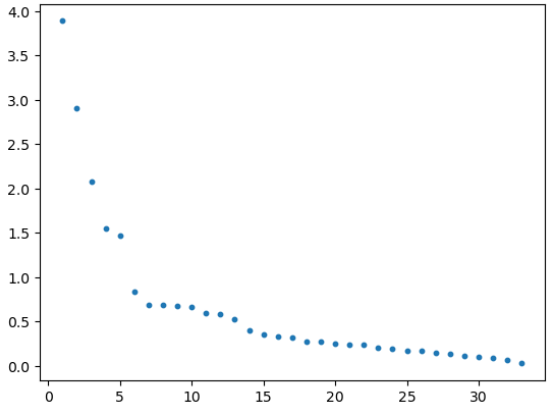


Figure 6.9: Plot of the ordered list of multiplicative values associated to points in the persistence diagram of 8-cluster filtration on iM with respect to cosine similarity. We plot the $\log_{10}$ of each multiplicative value against its position in the ordered list, skipping multiplicative value ∞, which would be first in the order. First five points (and ∞) are significantly higher than the rest, suggesting iM has 6 inherently present components with respect to cosine similarity.

- **k-means:** The implementation of k-means in Scikit-learn does not support working with cosine similarity, which is why we normalize each expression vector with respect to euclidean distance and also use euclidean distance on the result as a proxy for cosine similarity. We then run k-means on the normalized matrix iM with the parameter n, number of clusters we wished to obtain, equal to 6. As can be observed from Figure 6.8, this does not result in a good clustering, perhaps due to outliers that are still present in the data after normalization.
- **Spectral clustering:** We ran spectral clustering with the following parameters:

$$\texttt{n\_clusters} = 6,$$
$$\texttt{n\_neighbors} = 8,$$
$$\texttt{matrix} = DM,$$

where DM is the distance matrix between cells, computed based on their gene expression

vectors in iM in the cosine similarity. We see in Figure 6.8 that the obtained clustering is not the best, however, this is due to the fact that n was chosen to fit k-cluster. By setting parameter `n_clusters` to 9 instead, we obtain a clustering that fits nicely with the one obtained with k-cluster, see Figure 6.10.

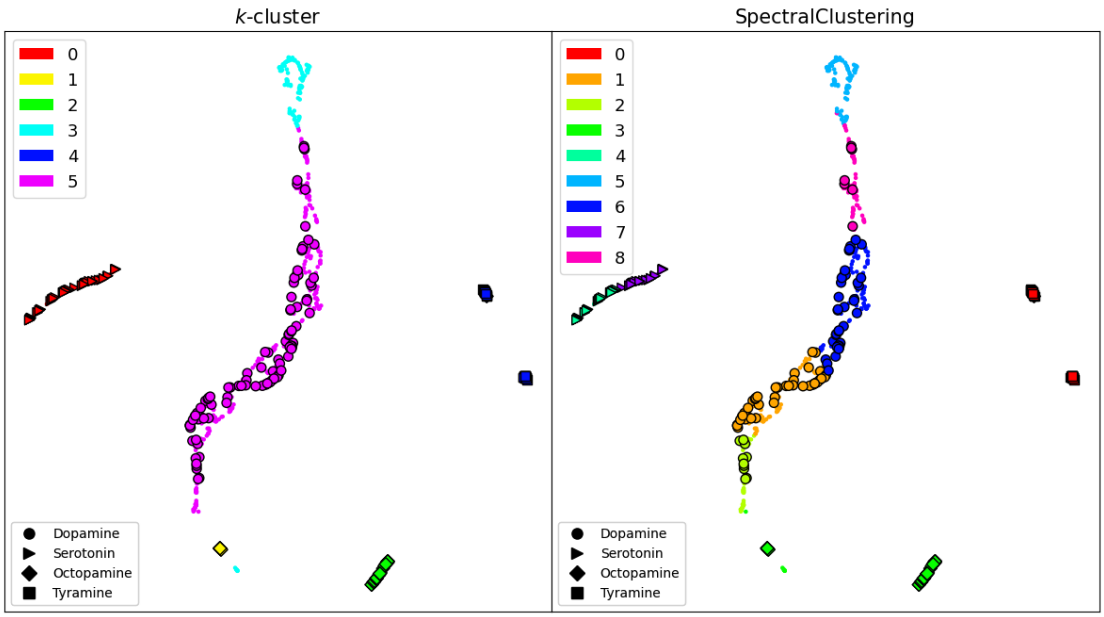


Figure 6.10: Comparison of clustering results on (not normalized) iM with respect to cosine similarity for k-cluster with parameters $N = 6$ and $k = 8$, and spectral clustering with `n_clusters`$= 9$ (instead of 6) and `n_neighbors`$= 8$. Different colors denote clusters given by each method, while the added annotations denote representatives of different classes identified a priori. Observe that each of the clusters obtained with spectral clustering contains representatives of only one class, and is roughly contained within a cluster obtained with k-cluster.

- **Louvain community detection:** We follow similar steps to define a graph on the point cloud iM as before, with weights of edges now being computed with respect to cosine similarity. Thus, we define the weight of edge (i, j) between cells i and j as

$$W(i, j) = C - \log_{10}(1 + d(i, j)),$$

where $d(i, j)$ is the cosine similarity between cells i and j, and C is the smallest constant so that all weights are non-negative. Again, we do not consider other methods of weight

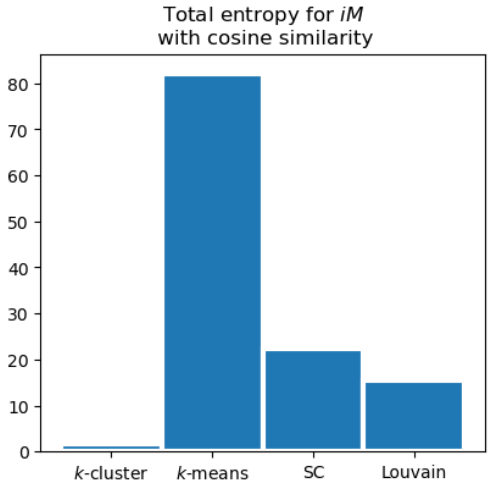


Figure 6.11: The total entropy for each of the tested clustering methods, computed on the set of representatives. As mentioned in Section 6.2.3, low values indicate the obtained clusters are homogeneous with respect to the cell subtype.

assignment. We construct a graph as

$$\texttt{G = networkx.from\_numpy\_array(W)},$$

where W is the matrix of weights. We then call the `algorithms.louvain` method on G with parameter `resolution` $= 1.01$ and visualize the results. We see that clusters 1 and 2 each contain representatives two classes. Note that we tested approximately 20 different resolutions values in the range of $[0.9, 1.5]$, and the clusters that are obtained are roughly the same with the exception of many additional tiny clusters appearing on the border of clusters 0 and 1.

As before, our qualitative observations are confirmed by total entropy of each clustering (see Figure 6.11):

	k-cluster	k-means	SC	Louvain
total entropy	1.60	82.17	22.41	15.50

With an even lower entropy than in the previous setting, k-cluster clearly performs best according to this metric. Due to the observed presence of mixed clusters, the total entropy of other clustering methods is significantly higher.

6.3.3 Clustering on log-transformed iM with euclidean distance

To give k-means a fighting chance, we log-transform iM by applying

$$iM(i,j) \mapsto \ln(1 + 10000 \frac{iM(i,j)}{\sum_k iM(i,k)})$$

to each entry of iM prior to clustering. Note that this is a standard transformation for gene expression data sets, and is the default normalization method in the Seurat package [128, 64, 129]. Let us abuse the notation slightly and denote the resulting matrix by $\ln(iM)$. The results of all tested clustering methods on $\ln(iM)$ are visualized in Figure 6.12. We list the chosen parameters and comment on their performance here.

- **k-cluster:** Again, we set the parameter k to 8 and determine the number of clusters, N, by studying the persistence diagram of 8-cluster filtration on $\ln(iM)$ with respect to euclidean distance, see Figure 6.13. We again choose 6 as the number of clusters. Observing Figure 6.12 we again notice a relatively good performance on other classes, but the method struggles with the class of tyraminergic neurons, expression vectors of which (in iM) are quite small in euclidean norm, see Figure 6.14. It is therefore not unlikely, that the log-transformation would exacerbate initially small differences on this class. However, as can be seen from Figure 6.12, k-means performs well on this class.
- **k-means:** We run k-means on $\ln(iM)$ with the parameter n, number of clusters we wished to obtain, equal to 6. As mentioned before, it can be observed in Figure 6.12 that k-means performs better than k-cluster on $\ln(iM)$. Only cluster 1 contains representatives of more than one class.
- **Spectral clustering:** Similarly as before, we ran spectral clustering with parameters

  $$\texttt{n\_clusters} = 6,$$
  $$\texttt{n\_neighbors} = 8,$$
  $$\texttt{matrix} = DM,$$

  where DM is the distance matrix between cells, computed based on their gene expression vectors in $\ln(iM)$ in the euclidean distance. The resulting clustering seems to be good when considered on its own, but cluster 5 contains many points that are consistently clustered together with dopaminergic representatives by other methods. We test the method with several other choices for parameters `n_clusters` and `n_neighbors`, but all results contain at least one cluster we assume to be mixed.

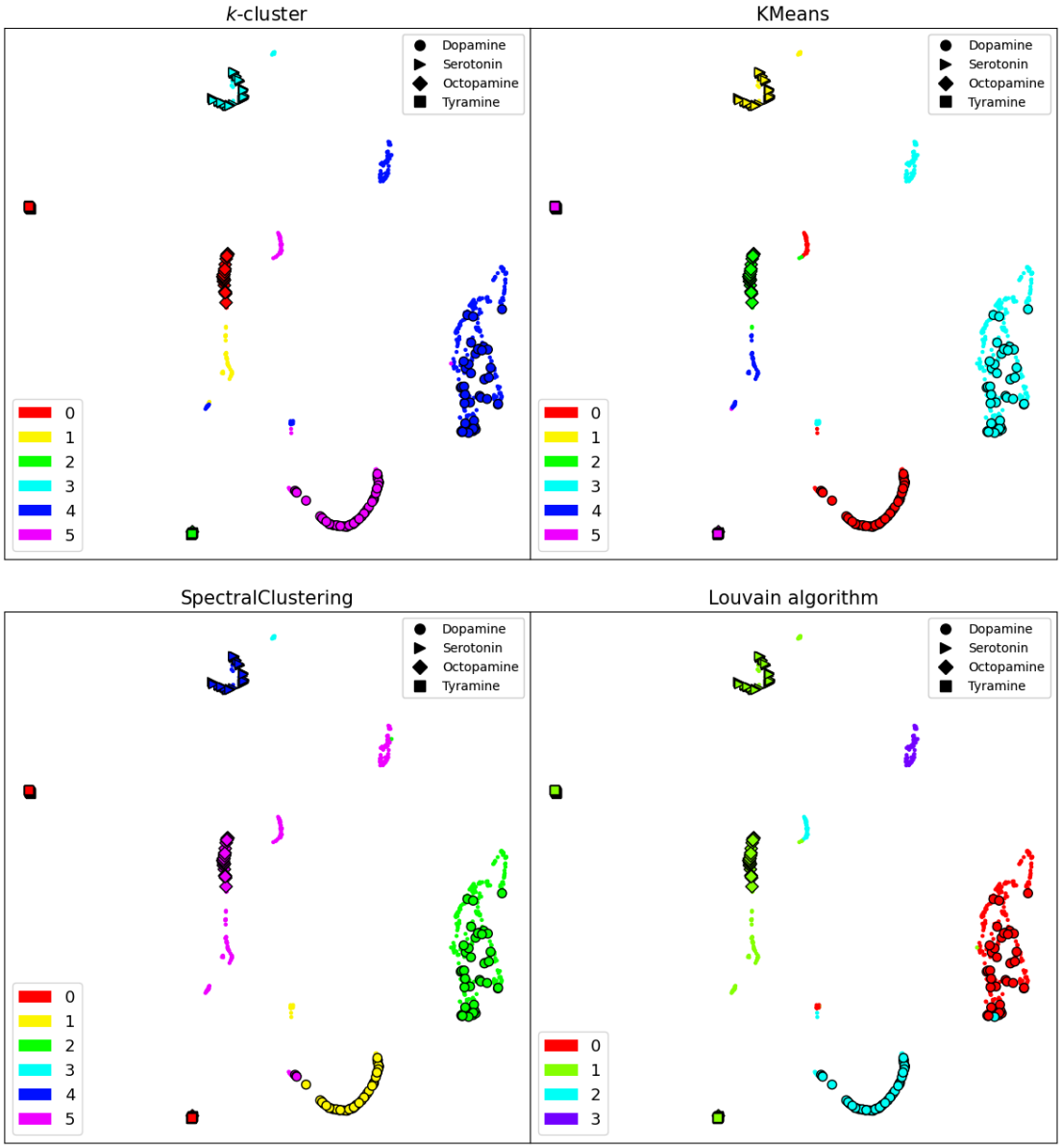


Figure 6.12: Comparison of clustering results on log-transformed iM with respect to euclidean distance for four chosen methods: k-cluster, k-means, spectral clustering and Louvain algorithm (in the order left to right, top to bottom). Different colors denote clusters given by each method, while the added annotations denote representatives of different classes identified a priori.

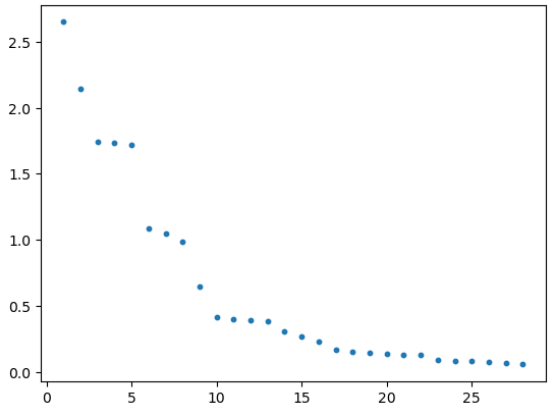


Figure 6.13: Plot of the ordered list of multiplicative values associated to points in the persistence diagram of 8-cluster filtration on $\ln(iM)$ with respect to euclidean distance. We plot the $\log_{10}$ of each multiplicative value against its position in the ordered list, skipping multiplicative value ∞, which would be first in the order. Many gaps can be observed, namely between points 1 and 2, 2 and 3, 5 and 6, 8 and 9, and 9 and 10. Thus, values $N = 2, 3, 6, 9, 10$ are all viable choices (taking into account ∞ as well), but we choose $N = 6$ for consistency sake.

- **Louvain community detection:** Define a weighted graph G on the point cloud $\ln(iM)$ by setting the weight of edge (i, j) between cells i and j to

$$W(i, j) = C - d(i, j),$$

where $d(i, j)$ is the euclidean distance between cells i and j in $\ln(iM)$, and C is the smallest constant so that all weights are non-negative. Note that due to the matrix being log-transformed, we do not need to apply the logarithm to the distance, as we did before. Again, we do not consider other methods of weight assignment. We construct a graph as

```
G = networkx.from_numpy_array(W),
```

where W is the matrix of weights. We then call the `algorithms.louvain` method on G with parameter `resolution` $= 1.01$. Observe in Figure 6.12 that cluster 1 contains representatives of three different classes. We tested approximately 10 different resolutions values in the range of $[0.8, 1.03]$, and the obtained clusters are roughly the same, with mixed clusters appearing for all tested resolutions.

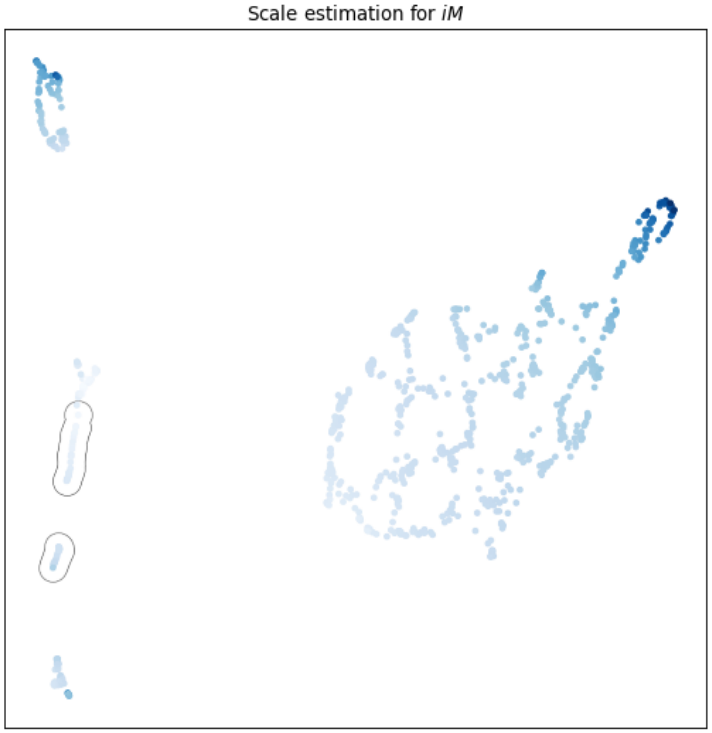


Figure 6.14: Illustration of the estimated scale of points in point cloud iM. The points are colored based on the root of their euclidean norm, $\sqrt{\|c\|}$ for cell c, where the darker color is used for points with higher norm. The circled points are representatives of tyraminergic neurons.

To evaluate and compare the performance of chosen clustering methods, we again compute their respective total entropy (see Figure 6.15):

	k-cluster	k-means	SC	Louvain
total entropy	13.07	1.60	18.68	44.67

As observed before, the performance of k-means on the set of representatives is significantly better than that of the other methods, which is not surprising, since we chose the geometric setting according to the assumptions made by k-means. On the other hand, k-cluster exhibits a higher total accuracy for the first time, reflecting the observed issues with classifying tyraminergic neurons.

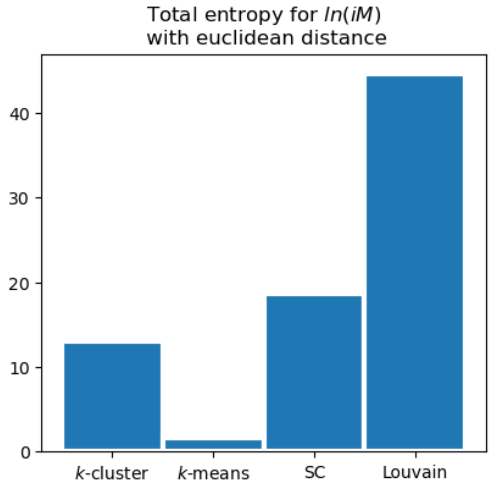


Figure 6.15: The total entropy for each of the tested clustering methods, computed on the set of representatives. As mentioned in Section 6.2.3, low values indicate the obtained clusters are homogeneous with respect to the cell subtype.

6.3.4 Marker detection on good clusterings

We present the results of marker detection analysis performed on the clustering method with smallest total entropy for each of the settings in the previous sections. In particular, the chosen methods are

- spectral clustering with `n_clusters`=6 and `n_neighbors`=8 on iM with respect to euclidean distance,
- k-cluster with $N = 6$ and $k = 8$ on iM with respect to cosine similarity,
- k-means with $n = 6$ and on $\ln(iM)$ with respect to euclidean distance.

First, we determine which cluster belongs to which class. For each of the methods, there is at least one cluster that either contains representatives of more than one class or it does not contain any class representatives. Thus, we compute the percentage of points within each cluster in which a gene is expressed (*i.e.* the chosen threshold vector in our marker detection method is the vector of ones). We base our decision on these percentages for important genes (with the exception of `'Fer2'`), which are summarized in Tables 6.1 to 6.3 for spectral clustering, k-cluster and k-means respectively.

Spectral clustering, iM, euclidean

	0	1	2	3	4	5
Fer2	43.6	64.4	18.8	11.4	0.0	58.0
Trh	0.0	0.3	0.0	75.2	0.0	1.0
ple	0.0	99.7	100.0	1.9	2.9	70.5
Tdc2	100.0	1.0	0.0	1.9	100.0	5.8
DAT	2.6	99.3	100.0	3.8	5.9	98.6
SerT	0.0	0.0	0.0	76.2	0.0	0.0
Tbh	5.1	5.8	12.9	14.3	100.0	1.9

Table 6.1: Percentages of points within each cluster of spectral clustering on iM with euclidean distance for which a specific gene is expressed. Since `DAT` and `ple` are high on clusters 1, 2 and 5, we argue the dopaminergic class comprises of points in these three clusters. In a similar way, we argue seratoninergic class consists only of cluster 3, the octopaminergic of cluster 4, and tyraminergic of cluster 0.

k-cluster, iM, cosine

	0	1	2	3	4	5
Fer2	1.2	0.0	0.0	68.3	41.5	52.9
Trh	98.8	0.0	0.0	0.8	0.0	0.4
ple	1.2	0.0	3.3	37.5	0.0	99.8
Tdc2	0.0	58.3	93.3	10.0	97.6	0.6
DAT	0.0	0.0	6.7	91.7	2.4	98.8
SerT	100.0	0.0	0.0	0.0	0.0	0.0
Tbh	11.2	91.7	93.3	2.5	4.9	6.2

Table 6.2: Percentages of points within each cluster of spectral clustering on iM with cosine distance for which a specific gene is expressed. Since `DAT` and `ple` are (relatively) high on clusters 3 and 5, we argue the dopaminergic class comprises of points in these two clusters. In a similar way, we argue seratoninergic class consists only of cluster 0, the octopaminergic of clusters 1 and 2, and tyraminergic of cluster 4.

k-means, $\ln(iM)$, euclidean

	0	1	2	3	4	5
Fer2	0.0	1.2	4.3	75.3	73.2	42.1
Trh	0.0	98.8	0.0	0.8	0.0	0.0
ple	98.0	1.2	2.1	96.2	0.0	0.0
Tdc2	0.7	0.0	78.7	1.0	19.6	97.4
DAT	98.0	0.0	4.3	99.2	87.5	0.0
SerT	0.0	100.0	0.0	0.0	0.0	0.0
Tbh	9.2	11.2	85.1	4.3	1.8	5.3

Table 6.3: Percentages of points within each cluster of k-means on $\ln(iM)$ with euclidean distance for which a specific gene is expressed. Since `DAT` (and partially `ple`) is high on clusters 0, 3 and 4, we argue the dopaminergic class comprises of points in these three clusters. In a similar way, we argue seratoninergic class consists only of cluster 1, the octopaminergic of cluster 2, and tyraminergic of cluster 5.

The clusters for which it is determined they belong to the same class are merged. For example, for the dopaminergic neurons, we form sets (merged clusters) D_{SC}, $D_{k\text{-cluster}}$, and $D_{k\text{-means}}$ with

- D_{SC} consisting of clusters 1, 2 and 5 of spectral clustering on iM with respect to the euclidean metric, see Table 6.4 and Figure 6.5,
- $D_{k\text{-cluster}}$ consisting of clusters 3 and 5 of k-cluster on iM with respect to cosine similarity, see Table 6.5 and Figure 6.8,
- $D_{k\text{-means}}$ consisting of clusters 0, 3 and 4 of k-means on $\ln(iM)$ with respect to the euclidean metric, see Table 6.6 and Figure 6.12,

In Section 6.4, further analysis is performed on their intersection, $D \coloneqq D_{\mathrm{SC}} \cap D_{k\text{-cluster}} \cap D_{k\text{-means}}$.

Expression profiles on the level of merged clusters (for all four classes) are analysed using our marker detection method, with the chosen threshold of gene's expression being the median on the original count matrix M. The results are summarized in Tables 6.4 to 6.6 for spectral clustering, k-cluster and k-means respectively. Based on the consistent appearance of genes `dac`, `CG34354`, `asRNA:CR44165`, `Ddc`, `Nep1`, `ct`, and `Lim1` in these summaries, we suggest them as potential not yet known markers for monoaminergic neuron subtypes.

6.3.5 Conclusion

While k-cluster did not achieve the best performance in every setting, it consistently delivered strong results with its total entropy never ranking worse than second best. Its ease of use, with only two parameters to configure, is a significant advantage. This is especially true considering that k, the parameter controlling the scale at which important sub-structures appear, can be chosen intuitively, and the inherent number of clusters detected via studying the multiplicative values of k-cluster filtration can inform our choice of the other parameter, the number of clusters N. Additionally, the flexibility of k-cluster to be applied across various metrics and distributional settings makes it highly adaptable. These attributes collectively make it an excellent candidate for wider use in diverse clustering applications.

Spectral clustering, iM, euclidean

	Dop	Ser	Oct	Tyr
Nep2	13.9	0.0	81.8	18.4
Ms	10.6	9.7	75.8	11.8
CG34354	41.1	77.8	9.1	25.0
CG13288	9.1	6.9	81.8	11.8
TfAP-2	2.4	11.1	78.8	3.9
asRNA:CR44165	2.4	0.0	97.0	50.0
rgr	1.5	1.4	78.8	25.0
CG10527	7.9	12.5	78.8	10.5
AANAT1	5.8	75.0	6.1	9.2
SIFa	2.6	2.8	78.8	19.7
lov	7.4	23.6	78.8	47.4
sdk	24.7	37.5	78.8	7.9
Nep1	8.9	34.7	90.9	36.8
ct	8.9	19.4	84.8	47.4
CG1572	2.1	5.6	75.8	11.8
CG32532	8.6	23.6	78.8	28.9
DIP-epsilon	2.2	18.1	78.8	15.8
dac	7.9	8.3	81.8	7.9
Ddc	51.2	97.2	0.0	13.2

Table 6.4: Percentages of points within each cluster of spectral clustering on iM with euclidean distance for which a specific gene is expressed higher than its median on the data set. We display the subset of genes for which there is a class with percentage smaller than $\beta_{\text{low}} = 10\%$ and a class with percentage bigger than $\beta_{\text{high}} = 75\%$, excluding important genes.

k-cluster, iM, cosine

	Dop	Ser	Oct	Tyr
trv	46.0	76.2	16.7	9.8
CG34354	41.5	72.5	14.3	9.8
asRNA:CR44165	2.3	0.0	81.0	87.8
lov	7.3	23.8	69.0	73.2
Nep1	8.6	31.2	81.0	58.5
ct	9.8	18.8	76.2	58.5
Lim1	44.0	10.0	64.3	92.7
dac	8.0	7.5	73.8	0.0
rdo	32.7	38.8	73.8	7.3
Ddc	50.0	93.8	4.8	2.4

Table 6.5: Percentages of points within each cluster of k-cluster on iM with cosine similarity for which a specific gene is expressed higher than its median on the data set. We display the subset of genes for which there is a class with percentage smaller than $\beta_{\text{low}} = 10\%$ and a class with percentage bigger than $\beta_{\text{high}} = 70\%$, excluding important genes.

k-means, $\ln(iM)$, euclidean

	Dop	Ser	Oct	Tyr
trv	46.0	76.2	21.3	5.3
CG34354	41.5	72.5	17.0	7.9
asRNA:CR44165	2.5	0.0	76.6	86.8
lov	7.3	23.8	66.0	73.7
Nep1	8.7	31.2	76.6	57.9
ct	10.0	18.8	72.3	55.3
Lim1	44.2	10.0	63.8	92.1
dac	7.7	7.5	70.2	0.0
Ddc	50.0	93.8	6.4	2.6

Table 6.6: Percentages of points within each cluster of k-means on $\ln(iM)$ with euclidean distance for which a specific gene is expressed higher than its median on the data set. Note that these percentages are computed based on the expression vectors in the original matrix M, and not the log-transformed version. We display the subset of genes for which there is a class with percentage smaller than $\beta_{\text{low}} = 10\%$ and a class with percentage bigger than $\beta_{\text{high}} = 70\%$, excluding important genes.

6.4 Classification of Dopaminergic Neurons

Similar clustering analysis as in Section 6.3 is carried out here for the subset

$$D \coloneqq D_{\text{SC}} \cap D_{k\text{-cluster}} \cap D_{k-\text{means}}$$

of neurons, for which spectral clustering, k-cluster and k-means all agree they belong to the dopaminergic class. This subset consisting of 587 points might not contain all dopaminergic neurons – the subset $D_{\text{SC}} \cup D_{k\text{-cluster}} \cup D_{k-\text{means}}$ in comparison contains 603 points – but it is the subset on which we have high confidence that the neurons truly belong to the dopaminergic class. Thus, we restrict the original count matrix M to the rows belonging to neurons in D. It is expected that their expression will be similar for the marker genes of monoaminergic neurons (and perhaps some other genes related to properties they have in common). We remove them from the analysis by discarding the columns belonging to genes whose expression is not sufficiently varied on D, measured in both the number of neurons a specific gene is expressed for, and the scale of its expression in terms of the maximum and minimum value. To be precise, we discard genes

- whose maximum expression is smaller than 10 (hinting that they are never highly expressed),
- whose minimum expression is larger than 2 (hinting that they are always expressed),
- that are expressed on less than 10 dopaminergic neurons, and
- that are expressed on more than $587 - 10$ dopaminergic neurons.

(These thresholds were not chosen following extensive analysis and can be reinvestigated.) Denote the resulting 587×999 matrix as dM (where "d" stands for "dopaminergic").

In comparison with classification on monoaminergic neurons, this task is completely unsupervised. We do not know how many clusters are present in the data (the conjectured number is 20–30, but it is unlikely we will be able to see all those in a data set of roughly 600 points) and we do not know, which genes play a role in the cell specialization for the expected subtypes. Thus, the only way to argue that a clustering is good, is to have it confirmed by other clustering results following different pipelines. Say $\mathcal{A} = \{A_1, \ldots, A_n\}$ and $\mathcal{B} = \{B_1, \ldots, B_m\}$ are two sets of clusters on a point cloud. We say they **agree**, if each cluster in $\mathcal{A}$ is either partitioned by a subset of clusters in $\mathcal{B}$ or is, together with some other clusters of $\mathcal{A}$, part of a partition of a cluster in $\mathcal{B}$. The reverse, of course, needs to hold for clusters in $\mathcal{B}$.

Since k-cluster proved to be the most consistently well-performing clustering method in Section 6.3, we use it exclusively here. We obtain clustering results in many settings, where we vary the geometry of the ambient space (cosine similarity *vs*. euclidean distance), the genes included (we test several criteria for feature selection), the parameters of k-cluster, and we test different methods for mitigating scale effects (either using none, log-transformation or applying square root to the data set). In most settings, the obtained sets of clusters do not agree with others. However, clustering results

1. on dM with respect to cosine similarity (Section 6.4.1),
2. on log-transformed dM with respect to euclidean distance (Section 6.4.2),
3. on a restriction of dM to genes chosen by a prior feature selection with respect to euclidean distance (Section 6.4.3),

exhibit interesting behavior.

6.4.1 Clustering on dM with respect to cosine similarity

As we have often done in Section 6.3, we set the parameter k to 8 and determine the number of clusters, N, by studying the persistence diagram of 8-cluster filtration on dM with respect to cosine similarity, see Figure 6.17. We choose $N = 2$ and run k-cluster to obtain clusters from Figure 6.18.

6.4.2 Clustering on log-transformed dM with respect to euclidean distance

Let us log-transform dM by applying

$$dM(i,j) \mapsto \ln(1 + 10000\frac{dM(i,j)}{\sum_k dM(i,k)})$$

to each entry of dM prior to clustering. With slight abuse of notation we denote the resulting matrix by $\ln(dM)$. Due to the fact that k-cluster detects very few clusters for $k = 8$, we set k to 6 instead. Studying the multiplicative values of the persistence diagram of 6-cluster filtration on $\ln(dM)$ with respect to euclidean distance, see Figure 6.19, we choose $N = 4$ and run k-cluster to obtain clusters from Figure 6.20.

6.4.3 Feature selection and subsequent clustering

Although we have already removed some genes from the analysis when constructing the matrix dM, further feature selection can be done following one of the standard methods. Namely, we plot the variance of each column (*i.e.* gene) of $\ln(dM)$ from Section 6.4.2 against its mean, see Figure 6.16. A trend curve φ, which we choose to be a polynomial of degree 3, is then fit to the plotted points. We discard genes whose variance is smaller than $0.05 + \frac{6}{5}\varphi(\mu)$, where μ is the gene's mean, resulting in a 587×112 matrix we denote by dMf. We further apply square root to each entry of dMf, which helps to mitigate scale effects.

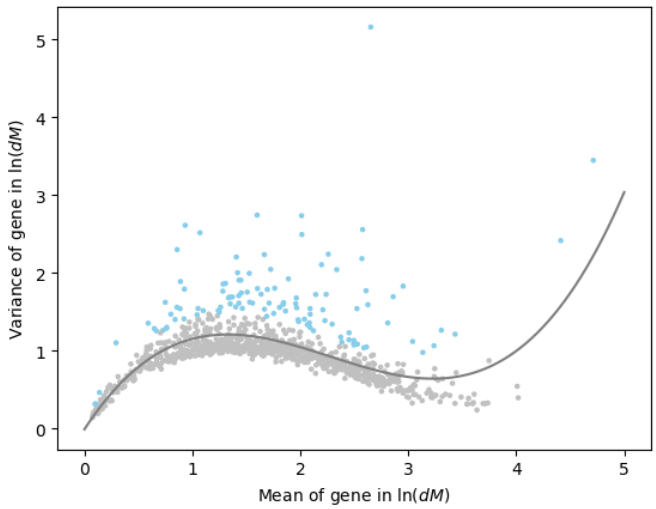


Figure 6.16: Plot of the variance against the mean for each column (*i.e.* gene) in $\ln(dM)$. The trend curve is modeled as a polynomial in degree 3 and fitted using least squares method (`numpy.polyfit`). Genes associated to points in gray are discarded, as their variance is smaller than $0.05 + \frac{6}{5}\varphi(\mu)$, where μ is their mean. The other 112 genes corresponding to points in blue are kept, and the projection of dM to these genes is denoted by dMf.

As often before, we choose to run k-cluster for $k = 8$. The plot of multiplicative values of points in the persistence diagram of 8-cluster filtration on $\sqrt{dMf}$ with respect to euclidean distance (see Figure 6.21) suggest possible choices for parameter N are 3, 5, 10 and 12. Since the numbers of clusters obtained in Sections 6.4.1 and 6.4.2 are quite low, we choose $N = 5$. The obtained clusters are visualized in Figure 6.22.

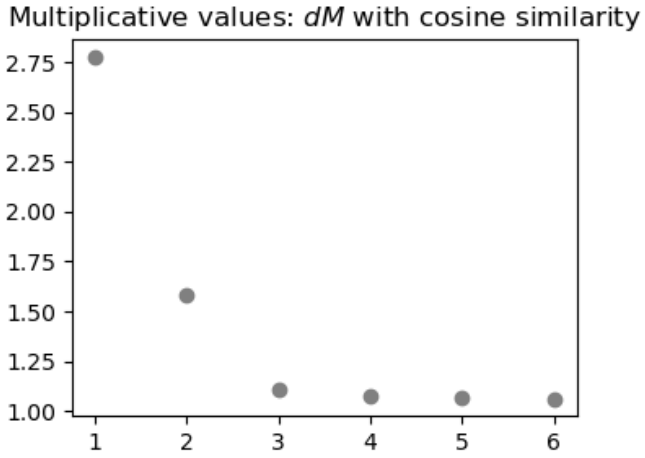


Figure 6.17: Plot of the ordered list of multiplicative values associated to points in the persistence diagram of 8-cluster filtration on dM with respect to cosine similarity. We plot each multiplicative value against its position in the ordered list, skipping multiplicative value ∞, which would be first in the order.

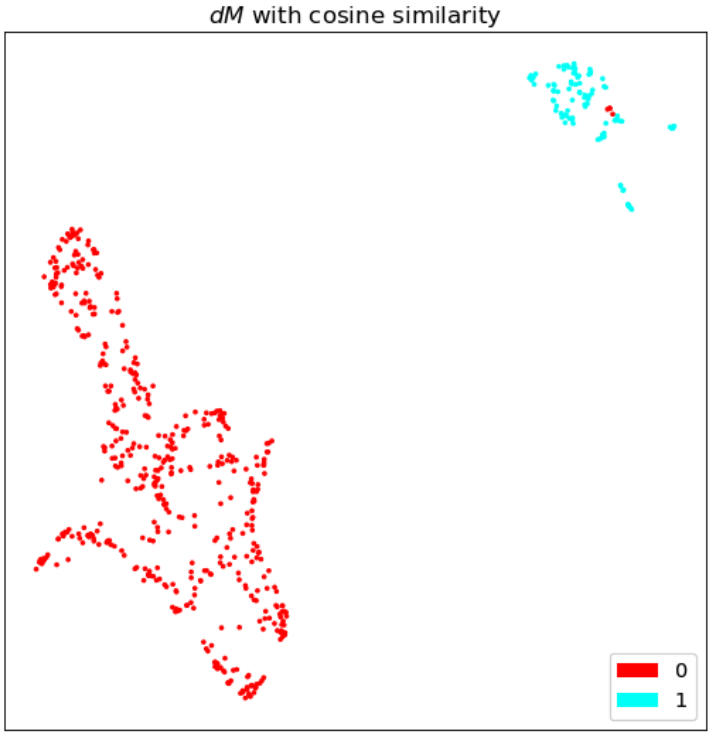


Figure 6.18: Results of clustering with k-cluster with parameters $k = 8$ and $N = 2$ on dM with respect to cosine similarity.

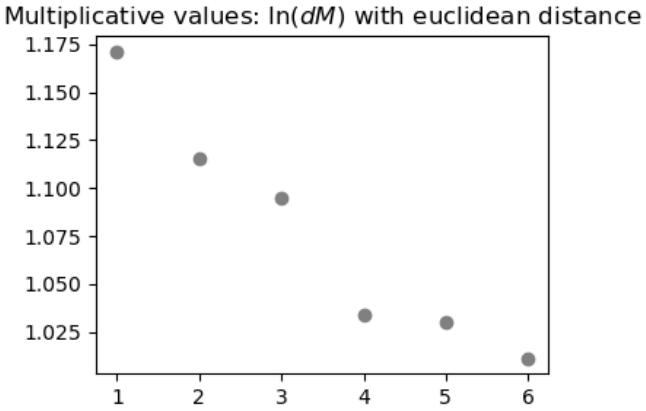


Figure 6.19: Plot of the ordered list of multiplicative values associated to points in the persistence diagram of 6-cluster filtration on $\ln(dM)$ with respect to euclidean distance. We plot each multiplicative value against its position in the ordered list, skipping multiplicative value ∞, which would be first in the order.

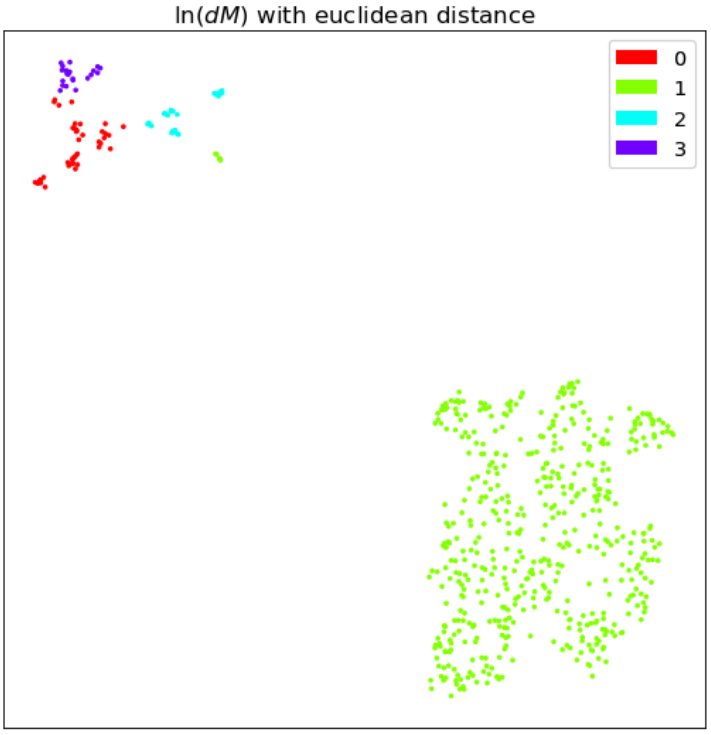


Figure 6.20: Results of clustering with k-cluster with parameters $k = 6$ and $N = 4$ on $\ln(dM)$ with respect to euclidean distance.

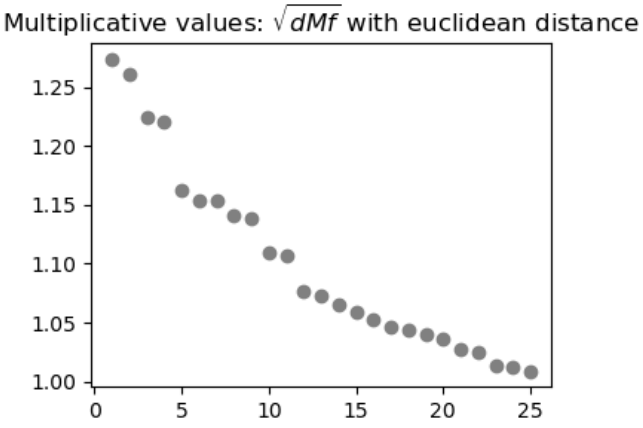


Figure 6.21: Plot of the ordered list of multiplicative values associated to points in the persistence diagram of 8-cluster filtration on $\sqrt{dMf}$ with respect to euclidean distance. We plot each multiplicative value against its position in the ordered list, skipping multiplicative value ∞, which would be first in the order.

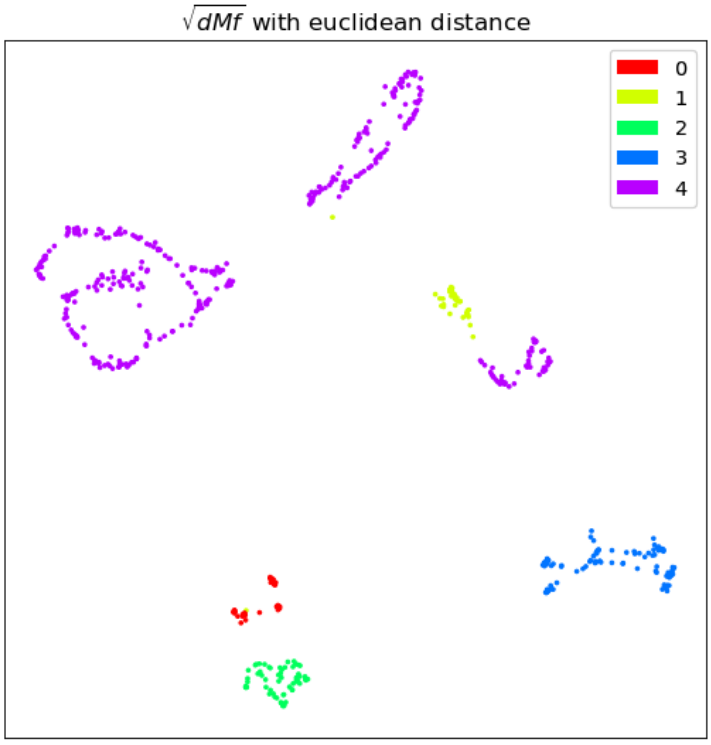


Figure 6.22: Results of clustering with k-cluster with parameters $k = 8$ and $N = 5$ on $\sqrt{dMf}$ with respect to euclidean distance.

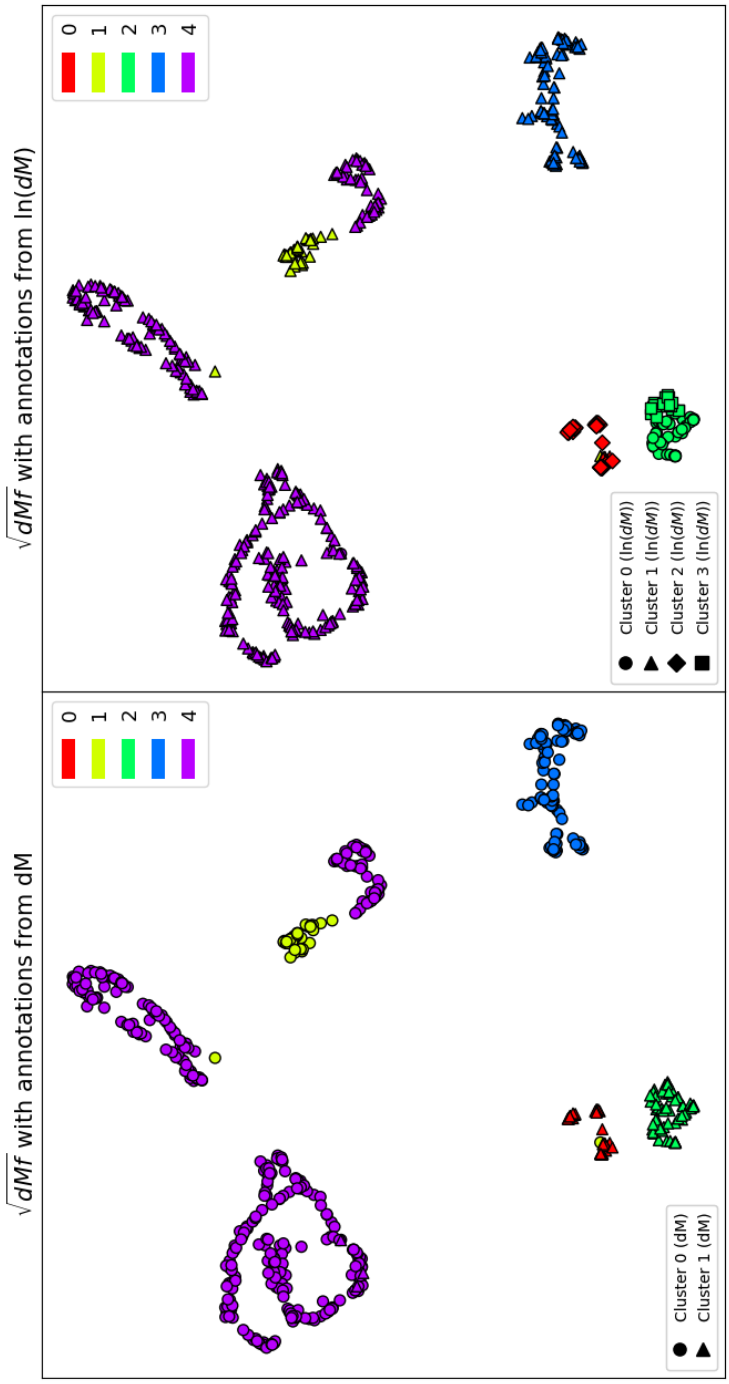


Figure 6.23: UMAP projection of $\sqrt{dMf}$ with two clustering results displayed on each plot. The plot on the left is colored with respect to the set of clusters obtained on $\sqrt{dMf}$ in Section 6.4.3 and annotated with respect to the set of clusters obtained on dM in Section 6.4.1. The plot on the right is colored with respect to the set of clusters obtained on $\sqrt{dMf}$ in Section 6.4.3 and annotated with respect to the set of clusters obtained on $\ln(dM)$ in Section 6.4.2.

6.4.4 Combining clustering results

To compare the sets of clusters obtained in Sections 6.4.1 to 6.4.3, we display pairs of them on the same plot, see Figure 6.23. Observe that these clustering results agree. We combine them by taking the coarsest set of clusters that is finer than all clustering results from Sections 6.4.1 to 6.4.3. Namely, define $\mathcal{A} = \{A_0, \ldots, A_5\}$ by setting

$$\begin{aligned} A_2 &= C_0(\ln(dM)) \cap C_2(\sqrt{dMf}), \\ A_5 &= C_3(\ln(dM)) \cap C_2(\sqrt{dMf}), \\ A_i &= C_i(\sqrt{dMf}) \text{ for } i = 0, 1, 3, 4, \end{aligned}$$

where $C_i(m)$ denotes the cluster i obtained from matrix m as denoted by the legends in Figure 6.23. They are illustrated in Figure 6.24.

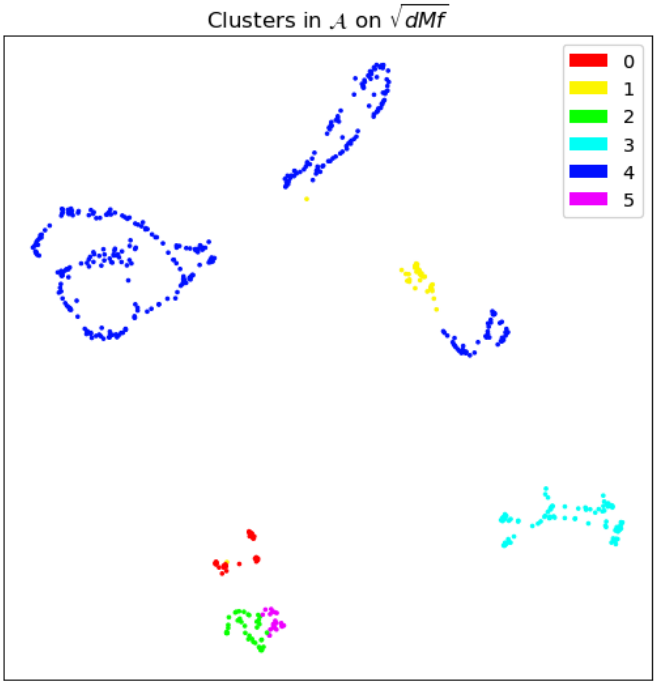


Figure 6.24: UMAP projection of $\sqrt{dMf}$ with colors corresponding to clusters in $\mathcal{A}$.

To argue about potential marker genes for clusters in $\mathcal{A}$ we again run our marker detection method on the restriction of the original count matrix M onto the rows belonging to dopaminergic neurons

(not to be confused with dM, where we also restrict to a subset of columns). We choose the vector of ones as the threshold for gene expression and summarize the results in Table 6.7, where we only display the rows belonging to genes that are expressed in less than $\beta_{\text{low}} = 5\%$ of points in one cluster while simultaneously being expressed for more than $\beta_{\text{high}} = 80\%$ of points in some other cluster. Observe that genes `CG15522` and `CG32532` can be used to distinguish clusters A_2 and A_5, which were originally identified as the same cluster by clustering on $\sqrt{dMf}$. The only gene that is a potential sole marker for its cluster is `Apoltp` (for cluster A_1). However, marker genes can be used in combinations. For example, if a neuron's expression of `CG32532` is higher and the expression of `Lim1` is lower than 1, we can claim with great confidence that it belongs to cluster A_5. In fact, we can correctly identify 95.2% of neurons in cluster A_5.

6.5 Conclusion

While small, the data set of monoaminergic neurons is big enough so that we were able to identify the four expected classes to our satisfaction. Unfortunately, we were not as successful in the subsequent analysis, where we attempted to divide the restriction of the data set to dopaminergic neurons into between 20 and 30 classes. As mentioned before, it is hardly reasonable to expect a data set with circa 600 points would support such fine fragmentation. Further, the only approach for validation of results that is available to us is comparing them to other results, which is why we cannot confirm the obtained clusters show true substructure in the data. However, as this is a very difficult task, we are satisfied with the little insight the methodology used provided. We hope to confirm that the 6 clusters we obtained in Section 6.4.4 represent true substructure in subsequent work, where we will analyse a new, bigger data set of dopaminergic neurons that is currently being preprocessed by our collaborators.

	0	1	2	3	4	5
mCherry	0.0	91.9	21.4	79.2	90.2	0.0
CG12594	48.5	2.7	81.0	30.2	18.4	28.6
Fer2	0.0	86.5	16.7	55.2	65.4	0.0
beat-IIa	24.2	18.9	23.8	2.1	10.6	81.0
Lgr1	0.0	81.1	14.3	16.7	17.0	0.0
CG17193	36.4	94.6	2.4	16.7	40.2	14.3
Sar1	57.6	27.0	59.5	2.1	11.2	95.2
orb	48.5	13.5	64.3	0.0	1.4	100.0
Rox8	24.2	2.7	35.7	7.3	12.8	81.0
CG15522	42.4	0.0	11.9	0.0	0.6	90.5
emc	78.8	5.4	54.8	2.1	2.8	90.5
GluRIA	54.5	0.0	69.0	2.1	7.8	95.2
GluRIB	54.5	5.4	66.7	1.0	9.8	100.0
dpr10	51.5	2.7	66.7	1.0	4.2	81.0
cmpy	97.0	8.1	45.2	3.1	41.6	42.9
Oct-TyrR	54.5	8.1	57.1	4.2	8.9	81.0
Ten-m	84.8	13.5	71.4	4.2	15.9	81.0
scro	0.0	83.8	0.0	85.4	79.6	0.0
pdm3	21.2	86.5	4.8	3.1	38.8	14.3
sli	0.0	24.3	47.6	30.2	26.3	90.5
side-VIII	3.0	8.1	21.4	17.7	15.4	95.2
dpr1	81.8	29.7	50.0	4.2	61.7	90.5
CG34370	36.4	0.0	40.5	8.3	2.5	85.7
Lim1	100.0	97.3	19.0	81.2	53.9	0.0
alpha-Man-Ia	45.5	18.9	52.4	4.2	9.5	85.7
Imp	97.0	16.2	97.6	0.0	3.6	95.2
mamo	100.0	16.2	83.3	3.1	8.1	100.0
CG43658	84.8	18.9	76.2	4.2	12.0	33.3
CG32532	66.7	2.7	0.0	0.0	0.0	95.2
ed	51.5	2.7	88.1	0.0	8.4	100.0
Apoltp	0.0	86.5	4.8	3.1	9.8	9.5
ab	21.2	21.6	38.1	2.1	31.8	85.7
dac	97.0	2.7	0.0	0.0	0.8	23.8
beat-IIIb	24.2	21.6	47.6	3.1	10.3	81.0
CG5758	87.9	10.8	83.3	4.2	7.0	76.2
zfh2	3.0	2.7	81.0	2.1	1.4	33.3
fd102C	0.0	0.0	50.0	0.0	0.3	90.5

Table 6.7: Percentages of points within each cluster in $\mathcal{A}$ for which a specific gene is expressed. We only display rows with at least one value lower than $\beta_{\text{low}} = 5\%$ and at least one value higher than $\beta_{\text{high}} = 80\%$.

Bibliography

[1] Ž. Urbančič and J. Giansiracusa, "Ladder decomposition for morphisms of persistence modules," *Journal of Applied and Computational Topology*, pp. 1–41, 2024.

[2] B. Hanin and D. Rolnick, "Deep ReLU networks have surprisingly few activation patterns," *Advances in neural information processing systems*, vol. 32, 2019.

[3] E. Miller, "Data structures for real multiparameter persistence modules," *arXiv preprint arXiv:1709.08155*, 2017.

[4] A. Adcock, D. Rubin, and G. Carlsson, "Classification of hepatic lesions using the matching metric," *Computer vision and image understanding*, vol. 121, pp. 36–42, 2014.

[5] X. Fernández and D. Mateos, "Topological biomarkers for real-time detection of epileptic seizures," *arXiv preprint arXiv:2211.02523*, 2022.

[6] N. Sale, J. Giansiracusa, and B. Lucini, "Quantitative analysis of phase transitions in two-dimensional XY models using persistent homology," *Physical Review E*, vol. 105, no. 2, p. 024121, 2022.

[7] R. J. Gardner, E. Hermansen, M. Pachitariu, Y. Burak, N. A. Baas, B. A. Dunn, M.-B. Moser, and E. I. Moser, "Toroidal topology of population activity in grid cells," *Nature*, vol. 602, no. 7895, pp. 123–128, 2022.

[8] A. Cerri, B. D. Fabio, M. Ferri, P. Frosini, and C. Landi, "Betti numbers in multidimensional persistent homology are stable functions," *Mathematical Methods in the Applied Sciences*, vol. 36, no. 12, pp. 1543–1557, 2013.

[9] M. Kerber, M. Lesnick, and S. Oudot, "Exact computation of the matching distance on 2-parameter persistence modules," in *SoCG 2019-International Symposium on Computational Geometry*, 2019.

[10] U. Bauer and M. Lesnick, "Induced matchings and the algebraic stability of persistence barcodes," *Journal of Computational Geometry*, vol. 6, no. 2, pp. 162–191, 2015.

[11] E. Jacquard, V. Nanda, and U. Tillmann, "The space of barcode bases for persistence modules," *Journal of Applied and Computational Topology*, pp. 1–30, 2022.

[12] A. De Gregorio, M. Guerra, S. Scaramuccia, and F. Vaccarino, "Parallel decomposition of persistence modules through interval bases," *arXiv preprint arXiv:2106.11884*, 2021.

[13] R. González Díaz and M. Soriano Trigueros, "Basis-independent partial matchings induced by morphisms between persistence modules," *ArXiv. org, ArXiv: 2006.11100*, 2020.

[14] K. Turner, "Representing vineyard modules," 2023.

[15] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *International conference on machine learning*. PMLR, 2013, pp. 1319–1327.

[16] Y. LeCun, C. Cortes, and C. J. Burges, "The MNIST database of handwritten digits," http://yann.lecun.com/exdb/mnist/.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017. [Online]. Available: https://arxiv.org/abs/1412.6980

[19] L. Fajstrup, E. Goubault, E. Haucourt, S. Mimram, and M. Raussen, *Directed algebraic topology and concurrency*. Springer, 2016, vol. 138.

[20] S. Lim, F. Mémoli, and Z. Smith, "The Gromov–Hausdorff distance between spheres," *Geometry & Topology*, vol. 27, no. 9, pp. 3733–380 0, 2023.

[21] A. Park, V. Croset, N. Otto, D. Agarwal, C. D. Treiber, E. Meschi, D. Sims, and S. Waddell, "Gliotransmission of D-serine promotes thirst-directed behaviors in Drosophila," *Current Biology*, vol. 32, no. 18, pp. 3952–3970, 2022.

[22] O. Bobrowski and P. Skraba, "Cluster persistence for weighted graphs," *Entropy*, vol. 25, no. 12, p. 1587, 2023.

[23] E. Deza and M. M. Deza, *Encyclopedia of distances*. Springer, 2009.

[24] D. Burago, Y. Burago, S. Ivanov *et al.*, *A course in metric geometry*. American Mathematical Society Providence, 2001, vol. 33.

[25] J. Munkres, *Topology*. Pearson, 2013.

[26] A. C. Mennucci, "On asymmetric distances," *Analysis and Geometry in Metric Spaces*, vol. 1, no. 2013, pp. 200–231, 2013.

[27] N. J. Kalton and M. I. Ostrovskii, "Distances between Banach spaces," *Forum Mathematicum*, 1999.

[28] H. Edelsbrunner and J. L. Harer, *Computational topology: an introduction*. American Mathematical Society, 2022.

[29] V. De Silva and G. E. Carlsson, "Topological estimation using witness complexes." in *PBG*, 2004, pp. 157–166.

[30] M. C. McCord, "Homotopy type comparison of a space with complexes associated with its open covers," *Proceedings of the American Mathematical Society*, vol. 18, no. 4, pp. 705–708, 1967.

[31] K. Borsuk, "On the imbedding of systems of compacta in simplicial complexes," *Fundamenta Mathematicae*, vol. 35, no. 1, pp. 217–234, 1948.

[32] A. Hatcher, *Algebraic Topology*. Cambridge University Press, 2002.

[33] P. Niyogi, S. Smale, and S. Weinberger, "Finding the homology of submanifolds with high confidence from random samples," *Discrete & Computational Geometry*, vol. 39, pp. 419–441, 2008.

[34] J. Kim, J. Shin, F. Chazal, A. Rinaldo, and L. Wasserman, "Homotopy reconstruction via the Cech Complex and the Vietoris-Rips complex," in *SoCG 2020 - 36th International Symposium on Computational Geometry*, 2020.

[35] V. De Silva and R. Ghrist, "Coverage in sensor networks via persistent homology," *Algebraic & Geometric Topology*, vol. 7, no. 1, pp. 339–358, 2007.

[36] D. Attali, A. Lieutier, and D. Salinas, "Vietoris-Rips complexes also provide topologically correct reconstructions of sampled shapes," in *Proceedings of the twenty-seventh annual symposium on Computational geometry*, 2011, pp. 491–500.

[37] A. E. Sizemore, C. Giusti, A. Kahn, J. M. Vettel, R. F. Betzel, and D. S. Bassett, "Cliques and cavities in the human connectome," *Journal of computational neuroscience*, vol. 44, pp. 115–145, 2018.

[38] J. W. Milnor, *Morse theory*. Princeton university press, 1963, no. 51.

[39] E. G. Escolar and Y. Hiraoka, "Persistence modules on commutative ladders of finite type," *Discrete & Computational Geometry*, vol. 55, no. 1, pp. 100–157, 2016.

[40] G. Carlsson and V. De Silva, "Zigzag persistence," *Foundations of computational mathematics*, vol. 10, pp. 367–405, 2010.

[41] W. Crawley-Boevey, "Decomposition of pointwise finite-dimensional persistence modules," *Journal of Algebra and its Applications*, vol. 14, no. 05, p. 1550066, 2015.

[42] D. Cohen-Steiner, H. Edelsbrunner, J. Harer, and Y. Mileyko, "Lipschitz functions have L p-stable persistence," *Foundations of computational mathematics*, vol. 10, no. 2, pp. 127–139, 2010.

[43] U. Bauer and M. Lesnick, "Persistence diagrams as diagrams: A categorification of the stability theorem," in *Topological Data Analysis*. Springer, 2020, pp. 67–96.

[44] L. Fajstrup and J. P. Costa, "On the hierarchy of d-structures," *Order*, vol. 34, no. 1, pp. 139–163, 2017.

[45] T. Nawy, "Single-cell sequencing," *Nature methods*, vol. 11, no. 1, pp. 18–18, 2014.

[46] A. Haque, J. Engel, S. A. Teichmann, and T. Lönnberg, "A practical guide to single-cell RNA-sequencing for biomedical research and clinical applications," *Genome medicine*, vol. 9, pp. 1–12, 2017.

[47] A. T. Lun, D. J. McCarthy, and J. C. Marioni, "A step-by-step workflow for low-level analysis of single-cell RNA-seq data with Bioconductor," *F1000Research*, vol. 5, 2016.

[48] R. A. Amezquita, A. T. Lun, E. Becht, V. J. Carey, L. N. Carpp, L. Geistlinger, F. Marini, K. Rue-Albrecht, D. Risso, C. Soneson *et al.*, "Orchestrating single-cell analysis with Bioconductor," *Nature methods*, vol. 17, no. 2, pp. 137–145, 2020.

[49] Orcestrating Single-Cell Analysis with Bioconductor. Visited on 2024-29-05. [Online]. Available: https://bioconductor.org/books/3.18/OSCA/

[50] X. Yu, F. Abbas-Aghababazadeh, Y. A. Chen, and B. L. Fridley, "Statistical and bioinformatics analysis of data from bulk and single-cell RNA sequencing experiments," *Translational Bioinformatics for Therapeutic Development*, pp. 143–175, 2021.

[51] S. C. Hicks, F. W. Townes, M. Teng, and R. A. Irizarry, "Missing data and technical variability in single-cell rna-sequencing experiments," *Biostatistics*, vol. 19, no. 4, pp. 562–578, 2018.

[52] *Mapping and sequencing the human genome.* National Academies Press, 1988.

[53] National Cancer Institute dictionary of cancer terms. Visited on 2024-27-05. [Online]. Available: https://www.cancer.gov/publications/dictionaries/cancer-terms/def/transcription

[54] A. T. Lun, S. Riesenfeld, T. Andrews, T. P. Dao, T. Gomes, P. in the 1st Human Cell Atlas Jamboree, and J. C. Marioni, "EmptyDrops: distinguishing cells from empty droplets in droplet-based single-cell RNA sequencing data," *Genome biology*, vol. 20, pp. 1–9, 2019.

[55] C. S. McGinnis, L. M. Murrow, and Z. J. Gartner, "DoubletFinder: doublet detection in single-cell RNA sequencing data using artificial nearest neighbors," *Cell systems*, vol. 8, no. 4, pp. 329–337, 2019.

[56] K. Mullis, F. Faloona, S. Scharf, R. Saiki, G. Horn, and H. Erlich, "Specific enzymatic amplification of DNA in vitro: the polymerase chain reaction," in *Cold Spring Harbor symposia on quantitative biology*, vol. 51. Cold Spring Harbor Laboratory Press, 1986, pp. 263–273.

[57] M. Fakruddin, K. S. B. Mannan, A. Chowdhury, R. M. Mazumdar, M. N. Hossain, S. Islam, and M. A. Chowdhury, "Nucleic acid amplification: Alternative methods of polymerase chain reaction," *Journal of Pharmacy and Bioallied Sciences*, vol. 5, no. 4, pp. 245–252, 2013.

[58] M. D. Robinson and G. K. Smyth, "Moderated statistical tests for assessing differences in tag abundance," *Bioinformatics*, vol. 23, no. 21, pp. 2881–2887, 2007.

[59] S. Anders and W. Huber, "Differential expression analysis for sequence count data," *Nature Precedings*, pp. 1–1, 2010.

[60] C. Hafemeister and R. Satija, "Normalization and variance stabilization of single-cell rna-seq data using regularized negative binomial regression," *Genome biology*, vol. 20, no. 1, p. 296, 2019.

[61] S. Choudhary and R. Satija, "Comparison and evaluation of statistical error models for scRNA-seq," *Genome biology*, vol. 23, no. 1, p. 27, 2022.

[62] R. Jiang, T. Sun, D. Song, and J. J. Li, "Statistics or biology: the zero-inflation controversy about scRNA-seq data," *Genome biology*, vol. 23, no. 1, p. 31, 2022.

[63] T. H. Kim, X. Zhou, and M. Chen, "Demystifying "drop-outs" in single-cell UMI data," *Genome biology*, vol. 21, no. 1, p. 196, 2020.

[64] Y. Hao, T. Stuart, M. H. Kowalski, S. Choudhary, P. Hoffman, A. Hartman, A. Srivastava, G. Molla, S. Madad, C. Fernandez-Granda *et al.*, "Dictionary learning for integrative, multimodal and scalable single-cell analysis," *Nature biotechnology*, vol. 42, no. 2, pp. 293–304, 2024.

[65] M. F. Freeman and J. W. Tukey, "Transformations related to the angular and the square root," *The annals of mathematical statistics*, pp. 607–611, 1950.

[66] P.-Y. Tung, J. D. Blischak, C. J. Hsiao, D. A. Knowles, J. E. Burnett, J. K. Pritchard, and Y. Gilad, "Batch effects and the effective design of single-cell gene expression studies," *Scientific reports*, vol. 7, no. 1, p. 39921, 2017.

[67] L. Haghverdi, A. T. Lun, M. D. Morgan, and J. C. Marioni, "Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors," *Nature biotechnology*, vol. 36, no. 5, pp. 421–427, 2018.

[68] A. Butler, P. Hoffman, P. Smibert, E. Papalexi, and R. Satija, "Integrating single-cell transcriptomic data across different conditions, technologies, and species," *Nature biotechnology*, vol. 36, no. 5, pp. 411–420, 2018.

[69] T. Stuart, A. Butler, P. Hoffman, C. Hafemeister, E. Papalexi, W. M. Mauck, Y. Hao, M. Stoeckius, P. Smibert, and R. Satija, "Comprehensive integration of single-cell data," *cell*, vol. 177, no. 7, pp. 1888–1902, 2019.

[70] Y. Zhang, G. Parmigiani, and W. E. Johnson, "ComBat-seq: batch effect adjustment for RNA-seq count data," *NAR genomics and bioinformatics*, vol. 2, no. 3, p. lqaa078, 2020.

[71] K. E. Witte, O. Hertel, B. A. Windmoeller, L. P. Helweg, A. L. Hoeving, C. Knabbe, T. Busche, J. F. Greiner, J. Kalinowski, T. Noll *et al.*, "Nanopore sequencing reveals global transcriptome signatures of mitochondrial and ribosomal gene expressions in various human cancer stem-like cell populations," *Cancers*, vol. 13, no. 5, p. 1136, 2021.

[72] S. Márquez-Jurado, J. Díaz-Colunga, R. P. das Neves, A. Martinez-Lorente, F. Almazán, R. Guantes, and F. J. Iborra, "Mitochondrial levels determine variability in cell death by modulating apoptotic gene expression," *Nature communications*, vol. 9, no. 1, p. 389, 2018.

[73] P. G. Ferreira, M. Muñoz-Aguirre, F. Reverter, C. P. Sa Godinho, A. Sousa, A. Amadoz, R. Sodaei, M. R. Hidalgo, D. Pervouchine, J. Carbonell-Caballero *et al.*, "The effects of death and post-mortem cold ischemia on human tissue transcriptomes," *Nature communications*, vol. 9, no. 1, p. 490, 2018.

[74] I. Gallego Romero, A. A. Pai, J. Tung, and Y. Gilad, "RNA-seq: impact of RNA degradation on transcript quantification," *BMC biology*, vol. 12, pp. 1–13, 2014.

[75] P. Qiu, "Embracing the dropouts in single-cell RNA-seq analysis," *Nature communications*, vol. 11, no. 1, p. 1169, 2020.

[76] P. V. Kharchenko, L. Silberstein, and D. T. Scadden, "Bayesian approach to single-cell differential expression analysis," *Nature methods*, vol. 11, no. 7, pp. 740–742, 2014.

[77] W. V. Li and J. J. Li, "An accurate and robust imputation method scImpute for single-cell RNA-seq data," *Nature communications*, vol. 9, no. 1, p. 997, 2018.

[78] D. Van Dijk, R. Sharma, J. Nainys, K. Yim, P. Kathail, A. J. Carr, C. Burdziak, K. R. Moon, C. L. Chaffer, D. Pattabiraman *et al.*, "Recovering gene interactions from single-cell data using data diffusion," *Cell*, vol. 174, no. 3, pp. 716–729, 2018.

[79] C. Chen, C. Wu, L. Wu, X. Wang, M. Deng, and R. Xi, "scRMD: imputation for single cell RNA-seq data via robust matrix decomposition," *Bioinformatics*, vol. 36, no. 10, pp. 3156–3161, 2020.

[80] A. Mongia, D. Sengupta, and A. Majumdar, "McImpute: matrix completion based imputation for single cell RNA-seq data," *Frontiers in genetics*, vol. 10, p. 9, 2019.

[81] T. Peng, Q. Zhu, P. Yin, and K. Tan, "SCRABBLE: single-cell RNA-seq imputation constrained by bulk RNA-seq data," *Genome biology*, vol. 20, pp. 1–12, 2019.

[82] D. Risso, F. Perraudeau, S. Gribkova, S. Dudoit, and J.-P. Vert, "A general and flexible method for signal extraction from single-cell RNA-seq data," *Nature communications*, vol. 9, no. 1, p. 284, 2018.

[83] M. Wang, J. Gan, C. Han, Y. Guo, K. Chen, Y.-z. Shi, and B.-g. Zhang, "Imputation methods for scRNA sequencing data," *Applied Sciences*, vol. 12, no. 20, p. 10684, 2022.

[84] S. Islam, U. Kjällquist, A. Moliner, P. Zajac, J.-B. Fan, P. Lönnerberg, and S. Linnarsson, "Characterization of the single-cell transcriptional landscape by highly multiplex RNA-seq," *Genome research*, vol. 21, no. 7, pp. 1160–1167, 2011.

[85] N. C. Chung and J. D. Storey, "Statistical significance of variables driving systematic variation in high-dimensional data," *Bioinformatics*, vol. 31, no. 4, pp. 545–554, 2014.

[86] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[87] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[88] P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti, "Generalized Louvain method for community detection in large networks," in *2011 11th international conference on intelligent systems design and applications*. IEEE, 2011, pp. 88–93.

[89] P. Pons and M. Latapy, "Computing communities in large networks using random walks," in *Computer and Information Sciences-ISCIS 2005: 20th International Symposium, Istanbul, Turkey, October 26-28, 2005. Proceedings 20*. Springer, 2005, pp. 284–293.

[90] Scikit-learn user guide: Clustering. Visited on 2025-23-02. [Online]. Available: https://scikit-learn.org/stable/modules/clustering.html

[91] C. Soneson and M. Delorenzi, "A comparison of methods for differential expression analysis of RNA-seq data," *BMC bioinformatics*, vol. 14, pp. 1–18, 2013.

[92] F. Chazal, D. Cohen-Steiner, M. Glisse, L. J. Guibas, and S. Y. Oudot, "Proximity of persistence modules and their diagrams," in *Proceedings of the Twenty-Fifth Annual Symposium on Computational Geometry*, ser. SCG '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 237–246. [Online]. Available: https://doi.org/10.1145/1542362.1542407

[93] V. De Silva, E. Munch, and A. Stefanou, "Theory of interleavings on categories with a flow," *Theory and Applications of Categories*, vol. 33, no. 21, pp. 583–607, 2018.

[94] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer, "Stability of persistence diagrams," in *Proceedings of the twenty-first annual symposium on Computational geometry*, 2005, pp. 263–271.

[95] M. Lesnick and M. Wright, "Interactive visualization of 2-D persistence modules," *arXiv preprint arXiv:1512.00180*, 2015.

[96] I. Steinwart, "A sober look at neural network initializations," *arXiv preprint arXiv:1903.11482*, 2019.

[97] R. Pascanu, G. Montufar, and Y. Bengio, "On the number of response regions of deep feed forward networks with piece-wise linear activations," in *International Conference on Learning Representations 2014*, 2014.

[98] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[99] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein, "On the expressive power of deep neural networks," in *international conference on machine learning*. PMLR, 2017, pp. 2847–2854.

[100] T. Serra, C. Tjandraatmadja, and S. Ramalingam, "Bounding and counting linear regions of deep neural networks," in *International conference on machine learning*. PMLR, 2018, pp. 4558–4566.

[101] B. Hanin and D. Rolnick, "Complexity of linear regions in deep networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2596–2604.

[102] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[103] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[104] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*. Springer, 2002, pp. 9–50.

[105] G.-H. Lee, D. Alvarez-Melis, and T. S. Jaakkola, "Towards robust, locally linear deep networks," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=SylCrnCcFX

[106] Initialization of deep neural networks with ReLU activation. [Online]. Available: https://github.com/ZivaUrbancic/Maxout_Initializations

[107] X. Zhang and D. Wu, "Empirical studies on the properties of linear regions in deep neural networks," *arXiv preprint arXiv:2001.01072*, 2020.

[108] D. Arpit, S. Jastrzębski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio *et al.*, "A closer look at memorization in deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 233–242.

[109] M. Grandis, *Directed algebraic topology: models of non-reversible worlds*. Cambridge University Press, 2009, vol. 13.

[110] F. Mémoli and G. Sapiro, "A theoretical and computational framework for isometry invariant recognition of point cloud data," *Foundations of Computational Mathematics*, vol. 5, pp. 313–347, 2005.

[111] D. Jungnickel and D. Jungnickel, *Graphs, networks and algorithms*. Springer, 2005, vol. 3.

[112] D. Mugnolo, "What is actually a metric graph?" *arXiv preprint arXiv:1912.07549*, 2019.

[113] M. Aanjaneya, F. Chazal, D. Chen, M. Glisse, L. J. Guibas, and D. Morozov, "Metric graph reconstruction from noisy data," in *Proceedings of the twenty-seventh annual symposium on Computational geometry*, 2011, pp. 37–46.

[114] Thirsty fly classification. [Online]. Available: https://github.com/ZivaUrbancic/ThirstyFlyClustering

[115] B. Sloley and A. Juorio, "Monoamine neurotransmitters in invertebrates and vertebrates: An examination of the diverse enzymatic pathways utilized to synthesize and inactivate blogenic amines," *International Review of Neurobiology*, vol. 38, pp. 253–303, 1995.

[116] E. R. Kandel, J. H. Schwartz, T. M. Jessell, S. Siegelbaum, A. J. Hudspeth, S. Mack *et al.*, *Principles of neural science*. McGraw-hill New York, 2000, vol. 4.

[117] O. Bobrowski and P. Skraba, "On the universality of random persistence diagrams," *arXiv preprint arXiv:2207.03926*, 2022.

[118] F. Chazal, L. J. Guibas, S. Y. Oudot, and P. Skraba, "Persistence-based clustering in Riemannian manifolds," *Journal of the ACM (JACM)*, vol. 60, no. 6, pp. 1–38, 2013.

[119] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 14, 2001.

[120] J. Shi and J. Malik, "Normalized cuts and image segmentation," in *Proceedings of IEEE computer society conference on computer vision and pattern recognition*. IEEE, 1997, pp. 731–737.

[121] Documentation for KMeans clustering within Scikit-learn library. Visited on 2024-19-06. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans

[122] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms," *Annals of data science*, vol. 2, pp. 165–193, 2015.

[123] Documentation for spectral clustering within Scikit-learn library. Visited on 2024-19-06. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html

[124] Documentation for Louvain algorithm within CDlib library. Visited on 2024-19-06. [Online]. Available: https://cdlib.readthedocs.io/en/0.2.0/reference/cd_algorithms/algs/cdlib.algorithms.louvain.html

[125] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[126] R. Lambiotte, J.-C. Delvenne, and M. Barahona, "Laplacian dynamics and multiscale modular structure in networks," *arXiv preprint arXiv:0812.1770*, 2008.

[127] Documentation of `from_numpy_array` method within networkx library. Visited on 2024-27-06. [Online]. Available: https://networkx.org/documentation/stable/reference/generated/networkx.convert_matrix.from_numpy_array.html

[128] Seurat: R toolkit for single cell genomics. Visited on 2024-27-06. [Online]. Available: https://satijalab.org/seurat/

[129] R. Satija, J. A. Farrell, D. Gennert, A. F. Schier, and A. Regev, "Spatial reconstruction of single-cell gene expression data," *Nature biotechnology*, vol. 33, no. 5, pp. 495–502, 2015.

[130] Bioconductor: Open source software for bioinformatics. Visited on 2024-27-06. [Online]. Available: https://www.bioconductor.org/

[131] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*. Cambridge University Press Cambridge, 2008, vol. 39.

[132] V. Croset, C. D. Treiber, and S. Waddell, "Cellular diversity in the Drosophila midbrain revealed by single-cell transcriptomics," *Elife*, vol. 7, p. e34550, 2018.

[133] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.