UNIVERSITAT DE BARCELONA

Facultat de Matemàtiques
i Informàtica

GRAU DE MATEMÀTIQUES

Treball final de grau

ESTIMATING THE DIMENSIONALITY OF COMPLEX NETWORKS USING PERSISTENT HOMOLOGY

Autor: Meritxell Vila Miñana

Directors: Carles Casacuberta Vergés
Aina Ferrà Marcús

Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 13 de juny de 2023

Contents

Acknowledgements **iii**

Introduction **v**
0.1 The problem of dimension v
0.2 Objectives vi
0.3 Hypothesis vi
0.4 The origins of this work vii
0.5 Work structure vii

1 Preliminary concepts **1**
1.1 Complex networks 1
1.1.1 Introduction to complex networks 1
1.1.2 Geometric model 2
1.2 Persistent homology in graph learning 7
1.2.1 Basic definitions 7
1.2.2 Topological features of graphs 8
1.2.3 Simplicial homology 8
1.2.4 Persistent homology 10
1.2.5 Persistence descriptors and comparison of diagrams 12
1.2.6 Kernels in Hilbert space 15
1.2.7 Extended persistence 17
1.2.8 Heat Kernel Signatures 19

2 Development **21**
2.1 *Detecting the ultra low dimensionality of real networks* 21
2.1.1 Methods 21
2.1.2 Discussion 22
2.2 Dimension detection using persistent homology 23
2.2.1 Surrogate generation and description 23
2.2.2 p_{ij} matrices as distance matrices 24

2.2.3 Euler characteristic 25
2.2.4 Shortest paths 27
2.2.5 Filtration based on the degree of vertices 29
2.2.6 Overview of results 39
2.3 Dimension estimate using graph theory 40

3 Conclusions **43**

Bibliography **45**

Annex **47**
3.1 Persistence diagrams of network surrogates 48
3.2 *Python* code generated to analyse the data using vertex filtrations . . 57

Abstract

In this work, a new interdisciplinary approach is presented to study the dimensionality of complex networks using techniques from topological data analysis (TDA) through a filtration of graphs by vertex degrees. For each of two real-world complex networks, 30 surrogate graphs were generated in each dimension from 1 to 10, and several TDA descriptors of graphs were compared with the corresponding values for the real networks in order to estimate their latent dimension. Total persistence, Wasserstein distance and scale-space kernel dissimilarity, among other descriptors, yielded consistent outcomes. The results of this study suggest that TDA is sensible to the latent dimension of complex networks, and provide conclusions consistent with those obtained in previous studies.

Resum

En aquest treball es presenta una nova metodologia interdisciplinària per estudiar la dimensionalitat de les xarxes complexes utilitzant tècniques d'anàlisi de dades topològica (TDA) mitjançant una filtració de grafs segons els graus dels vèrtexs. Per a cadascuna de les dues xarxes estudiades, es van generar 30 grafs sintètics en cada dimensió d'1 a 10, i s'han comparat diversos descriptors de TDA de grafs amb els valors corresponents de les xarxes reals per tal d'estimar-ne la dimensió latent. La persistència total, la distància de Wasserstein i la dissimilaritat de nuclis, entre d'altres descriptors, han donat resultats consistents. Els resultats d'aquest estudi suggereixen que la TDA és sensible a la dimensió latent de les xarxes complexes i proporcionen conclusions consistents amb les obtingudes en estudis anteriors.

2020 Mathematics Subject Classification. 55N31, 05C82, 62R40

Dedicated to Núria López Barrón

Acknowledgments

First and foremost, I want to thank my advisors Dr. Carles Casacuberta and Aina Ferrà for their persistent guidance in supervising this thesis. For their ambition and deep trust in my work, and for allowing me to grow as a researcher and as a person. I thank Carles Casacuberta for being patient and very generous with me; answering all my doubts and orienting me to new strategies when difficulties arised. I thank Aina Ferrà for introducing me to the computational part of topological data analysis, for being always ready to solve my programming doubts, and for motivating me during the process. I am especially grateful to Dr. M. Ángeles Serrano, who inspired me through the physical background of the problem, and trusted me her most recent advances to treat them from another mathematical approach. I also thank her for being always available to solve my doubts.

I want to thank my friends in the Mathematics and Physics departments, including Marc Ballestero, Beatriz Benavente, Belén Martín, Joan Núñez, Laura Ovejero, Neil Parker, Alba Vika Sánchez, Guillem Sans, Toni Soler, Beñat Susaeta, Lucía Tormo, Natàlia Puig, Vinyet Costa, Johan Sebastian Guzmán, Oriol Martínez, Andrea Pérez, Marc Lladó and Matías Viner for accompanying me and giving me support in the most difficult moments but also in the success during my undergraduate years. And to Àlex Garcia and Andriana Karuk, for reading a draft of this thesis and providing valuable feedback.

I am also grateful to my international friends Sebastián Fichendler, Tadeo Prada, Jacob Prins, Caitlin Gray, Pernilla van Duuren, Rintaro Kanaki, Carlo Sala, Queralt Codinach, Roberta De Matteis and Miriana Ardagna, which I made during my stay at Università di Bologna, and provided me a worldwide view of academic knowledge, and of life. I want to thank Aina Casassas for her inconditional support, and also Miguel Fernández, Dàlia Puig and Irene Forniés, for their endless encouragement. I especially want to thank Sílvia Casacuberta for inspiring me since the day we met, and always supporting and encouraging me; making me feel proud of who I am and what I accomplished.

Lastly, I want to thank my family for their unconditional love and generosity towards me. To my brother, for being the most important person in my life, constantly believing in me and always being there; to my mother, for being my most supportive person; and to my father, for always making me feel proud of myself. I extend the acknowledge to the rest of my family, especially to my grandparents, cousins and uncles, who always believe in me. This thesis is dedicated to the memory of Núria López, my grandmother, whose strength, positivism and perseverance inspired me throughout my life, and during this last month.

Introduction

0.1 The problem of dimension

The problem of dimension, that is, identifying the dimensionality of the relevant space associated with a given phenomenon, is common in many disciplines of the natural sciences. For example, in statistical physics and string theory, the number of dimensions is important for understanding critical properties and particle interactions. Other illustrative examples can be computer science and network science, where high-dimensional data is often simplified by reducing redundancy and finding patterns in a lower-dimensional space. However, defining similarity distances and determining the appropriate number of dimensions can be challenging. This is why, although it is not an easy task, identifying the dimensionality of a complex phenomenon is important for understanding and simplifying complex data and the interactions therein.

In [1], P. Almagro, M. Boguñá and M. A. Serrano introduced a model and a method to infer the dimensionality of the latent hyperbolic space underlying the connectivity of a complex network. It is a model-driven approach and it assumes that real-world networks are well described by the geometric $\mathbb{S}^D/\mathbb{H}^{D+1}$ model (see Section 1.1.2), in D dimensions. It is a generalization of the $\mathbb{S}^1/\mathbb{H}^2$ model, which is based on fundamental principles to describe the observed connectivity of unweighted and undirected networks, and it is described in detail in the first part of this work.

It is remarkable that the structure of a network can reveal its intrinsic dimensionality, since it is based on a latent space where nodes are more likely to be connected if they are closer to each other. In [1] the authors propose to measure dimensionality by analyzing the frequencies of chordless cycles of different lengths in the network. These are closed paths in the network that do not have any edges connecting nodes within the cycle. Their frequencies can provide insights into the network's underlying structure. We call *surrogates* the synthetic graphs generated by the $\mathbb{S}^D$ model for each D, imitating the properties of real-world networks. Surrogates can be used to test in which dimension the density of cycles is more similar to the one from the original network.

Persistent homology of complex networks (see Section 1.2) also focuses on cycles to obtain geometrical information from data. Therefore, in the current work, a further step of abstraction is taken to give a response on whether topological data analysis (TDA) can shed any light on the dimensionality problem, by studying cycles of surrogates. This work incorporates new ideas by establishing connections between the fields of persistent homology and complex networks. It explores an

uncharted territory by means of TDA techniques. This interdisciplinary approach provides fresh insights into the nature of complex networks and their underlying structures. By integrating methodologies from persistent homology and complex networks, this study not only contributes to both disciplines but also introduces novel ideas that could inspire further investigations in both areas.

0.2 Objectives

This work aims to acquire a solid background of complex networks and of persistent homology, for a further application of this knowledge to the problem of dimensionality. It is divided in two parts; the first one contains theoretical preliminary concepts. Those are used in the second part, which is more computational.

The main goal of this work is to seek for topological descriptors, in the context of persistent homology, that reveal the characteristics of the complex networks studied in [1]. To achieve this, we propose a suitable filtration for the persistent homology process, as well as appropriate dissimilarity measures between persistence diagrams. In the end, the results are compared with those obtained in [1]. The ultimate objective is to determine whether TDA is able to detect patterns caused by dimension differences in real-world networks.

0.3 Hypothesis

We work under the hypothesis that persistent homology should be able to detect the dimension of a complex network, provided that the dimension in which we work affects the probability of connection between nodes (see Equation 1.12 in Section 1.1.2) and therefore the distribution of cycles.

We propose that the best filtration (see Section 1.2.4) to analyse complex networks should be the one based on the degree of their nodes. Given the nature of these networks and their surrogates (see Section 1.1.2), the number of vertices gives little information. Instead, how edges are combined (and which cycles they produce) provides relevant information about the dimension of the network. This is suggested in [1], where the dimension is found to be related to the density of cycles. Therefore, it is strongly believed that a filtration of graphs based on the degree of nodes can be appropriate to treat this problem.

Moreover, when using TDA techniques, usually a study is performed in homological dimensions 0 and 1 (see Section 1.2.4). In this work, a hypothesis is made that H1 should give more relevant information than H0, since H1 focuses on the study of cycles. Using shortest-path distance in graphs and the corresponding

Vietoris-Rips complexes is not suitable for this kind of study due to the small-world phenomenon in complex networks (see Section 1.1.1).

0.4 The origins of this work

This research is framed in the context of a double bachelor degree, in Mathematics and Physics. A global analysis of the problem is pretended to be done from a mathematical point of view as well as from the physics perspective. That is why, further from the theoretical understanding of complex networks, persistent homology and graph theory, an experimental part is performed in the second part of this work in order to put in practice the acquired skills. Therewith, another perspective is added: the programming one, combining data analysis techniques with conceptual knowledge.

Very few articles in the literature [3, 4] combine the study of complex networks with TDA. However, to our knowledge, this is the first study attempting to estimate the dimensionality of complex networks using persistence descriptors of graphs. The methodological novelties in this work are the choice of a degree-based filtration and the selection of descriptors that best capture the distribution of cycles of the real-world networks on which our analysis focuses.

0.5 Work structure

This work is structured as follows. In Chapter 1 some preliminary concepts are introduced. In Section 1.1 complex networks are described, starting with an introduction in Section 1.1.1. In Section 1.1.2 a detailed description of the geometric model is provided. An explanation is given for the $\mathbb{S}^1$ model, the $\mathbb{H}^2$ model and the generalization to D dimensions with the $\mathbb{S}^D$ model. In Section 1.2 an overview to persistent homology in graph learning is given. Section 1.2.1 starts with some basic definitions from graph theory, and then in Section 1.2.2 some topological features of graphs are defined. *Euler's characteristic* and the *Betti numbers* are introduced. Then, in Section 1.2.3 the basic concepts of simplicial homology are reviewed. *Simplicial complexes* and the *Vietoris-Rips complex* are defined. Further, in Section 1.2.4 the concepts of *filtration*, *persistent homology group*, *persistence diagram* and *persistence barcode* are defined, followed, in Section 1.2.5, by a series of persistence descriptors and measures of distance between diagrams: the *bottleneck distance*, the *Wasserstein distance*, *total persistence*, *entropy*, *landscapes* and *persistence images*. A small introduction to *kernel* theory is given in Section 1.2.6. Section 1.2.7 defines the concept of *extended persistence* from a theoretical point of view. This section is ended with an illustrative example of how extended persistence has

been adapted for the purposes of this research, with a degree-based filtration. In the end of this chapter, a brief introduction to *heat kernel signatures*, using the *normalized graph Laplacian* is offered in Section 1.2.8.

In Chapter 2 a method to solve the problem of dimension is proposed and implemented using TDA techniques. Code has been programmed in *Python* to analyse the data, implementing resources from the **Gudhi** library. At the beginning of this chapter, in Section 2.1, the methods implemented in [1] to solve the problem of dimension are discussed. In Section 2.2 different approaches to solve the problem are analysed from the TDA perspective. Section 2.2.1 describes the data that is being analysed in this work. Then, Sections 2.2.2 and 2.2.4 study the possibility to infer a metric by studying the probability matrix from the model that generates the surrogates, as well as the *shortest path distance*. In Section 2.2.3 the relation between Euler's characteristic and the dimension D is studied in detail. And in Section 2.2.5 a filtration based on the degree of vertices is described and implemented. Two real-world networks are studied, and 30 surrogates are generated for each network and dimension using the $\mathbb{S}^D$ model. Some topological descriptors are used to compare each of the surrogates with the original network. Graphical results are drawn and discussed also in Section 2.2.5. In the end, a study is done using graph theory to detect the dimension of networks. The results are shown in Section 2.3.

Finally, in Chapter 3 this research is concluded with an overview on the present work and an outlook of new research lines that could derive from this one. All the figures and plots that are not referenced throughout this work have been done by the author, in the majority of the cases using *Python*. Two annexes are adjoined at the end. Annex 3.1 contains 60 persistence diagrams and 60 extended persistence diagrams of three example surrogates for each dimension $D \in \{1, \dots, 10\}$ of the *CElegans-C* and *Human1* networks. In Annex 3.2 the complete code in *Python* elaborated to analyse the data is given.

Chapter 1

Preliminary concepts

1.1 Complex networks

1.1.1 Introduction to complex networks

Many real complex networks are natural geometric objects and can be mapped into hidden low-dimensional metric spaces with hyperbolic geometry, where distances determine the likelihood of the interactions and encode the different intrinsic attributes determining how similar the elements of the system are. Network geometry has become one of the fundamental areas within the field of network science devoted to the discovery and modeling of nontrivial geometric properties of complex networks.

Complex networks typically have been studied as graphs where elements are represented as nodes and their interactions as links. Graphs of real networks, though, are not regular lattices nor are they completely random. The most important properties that we find are the *small-world* phenomenon, connecting every pair of nodes in a network, on average, by a small number of intermediate links; *scale-free* distributions of the number of connections per node (degree), usually being a power-law of the form $P(k) \sim k^{-\gamma}$ with $\gamma \in [2, 3]$; *heterogeneity* and the presence of many triangles, that is, *clustering* [5].

It can be thought that, because of the small-world effect, every pair of nodes is connected in a few steps and, therefore, the system lacks a metric structure defined. It is true that in this type of network, the distribution of shortest path lengths among pairs of nodes is sharply peaked around its average. Therefore, the minimum distance between two nodes is almost the same for every pair. For this reason, complex networks are often considered difficult to map. Nevertheless, some networks are embedded in metric spaces, like airport networks, trading routes, neural networks, etc. In this work we will take the particular example of

connectomes. This is, several neural networks representing an organism's nervous system, made up of neurons (nodes) which communicate through synapses (links). A connectome is constructed by tracing the neuron in a nervous system and mapping where neurons are connected through synapses.

Taking everything stated before into consideration, in [5] a model was developed to embed complex networks in *hidden metric space* models with an underlying *hyperbolic geometry*. In this hidden metric space, the probability of connection between two nodes depends on their distance in this space, and combines *popularity* and *similarity* dimensions of the nodes, such that more popular and similar nodes are more likely to interact. This model is able to reproduce universal features observed in real-world systems, such as the small-world effect, scale-free degree distributions or clustering, among others.

This model assumes that there exists some similarity between nodes, and the fact that geometry is an appropriate mathematical formalism. Another keystone is that clustering, or the number of triangles, can be considered as a consequence of the triangular inequality. Conferring, therefore, a connection between the topology of the network and the geometry of the underlying metric space.

In order to develop a model for these networks, the main difficulties arise as a consequence of the small-world effect. Since the distance between two nodes grows with $d \sim \log N$, where N is the number of nodes, this implies an exponential expansion of space, meaning that the number of nodes within a disk of a certain radius grows exponentially with the radius: $N \sim e^d$. From here an underlying hyperbolic geometry arises. It can be explained, though, with the networks being heterogeneous; since some nodes have high degree, and the others very low, nodes can be organized in such a way that the network has a tree structure [2].

1.1.2 Geometric model

Many real networks, indeed, share the hyperbolic space as their hidden metric space. It is the case of Internet, metabolic networks, trading maps or human brain structures. There are two presentations of the hyperbolic space: as the hyperbolic plane $\mathbb{H}^2$, usually with the Poincaré disk representation, or as its quasi-isomorphic version, the $\mathbb{S}^1$ model, which can then be generalized to the $\mathbb{S}^D$ model. This quasi-isomorphism allows us to use both models, depending on the particular application. The $\mathbb{S}^1$ model is convenient for theory development, analytical calculations or implementation of embedding techniques. Whereas the $\mathbb{H}^2$ version is best for visualization purposes or analyzing navigation properties. An interesting physical interpretation is that this model corresponds to an entropy-maximizing probabilistic mixture of grand canonical network ensembles. Here, links can be thought of as non-interacting fermions whose energies depend on distances [5].

The $\mathbb{S}^1$ model

In the $\mathbb{S}^1$ model, the hidden metric space is a circle of radius $R = N/2\pi$, being N the number of nodes. Without loss of generality, this sets density to 1. Every node i is defined by two variables: a *hidden degree* κ_i, related to its popularity, and an *angular position* θ_i. The probability of connection between nodes has to increase with the product of hidden degrees, and decrease as long as two nodes are far from each other. It is said that the probability of connection follows a gravity law, in which similar nodes are angularly closer and, in consequence, probably connected. In this way, two non similar nodes will only be likely to be connected if they have high popularity [5].

Specifically, nodes i and j are connected with probability

$$p_{ij} = \frac{1}{1 + \left(\frac{d_{ij}}{\mu\kappa_i\kappa_k}\right)^{\beta}} \tag{1.1}$$

where β controls the level of clustering of the network, d_{ij} is the angular distance between nodes i and j and μ controls the average degree. For $\beta < 1$, networks are unclustered in the infinite size limit, and for $\beta > 1$ we obtain networks with finite clustering in the thermodynamic limit. [1] Thus, there is a structural phase transition for $\beta = 1$. In [6] it is proved that Eq. (1.1) is the only possible choice for the connection probability that creates maximally random, clustered, small-world and heterogeneous networks, without two-point hidden degree correlations, corresponding, indeed, to the Fermi-Dirac form.

One can think of a simple algorithm to generate graphs from the $\mathbb{S}^1$ model, with uncorrelated hidden degrees, angular positions homogeneously distributed, and working in the $N \gg 1$ thermodynamic limit. First, we fix the number of nodes, $\beta > 1$, and the target average degree $\langle k \rangle$. Then μ is defined as follows

$$\mu = \frac{\beta}{2\pi\langle k \rangle} \sin\left(\frac{\pi}{\beta}\right). \tag{1.2}$$

Second, we assign each node i an angular position θ_i from a [0,1]-uniform distribution, and a hidden degree κ_i from a distribution $\rho(\kappa)$ so that $\langle \kappa \rangle = \langle k \rangle$. Finally, we connect every pair of nodes with the probability given by Eq. (1.1) [5].

In the thermodynamic limit, using this parametrization, the expected degree of a node with a hidden degree κ is k. The hidden degrees κ_i can be obtained

[1] In statistical mechanics, the thermodynamic limit of a system is the limit for a large number N of particles where the volume is taken to grow proportionally with N, that is, $N \to \infty$, $V \to \infty$, $N/V =$ constant.

from different distributions; however, in many real networks a Pareto distribution is observed:

$$\rho(\kappa) = (\gamma-1)\kappa_0^{\gamma-1}\kappa^{-\gamma};\ \ \kappa > \kappa_0 = \frac{\gamma-2}{\gamma-1}\langle k\rangle;\ \ \gamma > 2, \tag{1.3}$$

which results in a power-law degree distribution $P(k) \sim k^{-\gamma}$ for $N \gg 1$ [5].

To simulate networks with $2 < \gamma < 3$ and compensate finite size effects in networks with $\gamma \gtrsim 2$ we need to introduce a cutoff in $\rho(\kappa)$. For this, one uses a cutoff of the form

$$\rho(\kappa) = \frac{(\gamma-1)\kappa_0^{\gamma-1}}{1-(\frac{\kappa_c}{\kappa_0})^{1-\gamma}}\kappa^{-\gamma};\ \ \kappa_0 < \kappa < \kappa_c, \tag{1.4}$$

where $\kappa_0 = \dfrac{1-N^{-1}}{1-N^{\frac{2-\gamma}{\gamma-1}}}\dfrac{\gamma-2}{\gamma-1}\langle k\rangle$ and $\kappa_c = \kappa_0 N^{\frac{1}{\gamma-1}}$ [5].

The $\mathbb{H}^2$ model

In the $\mathbb{H}^2$ model, the hidden metric space is the two-dimensional hyperbolic disk with radius

$$R_{\mathbb{H}^2} = 2\ \ln\ \frac{N}{\pi\mu\kappa_0^2} \tag{1.5}$$

and metric tensor

$$ds^2 = dr^2 + \sinh^2 r d\theta^2. \tag{1.6}$$

In this model, points in $\mathbb{H}^2$ (of constant curvature $K = -1$) are characterized by two coordinates (r, θ). In this quasi-isomorphic geometric version of $\mathbb{S}^1$, popularity and similarity are combined into a single distance in the hyperbolic plane, such that closer nodes in this metric are more likely to be connected. In the limit where

$$\sin^2\frac{\Delta\theta_{ij}}{2} \gg \frac{\cosh(r_i - r_j)}{\cosh(r_i - r_j) + \cosh(r_i + r_j)},$$

the distance between two nodes can be approximated by

$$x_{ij} \approx r_i + r_j + 2\ln\sin\frac{\Delta_{ij}}{2} \approx r_i + r_j + 2\ln\frac{\Delta_{ij}}{2}, \tag{1.7}$$

which turns to be a very good approximation for almost all real networks which we are working with [5].

As in the $\mathbb{S}^1$ model, maximum entropy ensembles are obtained when the connection probability takes the Fermi-Dirac form, interpreting the network as a set of fermions (edges) that can be in different states (pairs of nodes), with energies

defined by the corresponding hyperbolic distances, and where β is the inverse of the system's temperature: [2]

$$p_{ij} = \frac{1}{1 + e^{\frac{\beta}{2}(x_{ij} - R_{\mathbb{H}^2})}}. \tag{1.8}$$

If the angular distribution of nodes is uniform and the radius of nodes are distributed with probability density

$$\rho(r) = \alpha \frac{\sinh \alpha r}{\cosh \alpha R_{\mathbb{H}^2} - 1}; \quad r \in [0, R_{\mathbb{H}^2}]; \quad \alpha \geq 1/2, \tag{1.9}$$

then the model generates graphs with a power-law degree distribution, with exponent, in this case, $\gamma = 2\alpha + 1$; see [5].

To establish a mapping between $\mathbb{S}^1$ and $\mathbb{H}^2$, the angular coordinates remain as in the $\mathbb{S}^1$ model, but the hidden degrees are transformed into radial coordinates:

$$(\kappa_i, \theta_i) \mapsto (r_i, \theta_i) = (R_{\mathbb{H}^2} - 2 \ln \frac{\kappa_i}{\kappa_0}, \theta_i), \tag{1.10}$$

where $R_{\mathbb{H}^2}$ is given by Eq. (1.5). This way, higher degree nodes are located closer to the center of the disk, and low degree nodes are placed near its boundary. Now, for large $R_{\mathbb{H}^2}$, i.e., for $N \gg 1$, if hidden degrees are power-law distributed according to Eq. (1.3), then the mapping Eq. (1.10) implies that the radial coordinates are distributed as in Eq. (1.9).

Substituting Eq. (1.5) and Eq. (1.10) into the probability in $\mathbb{S}^1$ from Eq. (1.1), we obtain the connection probability

$$p_{ij} = \frac{1}{1 + e^{\frac{\beta}{2}(\tilde{x}_{ij} - R_{\mathbb{H}^2})}}; \quad \tilde{x_{ij}} = r_i + r_j + \ln \frac{\Delta_{ij}}{2}. \tag{1.11}$$

As mentioned before, $\tilde{x_{ij}} \approx x_{ij}$ for almost all pairs of nodes, so we can conclude that in the thermodynamic limit both models are equivalent [5].

[2] For a system of identical fermions in thermodynamic equilibrium, the average number of fermions in a single state i is given by $\overline{n}_i = 1/(1 + e^{(\epsilon_i - \mu)/k_B T})$, where k_B is Boltzmann constant, T is the absolute temperature, ϵ_i is the energy of the state i, and μ is the chemical potential.

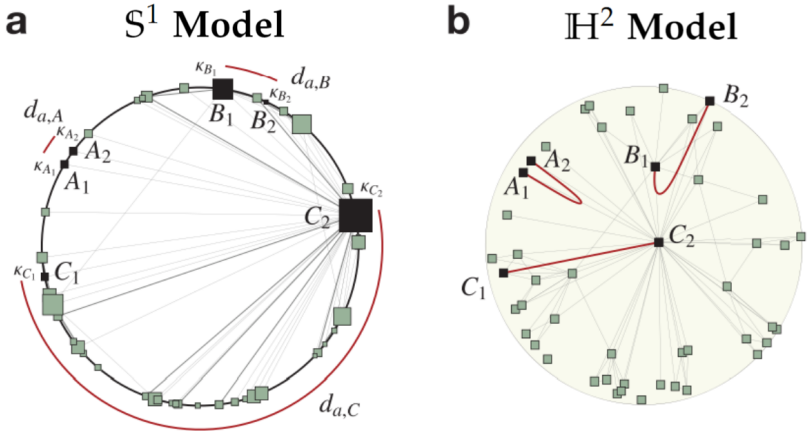

Figure 1.1: (a) $\mathbb{S}^1$ model. The similarity distances d_a between some pairs of nodes have been highlighted. The size of a node is proportional to its hidden degree κ_i. (b) $\mathbb{H}^2$ model in the hyperbolic plane. Nodes in the different pairs are separated by the same hyperbolic distance. Nodes are equally sized, but nodes with higher hidden degree are positioned closer to the center [5].

The $\mathbb{S}^D$ model

Analogously to the $\mathbb{S}^1$ model, in the $\mathbb{S}^D$ model a node i is assigned two hidden variables: a hidden degree κ_i, quantifying its popularity, and a position v_i in a similarity space (represented as a D-sphere, that is, a hypersphere in a $D+1$ Euclidean space). The probability of connection between any pair of nodes i and j takes the form of a gravity law, generalizing Eq. (1.1), so that, again, more popular and similar nodes are more likely to be connected:

$$p_{ij} = \frac{1}{1 + \left(\frac{R\Delta\theta_{ij}}{(\mu\kappa_i\kappa_j)^{1/D}} \right)^{\beta}}. \tag{1.12}$$

Again, κ_i is the *hidden degree* of node i because it coincides with the expected degree of node i in the ensemble of graphs produced by the model. To model the degree heterogeneity observed in real networks, $\rho(k)$ is chosen to be power-law distributed: $\rho(k) \sim \kappa^{-\gamma}$, with $\gamma > 2$. v_i and v_j are vectors indicating the position of nodes i and j in the D-sphere, with angular distance $\Delta\theta_{ij}$. And the radius of the sphere, set such that the density of nodes is 1, having N nodes, is

$$R = \left[\frac{N}{(2\pi)^{\frac{D+1}{2}}} \Gamma\left(\frac{D+1}{2} \right) \right]^{\frac{1}{D}}, \tag{1.13}$$

where Γ is the gamma function. The parameter β calibrates the coupling of the network with the underlying metric space and controls the level of clustering in the topology, and μ controls the average degree of the network.

As with in the $\mathbb{S}^1$ model, this has an isomorphic geometric formulation, where the popularity and similarity dimensions are combined into a single distance in a $D+1$ hyperbolic space: the $\mathbb{H}^{D+1}$ model.

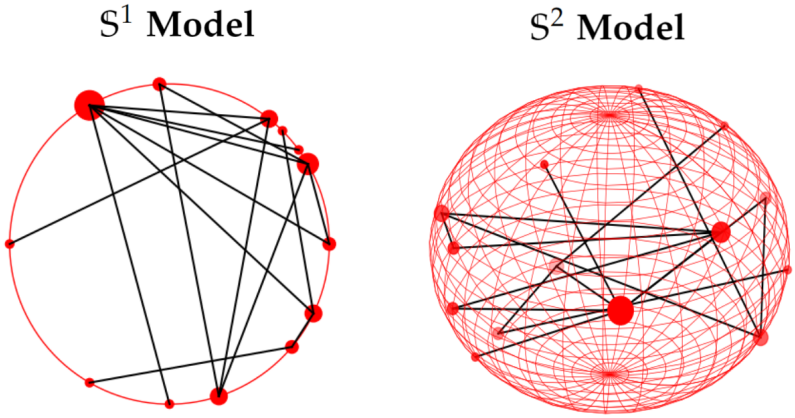

Figure 1.2: Representation of the $\mathbb{S}^D$ model for $D = 1$ and $D = 2$. The size of each node is proportional to its hidden degree κ_i.

1.2 Persistent homology in graph learning

1.2.1 Basic definitions

In this work we consider undirected graphs. An *undirected graph* is a pair $G = (V, E)$ of finite sets of n *vertices* and m *edges*, with $E \subseteq \{\{u, v\} \mid u, v \in V,\ u \neq v\}$. We will take for granted that (u, v) and (v, u) refer to the same edge. Two graphs $G = (V, E)$ and $G_0 = (V_0, E_0)$ are *isomorphic*, $G \cong G_0$, if there is a bijective function $\phi\colon V \to V'$ that preserves adjacency, this is, $(u, v) \in E$ if and only if $(\phi(u), \phi(v)) \in E'$. The isomorphism ϕ is thus preserving edges and connectivity. Since it is bijective, it has an inverse function ϕ^{-1}. We might also need the definition of a *multiset*, denoted by $\{\{\}\}$; this is a set whose elements are included with multiplicities.

Let S_n be the permutation group on n letters, and consider two graphs G and G' with n nodes. An element $\sigma \in S_n$ acts on a graph by permuting the order of vertices, and, by transitivity, the edges. If $G \cong G'$, there is a permutation $\sigma \in S_n$

such that $\sigma(G) = G'$. Assuming that all graphs have n vertices, we call a function $f\colon V \to \mathbb{R}^n$ *permutation-equivariant* if $f(\sigma(G)) = \sigma(f(G))$ for a permutation σ [7].

1.2.2 Topological features of graphs

The simplest topological features we can use to analyze graphs are *connected components* and *cycles*. Assuming that we are dealing with *planar graphs*, i.e., graphs that afford an embedding in the plane such that there are no overlaps between edges, *Euler's formula for planar graphs* holds:

$$|V| - |E| + |F| = 2, \tag{1.14}$$

where F are the faces of the graph and $|\cdot|$ denotes the cardinality of a set.

Another interesting observation is that we only need to know the number of connected components to calculate the number of cycles in a graph. If β_0 is the number of connected components and β_1 the number of cycles, we have

$$\beta_1 = m + \beta_0 - n, \tag{1.15}$$

where the number of vertices and edges are denoted, respectively, by n and m. The values β_0 and β_1 are also known as the first two *Betti numbers* of a graph [7].

1.2.3 Simplicial homology

The Betti numbers of a graph are part of a more generic concept, namely *simplicial homology*. Under this context, the Betti numbers are, in fact, the ranks of the zeroth and first homology group, respectively.

A *simplicial complex* K is a collection of sets that is closed under the subset operation. Thus, for any $\sigma \in K$ and $\tau \subseteq \sigma$, we have $\tau \in K$. An element $\sigma \in K$ with $|\sigma| = k + 1$ is also referred to as a *k-simplex*. We say that $\dim \sigma = k$. Moreover, if k is maximal among all the simplices of K, we call K a *k-dimensional simplicial complex*. We observe that a k-simplex has indeed $k + 1$ elements. This convention makes sense when we relate it to the concept of dimension. A 0-simplex, that is, a point or a vertex, has dimension 0. A typical example of a simplicial complex is a graph $G = (V, E)$. Putting $K := V \cup E$, we obtain a 1-dimensional simplicial complex.

The *Vietoris-Rips complex* $R_\epsilon(X)$ is the abstract simplicial complex with vertex set X and a k-face for each collection $x_{i_0}, \ldots, x_{i_k}$ such that $||x_{i_r} - x_{i_s}|| \leq \epsilon$ for all $r, s \in \{0, \ldots, k\}$. This is, $R_\epsilon(X)$ has a k-face $\{x_{i_0}, \ldots, x_{i_k}\}$ if and only if $\operatorname{diam}\{x_{i_0}, \ldots, x_{i_k}\} \leq \epsilon$, where $\operatorname{diam}(A) = \sup\{||a - b|| \mid a, b \in A\}$. Thus, there is an edge $\{x_0, x_1\}$ in $R_\epsilon(X)$ if and only if $||x_0 - x_1|| \leq \epsilon$. We observe that $R_\epsilon(X)$ is

determined by the *distance matrix* ($||x_i - x_j||$) for $x_i, x_j \in X$. The family $\{R_\epsilon(X)\}_{\epsilon \geq 0}$ is called a *filtered simplicial complex*, and ϵ is the *filtering parameter* [12].

The *Euler's characteristic* of a finite abstract simplicial complex K is defined as

$$\chi(K) = \sum_{n=0}^{\infty} (-1)^n c_n \tag{1.16}$$

where c_n denotes the number of n-faces of K. We note that, for surfaces (or for graphs), $\chi = V - E + F$. And this quantity is a topological invariant. Given a simplicial complex K, the vector space generated over $\mathbb{Z}_2$ coefficients whose elements are the k-simplices of K is called the *kth chain group*, denoted by $C_k(K)$. The elements of a chain group are also referred to as *simplicial chains*. Thus, sums of simplices of a compatible dimension form the elements of the chain group [7].

Given $\sigma = (v_0, \ldots, v_k) \in K$, the *kth boundary homomorphism* $\partial_k \colon C_k(K) \to C_{k-1}(K)$ is

$$\partial_k(\sigma) := \sum_{i=0}^{k} (v_0, \ldots, v_{i-1}, v_{i+1}, \ldots, v_k), \tag{1.17}$$

a sum of simplices with the ith vertex of the simplex missing.

Since ∂_k is a homomorphism, it extends to arbitrary simplicial chains, for which it is sufficient to define it on individual simplices [7]. The *kth homology group* of a simplicial complex K is the quotient group obtained from the two subgroups $\operatorname{Ker} \partial_k$ and $\operatorname{Im} \partial_{k+1}$ of $C_k(K)$:

$$H_k(K) := \operatorname{Ker} \partial_k \,/\, \operatorname{Im} \partial_{k+1}. \tag{1.18}$$

The kth homology group of a simplicial complex contains its k-dimensional topological features in the form of equivalence classes of simplicial chains, also known as *homology classes* [7].

The rank of the kth homology group $H_k(K)$ is known as the *kth Betti number*, denoted by β_k. Betti numbers can be very useful when simplicial complexes are compared. Noting that *Euler's characteristic* $\chi(K) = \sum_i (-1)^i |\{\sigma \mid \dim \sigma = i\}|$ can also be expressed as $\chi(K) = \sum_i (-1)^i \beta_i$, which is the sum of alternating Betti numbers, we can reproduce Eq. (1.15).

Homology groups are preserved under graph isomorphisms [7].

Lemma 1.1. Let G, G' be two isomorphic graphs, and $\phi \colon G \to G'$ an isomorphism. Then the homology groups of G and G' are isomorphic, that is, $H_p(G) \cong H_p(G')$ for all p.

From this fact we see that the Betti numbers of G and G' are the same. A similar property holds for isomorphic simplicial complexes [7].

1.2.4 Persistent homology

Let us consider an arbitrary graph $G = (V, E)$. We notice that the addition of an edge to G changes its Betti numbers, either by merging two connected components (decreasing β_0) or by creating an additional cycle (increasing β_1). In fact, we already saw this fact from Eq. (1.15); there it is clear that the insertion of a new edge $e = (u, v)$ with $u, v \in V$ either increases β_1 by 1 (m varies), or instead β_0 varies. We note that β_0 decreases at most by 1, given that a single edge may only merge two connected components into one. Because of that, Betti numbers are not so useful in large-scale graph analysis and we may want to formulate everything in terms of simplicial complexes, to get a more general description [7].

Given a simplicial complex K, we call a sequence of simplicial complexes a *filtration* if it affords a nesting property of the form

$$\emptyset = K_0 \subseteq K_1 \subseteq \cdots \subseteq K_{m-1} \subseteq K_m = K. \tag{1.19}$$

Since each element of this sequence is a simplicial complex, we can also think of this construction as building K by adding simplices one after the other. In the context of graphs, we can imagine filtrations as sequencing a graph based on some type of data, or function, assigned to its vertices. For example, we can build a filtration of a graph based on the degree of its vertices, by assigning to each edge the maximum of the weight (or degree) of its vertices. This construction is sometimes known as a *lower-star filtration* [7].

The boundary operators $\partial(\cdot)$, combined with the inclusion homomorphism between consecutive simplicial complexes, induce homomorphisms between corresponding homology groups of any filtration of simplicial complexes. Given $i \leq j$, we denote the homomorphism by $l^{i,j} \colon H_k(K_i) \to H_k(K_j)$. For every dimension k, this construction provides a sequence of homology groups:

$$0 = H_k(K_0) \xrightarrow{l_k^{0,1}} H_k(K_1) \xrightarrow{l_k^{1,2}} \cdots \xrightarrow{l_k^{m-2,m-1}} H_k(K_{m-1}) \xrightarrow{l_k^{m-1,m}} H_k(K_m) = H_k(K). \tag{1.20}$$

With this notation, the *kth persistent homology group* is

$$H_d^{i,j} := \operatorname{Ker} \partial_k(K_i) \,/\, (\operatorname{Im} \partial_{k+1}(K_j) \cap \operatorname{Ker} \partial_k(K_i)). \tag{1.21}$$

From the definition of the ordinary Betti numbers, we define the *kth persistent Betti number* as the rank of this group:

$$\beta_k^{i,j} := \operatorname{rank} H_k^{i,j}. \tag{1.22}$$

Let our filtration be associated to a set of values $a_0 \leq a_1 \leq \cdots \leq a_{m-1} \leq a_m$ (for example, the function values on the vertices). We are now able to compute some topological descriptors, such as *persistence diagrams* [7].

Let $\mathbb{F}$ be a field. Let $H_n(X;\mathbb{F})$ be the n-th dimensional homology $\mathbb{F}$-vector space of the topological space X, with coefficients in $\mathbb{F}$. Given a filtration of a finite abstract simplicial complex K, for all $i \leq j$ and $n \geq 0$, the inclusion $K_i \hookrightarrow K_j$ induces an $\mathbb{F}$-linear map

$$\phi_n^{i,j} \colon H_n(K_i;\mathbb{F}) \to H_n(K_j;\mathbb{F}). \tag{1.23}$$

A homology class $\alpha \in H_n(K_j;\mathbb{F})$ is said to be *born* at K_j if it does not belong to the image of $\phi_n^{i,j}$ for $i < j$. It is said to *die* at K_j for $j > i$ if $\phi_n^{i,j}(\alpha) = 0$ but $\phi_n^{i,j-1}(\alpha) \neq 0$. If α is born at K_i and dies at K_j, then the *life* or *persistence* of α is defined as $(j-i)$. The image of $\phi_n^{i,j}$ is an $\mathbb{F}$-subspace of $H_n(K_j;\mathbb{F})$, which is called *persistent homology*, denoted by $H_n^{i,j}(K;\mathbb{F})$.

Given a filtration $K_0 \subseteq K_1 \subseteq \cdots \subseteq K_{m-1} \subseteq K_m = K$ of an ordered abstract simplicial complex K, the associated *persistence barcode* has a horizontal segment from i to j for each homology generator born at i and dying at j. Homology classes are shown by increasing order of dimension. Longer segments are drawn under shorted ones, and segments starting later are above those starting earlier.

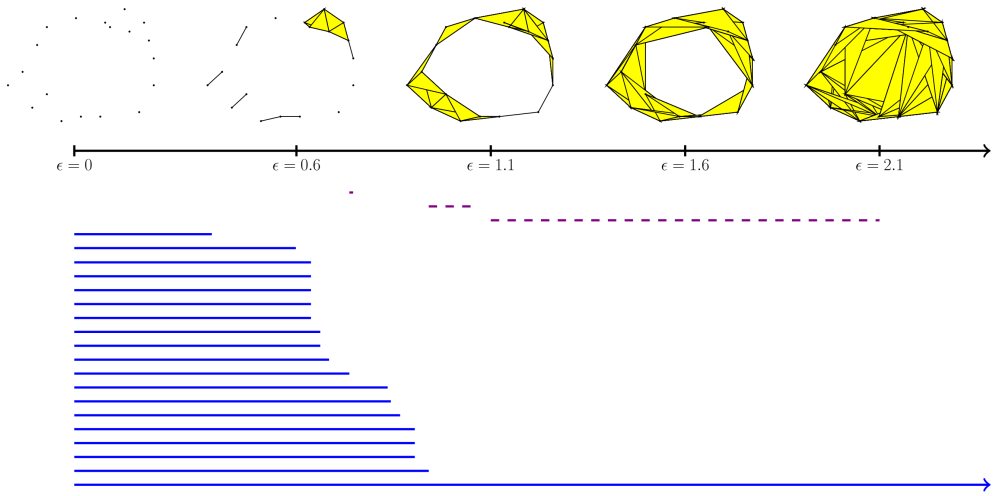

Figure 1.3: Example of a filtration with its associated persistence barcode. Solid lines represent the lifetime of connected components, and dashed lines represent the lifetime of holes [15].

The *persistence diagram* associated with a barcode has a point (b,d) in a coordinate plane for each finite line $[b,d)$ in the barcode. Thus a point (b,d) in a persistence diagram represents a homology class with birth parameter b and death parameter d. The rays $[b,\infty)$ in the barcode are represented as points (b,d_∞) in

the persistence diagram, where d_∞ is a chosen value above all finite death values in the barcode, representing infinity.

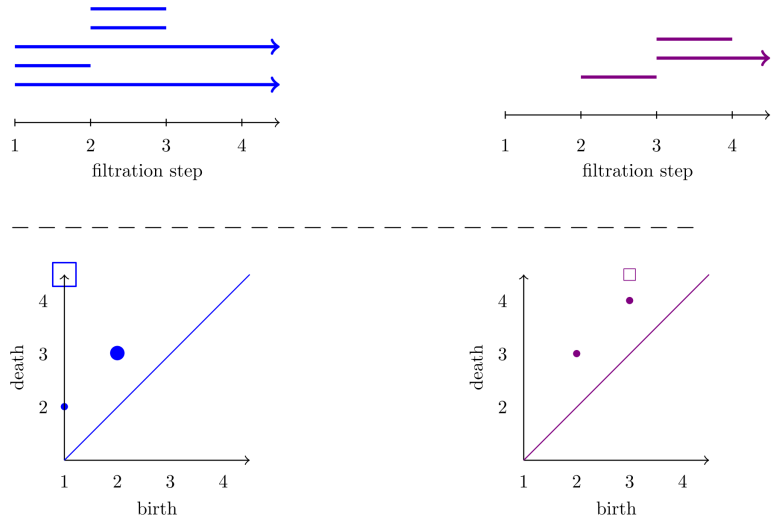

Figure 1.4: Example of a persistence barcode with its associated persistence diagram [15].

Thus, the topological activity of a filtration is contained in persistence diagrams. Persistence indicates if a feature created in a simplicial complex during the filtration is relevant or not. Typically, a feature is considered to be relevant when it shows a high persistence, while features with a low persistence might indicate low reliability or be considered as noise [7].

1.2.5 Persistence descriptors and comparison of diagrams

Bottleneck distance

The most common metric to compare two persistence diagrams D and D' is the *bottleneck distance*[3].

A *matching* between D and D' is a bijective function $\psi\colon D \to D'$ such that, for every $(x,x) \in \Delta$, either $\psi(x,x) = (x,x)$ or $\psi(x,x) = (b,d)$ with $b \neq d$. For each matching $\psi\colon D \to D'$, define

$$||\psi|| = \max\{d_\infty((x,y),\psi(x,y)) \mid (x,y) \in D\}$$

[3]Note that D is being used in this work for *persistence diagram* and for *dimension*, in order to preserve the notation from the bibliography. The two concepts are very different and, by context, they can always be distinguished.

where d_∞ is the l_∞ distance on $\mathbb{R}^2$,

$$d_\infty((x,y),\psi(x,y)) = \max\{|x-x'|,|y-y'|\}.$$

Definition 1.2. The *bottleneck distance* between two persistence diagrams D and D' is defined as

$$W_\infty(D,D') = \min\{||\psi|| \mid \psi\colon D \to D' \text{ matching}\}. \tag{1.24}$$

Therefore, $W_\infty(D,D')$ is the smallest $\epsilon \geq 0$ for which there exists a matching $\psi\colon D \to D'$ for which $d_\infty((x,y),\psi(x,y)) \leq \epsilon$ for all $(x,y) \in D$.

In general, the *Wasserstein distances* are defined for $p,q \geq 1$ as

$$W_p[q](D,D') = \min_{\psi\colon D\to D'} \left[\sum_{(x,y)\in D} d_q((x,y),\psi(x,y))^p\right]^{1/p} \tag{1.25}$$

where $d_q((x,y),(x',y')) = (|x-x'|^q + |y-y'|^q)^{1/q}$ [12].

Example 1.3. Consider the persistence diagrams D and D' in Figure 1.5. There we find that $||\phi|| = \max\{1.0, 0.2\} = 1.0$ and $||\psi|| = \max\{0.5, 0.5, 0.2\} = 0.5$. Therefore, $W_\infty(D,D') = 0.5$.

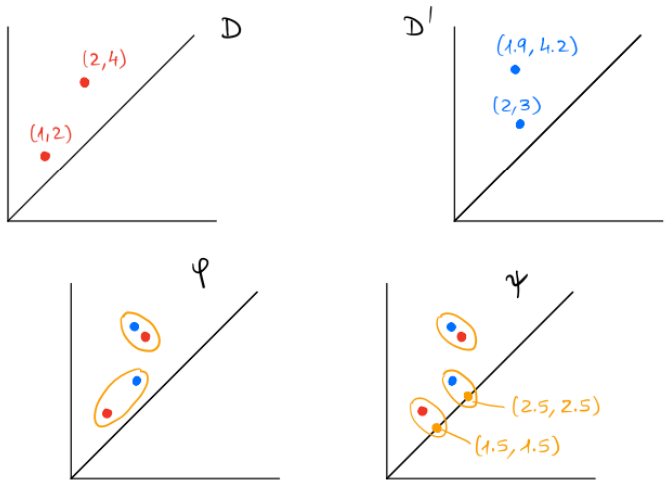

Figure 1.5: Example of two matchings between D and D'. [12]

Total persistence

Definition 1.4. Let D be a persistence diagram. For $x = (b, d) \in D$, the lifetime $l = d - b$ is called *persistence* of x. If $D = \{x_j\}_{j \in J}$, the *total persistence* of D is

$$L = \sum_{j \in J} (d_j - b_j). \tag{1.26}$$

Entropy

Definition 1.5. The *entropy* of a random variable is the average level of uncertainty inherent in its outcomes. For a persistence diagram, it is computed as follows:

$$E = -\sum_{i \in I} \frac{d_i - b_i}{L} \log_2 \left(\frac{d_i - b_i}{L} \right); \; L = \sum_{i \in I} (d_i - b_i). \tag{1.27}$$

Landscape

For real numbers $b < d$, consider the *tent* function

$$\Delta_{(b,d)}(t) = \max\{0, \; \min(t - b, d - t)\}. \tag{1.28}$$

Given a set of points $\{(b_i, d_i)\}_{i=1}^n$ defining a persistence diagram, with relative multiplicities m_i, its associated *landscape* over a field $\mathbb{F}$ is a collection of piecewise linear continuous functions $\lambda_k \colon \mathbb{R} \to \mathbb{R}$ defined by

$$\lambda_k(t) = \text{kmax}\{\lambda_{(b_i,d_i)}(t)\}_{i=1}^n \tag{1.29}$$

where kmax returns the k-th largest value in a given set of real numbers, counted with their multiplicities, or 0 if there is no k-th largest value [12].

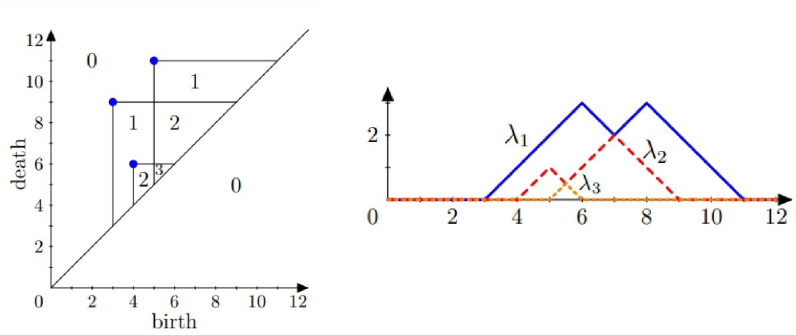

Figure 1.6: Example of a persistence landscape [13].

Persistence image

For a given persistence diagram, consider a function

$$\Phi(s,t) = \sum_{i=1}^{n} w_i G_i(s,t) \tag{1.30}$$

for (s,t) in a square, where each w_i is a weight and G_i is a 2-dimensional Gaussian function centered at (b_i, d_i). This yields a smoothing of the persistence diagram called a *persistence surface*. A *persistence image* is a discretization of Φ on a grid overlay [12].

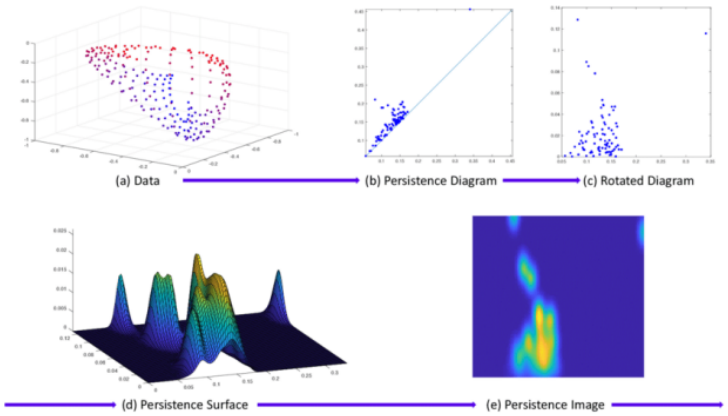

Figure 1.7: Process for the obtention of a persistence image [12].

1.2.6 Kernels in Hilbert space

Let X be any set. A *kernel* is a function $K\colon X \times X \to \mathbb{R}$ which is **symmetric**, that is, $K(x,y) = K(y,x)$ for all $x, y \in X$, and **positive definite**, $\sum_{i,j=1}^{n} c_i c_j K(x_i, x_j) \geq 0$; for all n and $c_1, \ldots, c_n \in \mathbb{R}$ and $x_1, \ldots, x_n \in X$, and, moreover, equality holds if and only if $c_i = 0$ for all i [12].

A function $K\colon X \times X \to \mathbb{R}$ is a kernel if and only if there exists a Hilbert space H and a map $\phi\colon X \to H$ such that $K(x,y) = \langle \phi(x), \phi(y) \rangle$ for all x, y. The Hilbert space H is called *feature space* and the map ϕ is called *feature map*.

Example 1.6. The following are kernels in Euclidean space $\mathbb{R}^d$ [12]:

- **Linear**: $K(x,y) = x^T y$. In this case, $\sum_{i,j=1}^{n} c_i c_j K(x_i, x_j) = K(\sum_i c_i x_i, \sum_i c_i x_i)$ by bilinearity.
- **Polynomial**: $K(x,y) = (1 + x^T y)^n$ with $n \geq 1$.

- **Gaussian**: $K(x,y) = e^{-\frac{||x-y||^2}{2\sigma^2}}$ with $\sigma > 0$.
- **Laplacian**: $K(x,y) = e^{-\alpha||x-y||}$ with $\alpha > 0$.

Example 1.7. The **heat kernel** $K_t(x,y) = \frac{1}{(4\pi t)^{d/2}} e^{-||x-y||^2/4t}$ solves the *heat equation* $\frac{\partial K_t}{\partial t}(x,y) = \Delta_x K_t(x,y)$ for $t > 0$ and $x, y \in \mathbb{R}^d$, with initial condition, where δ_x is the Dirac delta distribution centered at x, $\lim_{t\to 0} K_t(x,y) = \delta_x(y)$.

Every kernel $K : X \times X \to \mathbb{R}$ induces a *pseudometric* on X corresponding to the norm distance on the feature space:

$$d_k(x,y) = \sqrt{K(x,x) - 2K(x,y) + K(y,y)} = ||\phi(x) - \phi(y)||. \tag{1.31}$$

Note that here it is possible that $d_k(x,y) = 0$ with $x \neq y$ since the feature map ϕ need not be injective [12].

Scale-space kernel

Consider $K\colon \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ where $\mathcal{D}$ is a set of persistence diagrams. It is defined via a feature map $\phi\colon \mathcal{D} \to \mathrm{L}^2(\Omega)$, where $\Omega = \{(x,y) \in \mathbb{R}^2 \mid y \geq x\}$ is the half plane above the diagonal.

To each persistence diagram $D \in \mathcal{D}$ one assigns a sum $\sum_{p\in D} \delta_p$ of Dirac delta distributions. Here δ_p is viewed as a functional that evaluates each smooth function at $p = (b,d)$. However, the induced metric on D does not take into account the distance to the diagonal and hence it is not robust against noise [14].

Instead, the sum of Dirac deltas is taken as initial condition for a heat diffusion problem with a boundary condition on the diagonal. A solution $u\colon \Omega \times \mathbb{R}_+ \to \mathbb{R}$ of the Dirichlet problem is found:

$$\Delta_x u = \partial u \text{ in } \Omega \times \mathbb{R}_+,\ u = 0 \text{ on } \partial\Omega \times \mathbb{R}_+,\ u = \sum_{p\in D} \delta_p \text{ on } \Omega \times \{0\}.$$

For each $D \in \mathcal{D}$ and each scale parameter $\sigma > 0$, $\phi_\sigma(D) = u|_{t=\sigma}$ is defined. Thus, $K_\sigma(D_1, D_2) = \langle \phi_\sigma(D_1), \phi_\sigma(D_2) \rangle$. In this case, the feature map ϕ_σ is injective, so K_σ yields a metric. Explicitly, one obtains that

$$u(x,t) = \frac{1}{4\pi t} \sum_{p\in D} e^{-||x-p||^2/4t} - e^{-||x-\overline{p}||^2/4t},$$

where $\overline{p} = (d,b)$ if $p = (b,d)$ [14]. Thus, the *scale-space* or *Reininghaus kernel* is given by

$$K_\sigma(D_1, D_2) = \frac{1}{8\pi\sigma} \sum_{p\in D_1, q\in D_2} e^{-||p-q||^2/8\sigma} - e^{-||p-\overline{q}||^2/8\sigma}. \tag{1.32}$$

1.2.7 Extended persistence

Consider a graph $G = (V, E)$, with vertices V and (non-oriented) edges E. If $f\colon V \to \mathbb{R}$ is a function defined on its vertices (in our case, assigning to each vertex its degree), the *sublevel graphs* $G_\alpha = (V_\alpha, E_\alpha)$, where $\alpha \in \mathbb{R}$, are defined as $V_\alpha = \{v \in V\colon f(v) \leq \alpha\}$ and $E_\alpha = \{(v_1, v_2) \in E\colon v_1, v_2 \in V_\alpha\}$. In the sequence $(V_\alpha)_\alpha$, the loops persist forever since they never disappear from the sequence of sublevel graphs, and the same argument states for whole connected components of G. Moreover, branches pointing upwards (with respect to the orientation given by f) are missed, since they do not create connected components when they appear in the sublevel graphs [8].

Extended persistence refines the analysis by looking also at *superlevel sets* $V^\alpha = \{v \in V\colon f(v) \geq \alpha\}$, letting α decrease from $+\infty$ to $-\infty$. We restrict to the case in which $X = G$ is a graph. Now, death times can be defined for loops and whole connected components by picking the superlevel graphs in which the feature appears again, and using the corresponding α_d value as the death time for these features (analogously, the birth time is denoted α_b). This way, branches pointing upwards can be detected in the sequence of superlevel graphs, in an analogous way as downwards branches were detected in the sublevel graphs [8]. Figure 1.8 illustrates an example of sublevel and superlevel graphs.

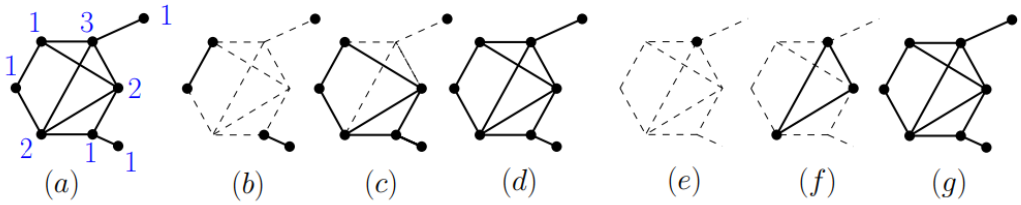

Figure 1.8: Example of sublevel and superlevel graphs. (a) Input graph with the values of a function $f\colon V \to \mathbb{R}$. (b, c, d) Sublevel graphs for $\alpha = 1, 2, 3$, respectively. (e, f, g) Superlevel graphs for $\alpha = 1, 2, 3$, respectively. [8]

Definition 1.8. The *extended persistence diagram of* f, denoted by $Dg(G, f) \subset \mathbb{R}^2$, is the family of intervals of the form $[\alpha_b, \alpha_d]$ which is turned into a multiset of points in the Euclidean plane $\mathbb{R}^2$ by using the interval endpoints as coordinates [8].

In comparison with ordinary persistence, extended persistence ensures that points have finite coordinates. This avoids losing information or having to give a special treatment to points with infinity coordinates. In practice, computing extended persistence diagrams can be done with the *Python* **Gudhi** library, and in Chapter 2 this is actually programmed and implemented. Those, however, can

be computed only after having defined a real-valued function on the nodes of the graphs. In the next section, a family of functions satisfying this from the Heat Kernel Signatures (HKS) for graphs is defined [8]. In this work, a *vertex degree filtration*, this is, $v \mapsto \deg(v)$, is proposed.

Definition 1.9. In the extended persistence context, using a degree-based filtration, we say that a cycle is *born* when the vertex with lower degree forming the cycle v_l is visited by the filtration function, that is, $v_l \in V_\alpha$. Analogously, we say that a cycle *dies* when the vertex with greatest degree forming the cycle v_g is visited by the filtration function, i.e., $v_g \in V_\alpha$.

In the problem that is treated here, extended persistence is especially useful for H1. The reason, as stated before, is that cycles live forever once they are detected, in ordinary persistence. Extended persistence, instead, will detect when the cycle is born and when it dies. Example 1.10 illustrates this phenomenon.

Example 1.10. Consider the graph in Figure 1.9. A visual examination evidences that it has two cycles.

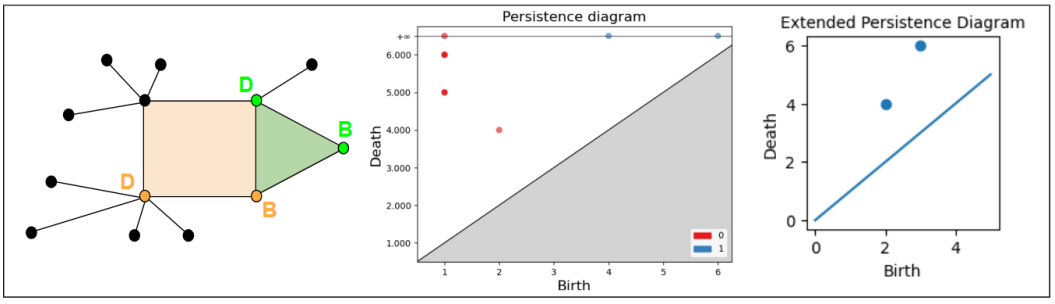

Figure 1.9: Persistence diagram and extended persistence diagram of an example graph. Birth (B) and death (D) vertices are painted in orange or green, for each cycle. In the ordinary persistence diagram, red points correspond to H0, and blue points to H1. Axes are in degree units.

The ordinary persistence diagram contains two blue points: $(4, \infty)$ and $(6, \infty)$. The x-coordinate corresponds to the higher vertex degree of each cycle, meaning that at this point the cycle is created. The y-coordinate, instead, is ∞, because the cycle will live forever for the next steps of the filtration. In this case, point $(4, \infty)$ corresponds to the triangle (in green) because its greatest vertex degree is 4. The same applies for the square (in orange), whose greatest vertex degree is 6 and corresponds to the point $(6, \infty)$. In this example, there are only two points at infinity, and cycles can be distinguished. When a complex network is analysed by this method, points in H1 appear at infinity. Extended persistence provides a

resolution of the points at infinity, by giving them finite coordinates corresponding to birth and death. The extended persistence diagram in Figure 1.9 has a point at $(2,4)$ and a point at $(3,6)$. The first one corresponds to the triangle, and the second one to the square. The y-coordinate now indicates the greatest vertex degree, but the birth information is added here. In the case of the triangle, the lowest degree is 2, while in the case of the square it is 3.

At this point the reader might ask why extended persistence is not used here for H0. The reason is that in a tree-like structure (as the ones studied in this work) there are no branches that are born from the top and then united to a larger component.

1.2.8 Heat Kernel Signatures

Heat Kernel Signatures (HKS) yield an example of family of functions derived from the spectral decomposition of Laplacian graphs, providing relevant features for graph analysis.

The *adjacency matrix* A of a graph G with vertices $V = \{v_1, \dots, v_n\}$ is $A := (1_{(v_i,v_j)\in E})_{ij}$. The *degree matrix* D is the diagonal matrix $D := \sum_j A_{ij}$, and the *normalized graph Laplacian* $L_w = L_w(G)$ is the linear operator acting on the space of functions defined on the vertices of G, which is represented by the matrix $L_w = I - D^{-1/2}AD^{-1/2}$.

The elements of L_w are given by

$$(L_w)_{ij} := \begin{cases} 1 & \text{if } i = j \text{ and } \deg(v_i) \neq 0, \\ -\dfrac{1}{\sqrt{\deg(v_i)\deg(v_j)}} & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j, \\ 0 & \text{otherwise.} \end{cases}$$

The normalized graph Laplacian admits an orthonormal basis of eigenfunctions $\Phi = \phi_1, \dots, \phi_n$ and its eigenvalues satisfy $0 \leq \lambda_1 \leq \cdots \leq \lambda_n \leq 2$. Since the eigenbasis Φ is not uniquely defined, ϕ_i are not useful to compare graphs. We use instead *Heat Kernel Signatures* [8].

Definition 1.11. Given a graph G and $t \geq 0$, the *Heat Kernel Signature* with diffusion parameter t is the function $hks_{G,t}$ defined on the vertices of G by

$$hks_{G,t} \colon v \mapsto \sum_{k=1}^{n} e^{-t\lambda_k} \phi_k(v)^2.$$

Chapter 2

Development

2.1 *Detecting the ultra low dimensionality of real networks*

In Section 0.1, we discussed the problem of dimension. This problem is successfully addressed in [1] from a physical and geometrical point of view, where the density of triangles, squares and pentagons play a fundamental role. The method and results obtained in [1] are briefly commented in this section.

The main goal of this chapter, though, is to present a new methodology used to address the problem of dimensionality by means of Topological Data Analysis (TDA), focusing on study of two real-world networks, and to compare our results with those from [1].

2.1.1 Methods

In [1] it is described how cycles of different lengths and dimensions in $\mathbb{S}^D$ networks are intertwined in a non-trivial way, which helps determine the dimensionality of the similarity space. Power-law distributions are used for the hidden degrees and the number of triangles incident on edges properly normalized is measured; that is, the *mean density of edge triangles*, C_t. Normalization is performed by dividing the number of triangles going through an edge by the maximum possible number given the degrees at the ends of the edge and then averaged over links that connect nodes in the network with a degree greater than one. The results show that the maximum values of clustering coefficients and densities of cycles decrease as the dimensionality of the similarity space increases. The dependency on dimensionality is also evident for the mean density of edge squares and pentagons, which display a maximum for a value of the dimension that increases with network heterogeneity [1].

The *phase space*, defined by (C_t, C_s, C_p), is then analyzed. When plotting the

results for different dimensions, fixing γ and D, and varying β, on the planes (C_x, C_y), $x, y \in \{t, s, p\}$ (where t, s, p denote triangles, squares and pentagons, respectively), the curves are found to be different, meaning that each dimension presents a characteristic profile.

Inferring hidden dimensions

According to this, a method is performed in [1] to determine the dimension of real-world networks. First, an ensemble of synthetic surrogates is created using the $\mathbb{S}^D$ model with different values of the inverse temperature β and dimension D. An iterative process is used to match the expected and observed degrees, ensuring that the model reproduces the degree distribution of the network. Next, the N nodes are assigned homogeneous random positions in the D-sphere. The ensemble of synthetic surrogates is restricted to feasible values of dimension and inverse temperature based on the maximum achievable density of edge triangles. At this point, the sample of random surrogates is generated using the connectivity law Eq. (1.12). Finally, a data-driven classifier (K-NN) is used to infer the dimension of the real network based on the densities of edge cycles (triangles, squares, and pentagons) in the synthetic surrogates that best approximate the real network. We note that the K-NN classifier identifies the K surrogates closest to the original network in the phase space (C_t, C_s, C_p) by minimizing the Euclidean distance. Other tests using decision trees and neural networks as classifiers are also tested in the article with similar results [1].

To quantify the goodness of their method, they computed *confusion matrices* for different types of topologies, using the inferred dimensionality. These are defined as the probability of predicting the dimensionality D' in a network generated with dimension D. An inference method is considered a good method when the confusion matrices are close to the identity matrix. In this case, the results happened to be very good predictions [1].

2.1.2 Discussion

The method presented in [1] infers the dimensionality of a graph by analyzing the densities of edge cycles. The results show that complex networks are well-represented in hyperbolic geometry with ultra low dimensionality, and the dimension of real-world networks (such as the Internet at the autonomous system level, the email network within the Enron company, and the human connectome) is obtained. In fact, it is found that many real-world networks, including biomolecular and social networks, have ultra-low dimensionality and can be well represented in hyperbolic geometry.

Even if a network is embedded in Euclidean space, the formalism of this method is also valid for geometric random graphs and non-small-world networks, which are effectively described by the geometry of the D-sphere. However, the model is not suited for describing D-dimensional lattices because their links are strongly correlated, unlike in this model where links are statistically independent. In [1] it is also suggested that multidimensional hyperbolic embeddings can provide a more accurate description of network complex systems and help reveal the correlation of factors that determine connectivity. Moreover, it seems clear that understanding network dimensionality is crucial for predicting network behavior.

2.2 Dimension detection using persistent homology

The main goal of this project is to search for topological invariants that can be computed in a complex network, and that provide meaningful information in order to study the dimensionality of the network. The final objective is to reproduce results found in [1] using an alternative method based on TDA.

2.2.1 Surrogate generation and description

From now on, we work with two different real-world networks, namely *C. Elegans-C*, representing the nervous systems network of the *Caenorhabditis elegans* worm, and *Human1*, representing a connectome of the human brain including one hemisphere [9].

Network	Type	$\|V\|$	$\|E\|$	av. deg.	C	D
CElegans-C	Biological - Brain	279	2287	16.39	0.34	1
Human1	Biological - Brain	493	7773	31.53	0.49	3

Table 2.1: Properties of real networks and their dimensionality, according to the method described in [1]. Here $|V|$ and $|E|$ denote the number of nodes and edges, respectively; C is the clustering coefficient, and D the dimension.

Table 2.1 shows the dimension obtained in [1] of each network, as well as other properties such as the number of nodes and edges, the average degree and the clustering coefficient.

In order to generate surrogates in different dimensions for the geometric model, we first need to infer the characteristic parameters from the given network. The hidden degrees (κ_i) and β (which in turn will fix the level of clustering of the network) are computed using *Mercator* [10], a tool to embed networks in their hidden

hyperbolic space. From equations (1.12) and (1.2) we see that the probability of connection between nodes is determined by the values of κ_i, β, and D (and a random angular position $\Delta\theta_{ij}$). Once κ_i and β are computed from the real network, 30 surrogates are generated for each dimension D using *Mercator*. Table 2.2 summarizes the inferred beta values for each network and for each dimension. In this surrogates, nodes are connected between each other according to Eq. (1.12).

Therefore, after this process, the data consists of 30 surrogates for each dimension D from 1 to 10, for the samples *CElegans-C* and *Human1*. At this point, the topological data analysis approach begins.

D	β *CElegans-C*	β *Human1*
1	1.4449	2.5165
2	3.1096	5.9064
3	4.5399	11.3629
4	5.8723	34.7880
5	7.1800	134.0782
6	9.1164	111.1748
7	9.1783	124.0354
8	10.4483	139.7730
9	11.9787	159.0632
10	12.3331	123.5125

Table 2.2: Inferred beta values for each dimension and both samples *CElegans-C* and *Human1*.

2.2.2 p_{ij} matrices as distance matrices

The typical approach when using TDA on data sets is to search for 0 and 1 dimensional homology by constructing a simplicial complex, based on distances between nodes. This, however, requires a notion of distance. As a first attempt to study these networks, p_{ij} values were considered. From Eq. (1.12) one can see that $1 - p_{ij}$ is proportional to a distance. These values, therefore, can be used to construct a distance matrix and proceed to compute the Vietoris-Rips complex.

Different patterns can be observed when persistence diagrams of the Vietoris-Rips filtration are represented for different surrogates in each dimension. However, further from the visual approach, it is difficult to determine any relationship between the diagrams and the dimension of the real network. This is because there is no notion of probability in the real network, and it does not make sense to compute $1 - p_{ij}$ on it. Therefore, there is no comparable item between the surrogates

and the original network. This approach, though, suggests that TDA might be able to enlighten some characteristics related with the dimension of the surrogates.

2.2.3 Euler characteristic

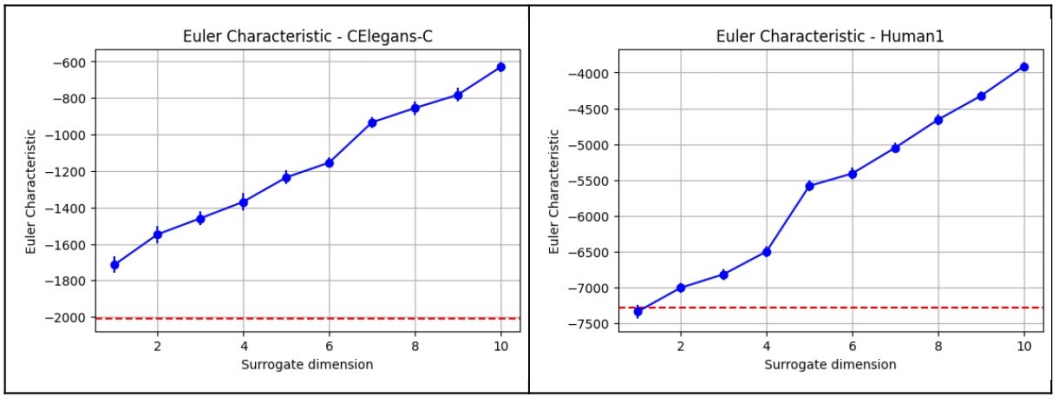

Figure 2.1: Average Euler characteristic of the surrogates for both *CElegans-C* and *Human1* samples, with the appropriate error bars, corresponding to the standard deviation interval. In red, the value of the original network is shown.

In Figure 2.1 the average Euler characteristic of the surrogates for both *CElegans-C* and *Human1* is represented. An increasing tendency is clearly noticed. In this section an explanation of this phenomenon is provided.

Notice that if p_{ij} decreases with increasing D, this implies that there is less probability of connection as D grows. Therefore, the number of edges decreases with D and $\chi = V - E + F$ is less negative (for a fixed number of V and F, as happens with the surrogates that vary very little among different D). In conclusion, if p_{ij} decreases with D, the Euler characteristic increases. We want to prove now that, for an increasing value of β, the value for p_{ij} decreases with D.

Proposition 2.1. Given Eq. (1.12), if β and D increase then p_{ij} decreases for all i, j.

Proof. From Eq. (1.12) and the expression of the radius in function of D we have:

$$p_{ij} = \frac{1}{1+\left(\frac{R\Delta\theta_{ij}}{(\mu\kappa_i\kappa_j)^{1/D}}\right)^{\beta}} = \frac{1}{1+\left(\frac{\left(\frac{N}{2\pi^{\frac{D+1}{2}}}\Gamma\left(\frac{D+1}{2}\right)\right)^{\frac{1}{D}}\Delta\theta_{ij}}{(\mu\kappa_i\kappa_j)^{1/D}}\right)^{\beta}} = \frac{1}{1+\Delta\theta_{ij}^{\beta}\left(\frac{N\,\Gamma\left(\frac{D+1}{2}\right)}{2\pi^{\frac{D+1}{2}}\mu\kappa_i\kappa_j}\right)^{\frac{\beta}{D}}}.$$

Let $C = \dfrac{N}{2\mu\kappa_i\kappa_j}$. Then,

$$p_{ij} = \frac{1}{1 + \Delta\theta_{ij}^{\beta}\left(C\dfrac{\Gamma(\frac{D+1}{2})}{\pi^{\frac{D+1}{2}}}\right)^{\frac{\beta}{D}}}.$$

We want to see now that $K(D) := \left(\dfrac{\Gamma(\frac{D+1}{2})}{\pi^{\frac{D+1}{2}}}\right)^{\frac{1}{D}}$ is an increasing function.

The behavior of $\Gamma(D)$ for an increasing positive real variable is given by *Stirling's formula* $\Gamma(D+1) \sim \sqrt{2\pi D}\left(\frac{D}{e}\right)^{D}$ where $\sim$ means asymptotic convergence. Therefore, $\Gamma(\frac{D+1}{2}) \sim \sqrt{\pi(D-1)}\left(\frac{D-1}{2e}\right)^{\frac{D-1}{2}}$.

Using this approximation we get

$$K(D) = \left(\frac{\Gamma(\frac{D+1}{2})}{\pi^{\frac{D+1}{2}}}\right)^{\frac{1}{D}} \sim \left(\frac{\sqrt{\pi(D-1)}\left(\frac{D-1}{2e}\right)^{\frac{D-1}{2}}}{\pi^{\frac{D+1}{2}}}\right)^{\frac{1}{D}} := f(D).$$

Therefore,

$$f(D) = \left(\sqrt{\frac{\pi(D-1)}{\pi^{D+1}}\left(\frac{D-1}{2e}\right)^{D-1}}\right)^{\frac{1}{D}} = \left(\sqrt{\frac{2e(D-1)^{D}}{\pi^{D}(2e)^{D}}}\right)^{\frac{1}{D}} = \sqrt{\frac{D-1}{2\pi e}}(\sqrt{2e})^{\frac{1}{D}}.$$

We now want to prove that $f(D)$ is an increasing function. We do so by computing its derivative.

$$f'(D) = \frac{1}{2}\left(\frac{D-1}{2\pi e}\right)^{-\frac{1}{2}}\frac{1}{2\pi e}(\sqrt{2e})^{\frac{1}{D}} - \sqrt{\frac{D-1}{2\pi e}}\frac{(\sqrt{2e})^{\frac{1}{D}}}{D^2}\ln(\sqrt{2e}).$$

Note that $f'(D) > 0$ if and only if $\frac{1}{2}\left(\frac{D-1}{2\pi e}\right)^{-\frac{1}{2}}\frac{1}{2\pi e}(\sqrt{2e})^{\frac{1}{D}} > \sqrt{\frac{D-1}{2\pi e}}\frac{(\sqrt{2e})^{\frac{1}{D}}}{D^2}\ln(\sqrt{2e})$. That is, $\sqrt{\frac{2\pi e}{D-1}}\frac{1}{4\pi e} > \sqrt{\frac{D-1}{2\pi e}}\frac{1}{D^2}\ln(\sqrt{2e})$ if and only if $\frac{1}{4\pi e} > \frac{D-1}{2\pi e}\frac{1}{D^2}\ln(\sqrt{2e})$, if and only if $\frac{1}{2\ln(\sqrt{2e})} > \frac{D-1}{D^2} := g(D)$.

We observe that $g(D) = \frac{D-1}{D^2}$ has derivative $g'(D) = \frac{2-D}{D^3}$, which is 0 for $D = 2$. Also, $g''(D) = \frac{2D-6}{D^4}$ and $g''(2) = -\frac{1}{8} < 0$. Therefore, $g(2) = \frac{1}{4}$ is a maximum. Hence, $g(2) = \frac{1}{4} \geq g(D)$ for all D.

Since $\frac{1}{2\ln(\sqrt{2e})} > \frac{1}{4} \geq \frac{D-1}{D^2}$, we see that $K := \left(\dfrac{\Gamma(\frac{D+1}{2})}{\pi^{\frac{D+1}{2}}}\right)^{\frac{1}{D}}$ is an increasing function. Now, back to our expression we have $p_{ij} = \dfrac{1}{1 + \Delta\theta_{ij}^{\beta}C^{\frac{\beta}{D}}K(D)^{\beta}}$. For β increasing, it is clear now that p_{ij} decreases when D is increased. □

At this point some remarks must be made. In this proof we have assumed that κ_i and κ_j are constant among the different surrogates. This is not exactly true, but in the thermodynamic limit in which the background theory is based, the assumption holds. In fact, the κ's are inferred with *Mercator* and fed into the generator of surrogates but, in theory, they intend to imitate the expected values of the original network.

Another assumption done here is that β is increasing. Table 2.2 shows that, in fact, it is not always the case (it holds true for *CElegans-C* but not for *Human1*, for example). What is the same for all dimensions is the clustering level controlled by β. However, β has an almost increasing tendency. This does not contradict the thesis that Euler's characteristic is an increasing function of D (when β is chosen increasing).

Among the consequences of Proposition 2.1, it is worth noting that Euler's characteristic χ cannot be considered as an appropriate descriptor to estimate the dimension of a network. In fact, χ performs a totally deterministic behaviour and reflects the growth of the number of edges.

2.2.4 Shortest paths

A second attempt to introduce a metric in this problem was to consider the *shortest path* between two nodes, which is the path that has the minimum number of edges or the minimum total weight among all possible paths connecting the nodes. In an unweighted graph, the shortest path length is simply the count of edges in the path. The algorithm used to find the shortest path in an unweighted graph is typically breadth-first search (BFS).

BFS explores the graph level by level, visiting vertices in a breadth-first order. This is, it starts at a given source vertex and systematically visits its neighbors, which in turn visits their neighbors, and so on, until all vertices are visited. BFS uses a queue data structure to keep track of the vertices to visit. The algorithm maintains a visited set to avoid revisiting vertices.

In our problem, a distance matrix can be created by assigning to each position a_{ij} in the matrix the shortest path between nodes i and j. This is clearly a distance matrix, and, again, the Vietoris-Rips complex can be computed. The complete code in *Python* elaborated to compute a persistence barcode and a persistence diagram from the shortest path distance matrix can be found in Annex 3.2.

In Section 1.1.1 we already discussed a consequence of the small-world effect of complex networks that can influence a study based on shortest path lengths. As mentioned there, having every pair of nodes connected in a few steps makes the distribution of shortest path lengths among pairs of nodes to be sharply peaked around its average [5]. Therefore, the minimum distance between two nodes is

almost the same for every pair. This theoretical observation has successfully been checked using TDA. Figure 2.2 is shown as an example of the persistence barcodes and persistence diagrams obtained. The diagrams obtained for the surrogates for the *CElegans-C* sample were very similar. Very few points are observed for H0 as well as for H1 in the persistence diagrams, but they all have very high multiplicity, as it can be seen in the persistence barcodes. This matches with the previous explanation, since nodes are connected in a few steps, and this is reflected in the nature of connected components and cycles.

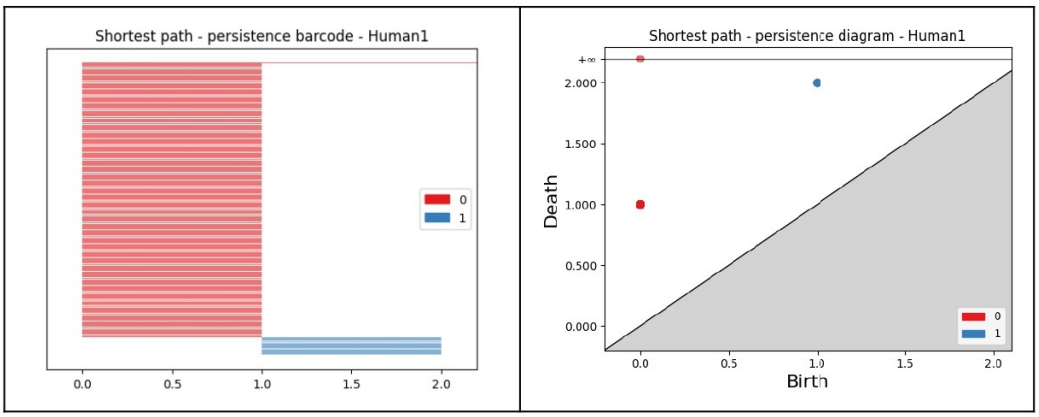

Figure 2.2: Persistence barcode and persistence diagram corresponding to *Human1* using the shortest path length to determine a distance matrix and compute the Vietoris-Rips complex. In red, H0 is represented, and in blue, H1 is shown. Axes are in distance units, where one distance unit corresponds to one edge length between two nodes.

Note that triangles are not detected by this method. This is due to the fact that nodes forming a triangle are at distance 1 between each other and, therefore, are born and die at the same step, for which reason TDA is unable to see them.

Using the shortest path to establish a metric and apply TDA methods to analyse the graphs could be used in other contexts. However, it is not appropriate when data come from complex networks. Not only because of the small-world phenomenon, but also because in this particular study we are very interested in the differences between dimensions among the embedded surrogates. And we have evidence from [1] that cycles, and in particular triangles, play a fundamental role in the assessment of dimensionality.

2.2.5 Filtration based on the degree of vertices

In the attempt to find a filtration based on intrinsic properties of the graph (as an alternative to basing it on distance matrices), the latest bibliography on the topic has been reviewed. In [7], persistent homology has shown strong empirical performance in the context of graph classification. Being able to capture long range graph properties via topological features, such as cycles of arbitrary length, in combination with other topological descriptors, it has improved predictive performance for data sets with prominent topological structures.

In practice, in [7], they comment on the empirical performance of several filtrations, and study to what extent they are capable of distinguishing between non-isomorphic graphs.

The *degree filtration* $v \mapsto \deg(v)$ showed surprising empirical performance in graph classification tasks in [7]. This is why the main part of the experimental work of this project has been carried out using a vertex degree filtration.

In practice, what this means is that in order to analyse a network using persistent homology, the graph is built by adding the vertices in increasing order of degree at each step.

Example 2.2. Consider the graph in Figure 2.3. The degree of each vertex is used to filter the graph. The calculation of persistent homology along this filtration involves counting the connected components and cycles, which are features that can only change whenever the filtration function changes. Figure 2.4 shows the persistence diagram arising from the filtration.

(a) Example graph (b) $f^{-1}((-\infty,0])$ (c) $f^{-1}((-\infty,1])$ (d) $f^{-1}((-\infty,2])$ (e) $f^{-1}((-\infty,3])$

Figure 2.3: Three different steps of a degree-based filtration for a simple graph. The preimage of the filtration function f is indicated at each step [7].

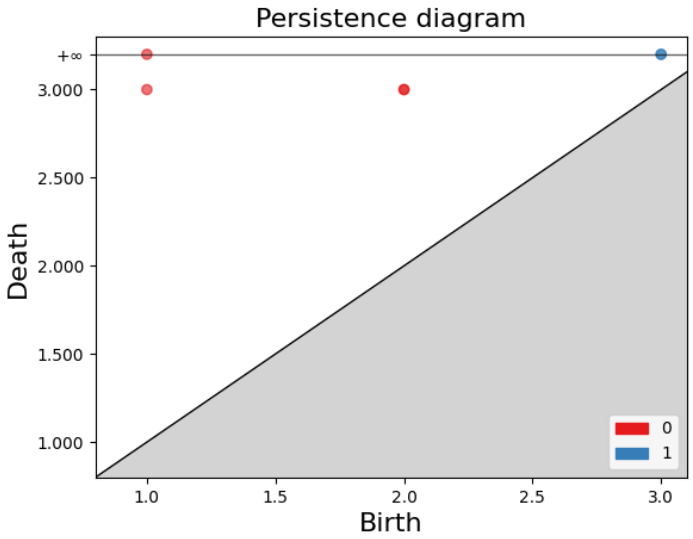

Figure 2.4: Persistence diagram corresponding to the filtration depicted in Figure 2.3. Axes are in degree units. Homology in dimension 0 is shown in red, and in blue in dimension 1. The multiplicity of points in the diagram is larger than one. The vertices with degree 1 correspond to the points $(1, 3)$ and $(1, \infty)$, since one of them merges into a larger connected component at step 3 and the other connected component lives forever. By contrast, the point $(2, 3)$ has multiplicity 2 and corresponds to the two connected components that appear at step 2 and merge into a larger connected component at step 3. The point $(3, \infty)$ also has multiplicity 2 and corresponds to the two cycles born in step 3, that live forever.

An important observation to be made here is that, while homology in dimension 0 provides meaningful information of the graph's structure, the information provided by points of homology in dimension 1 is limited. In fact, the first coordinate of the points corresponds to the maximum vertex degree forming a cycle, and the second coordinate is always infinite, since all cycles live forever.

To address this issue, and try to extract more information about cycles, the concept of *extended persistence* is explored. We already introduced extended persistence in Section 1.2.7. The idea is extracted from [8]. In our work, however, extended persistence is adapted to the vertex degree filtration that is being used. Therefore, the extended persistence diagram for homology in dimension 1 shows when cycles are born (counting as birth the moment in which the first vertex is included in the graph) and die (considering as death the moment in which the last vertex is included in the graph, and the cycle is completed). Naturally, this adds information on the cycle structure of the graph.

An extended persistence filtration based on the degree of vertices is a new tech-

nique in the TDA context, that is proposed in this work. This is the reason why a detailed theoretical analysis has been done (in this section and in Section 1.2.7), more in-depth than what it would be required for a project using more conventional TDA methods.

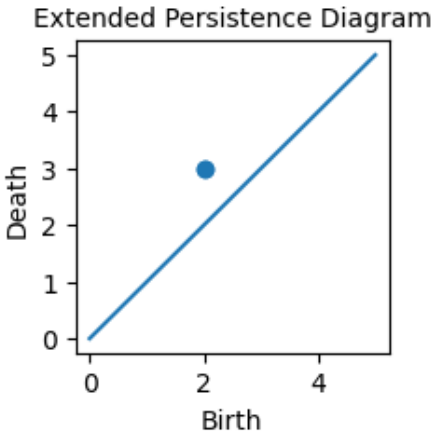

Figure 2.5: Extended persistence diagram corresponding to the graph depicted in Figure 2.3. Axes are in degree units. Notice that point $(2,3)$ has multiplicity 2, and corresponds to the two cycles, whose lowest vertex degree is 2 and whose highest vertex degree is 3.

The information provided by H0 is analysed now. Consider the visual representation of the distribution of degrees from the graph shown in Figure 2.6.

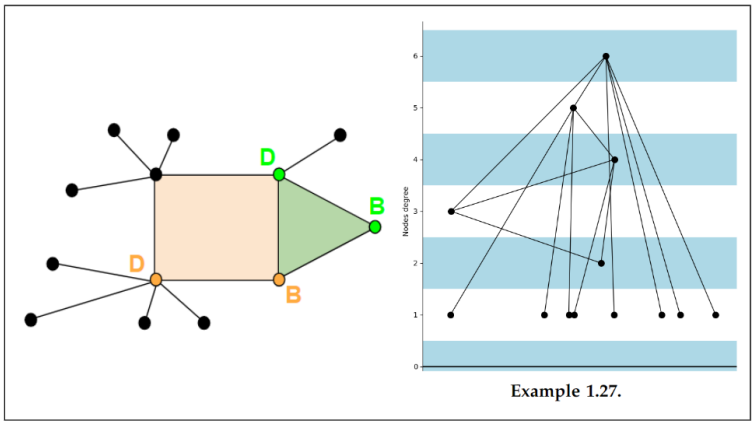

Figure 2.6: Visual representation of the tree-like structure of Example 1.10 and the degree based filtration. The lowest degree vertices are represented on the lower part of the figure, while vertices with greatest degrees are placed on top.

Recall Figure 1.9. We observe the following H0 points in the persistence diagram: $(2,4)$, $(1,5)$, $(1,6)$ and $(1,\infty)$. From Figure 2.6 it is clear that at the first step, eight connected components are created (those with degree 1). Another connected component is created at the second step (with degree 2). This one dies at step 4, leading to the point $(2,4)$. One connected component dies at step 5, and is represented by the point $(1,5)$. Finally, a great mortality is produced at step 6, killing all the connected components (point $(1,6)$), except the whole graph itself, which is the connected component that lives forever: $(1,\infty)$.

Persistence diagrams of real-world networks and their surrogates

As mentioned in Section 2.2.1, 30 surrogates were generated for each network. A persistence diagram was obtained for each surrogate and dimension, as well as for each of the given networks. The persistence diagram corresponding to each original network is represented in Figure 2.7. A sample of persistence diagrams and extended persistence diagrams of several surrogates from each network can be found in Annex 3.1, for each dimension from 1 to 10.

A visual examination of the diagrams represented in Figure 2.7, compared to the ones in Annex 3.1 corresponding to the surrogate, suggests that, in both cases, the low dimension diagrams look more similar to the original network than the ones with larger dimensions.

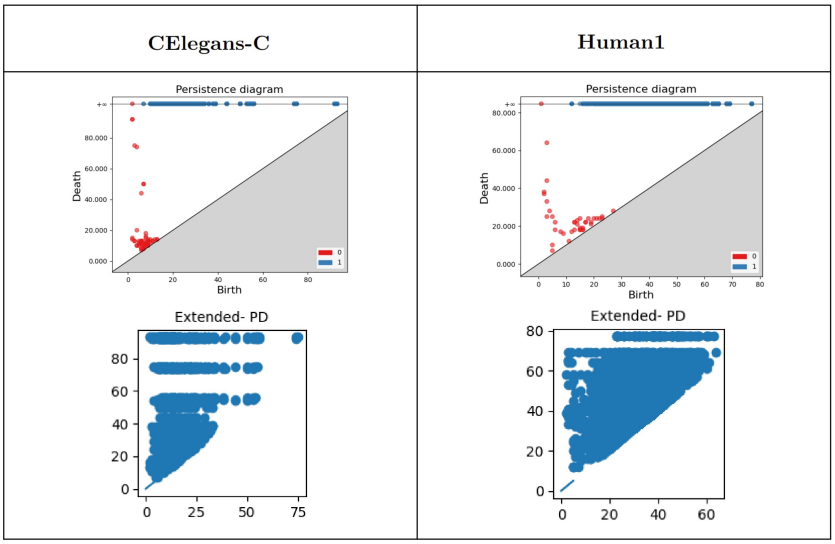

Figure 2.7: Persistence diagram and extended persistence diagram corresponding to the *CElegans-C* and *Human1* networks, respectively. Axes are in degree units.

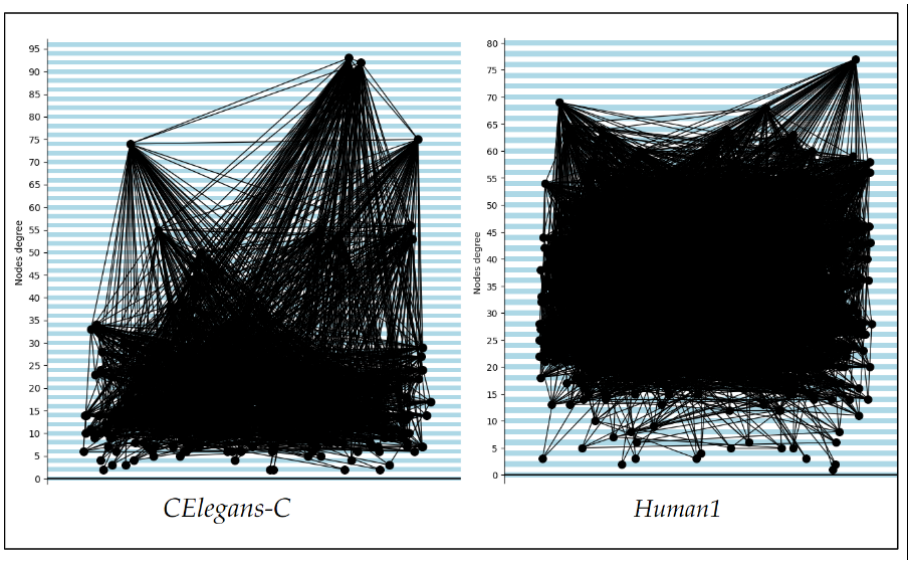

Figure 2.8: Visual representation of the tree-like structure of two networks and the degree based filtration. The lowest degree vertices are represented on the lower part of the figure, while vertices with greatest degrees are placed on top.

From Figure 2.8 the nature of the networks is captured. Points with death value at 93, 75 or 74, for example, are noticeable in the tree visualization of *CElegans-C*, as well as points with death value 65 or 44 in the case of *Human1*. These are also represented in the persistence diagrams in Figure 2.7, which indicates that the late appearance of some high degree nodes kills many connected components. Here the heterogeneity property of complex networks in the distribution of the number of contacts per node is clearly reflected, since many low-degree points can be seen, but there are only a few nodes with high degree. These high-degree points will play a significant role further in this work, and are analyzed in Section 2.2.6. The fact that a node with degree 77, for example, appears for *Human1* in the tree visualization and not in the persistence diagram only means that the other nodes contributing to this connected component were already connected between each other. Therefore, the appearance of this high-degree node does not kill any connected component.

Topological descriptors used to infer dimension

At this point, to compare each of the surrogate's diagram with the real one, a numerical quantification of the distance between diagrams is required. This will allow us to choose which dimension is more accurate to the original network. To do so, several topological descriptors have been considered. In this section it is

discussed whether the descriptors are appropriate or not, and in case they are, the results are shown. The complete code in *Python* elaborated to analyse the data can be found in Annex 3.2. Note that many computations are implemented from the *Python* **Gudhi** library [11].

In [1] the authors conclude that the *CElegans-C* network has dimension $D = 1$, and that *Human1* has dimension $D = 3$. Further from the specific number, what we aim with this section is to analyse whether TDA can distinguish between a low, a medium or a high dimension.

Figures 2.9 and 2.10 graphically summarize the results obtained. Note that there are two types of descriptors. The ones that are distances should have a minimum around the dimension of the real network. And the ones that show characteristic features of the networks should coincide, at the appropriate dimension, with the dashed red line, corresponding to the value of the corresponding feature for the real network.

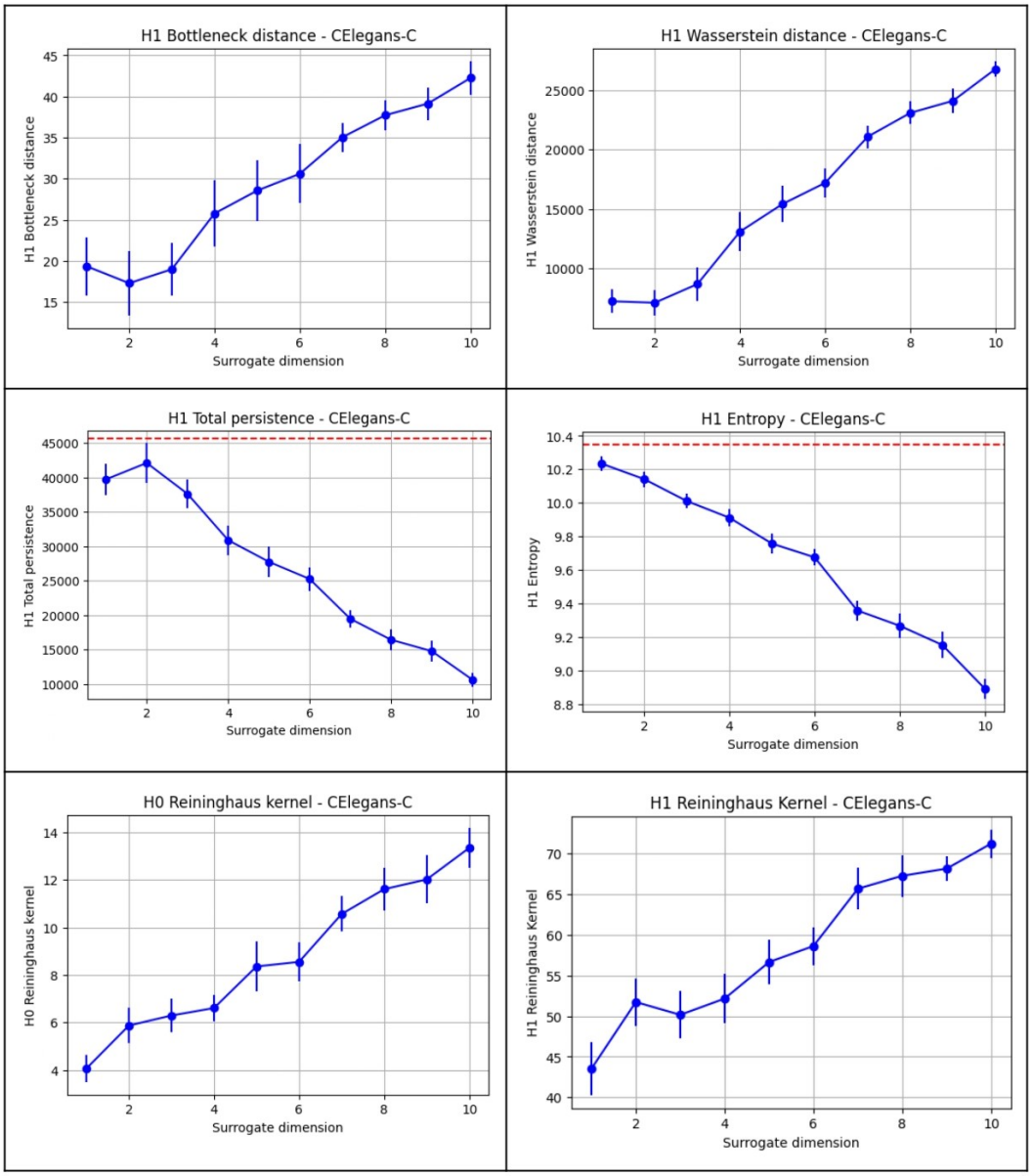

Figure 2.9: Topological persistence descriptors averaged over 30 surrogates of the *CElegans-C* sample in each dimension, with error bars corresponding to standard deviation. The x-axis shows the dimension of surrogates. When appropriate, the value of the original network is shown in red. Wasserstein distance is computed with $p = 1$ and $q = 2$. Reininghaus kernel is computed with $\sigma = 0.6$.

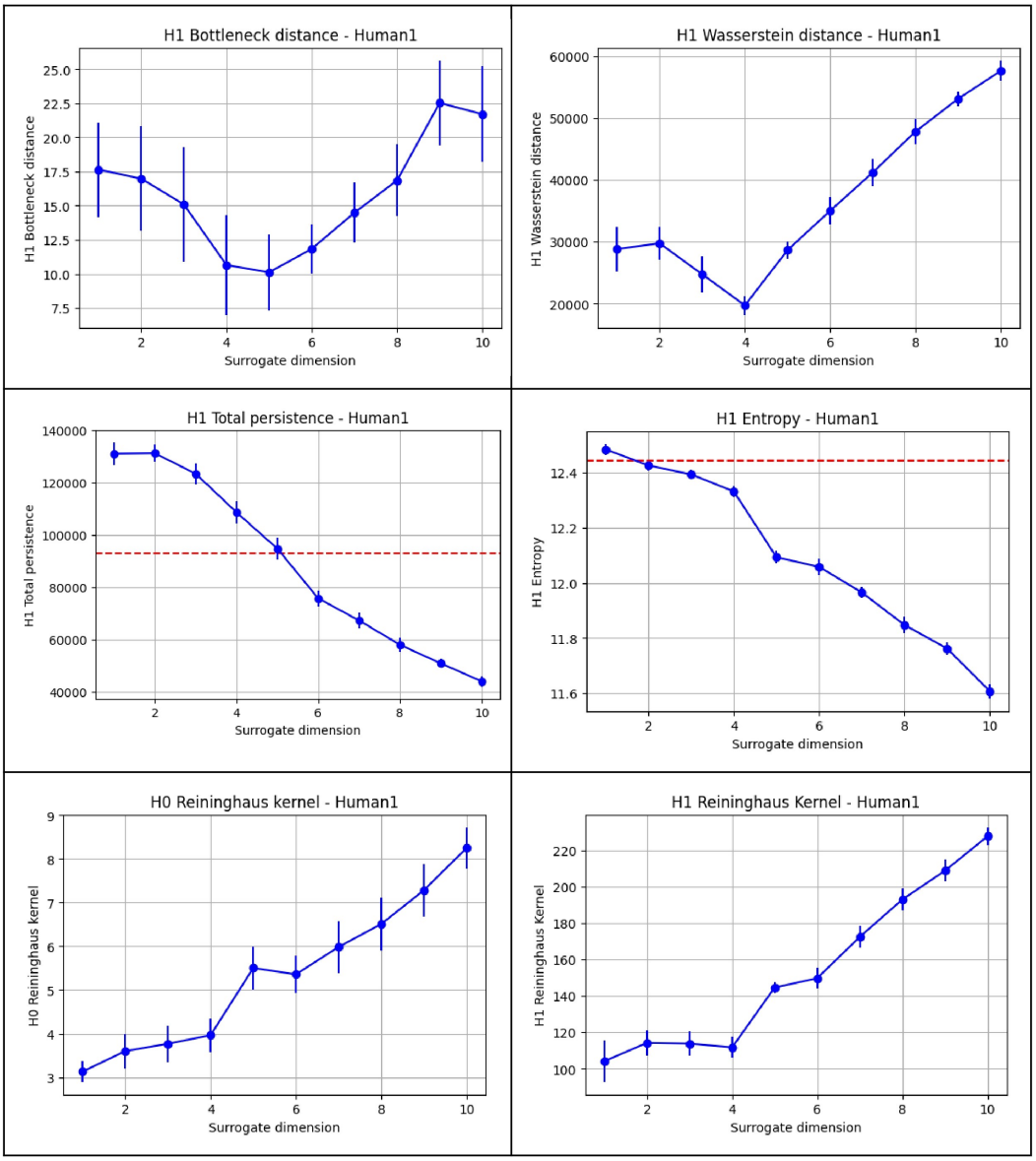

Figure 2.10: Topological persistence descriptors averaged over 30 surrogates of the *Human1* sample in each dimension, with error bars corresponding to the standard deviation interval. The x-axis shows the dimension of surrogates. When appropriate, the value of the original network is shown in red. Wasserstein distance is computed with $p = 1$ and $q = 2$. Reininghaus kernel is computed with $\sigma = 0.6$ for 10 surrogates in H1 for *Human1* due to computational constraints.

1. Bottleneck distance

Bottleneck distance, as defined in Section 1.2.5, is the most basic metric to compare persistence diagrams. If we look at the definition, it is easy to see that after choosing the best matching between pairs of points from each diagram, the distance corresponds to the maximum of the distances between points. If one looks at Figure 2.7, in both cases the persistence diagram for H0 has a point which is clearly separated from the rest, and near the infinity line.

Take the case of *CElegans-C*, for example, and consider the point $(1, 93)$. If we compare the original diagram to surrogate diagrams, in many matchings this point is paired with the diagonal, and is also the maximum distance between points in the matching. Therefore, in many cases, the distance d_∞ between two diagrams is reduced to the distance between $(1, 93)$ and the diagonal. This is independent of the other points, and therefore gives very little information of dissimilarity between the two diagrams. This result has been checked computationally. The same happens for the *Human1* sample.

Therefore, bottleneck distance has been discarded in H0. This shortcoming does not happen in H1, since there are many more points, and they are distributed through the whole diagram. Bottleneck distance has been computed to compare each of the 30 surrogates with the original network. The average distance is shown in Figure 2.9 and Figure 2.10, with the appropriate error bars, corresponding to the standard deviation interval. Bottleneck distance in H1 performs well and consistently with the expected results according to [1]. It can be seen in Figure 2.9 that *CElegans-C* is expected to have a very low dimension. The closest to the original would be $D = 2$. From Figure 2.10 we infer that *Human1* has a medium dimension. The results here suggest that the optimal dimension is $D = 4$ or $D = 5$.

2. Wasserstein distance

A similar argument applies for the Wasserstein distance, which refines the bottleneck distance. In this work, the Wasserstein distance between the real network and its surrogates has been computed with the standard parameters $p = 1$ (indicating a first-order Wasserstein distance) and $q = 2$ (which represents the internal norm used in the computation of the distance, being $q = 2$ the Euclidean norm). The results can be found in Figures 2.9 and 2.10.

Again, it can be seen that the expected dimension for *CElegans-C* is very low, between $D = 1$ and $D = 2$, and instead the supposed dimension for *Human1* is medium, around $D = 4$. It is visually clear that the Wasserstein distance (with $p = 1$ and $q = 2$) measures the same features as the bottleneck distance, but enhanced. Note, for instance, that the standard deviation bars are shorter.

3. Total persistence

For a better understanding of the data, other topological features have been computed, starting with total persistence. Total persistence sums among all the differences between death and birth of all points. The existence or not of points in the persistence diagram with very high persistence can strongly influence the results. Recall the example of point $(1, 93)$ from Figure 2.7. In Figures 2.9 and 2.10, H1 averages of the total persistence in each dimension are shown for *CElegans-C* and *Human1*, as well as the value for the original network in each case.

While the results in H0 are not accurate and strongly affected by the existence of points with very high persistence, the performance with H1 continues to be satisfactory and in agreement with previous calculations. Thus, H0 has been discarded for this study. The closest value to the original network for *CElegans-C* in this case continues to be $D = 2$, followed by $D = 1$. And looking at the total persistence graphic for *Human1* the value is close to $D = 5$. This is, in fact, a bit far from $D = 3$ as stated in [1], but still shows that the appropriate dimension for *Human1* is a medium value.

4. Entropy

As mentioned in Section 1.2.5, entropy provides a measure of the average amount of information or uncertainty in the variable's outcomes. Here entropy is measured in both H0 and H1, for the original networks and their surrogates. Again, H0 results were little informative and seemed to have a significant random component. The results suggested that the dimension is probably low or medium in both samples, although it is not conclusive. By the contrary, results for H1 are, again, satisfactory, and can be found in Figures 2.9 and 2.10. For *CElegans-C* it is clear that the most similar entropy value to the original is the one with $D = 1$. Moreover, *Human1* seems to have $D = 2$, with a value for $D = 3$ also quite close.

5. Other descriptors

Silhouette distance, being quadratic pointwise difference between the silhouette function of the original network and each of the surrogates, has been computed in this work. Landscapes and persistence images have also been obtained for every surrogate and the original networks. Results are not shown in this work because they did not add any better information with respect to the one provided by the other descriptors.

6. Reinighaus kernel

A more sophisticated distance between diagrams is computed using the scale-space kernel defined in Section 1.2.6. We will also refer to it as *Reininghaus kernel*. The value of the kernel evaluated at two persistence diagrams $K(D, D_{sur})$ is the scalar product of the vectors $\Phi(D)$ and $\Phi(D_{sur})$ and is given by Eq. (1.32). The distance between the diagrams is the norm of $\Phi(D) - \Phi(D_{sur})$ and is given by Eq. (1.31). The Reininghaus kernel distance is represented in Figures 2.9 and 2.10. Figure 2.9 indicates that *CElegans-C* has a very low dimension. In fact, it clearly suggests that the dimension of this network is 1. Computations are both done for H0 and H1 in this case. The results shown in Figure 2.10 are also very promising, since a low-to-medium dimension (1 to 4) is seen, with a large jump from 4 to 5 which discards any dimension greater than 4.

2.2.6 Overview of results

It is worth mentioning a possible reason why the results in H0 have been so imprecise in terms of discerning between dimension, and instead H1 seems to be very successful. It is clear, as it has been hypothesized at the beginning of this work, that there exists a relation between dimensionality of a real network and its cycles. This is exactly what we seek for with H1, and it is also the reason why the results are so conclusive.

H0, instead, has not performed accurately for this particular problem. Some insights of a theoretical explanation for this fact are given when the bottleneck distance is analyzed. As can be seen in Annex 3.1, the persistence diagrams of the surrogates share a "bubble pattern" concentrated at the left of each diagram. Point $(1, 93)$ is included there. These points are perfectly vertically aligned. This corresponds to the fact that x-values of the points indicate the degree of vertices, while y-values indicate the degree value in which two connected components have been merged, because an edge between them has appeared. Persistence diagrams in H0 provide little reliable information because points with high y-values have much weight on all persistence descriptors, but it only indicates that there is a connected component that has taken more time to connect with the rest. Whether this phenomenon happens or not is of random nature (since it depends on whether the p_{ij} matrix makes a particular edge appear or not), but it does not carry any information concerning the surrogate generation process.

2.3 Dimension estimate using graph theory

An alternative method based on graph theory has been used in order to compare it with our TDA approach. Heat Kernel Signatures have been introduced in Section 1.2.8. The analysis of surrogates in this section follows the same pipeline as the one described in the previous section. Thus, the same 30 surrogates that were generated for each sample and dimension are analysed here. The feature measured, though, has no longer a topological nature, but it is based on the normalized graph Laplacian.

Definition 1.11 gives an expression for the Heat Kernel Signature function. This function assigns a value to each vertex in the network. It requires a previous computation of the normalized graph Laplacian matrix, together with its eigenvalues and its eigenvectors. In the expression $hks_{G,t}\colon v \mapsto \sum_{k=1}^{n} e^{-t\lambda_k}\phi_k(v)^2$, λ_k represents the k-th eigenvalue and $\phi_k(v)$ corresponds to the v-th position of the k-th eigenvector.

The Heat Kernel Signature distance is then computed as the quadratic difference between the vector of values assigned to each vertex in the real network and the vector of values assigned to the surrogate's vertices. Figure 2.11 shows the results corresponding to the average distance between the real network Heat Kernel Signature vector and its surrogate vectors for both *CElegans-C* and *Human1*.

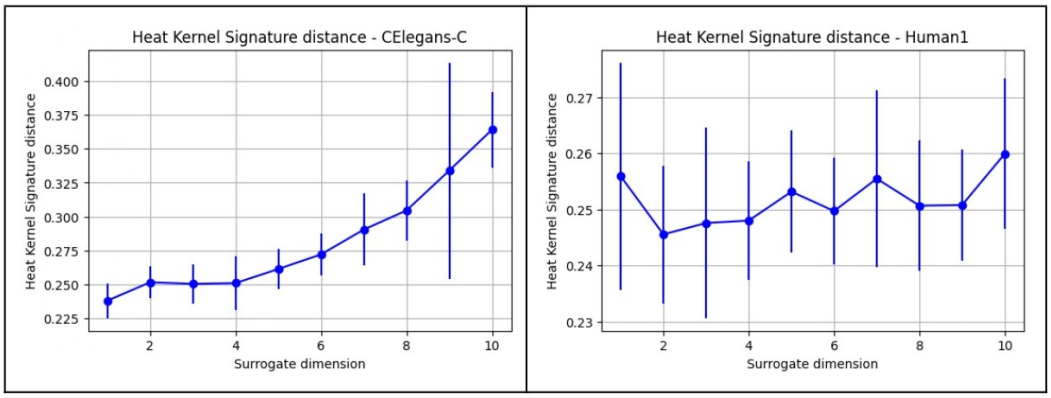

Figure 2.11: Average distance between the real network Heat Kernel Signature vector (with $t = 0.6$) and its surrogate vectors for *CElegans-C* and *Human1*, with error bars corresponding to standard deviation.

This method clearly assigns dimension 1, a very low dimension, to the *CElegans-C* sample, which is the expected result. By the contrary, the election of an accurate dimension is not so clear in the case of *Human1*. What is noticeable is the fact that HKS distance in this second case discards extreme dimensions (very low or

very high), and that a small increasing tendency can be seen. One concludes here that HKS distance is not an accurate analytic estimator of the dimensionality of networks.

In fact, HKS does not perform better than TDA persistence descriptors. In any case, the results are consistent between one method and the other, which supports the claim that graph theory and TDA are capable to detect differences between surrogates according to their dimension.

Chapter 3

Conclusions

In this work, the dimension of two complex networks is studied through a geometric network model. The model has been implemented in order to obtain 30 surrogates for each dimension and network. In [1] the dimension is found to be related to the density of cycles. Thus, it is plausible that persistent homology can detect dimensionality of networks. A degree-based filtration has been implemented together with an extended persistence technique. Bottleneck and Wasserstein (with $p = 1$ and $q = 2$) distances as well as Reininghaus kernel dissimilarities have been applied to compute differences between persistence diagrams from the real network and each surrogate. Total persistence and entropy have also been computed and compared with the corresponding values of the original network.

The results are promising and match the ones obtained in [1]. A very low dimension (1-2) has been obtained for *CElegans-C* in all cases. Similarly, *Human1* has been found to have a low-middle dimension, between 2 and 5.

The inference of a metric on graphs, seeking to apply traditional TDA techniques, has also been studied. Two approaches have been used; one based on the p_{ij} matrices, which are proportional to a distance in the model, and the other based on the distribution of shortest paths. The intrinsic properties of complex networks make both of these approaches less appropriate than the one based on the degree of vertices. The concept of hidden degree inferred to the surrogates by the geometrical model makes it ideal to study and compare different graphs with the chosen filtration function. Even though this focuses on the degree of vertices, it enhances differences between the surrogates at each step of the filtration because the appearance of a different number of edges per step gains importance.

In this sense the proposed study has successfully behaved for H1, as the focus is centered on the study of cycles. For H0, instead, little relevant information has been added to the one found from H1. Possible causes for this phenomenon have been discussed.

Other perspectives have been also implemented. The Euler characteristic has been studied. A mathematical proof is provided for the fact that an increasing tendency is expected as long as β and D are increasing. This has been checked computationally. Moreover, a graph theoretical point of view is added to the study. This is independent from TDA techniques, since it is based on properties of the graph Laplacian. The average distance between the real network Heat Kernel Signature vector and its surrogate vectors has been computed for both samples and the conclusion is that even if HKS can distinguish between high or low dimensions, it certainly does not perform better than TDA persistence descriptors.

There are several ways in which this work could be continued. Firstly, the methodology proposed here could be tested with synthetic networks generated by the model, by giving all the parameters manually instead of obtaining them from real-world networks. The advantage of it is that D could be exactly monitored, and the results would be more precise (because in real networks noise can be a factor). Secondly, networks with higher dimensions, such as Internet, should be tested. One must consider the impact of network complexity on computing times. The size of the network is tightly related to the computational demands, and dealing with networks of greater complexity presents additional computational restrictions. For example, even if the results are robust with the Reininghaus kernel, which vectorizes persistence diagrams into Hilbert spaces, total persistence is computationally faster and the results show that it is also a good analytic indicator. Finally, and most importantly, this research provides a bridge between network geometry and algebraic topology. By combining these two fields, the work pushes the boundaries of what we know, offering new insights and sparking curiosity for future exploration. It opens up exciting opportunities to delve deeper into the connections between persistent homology and complex networks, encouraging researchers to explore this fascinating domain.

Bibliography

[1] P. Almagro, M. Boguñá, M. A. Serrano, *Detecting the ultra low dimensionality of real networks*, Nature Communications vol. 13, 6096, Oct 2022.

[2] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. M. Boguñá, *Hyperbolic geometry of complex networks*, Physical Review E - Statistical, Nonlinear, and Soft Matter Physics, vol. 82, 9, Sep 2010.

[3] M. E. Aktas, E. Akbas, A El Fatmaoui, *Persistence homology of networks: methods and applications*, Applied Network Science, (2019) 4:61.

[4] D. Hernández, J. Hernández, D. Sánchez, *Simplicial degree in complex networks. Applications of topological data analysis to network science*, Chaos, Solitons and Fractals, vol. 136, Aug 2020.

[5] M. A. Serrano and M. M. Boguñá, *The Shortest Path to Network Geometry*, Cambridge University Press, Dec 2021.

[6] M. Boguñá, I. Bonamassa, M. D. Domenico, S. Havlin, D. Krioukov, and M. A. Serrano, *Network geometry*, Oct 2020. [Online]. Available: arXiv: 2001.03241.

[7] B. Rieck, *On the expressivity of persistent homology in graph learning*, Feb 2023. [Online]. Available: arXiv: 2302.09826.

[8] M. Carrière, F. Chazal, Y. Ike, T. Lacombe, M. Royer, Y. Umeda, *PersLay: A neural network layer for persistence diagrams and new graph topological signatures*, Apr 2019. [Online]. Available: arXiv: 1904.09378.

[9] Y. Ahn, H. Jeong, and B. J. Kim, *Physica A: Statistical Mechanics and its Applications*, 367, 531 (2006).

[10] G. García-Pérez, A. Allard, M. A. Serrano, M. Boguñá, *Mercator: Uncovering faithful hyperbolic embeddings of complex networks*, New Journal of Physics, vol. 21, 2019.

[11] C. Maria, F. Godi, T. Lacombe, M. Glisse, M. Carrière, M. Royer, G. Spreemann, W. Reise, V. Rouvreau, *Topological descriptors computation, Topological descriptors tools* GUDHI 3.1.0 [Online]. Available: https://gudhi.inria.fr/python/latest/.

[12] C. Casacuberta, *Topological Data Analysis; Matemàtica avançada per a reptes científics*, 2022.

[13] P. Bubenik, *Statistical topological data analysis using persistence landscapes*, J. Mach. Learn. Res., 16 (2015).

[14] J. Reininghaus, S. Huber, U. Bauer, R. Kwitt, *A stable multi-scale kernel for topological machine learning*, 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, 4741-4748.

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

Annex

3.1 Persistence diagrams of network surrogates

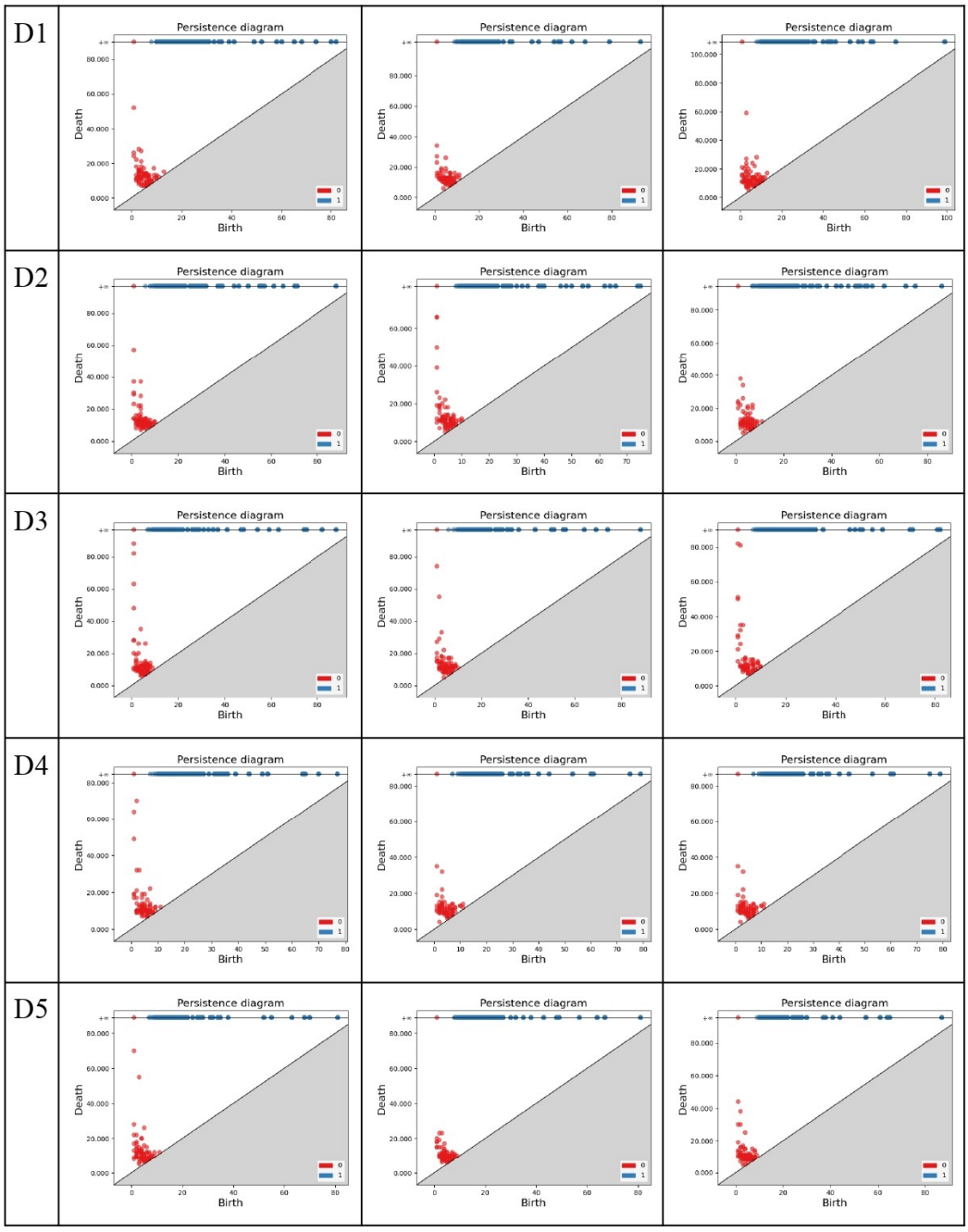

Figure 3.1: Persistence diagrams of three example surrogates for each dimension D from 1 to 5 of the *CElegans-C* network. Axes are in degree units. Red points correspond to H0 and blue points to H1.

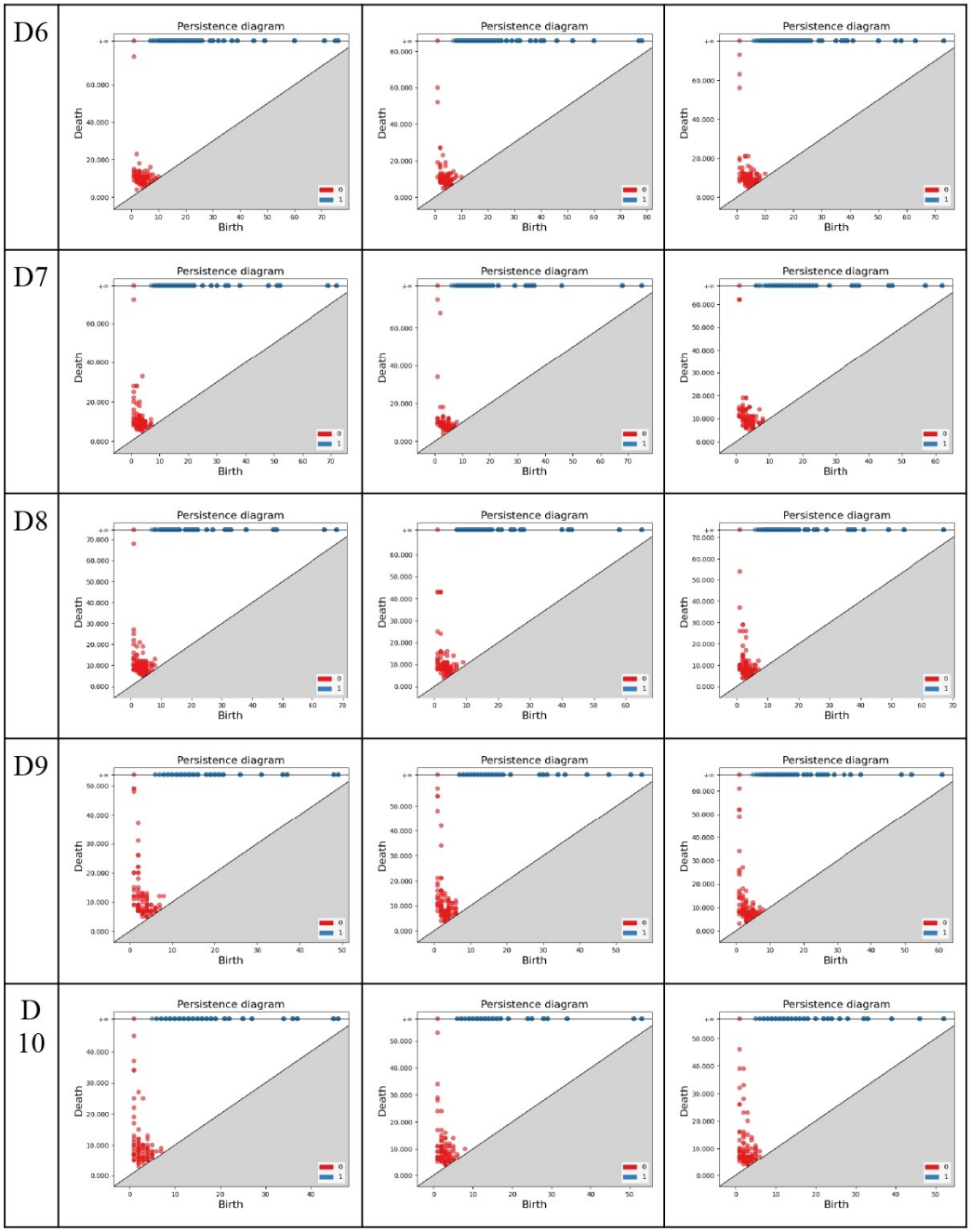

Figure 3.2: Persistence diagrams of three example surrogates for each dimension D from 5 to 10 of the *CElegans-C* network. Axes are in degree units. Red points correspond to H0 and blue points to H1.

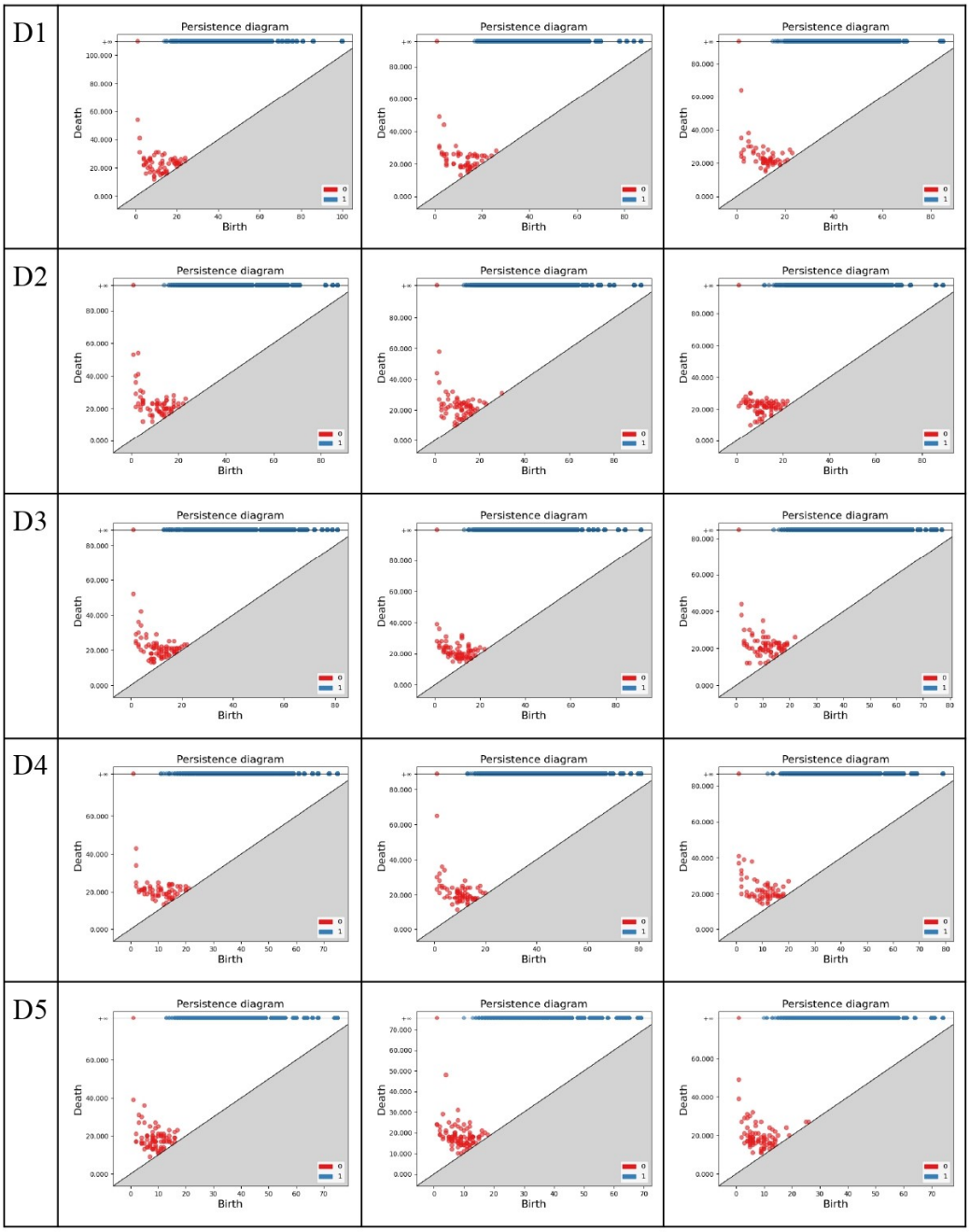

Figure 3.3: Persistence diagrams of three example surrogates for each dimension D from 1 to 5 of the *Human1* network. Axes are in degree units. Red points correspond to H0 and blue points to H1.

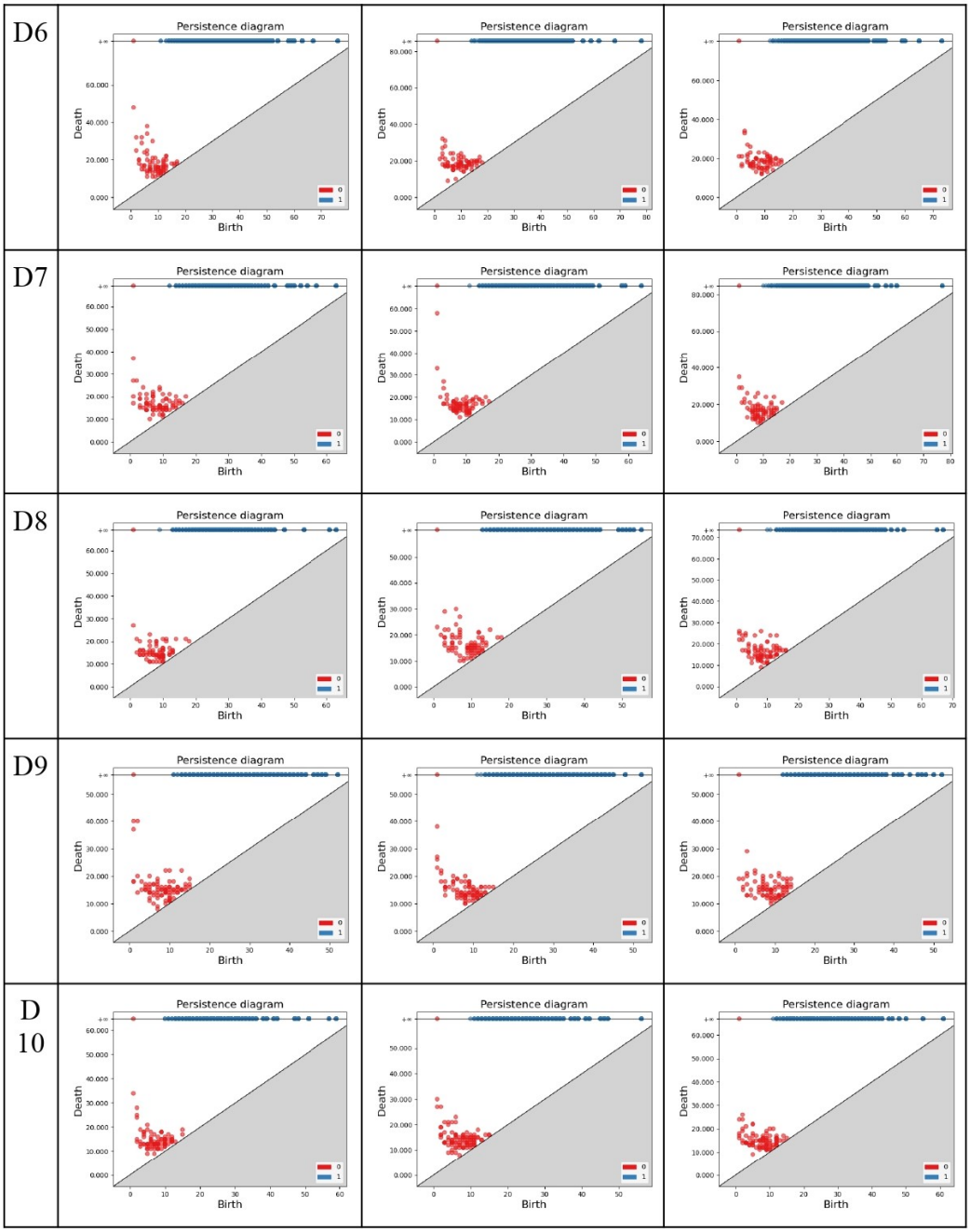

Figure 3.4: Persistence diagrams of three example surrogates for each dimension D from 5 to 10 of the *Human1* network. Axes are in degree units. Red points correspond to H0 and blue points to H1.

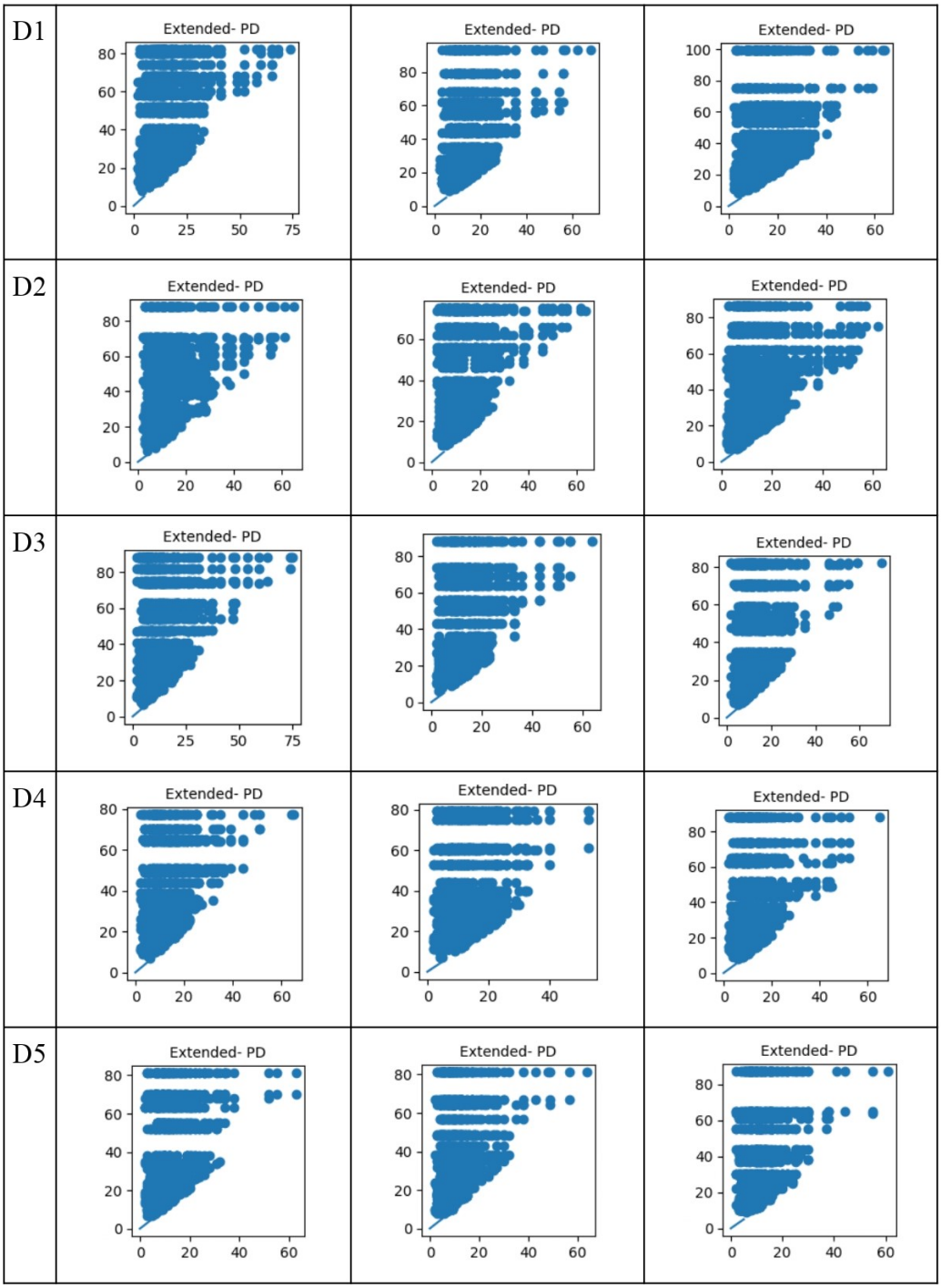

Figure 3.5: Extended persistence diagrams for H1 of three example surrogates for each dimension D from 1 to 5 of the *CElegans-C* network. Axes are in degree units.

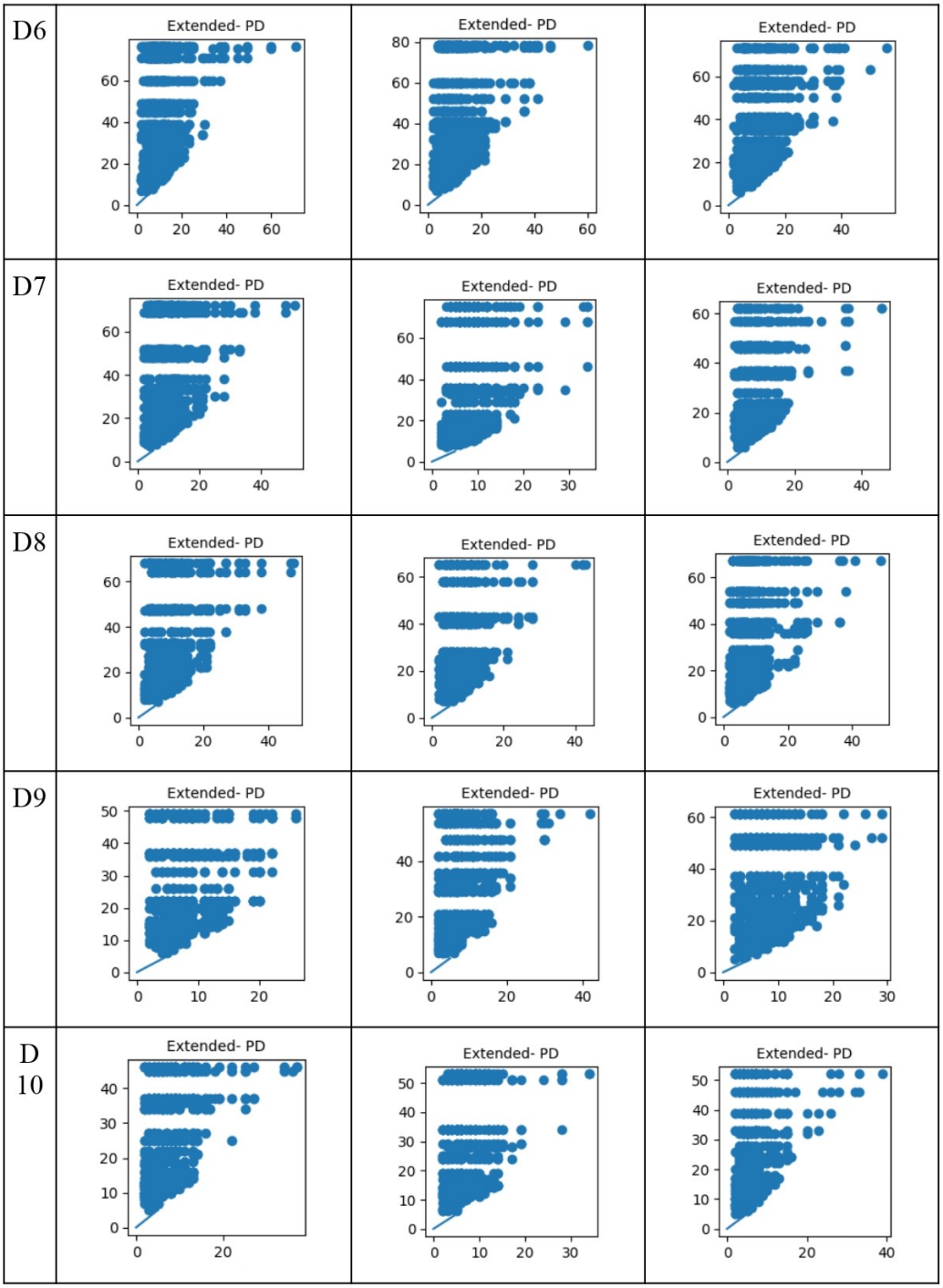

Figure 3.6: Extended persistence diagrams for H1 of three example surrogates for each dimension D from 5 to 10 of the *CElegans-C* network. Axes are in degree units.

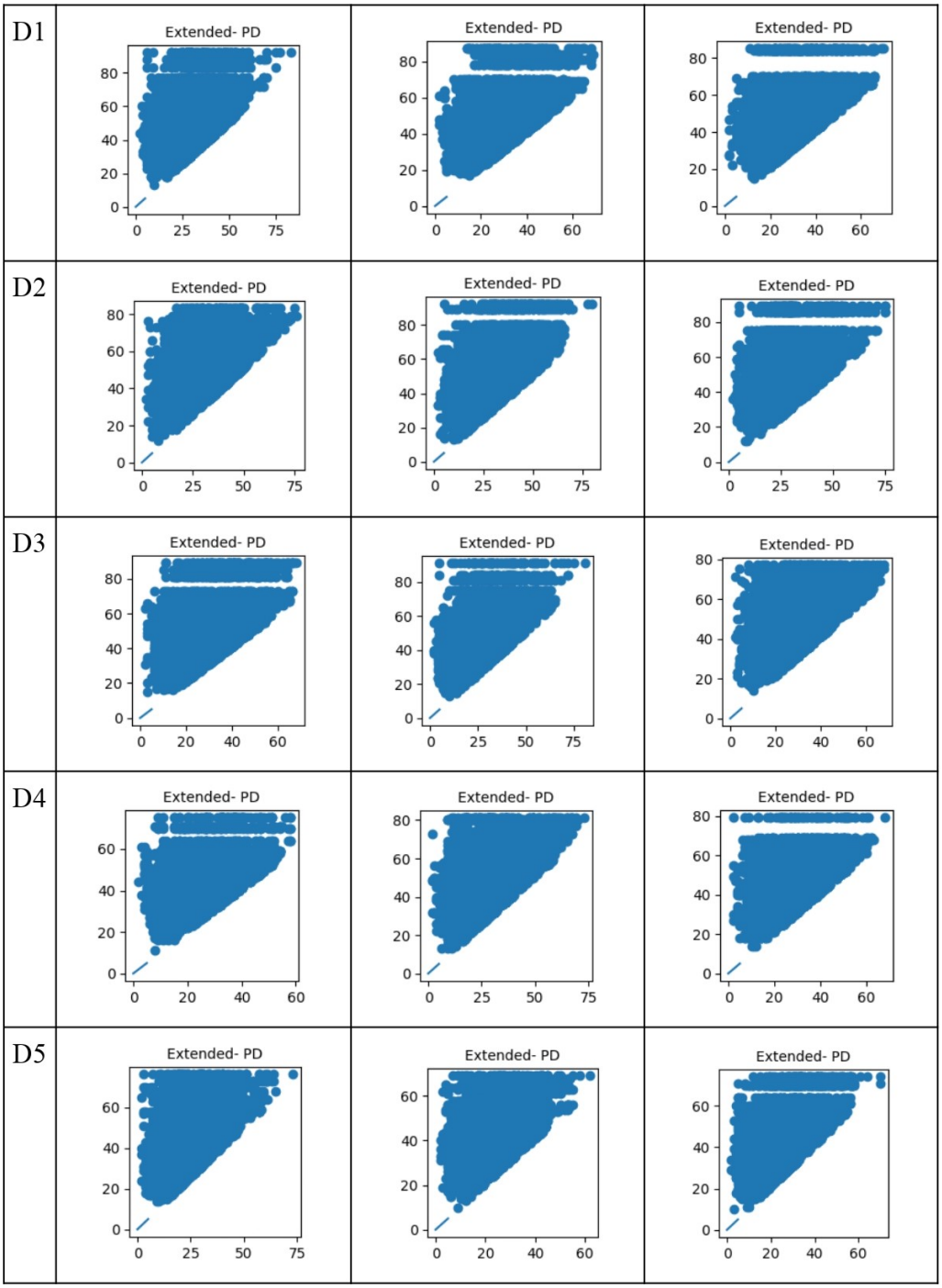

Figure 3.7: Extended persistence diagrams for H1 of three example surrogates for each dimension D from 1 to 5 of the *Human1* network. Axes are in degree units.

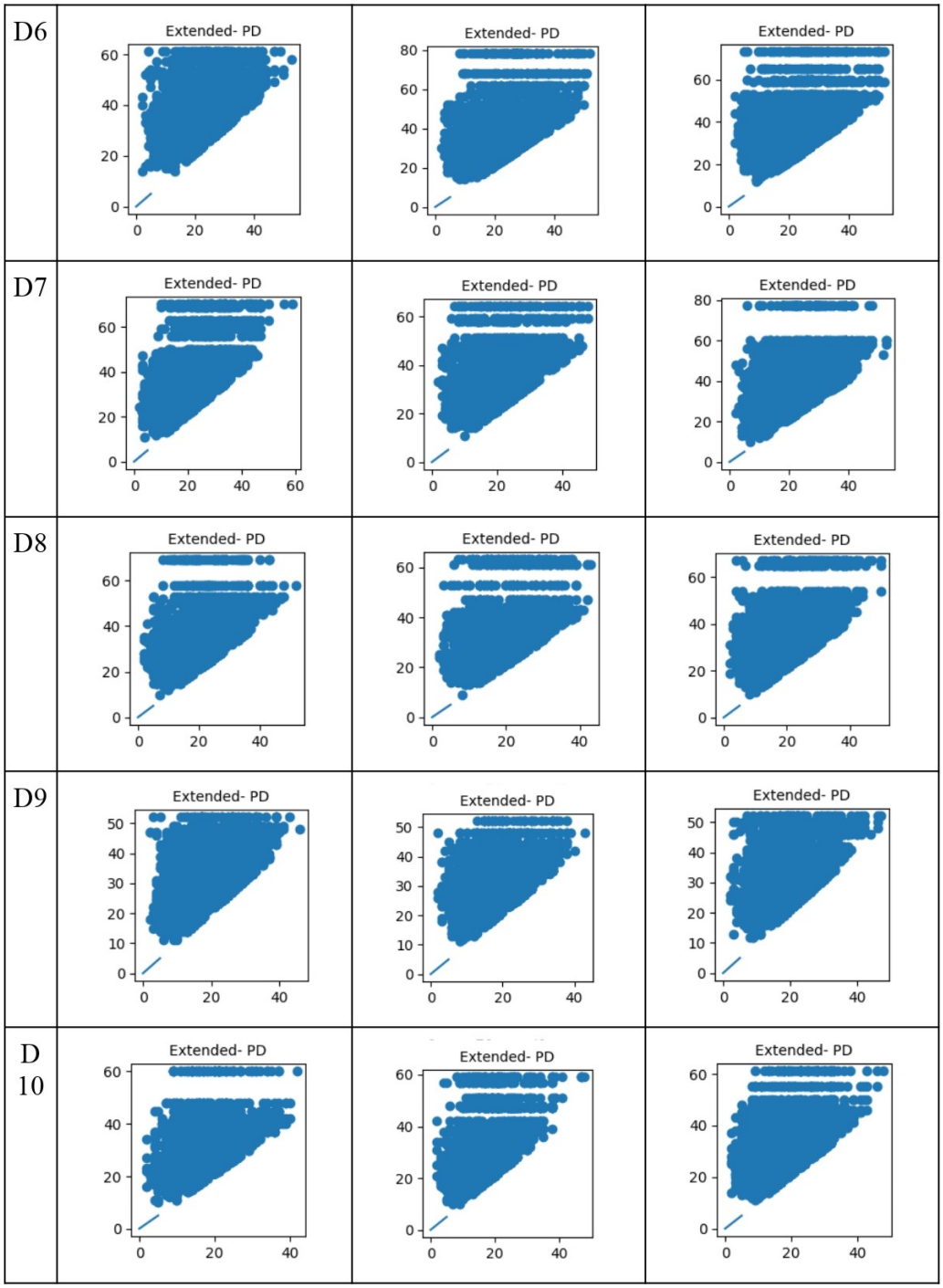

Figure 3.8: Extended persistence diagrams for H1 of three example surrogates for each dimension D from 5 to 10 of the *Human1* network. Axes are in degree units.

The reader should notice that Figures 3.5, 3.6, 3.7 and 3.8 do not necessarily share the same scale in the axes (although in the vast majority, they do). This is done on purpose, in order to help the reader distinguish the features between dimensions, and at the same time to make the diagrams readable.

3.2 *Python* code generated to analyse the data using vertex filtrations

The following code was generated to obtain the figures in Section 2.2.4. Note that *Networkx* internally uses BFS to calculate the shortest path length between the source and target nodes in unweighted graphs.

```
!pip install networkx gudhi
!pip install gudhi

from sklearn import datasets
import pandas as pd # pandas for data bases
import numpy as np # numpy for vectors and matrices
import matplotlib.pyplot as plt # to draw graphics
import gudhi as gd #for the TDA computations
import networkx as nx #to manipulate complex graph networks
import gudhi

# Read edges from file and create a graph
G = nx.Graph()
with open('Human1_3_10.edge', 'r') as f:
    for line in f:
        if line.startswith('#'):
            continue
        u, v = line.strip().split()
        u = int(u[1:])
        v = int(v[1:])
        G.add_edge(u, v)

# Sort the nodes numerically
nodes = sorted(G.nodes())

# Print the nodes and edges of the graph
print("Nodes: ", nodes)
print("Edges: ", G.edges())

# Draw the graph
nx.draw(G, with_labels=True, node_size=70)
plt.show()

# Build the associate matrix using the shortest path lengths
n = len(nodes)
matriu = np.zeros((n, n))
```

```
for i, node1 in enumerate(nodes):
    for j, node2 in enumerate(nodes):
        if i != j:
            try:
                matriu[i][j] = matriu[j][i] = nx.shortest_path_length(G,
                    source=node1, target=node2)
            except nx.NetworkXNoPath:
                matriu[i][j] = matriu[j][i] = float('inf')

print(matriu)

rips_complex_sample = gd.RipsComplex(distance_matrix = matriu)
rips_complex_tree = rips_complex_sample.create_simplex_tree(max_dimension
    =2)
persistence = rips_complex_tree.persistence()

bx = gd.plot_persistence_barcode(persistence)
ax = gd.plot_persistence_diagram(persistence)
```

The code used to analyse the networks using topological data analysis tools is also available in this section. This first part corresponds to the packages that need to be installed or imported.

```
!pip install networkx gudhi
!pip install gudhi
!pip install tqdm
!pip install POT

import numpy as np
import pandas as pd
import gudhi as gd
import math
import matplotlib.pyplot as plt
import networkx as nx
import gudhi.wasserstein
import ot

from mpl_toolkits.mplot3d import Axes3D
from sklearn import metrics
from numpy.linalg import eig
from gudhi import representations as gdr
from tqdm.notebook import tqdm
```

Next, several functions are defined in order to compute different topological descriptors.

```
1
2  def compute_Euler_characteristic(G):
3    return G.number_of_nodes() - G.number_of_edges()
4
5  def total_persistence(persistence, dim):
6      summ = 0
7      for i in range(len(persistence)):
8          if persistence[i][0] == dim:
9              if persistence[i][1][1] < float('inf'):
10                 summ += persistence[i][1][1] - persistence[i][1][0]
11     return summ
12
13 def compute_laplacian_spectrum(G):
14   nodes = G.nodes
15   laplaciana = np.zeros((G.number_of_nodes(), G.number_of_nodes()))
16   for i, n in enumerate(nodes):
17     for j, m in enumerate(nodes):
18       if i == j:
19         laplaciana[i][i] = G.degree(n)
20       elif n in G.neighbors(m):
21         laplaciana[i][j] = laplaciana[j][i] = -1
22   w,v = eig(laplaciana)
23   return np.sort(w)
24
25 def compute_Euler_characteristic_distance(E1, E2):
26   return E1 - E2
27
28 def compute_normalized_laplacian_spectrum(G):
29   #Laplaciana
30   nodes = G.nodes
31   #n = 280
32   laplaciana = np.zeros((G.number_of_nodes(), G.number_of_nodes()))
33   for i, n in enumerate(nodes):
34     for j, m in enumerate(nodes):
35           if i == j and G.degree(n) != 0:
36             laplaciana[i][i] = 1
37           elif n in G.neighbors(m):
38               laplaciana[i][j] = laplaciana[j][i] = -1.0/np.sqrt(G.degree(n)*
                     G.degree(m))
39   w,v = eig(laplaciana)
40   return np.sort(w)
```

```
41
42 def compute_landscape(dirty_diagram, dim = 0, k = 50, res = 200):
43   clean_dgm= [np.array(dirty_diagram[i][1]) for i in range(len(dirty_diagram
         )) if dirty_diagram[i][0] == dim and dirty_diagram[i][1][1] != float('
         inf')]
44   L = gdr.Landscape(num_landscapes=k, resolution=res).fit_transform([np.
         array(clean_dgm)])
45   return L[0]
46
47 def compute_silhouette(dirty_diagram, dim = 0, p = 1, res = 200):
48   clean_dgm= [np.array(dirty_diagram[i][1]) for i in range(len(dirty_diagram
         )) if dirty_diagram[i][0] == dim and dirty_diagram[i][1][1] != float('
         inf')]
49   sil = gdr.Silhouette(weight = lambda x: np.power(x[1]-x[0], p), resolution
         =200)
50   silueta = sil.fit_transform([np.array(clean_dgm)])
51   return silueta[0]
52
53 def compute_persistence_image(dirty_diagram, dim = 0, sigma = 1, res = 200)
       :
54   clean_dgm= [np.array(dirty_diagram[i][1]) for i in range(len(dirty_diagram
         )) if dirty_diagram[i][0] == dim and dirty_diagram[i][1][1] != float('
         inf')]
55   img = gdr.PersistenceImage(bandwidth = sigma, resolution=[res, res])
56   imatge = img.fit_transform([np.array(clean_dgm)])
57   return imatge[0]
58
59 def compute_bottleneck_distance(dgm1, dgm2):
60   diag1 = [[bd[1][0], bd[1][1]] for bd in dgm1 if bd[0] == 0]
61   diag2 = [[bd[1][0], bd[1][1]] for bd in dgm2 if bd[0] == 0]
62   return gd.bottleneck_distance(diag1, diag2)
63
64 def compute_wasserstein_distance(dgm1, dgm2):
65   diag1 = [[bd[1][0], bd[1][1]] for bd in dgm1 if bd[0] == 0]
66   diag1.pop(0)
67   diag1 = np.array(diag1)
68   diag2 = [[bd[1][0], bd[1][1]] for bd in dgm2 if bd[0] == 0]
69   diag2.pop(0)
70   diag2 = np.array(diag2)
71   return gd.wasserstein.wasserstein_distance(diag1, diag2, order=1.,
         internal_p=2.)
72
73 def compute_landscape_distance(L1, L2, k = 50, res = 200):
74   diff_area = []
```

```
  for i in range(k):
      auc1_i = metrics.auc(range(res), L1[i*res:(i+1)*res])
      auc2_i = metrics.auc(range(res), L2[i*res:(i+1)*res])
      diff_area.append((auc1_i - auc2_i)**2)
  return np.sqrt(np.sum(diff_area)) / res

def compute_silhouette_distance(S1, S2):
  return np.sqrt(np.sum(np.square(S1-S2)))

def Reininghaus_kernel(sigma, dgm1, dgm2, dim=1):
    dgm1 = np.array(dgm1)
    dgm2 = np.array(dgm2)
    n = dgm1.shape[0]
    m = dgm2.shape[0]
    K = np.zeros((n, m))
    for i in range(n):
        for j in range(m):
            if dgm1[i][0] == dim or dgm2[j][0] == dim:
                continue
            p = np.array(dgm1[i][1])
            q = np.array(dgm2[j][1])
            if np.isfinite(p[1]) and np.isfinite(q[1]):
                d2 = (p[0]-q[0])**2 + (p[1]-q[1])**2
                d_bar2 = (p[0]-q[1])**2 + (p[1]-q[0])**2
                K[i,j] = np.exp(-d2/(8*sigma)) - np.exp(-d_bar2/(8*sigma))
    total = np.sum(K) / (8*np.pi*sigma)
    return total

def compute_reininghaus_distance(sigma, dgm1, dgm2, dim=1):
  d = np.sqrt(Reininghaus_kernel(sigma, dgm1, dgm1)-2*Reininghaus_kernel(
      sigma, dgm1, dgm2)+Reininghaus_kernel(sigma, dgm2, dgm2))
  return d

def compute_entropy(dgm1, dim=0):
    dgm1 = np.array(dgm1)
    n = dgm1.shape[0]
    L = total_persistence(dgm1, dim)
    entropy = 0
    for i in range(n):
        if dgm1[i][0] == 1:
            continue
        di = dgm1[i][1][1]
        if np.isfinite(di):
          bi = dgm1[i][1][0]
```

```
        p = (di - bi) / L
        entropy -= p * np.log2(p)
    return entropy

def compute_heat_kernel_signature(G):
  n = G.nodes
  H = []
  t = 0.6
  vaps, veps = compute_normalized_laplacian_spectrum(G)
  for v in range(len(n)):
    sum=0
    for k in range(len(n)):
      l = vaps[k]
      phi = veps[k][v] #k-th vector at v position
      c = math.exp(-t*l)*(phi**2)
      sum += c
    H.append(sum)
  return H

def read_graph(path):
  # read edges from file and create an undirected graph G
  G = nx.Graph()
  with open(path, 'r') as f:
    for line in f:
      u, v = line.strip().split()
      G.add_edge(u, v)
  return G

def read_graph_modified(path):
    G = nx.Graph()
    with open(path, 'r') as f:
        for line in f:
            if line.startswith('#'):
                continue
            u, v = line.strip().split()
            u = int(u[1:]) # convert node labels to integers by removing the
                'v' prefix
            v = int(v[1:])
            G.add_edge(u, v)
    return G

def compute_degree_filtration_persistence(G):
  #A dictionary called node_index_map is created to map nodes to their
     indices in the simplex tree.
```

```
  node_index_map = {node: i for i, node in enumerate(G.nodes())}

  st = gd.SimplexTree()
  for (u, v) in G.edges():
      st.insert([node_index_map[u], node_index_map[v]])
  for i, node in enumerate(G.nodes()):
      st.insert([node_index_map[node]])
      st.assign_filtration([node_index_map[node]], G.degree[node])

  _ = st.make_filtration_non_decreasing()

  dgm = st.persistence(persistence_dim_max=True)

  return dgm
```

The following code corresponds to the main program in homological dimension 0 (H0). At first, it reads and computes topological descriptors for an original network. Then, in a loop over the 10 dimensions and the 30 surrogates for each network, it computes the same topological descriptors for surrogates, and compares them to the values for the original network.

```

G = read_graph('Human1.edge')
dgm = compute_degree_filtration_persistence(G)

caract = compute_Euler_characteristic(G)
entropy = compute_entropy(dgm)
laplaciana_spec = compute_laplacian_spectrum(G)
laplaciana_spec_norm = compute_normalized_laplacian_spectrum(G)
totpers = total_persistence(dgm, 0)
silhouette = compute_silhouette(dgm)
landscape = compute_landscape(dgm)
imatge = compute_persistence_image(dgm)

#BEGINING OF THE MAIN PROGRAM

# DataFrame to store the values
df = pd.DataFrame(columns = ['Nom', 'Euler Caract', 'Wasserstein Dist', '
    Entropy', 'TotPers Dist', 'Sil Dist', 'Rein dist 0.6', 'Sign dist'])
base_name = 'Human1_'
for dim in tqdm(range(1, 11)):
  for i in tqdm(range(1, 31)):
    name = base_name + str(i) + "_" + str(dim) +".edge"
```

```
    print(name)
    G_sur = read_graph_modified(name)
    caract_sur = compute_Euler_characteristic(G_sur)
    caract_dist = compute_Euler_characteristic_distance(caract, caract_sur)
    #laplaciana_spec_sur = compute_laplacian_spectrum(G_sur)
    #laplaciana_spec_norm_sur = compute_normalized_laplacian_spectrum(G_sur)
    dgm_sur = compute_degree_filtration_persistence(G_sur)
    signature_sur = compute_heat_kernel_signature(G_sur)

    # compute the bottleneck distance
    bottleneck_dist = compute_bottleneck_distance(dgm, dgm_sur)

    # compute the wasserstein distance
    wasserstein_dist = compute_wasserstein_distance(dgm, dgm_sur)

    # Compute the total persistence
    totperszero_sur = total_persistence(dgm_sur, 0)
    totpers_dist = abs(totperszero_sur - totpers)

    # Compute the entropy
    entropy_sur = compute_entropy(dgm_sur, 0)
    entropy_dist = abs(entropy_sur - entropy)

    # Compute the silhouette
    silhouette_sur = compute_silhouette(dgm_sur)
    silhouette_dist = compute_silhouette_distance(silhouette, silhouette_sur
        )

    # Compute distance between landscapes by summing quadraticaly the
        difference between areas
    landscape_sur = compute_landscape(dgm_sur)
    landscape_dist = compute_landscape_distance(landscape, landscape_sur)

    # Compute the persistence image
    imatge_sur = compute_persistence_image(dgm_sur)
    imatge_dist = compute_persistence_image_distance(imatge, imatge_sur)

    # Distance between images using Reininghaus Kernel's
    Rein_dist_06 = compute_reininghaus_distance(0.6, np.array(dgm), np.array
        (dgm_sur))

    # Euclidean distance between the two heat signature vectors
    min_length = min(len(signature), len(signature_sur))
    signature_dist_complex = np.sqrt(np.sum(np.square(np.array(signature[:
```

```
        min_length]) - np.array(signature_sur[:min_length])))
    signature_dist = np.real(signature_dist_complex)

    df = df.append({'Nom': name, 'Euler Caract': caract_sur, 'Wasserstein
        Dist': wasserstein_dist, 'Entropy': entropy_sur, 'TotPers Dist':
        totperszero_sur, 'Sil Dist': silhouette_dist, 'Rein dist 0.6':
        Rein_dist_06, 'Sign dist': signature_dist}, ignore_index = True)
  df.to_csv('signhuman1_'+str(dim)+'.csv', index=False)
```

Finally, the data is averaged for each dimension, and standard deviation is computed. This is used to plot a comparison between the values for the 10 dimensions and the original value. An example is provided here for the Euler characteristic. The other descriptors are plotted analogously.

```
from google.colab import drive
drive.mount('/content/drive')

interval_size = 30 # Size of each interval
num_intervals = 10 # Number of intervals

for dim in range(1, 11):
    df = pd.read_csv('prova_cuc_' + str(dim) + '.csv')

    slices = []
    for i in range(num_intervals):
        start = i * interval_size
        end = (i + 1) * interval_size
        interval_slice = df.iloc[start:end]
        slices.append(interval_slice)

    euler_dim = []
    euler_var_dim = []
    entr_dim = []
    entr_var_dim = []
    perst_dim = []
    perst_var_dim = []
    sil_dist_dim = []
    sil_dist_var_dim = []
    rein_dist_dim = []
    rein_dist_var_dim = []

    for interval_slice in slices:
        euler_dim.append(interval_slice['Euler Caract'].mean())
```

```
        euler_var_dim.append(interval_slice['Euler Caract'].std())
        entr_dim.append(interval_slice['Entropy'].mean())
        entr_var_dim.append(interval_slice['Entropy'].std())
        perst_dim.append(interval_slice['TotPers Dist'].mean())
        perst_var_dim.append(interval_slice['TotPers Dist'].std())
        sil_dist_dim.append(interval_slice['Sil Dist'].mean())
        sil_dist_var_dim.append(interval_slice['Sil Dist'].std())
        rein_dist_dim.append(interval_slice['Rein dist 0.6'].mean())
        rein_dist_var_dim.append(interval_slice['Rein dist 0.6'].std())

#Euler's characteristic plot - 10 dimensions comparison to the original
    value
fig, ax1 = plt.subplots(figsize=(6, 4))
ax1.errorbar(x = [i for i in range(1, 11)], y = euler_dim, yerr =
    euler_var_dim, color='blue', fmt='-o')
ax1.axhline(caract, ls='--', color='red')
ax1.set_ylabel('Euler Characteristic')
ax1.set_xlabel('Surrogate dimension')
plt.grid(True)
plt.title("Euler Characteristic - Human1")
plt.show()
```

For H1, the code to compute topological descriptors is analogous. However, as mentioned during the project, extended persistence is used in H1 to obtain more information about the cycles. The following code computes extended persistence and visualizes it.

```
#read edges from file and create an undirected graph G
G = nx.Graph()
with open('Human1.edge', 'r') as f:
    for line in f:
        u, v = line.strip().split()
        G.add_edge(u, v)

node_index_map = {node: i for i, node in enumerate(G.nodes())}

# Construct the simplex tree and assign filtration values
# The simplex tree st is constructed by inserting nodes and edges from the
    graph G.
# Filtration values are assigned to nodes based on their degree in the
    graph
st = gd.SimplexTree()
for (u, v) in G.edges():
```

```
    st.insert([node_index_map[u], node_index_map[v]])
for i, node in enumerate(G.nodes()):
    st.insert([node_index_map[node]])
    st.assign_filtration([node_index_map[node]], G.degree[node])

_ = st.make_filtration_non_decreasing()

nx.draw(G, with_labels=True, node_size = 70)
plt.show()

# Visualize the graph
pos = {}
for idxv, v in enumerate(G.nodes()):
    y = st.filtration([node_index_map[v]])
    x = random.uniform(-1, 1)
    pos[v] = [x,y]
fig, ax = plt.subplots(figsize=(5,5))
nx.draw(G, pos=pos, node_size=70, ax=ax)
limits=plt.axis('on')
ax.tick_params(left=True, bottom=False, labelleft=True, labelbottom=False)
plt.show()


# Compute the extended persistence diagram and visualize it
st.extend_filtration()
dgms = st.extended_persistence(min_persistence=1e-5)

#make extended persistence for H1 symmetric respect the diagonal
invext_dgm = [np.array([p[1], p[0]]) for _, p in dgms[3] if np.isfinite(p
    [1])]
invext_dgm_tuples = [tuple(p) for p in invext_dgm]

# Compute the extended persistence diagram and visualize it
fig, axs = plt.subplots(2, 2, figsize=(5,5))
plt.subplots_adjust(hspace=0.3, wspace=0.3)
axs[0,0].scatter([dgms[0][i][1][0] for i in range(len(dgms[0]))], [dgms[0][
    i][1][1] for i in range(len(dgms[0]))])
axs[0,0].plot([0,5],[0,5])
axs[0,0].set_title("Ordinary PD", fontsize=10)
axs[0,1].scatter([dgms[1][i][1][0] for i in range(len(dgms[1]))], [dgms[1][
    i][1][1] for i in range(len(dgms[1]))])
axs[0,1].plot([0,5],[0,5])
axs[0,1].set_title("Relative PD", fontsize=10)
axs[1,0].scatter([dgms[2][i][1][0] for i in range(len(dgms[2]))], [dgms[2][
```

```
    i][1][1] for i in range(len(dgms[2]))])
axs[1,0].plot([0,5],[0,5])
axs[1,0].set_title("Extended+ PD", fontsize=10)
invext_dgm_x = [p[0] for p in invext_dgm_tuples]
invext_dgm_y = [p[1] for p in invext_dgm_tuples]
axs[1,1].scatter(invext_dgm_x, invext_dgm_y)
axs[1,1].plot([0,5],[0,5])
axs[1,1].set_title("Extended Persistence Diagram", fontsize=10)
axs[1, 1].set_xlabel("Birth")
axs[1, 1].set_ylabel("Death")
plt.show()

arr = np.array(invext_dgm)
diagram = [(arr[i][0], arr[i][1]) for i in range(arr.shape[0])] # Convert
    the array to a list of tuples
extended_diagram = [(1, (p[0], p[1])) for p in diagram]
```