Sapienza
Università di Roma

Bridging Topological Persistence and Machine Learning for Music Information Retrieval

Facoltà di Scienze Matematiche Fisiche e Naturali

Corso di Laurea Magistrale in Matematica

Candidate

Luca Sassone
ID number 1544447

Thesis Advisor	Co-Advisors
Prof. Marco Manetti	Dr. Mattia G. Bergomi
	Prof. Massimo Ferri

Academic Year 2020/2021

Bridging Topological Persistence and Machine Learning for Music Information Retrieval
Master's thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: sassone.1544447@studenti.uniroma1.it

Abstract

How can we leverage topological persistence to inject knowledge in machine learning algorithms, and in particular apply persistence-based pipelines to classification tasks, such as Music Information Retrieval? We propose a strategy based on persistent homology, and more specifically persistence images, to tackle the problem of music genre classification. We test our pipelines on GTZAN—a widely used, publicly available dataset—showing that persistence images improve a simple machine learning algorithm's classification performance. Thereafter, we introduce a complementary approach to persistent images for vectorizing persistence diagrams. The algorithm ranks the points of a persistence diagram, allowing us to consider only its n most relevant elements. This approach is complementary to persistent images: it generates lighter fingerprints and depends on fewer hyperparamenters. We focus mostly on the conceptual implications of the theories and methods mentioned above, rather than compare performance with state-of-the-art methods.

Contents

Chapter 1

Introduction

Topological persistence and persistent homology rapidly became a valuable tool in numerous applications, see [Fer17]. Indeed, persistent homology allows the user to extract specific features from data points $\{x_1, \ldots, x_n\}$ under the form of a continuous function $f : x_i \to \mathbb{R}$. On the other hand of the data-analysis spectrum, machine learning allows for *free* modelling of a dataset, given a task: the algorithm learns optimal parameters to solve a certain task (typically minimizing an error function) given an initial set of data points.

We review state-of-the-art methods and investigate original ways to merge these two approaches, aiming to maintain the control granted by topological persistence and flexibility of machine learning methods.

As a reference application, we consider the music genres classification task: an open problem currently investigated by the Music Information Retrieval (MIR) community, see [Dow03]. In our setting, music genres classification is a particularly relevant problem: on one hand, specific features need to be extracted from audio data, however time-frequency analysis is prone to noise, thus topological persistence can be leveraged to *forget* noisy components of such features. On the other hand, machine learning tools—ranging from simple regressions to deep neural networks—proved to be highly effective on audio-classification tasks.

In chapter 2, we introduce the music genres classification problem and the needed mathematical tools from signal analysis and topological persistence. In chapter 3 we shall compute persistence diagrams on audio features and transform them into persistent images to tackle the music genres classification problem selecting a standard machine learning (ML) algorithm. We shall compare the performance of such persistence-based pipeline with the naive application of the selected ML algorithm. Finally, in chapter 4, we will propose a cardinality reduction algorithm for persistence diagrams. We did not test this latest algorithm on the music genres classification task. However, as discussed in the conclusions of this work, we believe our solution could be complementary to persistent images and suitable for injecting knowledge in standard machine learning pipelines.

The Python package developed to validate the methods described in this work is available at https://github.com/luca9433/$_codice_tesi$.

Chapter 2

Preliminary notions

In this chapter, we introduce the preliminary notions concerning both the applied and theoretical sides of this work. We start by presenting an overview of the music genres classification problem. Then, we will give a more mathematically detailed description of the feature we want to extract from musical audio, and discuss the theoretical aspects which we will apply to select certain interesting topological properties of the extracted information.

2.1 Music genres classification

We can introduce the problem of music genres classification by citing the introduction to the article [TC02], where, twenty years ago, George Tzanetakis and Perry Cook, considering the ever-wider availability of music on the Web since then, emphasized the consequent increasing importance of structuring and organizing it:

"Musical genres are labels created and used by humans for categorizing and describing the vast universe of music. Musical genres have no strict definitions and boundaries as they arise through a complex interaction between the public, marketing, historical, and cultural factors. This observation has led some researchers to suggest the definition of a new genre classification scheme purely for the purposes of music information retrieval. However, even with current musical genres, it is clear that the members of a particular genre share certain characteristics typically related to the instrumentation, rhythmic structure, and pitch content of the music. Automatically extracting music information is gaining importance as a way to structure and organize the increasingly large numbers of music files available digitally on the Web. It is very likely that in the near future all recorded music in human history will be available on the Web. Automatic music analysis will be one of the services that music content distribution vendors will use to attract customers. Another indication of the increasing importance of digital music distribution is the legal attention that companies like Napster have recently received. Genre hierarchies, typically created manually by human experts, are currently one of the ways used to structure music content on the Web. Automatic musical genre classification can potentially automate this process and provide an important component for a complete music information retrieval system for audio signals. In addition, it provides a framework for developing and evaluating features for describing musical content. Such features can be used for similarity retrieval, classification, segmentation, and audio thumbnailing and form the foundation of most proposed audio analysis techniques for music. ”

The basis of any type of automatic audio analysis system is the extraction of feature vectors. Feature extraction is the process of computing a compact numerical representation that can be used to characterize a segment of audio. The design of descriptive features for a specific application is the main challenge in building pattern recognition systems. Once the features are extracted, standard machine learning techniques which are independent of the specific application area can be used.

2.1.1 State of the art

A large number of different feature sets have been proposed to represent audio signals. Typically, they are based on some form of time-frequency representation. Mel-frequency cepstral coefficients (MFCCs) are a set of perceptually motivated features that have been widely used in speech recognition, which automatic classification of audio has also a long history originating from, and are based on the fast Fourier transform (FFT). After taking the log-amplitude of the magnitude spectrum, the FFT bins are grouped and smoothed according to the perceptually motivated Mel-frequency scaling. Finally, a discrete cosine transform is performed to decorrelate the resulting feature vectors.

MFCCs provide a compact representation of the spectral envelope, such that most of the signal energy is concentrated in the first coefficients.

Audio classification techniques that include non-speech signals have also been proposed. Most of these systems target the classification of broadcast news and video in broad categories like music, speech, and environmental sounds. For example, [Rob+19] presents an analysis of a multiclass classification problem to identify queenless states by monitoring bee sound in two possible cases: a strong and healthy colony that looses its queen and a reduced population queenless colony. Extracting features by MFCCs and using a Lasso Logistic model for feature selection and regularization, the authors show that is possible to detect the queenless state in both cases: queenless or healthy colonies can generate slightly different patterns and the data clusters of the same condition tend to be close.

More in general, the features used in this kind of works are statistics (mean, variance, autocorrelation) over the whole sound file of short-time features such as pitch, amplitude, brightness, and bandwidth. However, these do not directly attempt to model musical signals and therefore they are not suitable for the automatic classification of musical genres. For example, for this aim, we also need information on the rhythmic structure of music.

Research in the areas of automatic beat detection and multiple pitch analysis can provide ideas for the development of novel features specifically targeted to the analysis of music signals. In [Sch98], Scheirer describes a real-time beat tracking system for audio signals with music. In this system, a filterbank is coupled with a network of comb filters that track the signal periodicities to provide an estimate of the main beat and its strength.

In much more recent times, Castellon, Donahue, and Liang demonstrate in [CDL21] that language models pre-trained on codified (discretely-encoded) music audio learn representations that are useful for downstream music information retrieval (MIR) tasks. Specifically, they explore representations from a music generation system containing a language model trained on codified audio from 1M songs. To determine if this system's representations contain useful information for MIR, they use them as input features to train shallow models on several MIR tasks. Relative to representations from conventional MIR models which are pre-trained on tagging, they find that using representations from their system as input features yields 30% stronger performance on average across four MIR tasks: tagging, genre classification, key detection, and emotion recognition. In particular, tagging involves determining which tags from a fixed set of tags apply to a particular song. Categories of tags include the genre (e.g., jazz), instrumentation (e.g., violin), emotions (e.g., happy), and characteristics (e.g., fast); genre classification involves assigning the most appropriate genre from a fixed list for a given song. They report accuracy on the GTZAN dataset [TC02], which contains 30-second clips from ten distinct genres. They note that this task has a high degree of overlap with tagging, as tagging datasets typically have several genres within their tag vocabulary. In fact, seven of ten genres in GTZAN are present in the tag list of the tagging dataset they use.

Getting closer to our particular point of view on the classification problem, we can find enlightening examples of applications of topological tools to classification problems in music in Mattia Bergomi's works: already in his Ph.D. thesis [Ber15], and later in [BBD16], he proposes a strategy to describe some music features as a polyhedral surface obtained by a simplicial interpretation of the Tonnetz, a graph largely used in computational musicology to describe the harmonic relationships of notes in equal tuning. In particular, he uses persistent homology in order to describe the persistent properties of music encoded in that model. The task of automatic music style classification is addressed by computing the hierarchical clustering of the topological fingerprints associated with some collections of compositions.

2.1.2 Suggested solution

Carrying out the task of processing extracted data, as in our case, from audio tracks, there are conceptually two possible ways to follow: the first one is to classify and predict. In our case this approach has certainly, as we will see, a solid theoretical basis which we will see confirmed by the results, but it presupposes the use of very heavy objects (such as persistence diagrams and persistence images) which often involve many parameters and, from a computational point of view, they can require a lot of memory.

The second one is to operate a cardinality reduction selecting point from persistence diagrams based on certain appropriate criterions. This certainly involves a loss of information but it can have a gain in classification: if the selection criterion is good enough, you can aim as much as possible to eliminate noisy data, leaving only the signal. Object such as persistence diagrams, obtained from certain filtrations of the extracted feature, such as MFCCs, can undergo a cardinality reduction and then be compared with each other.

2.2 Mel-Frequency Cepstral Coefficients

Mel-Frequency Cepstral Coefficients (MFCCs) are short-term spectral-based features, widely used in automatic speech and speaker recognition. They were introduced in [DM80] in the 1980s, and have been state-of-the-art ever since. There are also several examples of authors who have tried to apply them in the musical field, e.g. Beth Logan, in [Log00], investigates the appropriateness of using the Mel-frequency scale to model the musical spectra and the *Discrete Cosine Transform* (DCT) to decorrelate the Mel-spectral vectors. DCT is the most common linear, invertible function $\mathbb{R}^N \to \mathbb{R}^N$ (i.e. invertible $N \times N$ matrix) that provides for spatial compression, capable of detecting changes in the information on an area and the contiguous one of a digital image, neglecting repetitions; the function that supports temporal compression is instead entrusted to a special "motion vector", which identifies the dynamic components while leaving out the static ones.

We assume that an audio signal is, on short time scales, statistically stationary. This is why we frame the signal 20-40ms frames. Shorter frames would not yield reliable spectral estimates. Longer signals would change too much throughout the frame.

The next step is to calculate the power spectrum of each frame. The power spectrum of a time series describes the distribution of power into frequency components composing that signal. According to Fourier analysis, any physical signal can be decomposed into a number of discrete frequencies, or a spectrum of frequencies over a continuous range. The statistical average of a certain signal or sort of signal (including noise) as analyzed in terms of its frequency content, is called its spectrum. When the energy of the signal is concentrated around a finite time interval, especially if its total energy is finite, one may compute the energy spectral density. More commonly used is the power spectral density (or simply power spectrum), which applies to signals existing over all time, or over a time period large enough (especially in relation to the duration of measurement) that it could as well have been over an infinite time interval. This step is motivated by the human cochlea (an organ in the ear) which vibrates at different spots depending on the frequency of the incoming sounds. Depending on the location in the cochlea that vibrates [1], different nerves fire informing the brain that certain frequencies are present. MFCCs perform a similar task, identifying which frequencies are present in the frame.

Let s be a signal in time-domain and $\{s_j\}_{j \in J}$ a windowing of s. The discrete Fourier transform (DFT) is:

$$DFT(s_j) = \sum_{n=1}^{N} s_j h e^{-i\frac{2\pi kn}{N}} \qquad 1 \leqslant k \leqslant K,$$

where h is an N samples long analysis window (e.g. hamming window), n ranges over the number of samples and K is the length of the DFT. The periodogram estimate of the power spectrum P_j for the frame s_j is defined taking the absolute value of the complex DFT and squaring the result:

$$P_j = \frac{1}{N}|DFT(s_j)|^2$$

[1]The reader can refer to [ABM12] for anatomic details.

The periodogram spectral estimate still contains a lot of information not required for MIR. In particular, the cochlea cannot discern the difference between two closely spaced frequencies, especially when they are high. For this reason, we take clumps of periodogram bins and sum them up to get an idea of how much energy exists in various frequency regions. This is performed by the Mel filterbank: the first filter is very narrow and indicates how much energy exists near 0 Hertz. As the frequencies get higher, filters get wider, thus less sensitive to small variations. We are only interested in roughly estimating how much energy is carried by the signal per frequency band. The tool that one can use to space filterbanks exactly and figure out how wide to make them is the Mel scale. It relates the perceived frequency, or pitch, of a pure tone to its actual measured frequency. Humans are much better at discerning small changes in pitch at low frequencies than they are at high frequencies. Incorporating this scale makes our features match more closely what humans hear. The formula for converting from frequency to Mel scale is:

$$M(f) = 1125 \ln(1 + f/700)$$

To go from Mels back to frequency:

$$M^{-1}(m) = 700(e^{m/1125} - 1).$$

We then take the logarithm of filterbank energies. This step is also motivated by human perception: if we mean *loudness* as the acoustic and psychoacoustic quality associated with the strength of a sound, determined by the pressure that the sound wave exerts on the eardrum, we do not hear it on a linear scale. Generally, to double the perceived volume of a sound we need to put 8 times as much energy into it. This means that large variations in energy may not sound as huge if the sound is loud to begin with. This compression operation makes our features match more closely what humans hear.

The final step is to compute the DCT of the log filterbank energies. It is defined, in one dimension, for an N samples long sequence $\{x_n\}_{n=0,\dots,N-1}$, as the linear, invertible function $X : \mathbb{R}^N \to \mathbb{R}^N$ given by [2]:

$$X_u = [DCT(\{x_n\})]_u = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi(2n+1)u}{2N}\right], \qquad for\ u = 0, \dots, N-1.$$

This step is performed because filterbanks are overlapping, so filterbank energies are correlated with each other. The DCT decorrelates them, but notice that only some of the 26 DCT coefficients are kept because the higher DCT coefficients represent fast changes in the filterbank energies and it turns out that these fast changes degrade MIR performance, so we get a small improvement by dropping them. The resulting features are called Mel-Frequency Cepstral Coefficients (MFCCs).

Therefore, we can think of MFCCs as something analogous to Fourier coefficients with respect to a Fourier basis. In analogy with the Fourier transform, MFCCs transform a signal expressed as a function of a variable in the time-domain into a quantity expressed as a function of a variable in the frequency-domain. Thus we can represent them in time-frequency coordinates. For further details on calculating

[2]See [Kha03] for more details.

MFCCs, the reader can consult [Lav20], from which we borrowed heavily for this section. In [Log00], the steps of calculation of MFCCs are discussed, seeking to determine if the process is suitable for creating features to model music. In the next section, we will introduce the tools that will allow us to specify in what sense we want to do that.

2.3 Introduction to Persistent Topology

We largely borrow this section from [Fer17].
If one aims to analyze and classify images of rigid objects, linear algebra and geometry satisfy almost all needs. However, the rigidity of geometry could be an obstacle when studying, for instance, natural images. Even harder is to apply geometry-based analysis to biomedical data or, as in our case, to feature extracted from audio tracks. Topology seems to be better suited to our needs, due to its flexibility: in many cases, it is possible to find a homeomorphism between two objects, which superimposes one to the other, whereas no invertible matrix will ever be able to do that. Algebraic topology is helpful to discover when objects are not homeomorphic, associating invariants to topological spaces, which are not homeomorphic if corresponding invariants are not identical. But topology also has a limit: if geometry is too rigid, topology is too free. A middle ground between the two approaches is persistent topology, which yields topological descriptors preserving some selected geometric features through filtering functions. A filtering function, i.e. a real-valued continuous function on a topological space, represents the point of view according to which we want to compare two objects. The idea behind persistent topology is to associate the concept of shape not only with a topological space, but with a pair topological space, filtering function: to compare two objects X and Y, persistent homology does not analyze X and Y simply as they are as topological spaces, but considers the two pairs (X, f) and (Y, g), where f and g can be adapted from time to time in order to capture certain interesting features of the topological spaces.

2.3.1 Basic notions

Now, we define the principal concepts in persistent topology, which we shall use in the remainder of this work.

Definition 2.3.1. *Let X and Y be topological spaces and $\varphi : X \rightarrow Y$ a function; φ is said to be a homeomorphism from X to Y if it is continous, invertible and its inverse is also continous. If φ exists with these characteristics we say that X and Y are homeomorphic.*

For each non-negative integer k, the k-th homology group $\mathcal{H}_k(X)$ of a topological space X is a powerful homeomorphism invariant. [3]

Definition 2.3.2. *For each k non-negative integer, rank $\mathcal{H}_k(X)$ is called the k-th Betti number of the topological space X. We denote it with $\beta_k(X)$.*

[3]The reader can refer to [Rot13] or to any introductory text to algebraic topology for a valid theoretical introduction to homology.

Intuitively, $\beta_0(X)$ counts the number of path-connected components of X, $\beta_1(X)$ counts its 1-dimensional holes (independent 1-cycles), $\beta_2(X)$ counts its 2-dimensional holes (independent 2-cycles).

Definition 2.3.3. *Let X be a topological space. A filtering function on X is a continous function $f : X \to \mathbb{R}$.*

Definition 2.3.4. *Let us consider the pair (X, f), where X is a topological space and $f : X \to \mathbb{R}$ is a filtering function. The sublevel set relative to a given $l \in \mathbb{R}$ is the set $X_l = \{x \in X | f(x) \leqslant l\}$.*

From now on, we will indicate with k a non-negative integer and with (X, f) the pair topological space, filtering function.

Definition 2.3.5. *For all $u, v \in \mathbb{R}, u < v$, the inclusion function $i^{u,v} : X_u \to X_v$ is continous and induces, at each degree k, a group homomorphism $i_*^{u,v} : \mathcal{H}_k(X_u) \to \mathcal{H}_k(X_v)$. k-Persistent Betti number (k-PBN) functions assign to a pair (u, v) the number rank($Im\ i_*^{u,v}$), which we also are going to indicate with $\beta_k^{u,v}(X, f)$, namely the number of k-homology classes of $\mathcal{H}_k(X_u)$ which "survive" in $\mathcal{H}_k(X_v)$ [4].*

According to this notation, u and v represent the levels of "birth" and "death" of a generator, respectively. (u, v) is a point in the plane which we call a *cornerpoint*. If a generator never dies and w is the level of its "birth", we are going to represent it with a vertical line at the abscissa w, which we call a *cornerline* (or, often, *cornerpoint at infinity*). The *persistence* of a cornerpoint (u, v) is $v - u$ and a cornerline is a cornerpoint with an infinite persistence.

Definition 2.3.6. *A persistence module is given by:*

- *a closed, discrete and lower-bounded subset of real numbers:*

$$T = \{u_0 < u_1 < u_2 < \dots\} \subset \mathbb{R}.$$

- *a sequence of abelian group homomorphisms:*

$$0 \xrightarrow{p_{-1}} P(u_0) \xrightarrow{p_0} P(u_1) \xrightarrow{p_1} P(u_2) \xrightarrow{p_2} \dots.$$

According to the sense of the definition above, the sequence of group homomorphisms $i_*^{u,v}$ in Definition 2.3.5 defines a persistence module, with $T = \{\dots < u < v < \dots\}$.

Definition 2.3.7. *The k-th persistence diagram $\mathcal{D}_k(X, f)$ consists of cornerpoints and cornerlines which characterize the k-PBN function.*

More precisely, we have to consider cornerpoints and cornelines with their multiplicity. For this purpose, we are going to introduce the following notation.

$$\Delta^+ = \{(u, v) \in \mathbb{R}^2 | u < v\}, \qquad \Delta = \{(u, v) \in \mathbb{R}^2 | u = v\}, \qquad \bar{\Delta}^+ = \Delta^+ \cup \Delta$$

[4] A classical reference for these concepts is, for instance, [EH10]

Definition 2.3.8. *The multiplicity $\mu_k(u, v)$ of a cornerpoint $(u, v) \in \Delta^+$ is the finite and non-negative number defined by the following limit:*

$$\lim_{\epsilon \to 0^+} \beta_k^{u+\epsilon, v-\epsilon}(X, f) - \beta_k^{u-\epsilon, v-\epsilon}(X, f) - \beta_k^{u+\epsilon, v+\epsilon}(X, f) + \beta_k^{u-\epsilon, v+\epsilon}(X, f).$$

Remark 2.3.1. *We are considering all cornerpoints laying in the square with center (u, v) and side 2ϵ, counting how many cornerpoints are in that square when ϵ tends to 0 (in the expression we are subtracting two times $\beta_k^{u-\epsilon, v+\epsilon}(X, f)$ from $\beta_k^{u+\epsilon, v-\epsilon}(X, f)$ with the second and the third term; so we have to add again it one time, and this explains the presence of the fourth term in the expression).*

We can analogously define the multiplicity of a cornerline.

Let us now give another definition of a persistence diagram.

Definition 2.3.9. *The k-th persistence diagram $\mathcal{D}_k(X, f)$ is the union of the set of all points $(u, v) \in \Delta^+$ such that $\mu_k(u, v) > 0$, considered with their multiplicities, and points in Δ, considered with infinite multiplicity. Points of a persistence diagram belonging to Δ^+ are often called proper points.*

Definition 2.3.10. *Given the pairs (X, f) and (Y, g) such that the respective persistence diagrams $\mathcal{D}_k(X, f)$ and $\mathcal{D}_k(Y, g)$ have a finite number of points $(u, v) \in \Delta^+$ whose multiplicity $\mu_k(u, v)$ is greater than 0, match the cornerpoints of $\mathcal{D}_k(X, f)$ either with cornerpoints of $\mathcal{D}_k(Y, g)$ or with their own projections on the "diagonal" Δ; the weight of this matching is the upper bound of the L^∞-distances of matching points. The matching distance (or Bottleneck distance) of $\mathcal{D}_k(X, f)$ and $\mathcal{D}_k(Y, g)$ is the lower bound of such weights among all possible such matchings.*

Using mathematical formalism, the Bottleneck distance can be defined as follows:

$$d_B(\mathcal{D}_k(X, f), \mathcal{D}_k(Y, g)) = \inf_{\sigma} \sup_{P \in \mathcal{D}_k(X,f)} \hat{d}(P, \sigma(P))$$

with σ moving among all possible bijections between $\mathcal{D}_k(X, f)$ and $\mathcal{D}_k(Y, g)$, and where, given $(u, v) \in \mathcal{D}_k(X, f)$ and $(u', v') \in \mathcal{D}_k(Y, g)$,

$$\hat{d}((u, v), (u', v')) = \min\left\{\max\{|u - u'|, |v - v'|\}, \max\left\{\frac{v - u}{2}, \frac{v' - u'}{2}\right\}\right\}$$

Definition 2.3.11. *Given two pairs (X, f), (Y, g), with X and Y homeomorphic, the weight of a given homeomorphism $\varphi : X \to Y$ is $\sup_{x \in X} |g(\phi(x)) - f(x)|$. The natural pseudo-distance of (X, f) and (Y, g) is the lower bound of these weights among all possible homeomorphisms. If the k-persistence diagrams of the two pairs are given, their matching distance is a lower bound for the natural pseudo-distance of the two pairs, and it is the best possible obtainable from the two k-persistence diagrams.*

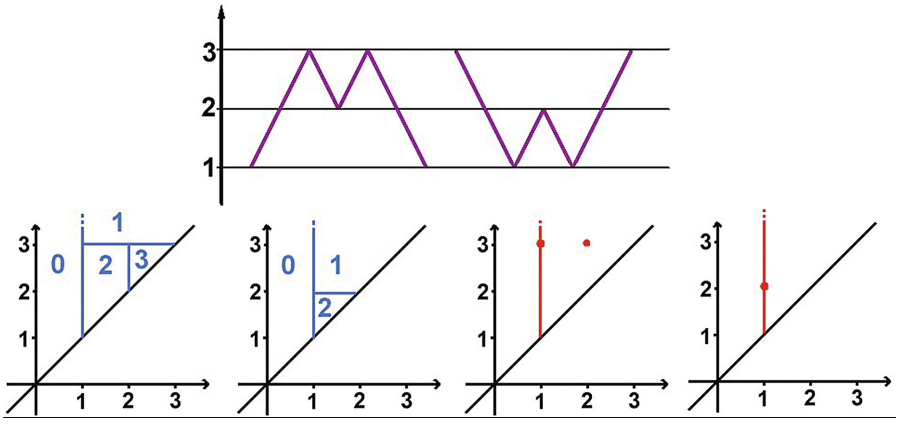


Figure 2.1. 0-PBN functions (on the left) e 0-persistence diagrams (on the right) for M and W.

Example Now, we can try to visualize the concepts we introduce in the previous section comparing two homeomorphic objects, the letters M and W, seen as topological spaces, using "height" as a filtering function, that is, operating a *sublevel sets filtration* on the two spaces, which we indicate with the same letters they represent. If we think of them, for instance, as in fig. 2.1, we can get from each other with a rigid motion, then they are certainly isomporphic and their Betti numbers are equal to each other:

$$\beta_k(M) = \beta_k(W) = \begin{cases} 1 & if\ k = 0 \\ 0 & if\ k = 1 \end{cases}$$

We now want to determine the 0-PBN functions and 0-persistence diagrams for M e W. So, we focus on the path-connected components of their sublevel sets as the height varies.
For M, at height $l = 1$ two connected components arise and persist for heights $l < 3$. A third one arises at height $l = 2$ and merges with the others at height $l = 3$, when only one component persists.
For W, at height $l = 1$ two connected components arise, persist for heights $1 \leqslant l < 2$ and merge at height $l = 2$, when only one component persists.

Thus, the different 0-PBN functions and 0-persistence diagrams we obtained, see fig. 2.1, associated with the sublevel sets filtration of M and W, allow one to distinguish the two objects, which are indistinguishable from a purely topological point of view.

2.4 Persistence Images

The introduction of persistence images in [Ada+17] is due to the need to solve the problem which is summarized in the following statements:

"How can we represent a persistence diagram so that:
(i) the output of the representation is a vector in $\mathbb{R}^n$,
(ii) the representation is stable with respect to input noise,
(iii) the representation is efficient to compute,

(iv) the representation maintains an interpretable connection to the original persistence diagram, and
(v) the representation allows one to adjust the relative importance of points in different regions of the persistence diagram."

It is possible to construct a finite-dimensional-vector representation of a persistence diagram called a **persistence image** (PI). We first map a persistence diagram B to an integrable function $\rho_B : \mathbb{R}^2 \to \mathbb{R}$ called a **persistence surface**. The surface ρ_B is defined as a weighted sum of probability density functions, typically Gaussian functions, one centered at each point in B. The idea of persistence surfaces has already appeared even before the development of persistent homology in [Fer+98] and [DFL98]. Taking a discretization of a subdomain of ρ_B defines a grid. A persistence image, i.e., a matrix of pixel values, can be created by computing the integral of ρ_B on each grid box. This PI is "vectorization" of the persistence diagram, and provides a solution to the problem statement above, as explained in detail in [Ada+17].

Precisely, let B be a persistence diagram in birth-death coordinates. Let $T : \mathbb{R}^2 \to \mathbb{R}^2$ be the linear transformation $T(u,v) = (u, v-u)$, and let $T(B)$ be the transformed set in birth-persistence coordinates, where each point $(u,v) \in B$ corresponds to a point $(u, v-u) \in T(B)$.

Let $\phi : \mathbb{R}^2 \to \mathbb{R}$ be a differentiable probability distribution with mean $m = (m_u, m_v) \in \mathbb{R}^2$. In applications, this distribution is usually chosen to be the normalized symmetric Gaussian $\phi_m = g_m$ with mean m and variance σ^2, defined as

$$g_m(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{[(u-m_u)^2+(v-m_v)^2]}{2\sigma^2}}.$$

We fix a non-negative weighting function $f : \mathbb{R}^2 \to \mathbb{R}$ that is zero along the horizontal axis, continuous, and piecewise differentiable. With these ingredients, we transform the persistence diagram into a scalar function over the plane.

Definition 2.4.1. *For B a persistence diagram, the corresponding persistence surface $\rho_B : \mathbb{R}^2 \to \mathbb{R}$ is the function*

$$\rho_B(z) = \sum_{w \in T(B)} f(w)\phi_w(z)$$

The reason for choosing differentiable distributions is we want to "smooth out" any "jumps" between values assumed on different points of B, blurring the transition between values around different cornerpoints.

The weighting function f is critical to ensure the transformation from a persistence diagram to a persistence surface is stable: for the proof, we refer the reader to [Ada+17]. Finally, the surface $\rho_B(z)$ is reduced to a finite-dimensional vector by discretizing a relevant subdomain and integrating $\rho_B(z)$ over each region in the discretization. In particular, we fix a grid in the plane with n boxes (pixels) and assign to each the integral of ρ_B over that region.

Definition 2.4.2. *For B a persistence diagram, its persistence image is the collection of pixels*

$$I(\rho_B)_p = \iint_p \rho_B(u,v)\, dv\, du$$

When generating a PI, the user makes three choices: the resolution, the distribution (and its associated parameters), and the weighting function. A strength of PIs is that they are flexible; a weakness is that these choices are non-canonical. The interested reader in a more in-depth discussion of these aspects can refer to [Ada+17].

Chapter 3

A Persistence-Image-based approach for Music Information Retrieval

The GTZAN dataset [TC02] is a widely used dataset for validating music genre recognition algorithms. We relied on MFCCs (see section 2.2) as audio descriptors. Then, building on top of the MFCC-based representation, we leveraged Topological Data Analysis (TDA) and Persistent Homology (PH) to create more robust and concise descriptors.

In this chapter, we describe how we extracted and analyzed features from audio tracks of the GTZAN dataset. First we analyzed the extracted information performing a sublevel sets filtration on MFCCs and compute their 0-persistence diagrams (PDs). The filtration is performed, as in section 2.3.1, using *height* as a filtering function, based on the number of independent 0-cycles of MFCCs' sublevel sets, that is, the number of path-connected components. After computing the PDs, we considered the corresponding PIs, see section 2.4, and applied appropriate machine learning techniques to learn how to appropriately associate a class (music genre) to a given PI.

Let us first give an overview of the GTZAN, before describing in detail TDA computations and machine learning techniques we applied for our classification purposes.

3.1 The GTZAN dataset

The gtzan8 audio dataset [TC02] contains 1000 tracks of 30-second length. Tracks are labelled according to ten genres, each containing 100 tracks which are all 22050Hz Mono 16-bit audio files in .wav format. The genres are blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, rock.

The dataset was introduced in 2002 and appears in at least 100 published works. It is the most-used public dataset in machine listening research for music genre recognition. Though its use is so widespread, GTZAN has always been missing metadata identifying its contents, until [Stu12] provided them for the first time.

In [Stu13] a partial reconstruction of GTZAN's excerpts is provided. Furthermore, in the same article, the ways people describe the music or artist of each excerpt are surveyed, querying the application programming interface provided by last.fm, and retrieving the tags.

The reader is referred to [Stu13] for a detailed discussion of limits and faults of GTZAN for MIR, particularly for music genre recognition.

Data folders

- "genres original". A collection of ten genre folders with 100 audio files each, all having a length of 30 seconds.
- "images original". For each audio file, a visual representation of the corresponding Mel spectrogram.
- Two CSV files containing features of the audio files. One file has for each song mean and variance computed over multiple features that can be extracted from an audio file. The other file has the same structure, but each song is split into three-seconds audio files.

3.2 PIs computation

We now describe in detail the steps we performed to extract features from the GTZAN's tracks, analyze them with TDA tools, and classify the TDA-based signatures with machine learning methods. See https://github.com/luca9433/$_codice_tesi$ for Python code.

Feature extraction The first step concerns the extraction of MFCCs from audio tracks. We used the Python library *Librosa* [McF+21], which provides functions generating MFCCs from an audio signal. For each track we set the sample rate parameter to the value we read by loading the sampling rate with librosa itself, and we set to 128 the number of MFCCs to be computed.

Librosa also allows one to visualize the input data. In fig. 3.1 we report a visualization of MFCCs of a track from the "blues" folder of GTZAN dataset.

Sublevel sets filtrations and 0-PDs Once we obtained MFCCs of all the audio tracks, we performed their sublevel sets filtrations, using the special function *lower_star_img* imported from the Python package *Ripser* [TSB18]. It returns the 0-PDs associated with the filtrations, stored as NumPy arrays with shapes $(-, 2)$. The machine interprets as infinity the ordinates of cornerlines or cornerpoints with very big death-coordinates. Therefore, in each PD it is necessary to replace these elements with cornerpoints having the same abscissa and ordinate equal to the maximum finite life +1 of all the cornerpoints in the diagram. So, given a cornerline - or a cornerpoint having a too much big death-coordinate - with abscissa w, we replaced it with the point

$$(w, max\{v : (u, v) \ is \ a \ proper \ cornerpoint\} + 1).$$

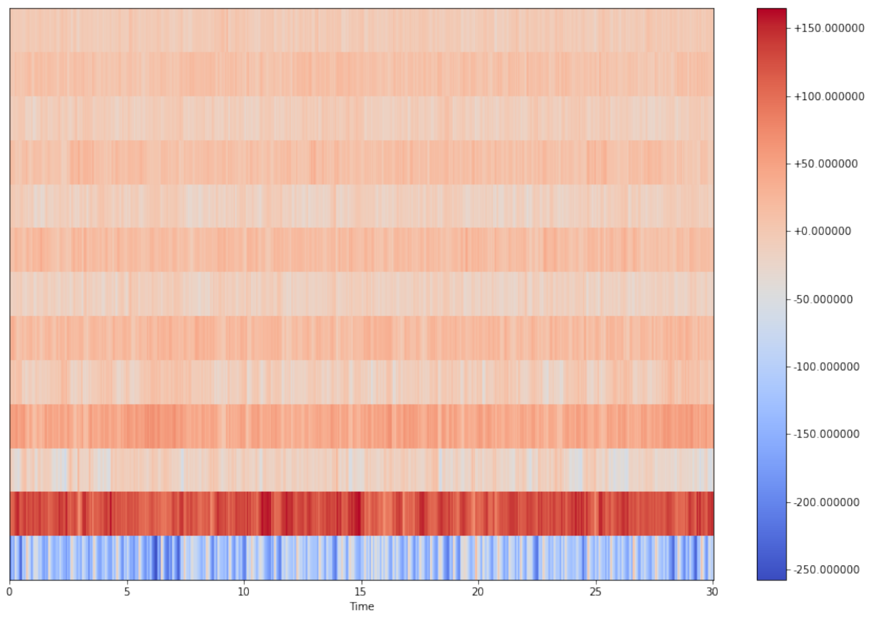


Figure 3.1. An example of visualization of MFCCs extracted from audio file *blues.00000* from the blues folder of GTZAN, recognized as *One Bourbon, One Scotch, One Beer* by John Lee Hooker. For better readability, we set to 13 the number of MFCCs.

PIs computation We then wanted to obtain the PIs corresponding to the 0-PDs, according to section 2.4. We computed them with the function *PersistenceImager* imported from the Python package *Persim*. It provides a *fit* method, which can be called on one or more arrays to automatically determine the minimum birth and persistence ranges needed to capture all persistence pairs. The ranges and resolution are automatically adjusted to accommodate the specified pixel size. Once we have 0-PDs associated with MFCCs' filtrations we can fit them and then generate PIs with the *trasform* method, which can be called on the fitted diagrams to generate the corresponding PIs.

PIs visualization Having fitted the 0-PDs before computing the corresponding PIs, their shapes are likely all the same or very similar to each other. However, we reshaped them to the shape of a PI with the minimum number of elements in each dimension of all computed PIs and we obtained PIs with shapes $(763, 768)$. We can thus think of PIs as points of a submanifold of $\mathbb{R}^{763\times 768}$. The UMAP algorithm for dimension reduction [MHM20] allowed us to visualize their projections on a plane, see fig. 3.3, to try to guess any regularity in their distribution with respect to how they are divided per genre. Before performing the projection, we need to flatten the arrays representing PIs with the *.flatten* method, that is, creating a copy of each array collapsed into one row. Then we can fit the flattened arrays with the UMAP *fit* method and transform them with the UMAP *transform* method for dimension

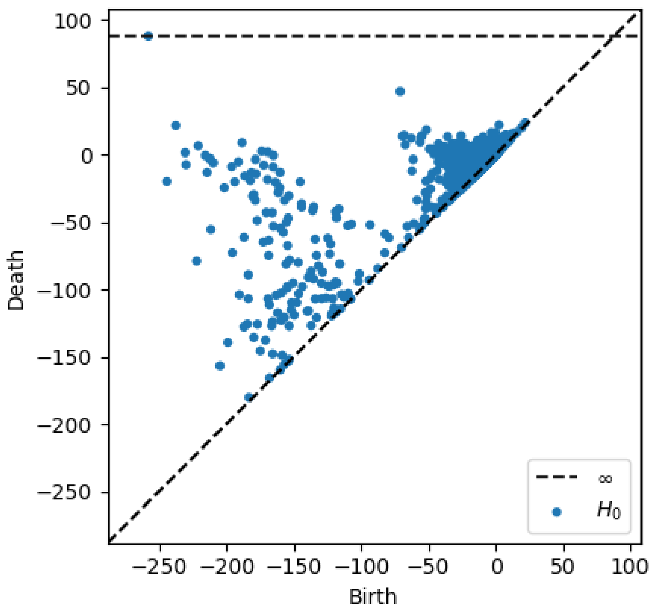


Figure 3.2. The 0-PD corresponding to the sublevel sets filtration of MFCCs extracted from the same audio file of fig. 3.1, but setting the number of MFCCs to 128.

reduction.

It is clear that a dimension reduction involves a loss of information, but it is however possible to establish some trends by observing the obtained image.

For instance, it is possible to notice in the middle upper part a higher concentration of points corresponding to the rock genre, while metal is particularly present and concentrated in the right part of the image. On the left side, there is a higher concentration of the classical genre. The disco genre would seem to prevail in the upper right.

3.3 Classification and cross-validation

We need to check the validity of our approach for MIR, in particular for music genre recognition, by trying an automatic classification with a machine learning pipeline, comparing the accuracy we obtain by applying it first to the starting dataset and then to the PIs.

For our automatic classification problem we used Support-Vector Machines (SVMs). SVMs are supervised learning methods used for classification and regression [Ped+11]. These robust prediction methods based on statistical learning frameworks were introduced in [BGV92] in the 1990s[1]

[1] A good reference for a brief introduction is also [EP99].

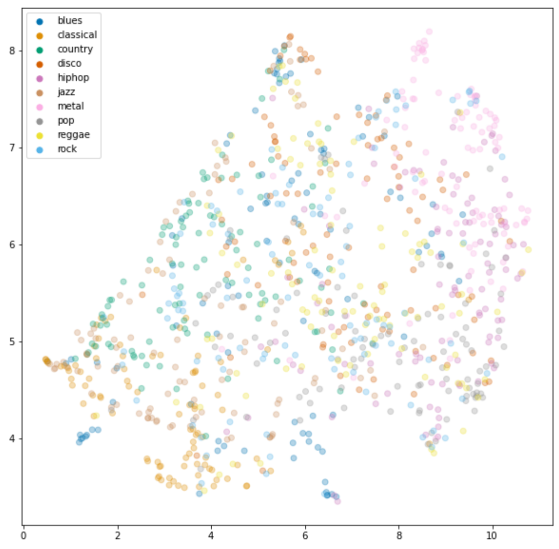


Figure 3.3. UMAP scatter plot of PIs' projection. Despite a considerable dispersion and overlapping of colors, it is possible to identify areas where certain colors are significantly more prevalent and concentrated than others.

The machine learning library [Ped+11] provides classes capable of performing binary and multi-class classification on a dataset. We used *SVC* (Support Vector Classification)[2].

To avoid overfitting, it is common practice, when performing a machine learning experiment, to hold out part of the available data as a test set X_{test}, y_{test}. The problem remains that the results may depend on a particular random choice for the pair of sets (train, validation). A solution to this problem is a procedure called cross-validation (CV). In the basic approach, called k-fold CV, the training set is split into k smaller sets and the following procedure is followed for each of the k "folds"[3]:

- A model is trained using $k - 1$ of the folds as training dataset;
- the resulting model is validated on the remaining part of the dataset (i.e., it is used as a testing dataset to compute a performance measure such as accuracy).

In particular, we used *Stratified k-fold* for cross-validation of SVC on our datasets. We set the number of folds to 5. See fig. 3.4 for a flowchart of the pipeline.

[2]For the mathemtical formulation the reader can also refer to [Ped+11].
[3]See [Ped+11] for a more in-depth discussion.

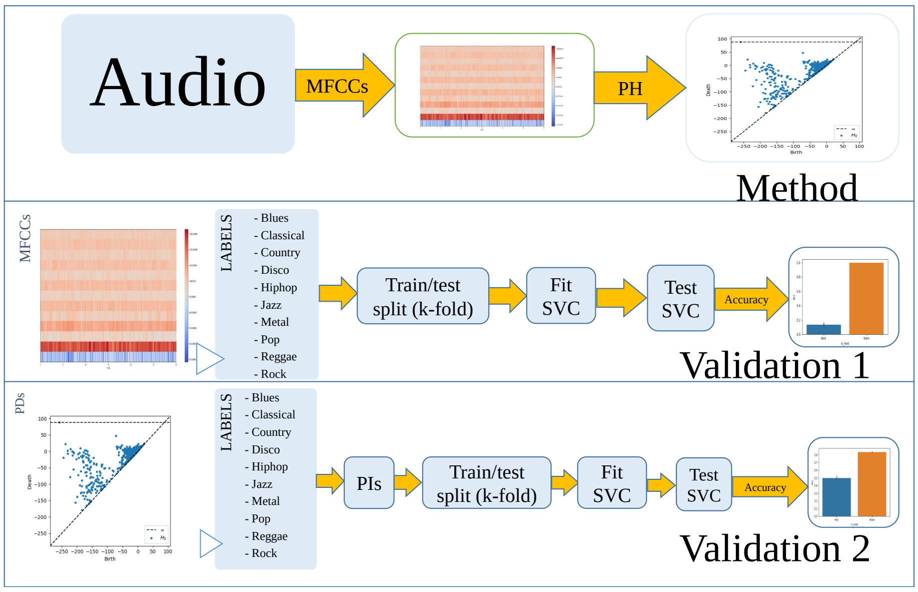


Figure 3.4. A flowchart showing the flow of the algorithm. In the upper part of the figure, the extraction of the MFCCs from the audio track and the passage to the corresponding persistence diagram is schematized. In the central and the lower part, the phases of the classification and validation process performed respectively on the raw MFCCs and then on the corresponding persistence images are schematized.

3.4 Results

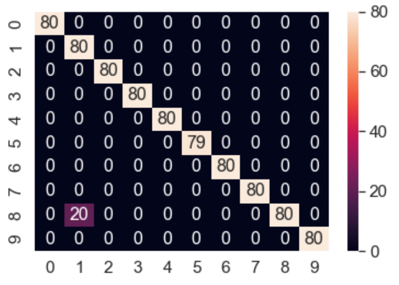
We report *confusion matrices* and *accuracies* we obtained applicating the model SVC both to MFCCs and then to PIs, with music genres as classes. In a confusion matrix, the columns correspond to the actual class and the rows to the predicted class. By definition, the element on row i and column j is the number of cases in which the classifier classified the "true" class i as class j.

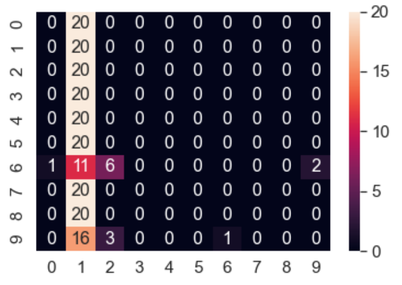
The accuracy is obtained as the number of classes that the classifier correctly predicts divided by the total number of predictions made. The *train accuracy* is the accuracy of a model on examples it was constructed on. The *test accuracy* is the accuracy of a model on examples it has not seen yet.

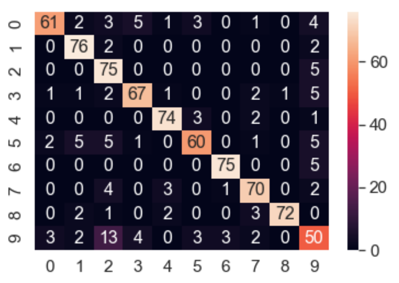
fig. 3.5 contains confusion matrices and bar plots reporting train and test accuracies for SVC performed on MFCCs and on PIs respectively.

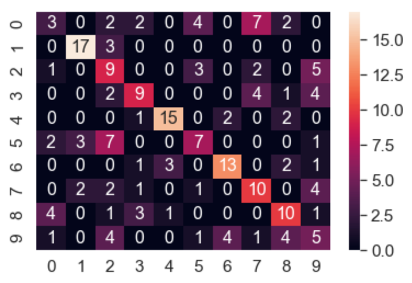
Against an extremely low accuracy (less than 20%) obtained simply performing the method SVC on MFCCs, the value rises significantly (around 50%) considering PIs.

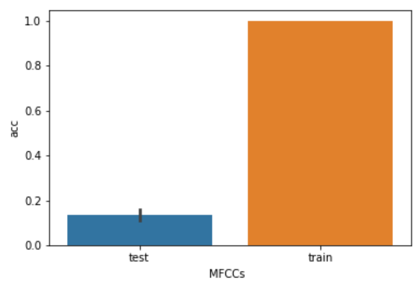
The significant improvement obtained in the test accuracy by performing a standard classifying method as SVC on PIs with respect to MFCCs is in favor of the validity of our theory: the functor that maps the image (MFCCs), thought as a topological space, to the persistence modules is necessary to extract information, which would otherwise be drowned in noise.


(a) A confusion matrix obtained for a training dataset of MFCCs.

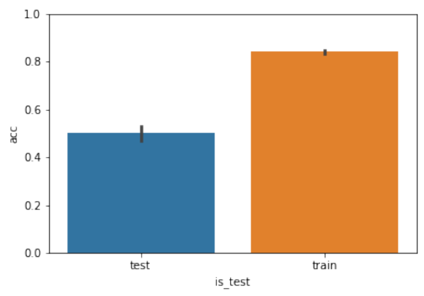

(b) A confusion matrix obtained for a testing dataset of MFCCs.


(c) A confusion matrix obtained for a training dataset of PIs.


(d) A confusion matrix obtained for a testing dataset of PIs.



(e) Accuracy in test and train of SVC performing on MFCCs.



(f) Accuracy in test and train of SVC performing on PIs.

Figure 3.5. figs. 3.5(a) and 3.5(b) report train and test confusion matrices respectively for the method SVC performed on MFCCs, using Stratified k-fold as dataset splitting method. They reveal a very low accuracy score (10 - 15%), which makes any attempt at classification impossible. By applying the same classification method to persistence images, however, accuracy scores increase significantly, as can be seen by observing confusion matrices in figs. 3.5(c) and 3.5(d) (setting to 5 the number of folds, the testing dataset has about 200 elements - 20 per genre - one fifth of the elements of the entire dataset), confirming values around 50%. Finally, barplots in figs. 3.5(e) and 3.5(f) provide an estimate of the central tendency for test and train accuracy of SVC through the 5 fits in which it is implemented on MFCCs and PIs respectively.

Chapter 4

A cardinality reduction algorithm for persistence diagrams

We are currently working on an alternative tool that we would like to apply to the MIR classification application described in chapter 3. In this chapter, we shall describe the algorithm and analyze some preliminary results.

Instead of considering the persistence images corresponding to PDs, we consider for each diagram D a subdiagram D' of relevant cornerpoints selected through an algorithm dubbed Ziggurat, that was firstly introduced in [Gur21]. The essence of this algorithm is to provide a more compact version of a given persistence diagram ranking and then selecting the most *relevant* cornerpoints. Intuitively and on the computational side, we aim to reduce the number of cornerpoint, this would allows for taking advantage of the powerful machine learning tools that, as mentioned in chapter 3, need vector of fixed length as input. On the geometrical side, the Ziggurat algorithm acts as to restore symmetry lost during noisy data acquisition.

4.1 Cornerpoints selection

Given a pair (X, f), k-persistent Betti numbers functions (k-PBN) and k-persistence diagrams (k-PD) have been defined in chapter 2. These mathematical tools are very useful in applications, because they yield a concise signature that is representative of the pair (X, f), i.e. respectful of the data structure X, and the features the user desires to stress f.

Persistence diagrams are multisets whose elements are cornerlines and cornerpoints. We recall that a cornerpoint's coordinates are the moments of birth and death of homological classes computed considering the filtration induced by $f : X \to \mathbb{R}$. However, not all cornerpoints have the same relevance: if the starting object is complex—e.g., of medical nature—part of the cornerpoints could be associated to noisy features. Experimental evidence suggests that, in most cases, such cornerpoints are very close to the diagonal, although this is not always the case: it is possible that the presence of apparently noisy cornerpoints is the result of noisy data acquisition. For this reason, it is essential to devise algorithms to rank cornerpoints assigning to each one a certain degree of relevance, to obtain a hierarchical classification of all points.

Consider fig. 4.1(a), the filtered object X is represented on the left of the figure and the corresponding 0-persistence diagram on the right. The persistence diagram is obtained through the filtering function "height". The *elderly rule* is used to construct the diagram, that is, a 0-cycle (path-connected component) dies only when it merges with an older 0-cycle, i.e. a cycle that was born earlier throughout the filtration. In the figure, the absolute minimum of X causes a 0-cycle to be born. This first cycle, being the first to be born, will never die, thus creating, in the figure, the cornerline with abscissa 0. Continuing along the filtration, the second 0-cycle is born at height 1 and it dies at 23. In particular, the 0-cycle represented by the cornerpoint c arises at height 10 and, immediately after, the one represented by the cornerpoint d is born at height 11. In this area of the starting object, we can notice a sudden alternation of maximum and minimum peaks, which could be caused by noisy data acquisition. Analyzing better the trend followed by X, the second oscillation is negligible compared to the first, at least in terms of amplitude. At the level of the persistence diagram, this means that the cornerpoint d should be considered *less important* than c.

4.2 Ziggurat

Definition 4.2.1. *Let D be a persistence diagram with a finite set of cornerpoints. Let us assume that all points on the diagonal belong to D. To make the definition simpler, add a cornerpoint at $(-\infty, +\infty)$. Let us define the function legacy $\eta : \mathbb{R}^2 \to \mathbb{R}$ such that*

$$\eta((u, v)) = max\{\overline{u} - \overline{v} \ |(\overline{u}, \overline{v}) \in D, \overline{u} \leqslant u, \overline{v} \leqslant v\}$$

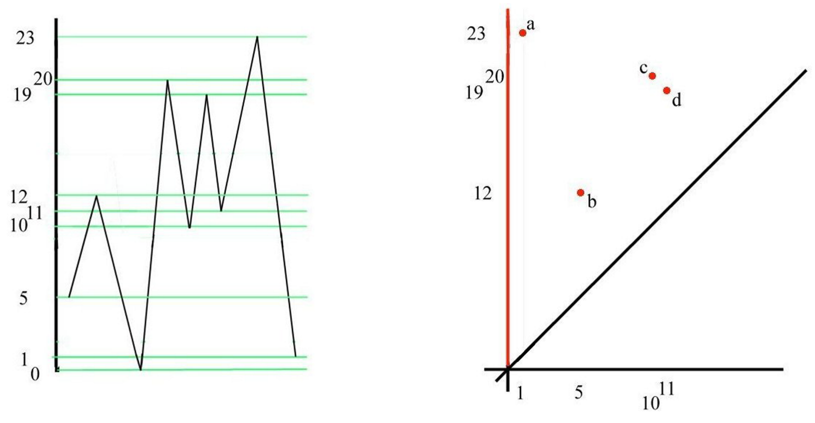
Definition 4.2.2. *For D a persistence diagram as above and η legacy, Ziggurat is defined as*

$$Z_D = \{(u, v, w) \in \mathbb{R}^3 | \eta((u; v)) > -\infty, w \leqslant \eta((u, v))\}$$

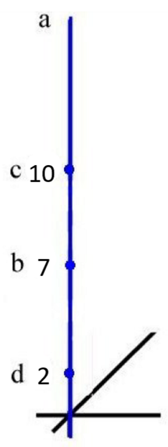
.

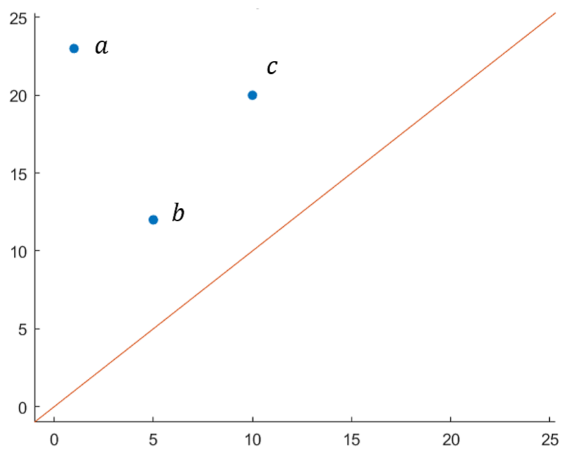
Before giving the selection algorithm based on the Ziggurat's construction, it is useful to give a more qualitative explanation of the Ziggurat's structure. It develops in the three-dimensional space $\mathbb{R}^3$ and it is constructed starting from a persistence diagram D, which is instead represented in the two-dimensional space $\mathbb{R}^2$. See fig. 4.2 for a representation of the Ziggurat. Each cornerpoint of with coordinates (u, v) has positive persistence $p = v - u$. To represent the Ziggurat graphically, a prism is associated to each cornerpoint. The upper base of each prism is defined by the triangle generated starting from the vertical and horizontal extensions on the bisector of the first quadrant of each cornerpoint. The upper base of the prisms is positioned at a height $-p$. Every prism has depth extending to $-\infty$. Referring to fig. 4.2,

the cornerpoint with the highest persistence is the one corresponding to the cornerline of abscissa 2, whose coordinates are $(2, 15)$: for computational purposes the height of cornerlines is set to $\max_{c \in D} v_c - u_c + 1$ where c ranges across all cornerpoints in D. This cornerpoint has a persistence $15 - 2 = 13$, and therefore the maximum depth of all prisms is $(13 + 1) = 14$.



(a)





(b) (c)

Figure 4.1. The Ziggurat-based ranking and selection procedure. Panel (a) Image credits: [Gur21]. Initial object (on the left) and the corresponding 0-PD through the filtering function "height" (right). Panel (b) Image credits: [Gur21]. 0-persistence diagram of the pair $(Z_D, u - v - w)$ where D refers to the diagram in fig. 4.1(a). Panel (c) Image credits: [Gur21]. Selected cornerpoints with Kurlin's rule from the diagram on the left side of fig. 4.1(a).

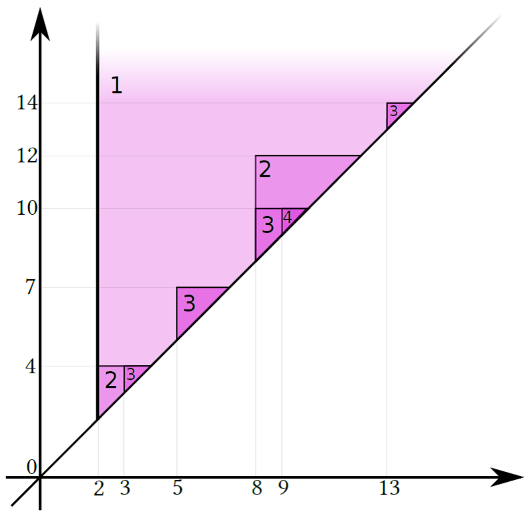


Figure 4.2. Persistence Diagram D. Image credits: [Gur21].

An additional triangular prism is also added at 0, with the upper base constructed starting from the bisector of the first quadrant. In figs. 4.3 to 4.5 a construction of the Ziggurat corresponding to the persistence diagram represented in fig. 4.2, performed by Davide Gurrieri in [Gur21].

Let D be a persistence diagrams as in the definitions above, and Z_D the Ziggurat associated with D. We can now consider the 0-persistence diagram $\mathcal{D}$ obtained by considering the pair (Z_D, f_{Z_D}), where $f_{Z_D}(u,v,w) = u - v - w$, see Definition 4.2.2.

Figure 4.6 shows the front view of the Ziggurat cut from the plane corresponding to the level $k = 4$ of the filtering function $u - v - w$.

4.2.1 Cornerpoints selection in the Ziggurat's persistence diagram

As mentioned previously, the idea behind the cornerpoints selection process in a persistence diagram D is to use the pair (Z_D, f_{Z_D}), where $f_{Z_D}(u,v,w) = u - v - w$, which, for a given k value, defines an oblique plane $u - v - w = k$. In particular, for $k = 0$, we obtain a plane intersecting all the prisms vertices of the Ziggurat Z_D. The 0-persistence diagram $\mathcal{D}$ generated by the pair (Z_D, f_{Z_D}) can be used to obtain a relevance ranking of D's cornerpoints. As $k > 0$, the plane defined by $u - v - w = k$ moves parallel to itself, intersecting the prisms of the Ziggurat. Initially, the intersection generates n pyramids, each corresponding to one of the n cornerpoints of D. As k increases, pyramids start to merge with each other. Unlike the example in fig. 4.1(a), in this filtration pyramids are all born at the same time. In fact, for $k = 0$, we have $u - v - w = 0$, hence $w = u - v$, which is the persistence of the cornerpoint (u, v) changed in sign. Therefore, the plane intersects the prism associated with the cornerpoint (u, v) at the point with coordinates $(u, v, u - v)$, corresponding by construction to the vertex of the upper base of the prism lying on the "vertical" line of (u, v) itself, which is the vertex of the pyramid corresponding to (u, v). Therefore, it is not possible to use the elderly rule we introduced previously. It was therefore decided to consider *older* pyramids associated with cornerpoints

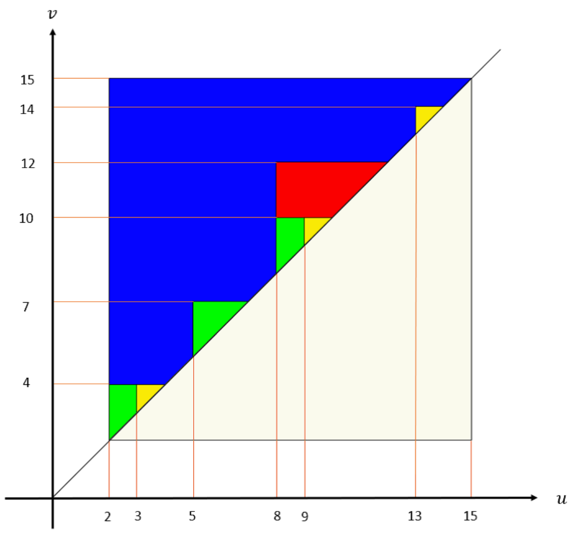


Figure 4.3. Figure borrowed from [Gur21]. View from top. In yellow, the prisms corresponding to cornerpoints with persistence 1, in green persistence those with 2, in red those with persistence 4, in blue the prism relative to the cornerpoint with the greatest persistence.

with higher persistence, i.e., the first pyramid to be born is the one with the highest persistence. Once assigned the order of birth, we proceed according to the classical algorithm. If two pyramids are associated with cornerpoints having equal persistence, the pyramid corresponding to the cornerpoint with the highest abscissa is considered to be older than the other one. Also, by convention, a pyramid dies when it merges with the plateau at height 0. This allows attributing less relevance to cornerpoints close to the diagonal, which typically represent noise. To understand better the concepts exposed above, consider the persistence diagram of fig. 4.1(a) where, for simplicity, the cornerline is neglected.

figs. 4.7 to 4.10 show the birth and the death of pyramids for different values of k. For $k > 0$ all the pyramids are born. As mentioned above, even if the pyramids are born at the same time, one is considered older than another based on the persistence of the corresponding cornerpoints. Thus, the oldest pyramid is consider to be a, followed by c, d, and b. Then we apply the elderly rule convention. For $k = 2$ the pyramid d dies merging with c. For $k = 7$ the pyramid b dies merging with d (and with the plateau at the same time). In the end, for $k = 10$ the pyramid c dies, merging with the plateau. The oldest pyramid a is considered to be never-dying, even if it would melt with the plateau for $k = 22$. As a result of this operation, we obtain the D persistence diagram in fig. 4.1(b), which provides the desired relevance ranking.

Referring to fig. 4.1(a), we obtained that point d had to be considered less relevant, as it derives from a disturbance of the starting object. Therefore, the result obtained is coherent with our expectations: point d is at the bottom of the relevance ranking.

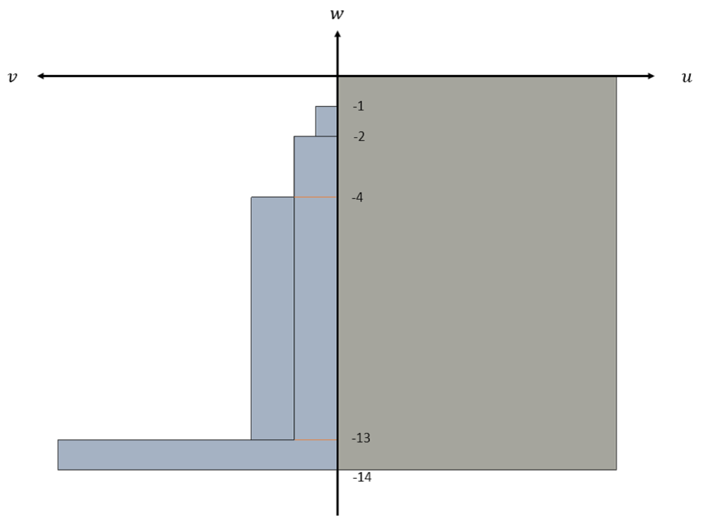


Figure 4.4. Figure borrowed from [Gur21]. Side view.

4.3 Algorithm

It is necessary to automatize the ranking operation via an algorithm taking in input the coordinates and other attributes of the cornerpoints, and, for each persistence diagram, returns the relevance ranking calculated as in section 4.2.1. The algorithm was designed in Python language. The code is available at:

https://github.com/luca9433/$_{c}odice_{t}esi_{/}blob/main/Ziggurat.py.$

Let D be a persistence diagram consisting of n cornerpoints with coordinates (x_i, y_i), with $i = 1, \ldots, n$. Consider the Ziggurat Z_D.

- When a pyramid generated by a cornerpoint merges with the plateau, it is considered dead.
- To apply the elderly rule, it is assumed that a cornerpoint is born before than another one if it has the highest persistence.
- In the case two cornerpoints have the same persistence, the cornerpoint considered to be older is the one with the highest abscissa.
- By convention, the cornerpoint with the highest persistence never dies (it becomes a cornerline in the persistence diagram of Z_D).

Python language allowed us to treat cornerpoints as objects of a class characterized by different attributes: an id number, coordinates (x, y), the level of the Ziggurat filtering function when the corresponding pyramid merges with another one for the first time, and multiplicity are the main ones.

For simplicity, from now on we will say that a cornerpoint merges with another one, is younger/older than another one, persists or dies, meaning that the corresponding

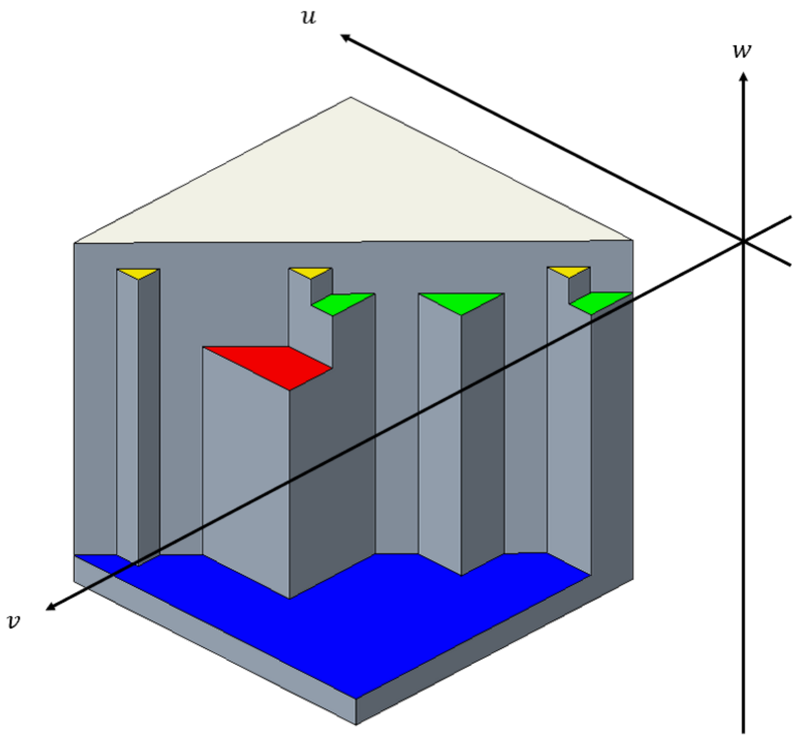


Figure 4.5. Figure borrowed from [Gur21]. Front view.

pyramid merges with another one, is younger/older than another one, persists or dies respectively.

As seen in section 4.2.1, it is necessary to compute all k_i, with $i = 1, \dots n$, at which the pyramids generated by the n cornerpoints with coordinates (x_i, y_i) die. A pyramid can die only if it merges with an older pyramid or it merges with the plateau. Given a generic cornerpoint with coordinates (x_i, y_i), to find the corresponding level k_i (the attribute level defined in the function "merging level" in the code) it suffices to verify if this cornerpoint merges with cornerpoints belonging to the following set:

$$A_i = \{(x, y) \in \mathbb{R}^2 | y > x + (y_i - x_i), x > x_i - (y_i - x_i), y < y_i + (y_i - x_i),$$

$$y < x + 2(y_i - x_i), y = x + (y_i - x_i), x_i < x \leqslant y_i)\}$$

If a cornerpoint does not belong to A_i, it means that it is younger than (x_i, y_i), or it is older and has no time to merge with (x_i, y_i) because (x_i, y_i) has already merged with the plateau.
Comparing (x_i, y_i) with any other cornerpoint $(x_h, y_h) \neq (x_i, y_i)$ we have:

- If $(x_h, y_h) \notin A_i$, then $k_{i,h} = y_i - x_i$

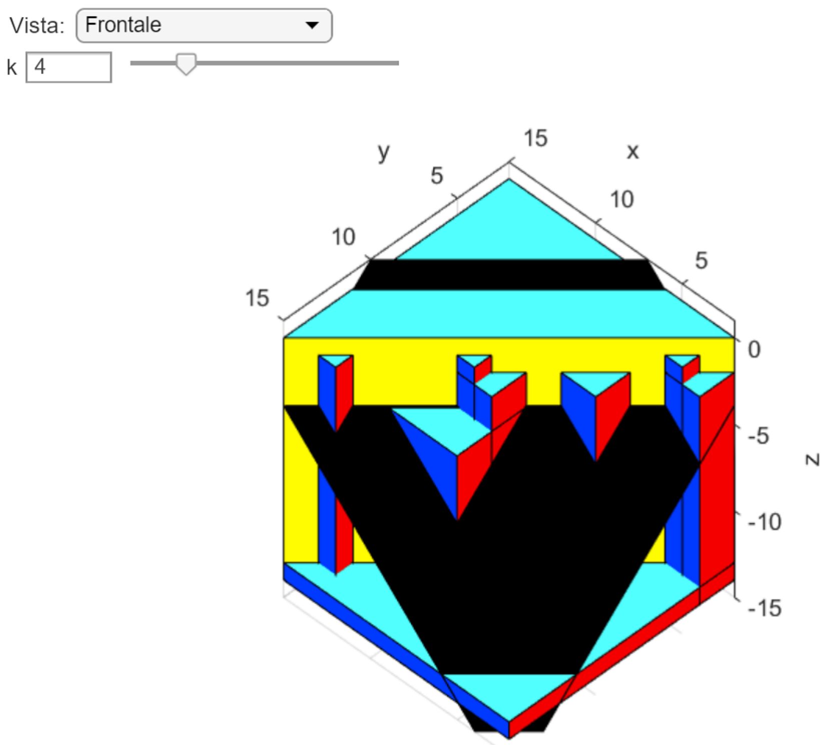


Figure 4.6. Figure borrowed from [Gur21]. Frontal view of the Ziggurat cutted by the plane of equation $f_{Z_D}(u, v, w) = u - v - w = 4$.

- If $(x_h, y_h) \in A_i$, then

$$k_{i,h} = \begin{cases} x_i - x_h & if\ (x_h, y_h) \in B_i \\ y_h - y_i & if\ (x_h, y_h) \in C_i \\ x_i - x_h + y_h - x_i & if\ (x_h, y_h) \in D_i \end{cases}$$

where:

- $B_i = \{(x, y) \in \mathbb{R}^2 | y > x + (y_i - x_i), x > x_i - (y_i - x_i), y \leqslant y_i\}$
- $C_i = \{(x, y) \in \mathbb{R}^2 | y \geqslant x + (y_i - x_i), y < y_i + (y_i - x_i), x \geqslant x_i\}$
- $D_i = \{(x, y) \in \mathbb{R}^2 | y > y_i, x < x_i, y < x + 2(y_i - x_i)\}$

For a given $i = 1, \ldots, n$, we consider the cornerpoint (x_i, y_i). We have a trapezoidal region $A_i = B_i \cup C_i \cup D_i$, union of a downside triangle B_i, an upside triangle C_i and a third middle triangle (case 3 in the code) D_i, such that, given another cornerpoint $(x_h, y_h) \in A_i$, (x_i, y_i) merges with (x_h, y_h) before it merges with the plateau and we can compute its merging level (the lowest level for which it merges with another cornerpoint) as follows: we consider the cornerpoint (x_i, y_i) and the corresponding set A_i. We calculate $k_{i,h}$ for all h such that $(x_h, y_h) \in Ai$ and finally we define $k_i = min(k_{i,h})$. If none $(x_h, y_h) \in A_i$, we set $k_i = yi - xi$ (the cornerpoint (x_i, y_i) merges with the plateau). The result is the assignment of a level for each cornerpoint. Each level is an ordinate of the persistence diagram of the

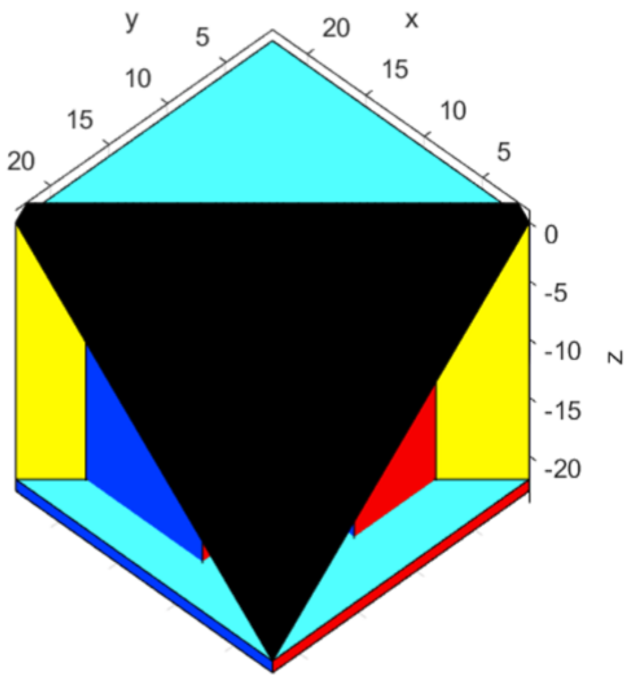


Figure 4.7. Image credits: [Gur21]. Frontal view of the Ziggurat cutted by the plane of equation $f_{Z_D}(u, v, w) = u - v - w = 0$ $(k = 0)$.

Ziggurat (the abscissas are all zero). Thus, we can define an order of cornerpoints based on level and sort them.

A "merges with" attribute is also defined for a given cornerpoint, consisting of the list of cornerpoints which it is compared with, analysing all the possible disposition of that cornerpoint with another one. It should be noted that, in a certain sense, we attribute a "verse" to the "merges with" attribute, since when a cornerpoint merges with another one, we mean that it is the first one to die being absorbed by the second one, according to the elderly rule. Then, we obtain, for a given cornerpoint, the consecutive merging with other older cornerpoints (in the Python code see functions "merge" and "merging list") starting from itself up to the oldest one, which merges with the plateau. The hypothesis based on which we obtain the list of consecutive mergings consists in assuming that for each cornerpoint we take, from its "merges with" list, the minimum cornerpoint among those greater than it. We then do the same with this minimum cornerpoint, and so on, until we arrive at a cornerpoint whose "merges with" list contains itself only.

Once the relevance ranking of the cornerpoints has been obtained through the persistence diagram $\mathcal{D}$ of the Ziggurat, it is necessary to define which cornerpoints to select and which ones to consider noise. Therefore, it is necessary to define a rule that establishes a level (ordinate of $\mathcal{D}$) to consider noise cornerpoints below it. To this end, the rule set out in article [Kur16] used in [BFT20] turns out to be very useful. In general, given a persistence diagram, the idea is to define "diagonal gaps", that are diagonal bands included between two lines parallel to the diagonal, which

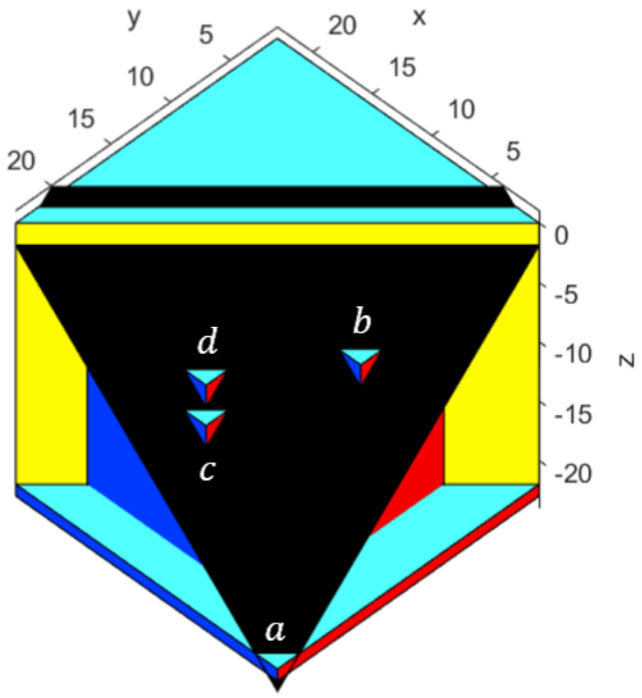


Figure 4.8. Image credits: [Gur21]. Frontal view of the Ziggurat cutted by the plane of equation $f_{Z_D}(u, v, w) = u - v - w = 2$ $(k = 2)$.

do not contain cornerpoints, but have them on both sides. Later it is established a diagonal gap hierarchy, using the decreasing width of the bands as a criterion. The assumption of this procedure is that cornerpoints below the widest diagonal gap are noisy. In the context of the persistence diagram $\mathcal{D}$ of the Ziggurat, the application of this rule is very simple. All cornerpoints have the same abscissa and therefore they lie on a single vertical line. The diagonal gaps become simply the ordinate differences of consecutive cornerpoints. It is therefore possible to carry out the selection considering the points above the widest gap. Referring to the persistence diagram in fig. 4.1(b), the largest gap is the one with width 5 between ordinates 2 and 7. Coherently with the comments made in relation to this example, points a, c and b are selected, while point d is discarded as noise. The final result of the selection process made on the diagram on the right side of fig. 4.1(a) is shown in fig. 4.1(c)

Example Let us consider the persistence diagram represented in fig. 4.12(a) and consisting of the cornerpoints in table 4.1. If we consider the corresponding Ziggurat and the selection algorithm described above, we obtain a relevance ranking of its cornerpoints, which allows us to select a subset of cornerpoints of the original persistence diagram, thus obtaining a new persistence diagram with reduced cardinality. In fig. 4.12, we color-code the correspondence between cornerpoints in the starting persistence diagram and cornerpoints in the Ziggurat's persistence diagram. We can better see the selection made with Kurlin's rule by selecting the cornerpoints

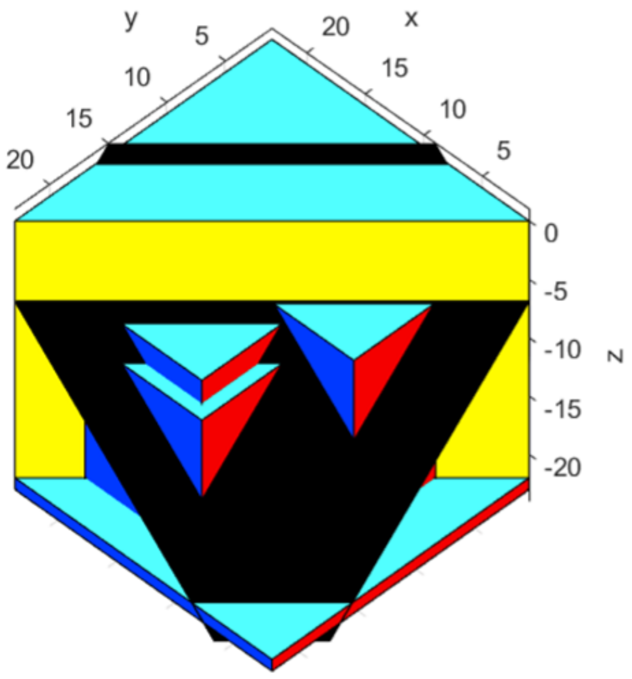


Figure 4.9. Image credits: [Gur21]. Frontal view of the Ziggurat cutted by the plane of equation $f_{Z_D}(u, v, w) = u - v - w = 7$ ($k = 7$).

above the widest gap, representing with different colors the points above the gap and those below. In this case, the selection is particularly interesting, as in the starting diagram we can recognize clusters, and from each a cornerpoint is selected (we could also have chosen, for instance, a priori the number of cornerpoints to select, without invoking Kurlin's rule). The relevance ranking of the cornerpoints, decreasingly ordered by level, is reported in table 4.2. Kurlin's rule in this case selects the first three cornerpoints.

4.4 A possible extension of the elderly rule

We have just seen how it is possible to obtain a cardinality reduction of a persistence diagram using the Ziggurat selection algorithm and a selection criterion. This involves a loss of information but could lead to a gain in classification power by discarding irrelevant or noisy information. In this regard, one might wonder if there is an optimal way to select cornerpoints of a persistence diagram and if the one treated just now comes close to being so. An idea still under development is that the cardinality reduction can indeed be seen as a recovery of symmetry possibly lost during the extraction of information. In that context, the selected cornerpoints would act as sort of *symmetry attractors* by accumulating the multiplicity of nearby cornerpoints and thus allowing for symmetry discovery.

Following this latter approach means rethinking the elderly rule, which until now

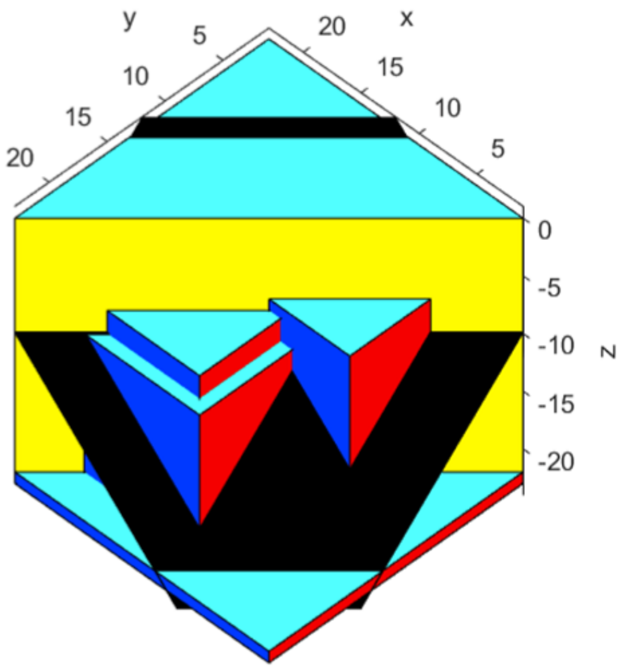


Figure 4.10. Image credits: [Gur21]. Frontal view of the Ziggurat cutted by the plane of equation $f_{Z_D}(u, v, w) = u - v - w = 4$ ($k = 4$).

took into account only the cornerpoints' persistence and abscissa. Indeed, we should also consider the cornerpoints' multiplicity.

Let D be a persistence diagram. Initially, all cornerpoints of D have cumulative multiplicity equal to their multiplicity, except for the diagonal, which has infinite cumulative multiplicity (putting the diagonal to infinity has the intended effect that all of Z_D's cornerpoints die at finite value). At the merging of two components of the Ziggurat, corresponding to two cornerpoints of D, the one to be considered older is:

- the one with the highest cumulative multiplicity;
- if they have equal cumulative multiplicity, the one with the highest persistence;
- if they have equal cumulative multiplicity and persistence, the one with the highest abscissa.

The cumulative multiplicity of the eldest component is the sum of the cumulative multiplicities of the merging components.

In a cluster of cornerpoints, this should privilege not necessarily the one with the highest persistence, but the one in the center of the densest part of the cluster. While this version of the algorithm better clarifies, in principle, which cornerpoint is selected, it is also true that the algorithm would seem much more complicated to manage in this version, due to the fact that the cumulative multiplicities of each

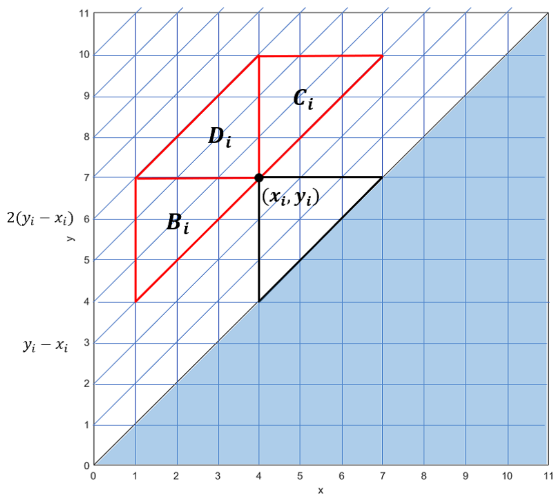


Figure 4.11. Image credits: [Gur21]. Graphic display of the selection algorithm.

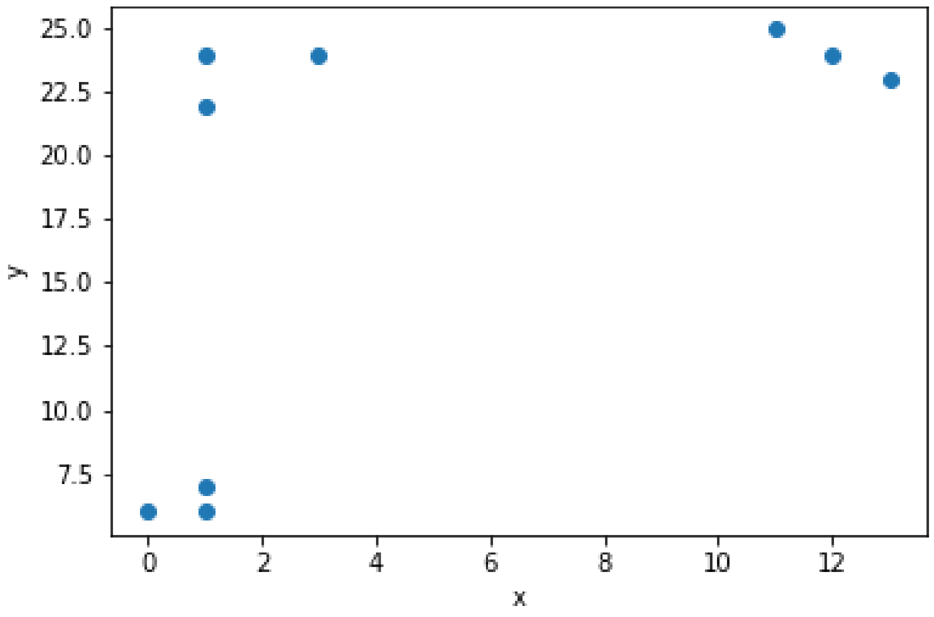
id	x	y
1	0	6
2	1	6
3	1	7
4	1	22
5	1	24
6	3	24
7	11	25
8	12	24
9	13	23

Table 4.1. Coordinates of cornerpoints of persistence diagram displayed in fig. 4.12(a); id is the identificative number of a cornerpoint.

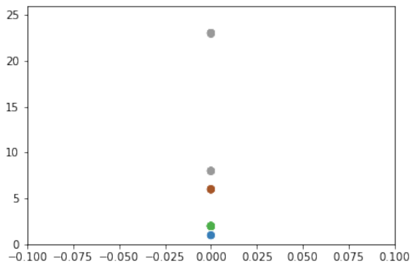
id	x	y	level
5	1	24	23
7	11	25	8
3	1	7	6
9	13	23	2
8	12	24	2
6	3	24	2
4	1	22	2
2	1	6	1
1	0	6	1

Table 4.2. Coordinates of cornerpoints in table 4.1 sorted by level. According to Kurlin's rule the first three cornerpoints are selected.

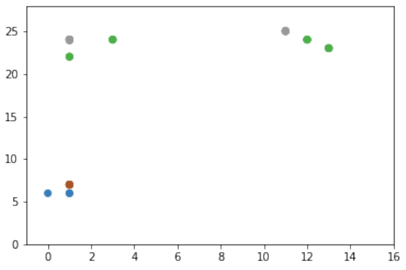
cornerpoint they would continually update. In the code we tried an implementation of this second elderly rule, which would seem to have, at most, only local and not global applicability, based on the fact that by restricting to cornerpoint clusters sufficiently close to each other, we are able to avoid comparing cornerpoints very "distant" but having similar level.



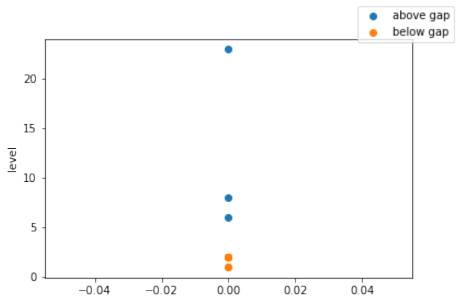
(a)



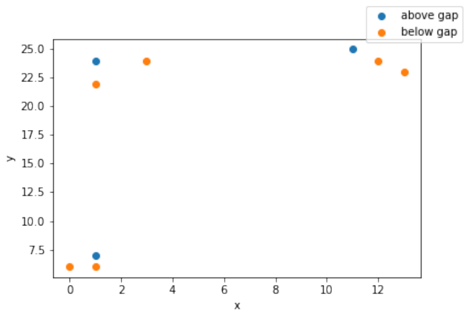
(b)



(c)



(d)



(e)

Figure 4.12. In fig. 4.12(a), the persistence diagram whose cornerpoints have coordinates shown in table 4.1; figs. 4.12(b) and 4.12(c) show the correspondence of cornerpoints in the corresponding Ziggurat's persistence diagram with the respective ones in the original diagram, colored according to cornerpoints' levels, and provide a relevance ranking for the original diagram. Finally figs. 4.12(d) and 4.12(e) show the cornerpoints seletion according to Kurlin's rule in the Ziggurat persistence's diagram and the effects of this selection on the original diagram respectively: cornerpoints "above the gap" are those selected and we can see that for each "cluster" exactly one cornerpoint is selected.

Conclusions and future work

We explored the crossroad between topological persistence and machine learning. Persistence homology allows for concisely representing specific, user-driven features of the data, as persistence diagrams. However, by their own nature, persistence diagrams cannot be fed directly to machine learning algorithms, that are highly efficient in solving relevant task, such as data classification.

First, we considered persistence images, a state-of-the-art transformation that allows for vectorizing persistence diagrams. We implemented the necessary algorithms to apply this technique to the music genres classification task, and showed how topologically relevant information translates in a performance improvement of basic machine learning strategies. While developing this pipeline, we realized that persistent images require the user to choose additional hyperparameters and are—being images—computationally heavy to analyze.

For this reason, we introduced a second method to perform persistence diagram vectorization. This latter method ranks by relevance, and thus allows us to select the cornerpoints of a persistence diagram. Cornerpoints are ranked through the Ziggurat algorithm, presented in chapter 4. Intuitively, given a collection of persistence diagrams, this algorithms yields the n most relevant cornerpoint per diagram, allowing us to create a lightweight, vectorizable representation of each diagram. We tested the Ziggurat-based ranking and selection pipeline on several toy examples showing how this method is able to indeed detect representative cornerpoints that well summarize the content of the original persistence diagram.

In a forthcoming paper, we shall validate this approach on the music genres classification task, both in the version described in chapter 4, and with an original elderly rule which more explicitly associates the process of cardinality reduction with a recovery of symmetry lost during the extraction of information, in the wake of what has been said in section 4.4.

The Python package developed to validate the methods described in this work is available at $https : //github.com/luca9433/_codice_tesi$.

Bibliography

[DM80] Steven Davis and Paul Mermelstein. “Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences”. In: *IEEE transactions on acoustics, speech, and signal processing* 28.4 (1980), pp. 357–366.

[BGV92] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. “A training algorithm for optimal margin classifiers”. In: *Proceedings of the fifth annual workshop on Computational learning theory.* 1992, pp. 144–152.

[DFL98] Pietro Donatini, Patrizio Frosini, and Alberto Lovato. “Size functions for signature recognition”. In: *Vision Geometry VII.* Vol. 3454. International Society for Optics and Photonics. 1998, pp. 178–183.

[Fer+98] Massimo Ferri et al. “Point selection: A new comparison scheme for size functions (with an application to monogram recognition)”. In: *Asian Conference on Computer Vision.* Springer. 1998, pp. 329–337.

[Sch98] Eric D Scheirer. “Tempo and beat analysis of acoustic musical signals”. In: *The Journal of the Acoustical Society of America* 103.1 (1998), pp. 588–601.

[EP99] Theodoros Evgeniou and Massimiliano Pontil. “Support vector machines: Theory and applications”. In: *Advanced Course on Artificial Intelligence.* Springer. 1999, pp. 249–257.

[Log00] Beth Logan. “Mel frequency cepstral coefficients for music modeling”. In: *In International Symposium on Music Information Retrieval.* Citeseer. 2000.

[TC02] George Tzanetakis and Perry Cook. “Musical genre classification of audio signals”. In: *IEEE Transactions on speech and audio processing* 10.5 (2002), pp. 293–302.

[Dow03] J Stephen Downie. “Music information retrieval”. In: *Annual review of information science and technology* 37.1 (2003), pp. 295–340.

[Kha03] Syed Ali Khayam. “The discrete cosine transform (DCT): theory and application”. In: *Michigan State University* 114 (2003), pp. 1–31.

[EH10] Herbert Edelsbrunner and John Harer. *Computational topology: an introduction.* American Mathematical Soc., 2010.

[Ped+11] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[ABM12] G Anastasi, G Balboni, and P Motta. “Trattato di Anatomia Umana, vol. III”. In: *Milan, Italy: Ermes* (2012).

[Stu12] Bob L Sturm. “An analysis of the GTZAN music genre dataset”. In: *Proceedings of the second international ACM workshop on Music information retrieval with user-centered and multimodal strategies*. 2012, pp. 7–12.

[Rot13] Joseph J Rotman. *An introduction to algebraic topology*. Vol. 119. Springer Science & Business Media, 2013.

[Stu13] Bob L Sturm. “The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use”. In: *arXiv preprint arXiv:1306.1461* (2013).

[Ber15] Mattia Giuseppe Bergomi. “Dynamical and topological tools for (modern) music analysis”. PhD thesis. Università degli Studi di Milano; Université Pierre et Marie Curie, 2015.

[BBD16] Mattia G Bergomi, Adriano Baratè, and Barbara Di Fabio. “Towards a topological fingerprint of music”. In: *International Workshop on Computational Topology in Image Context*. Springer. 2016, pp. 88–100.

[Kur16] Vitaliy Kurlin. “A fast persistence-based segmentation of noisy 2D clouds with provable guarantees”. In: *Pattern recognition letters* 83 (2016), pp. 3–12.

[Ada+17] Henry Adams et al. “Persistence images: A stable vector representation of persistent homology”. In: *Journal of Machine Learning Research* 18 (2017).

[Fer17] Massimo Ferri. “Persistent topology for natural data analysis—A survey”. In: *Towards integrative machine learning and knowledge extraction*. Springer, 2017, pp. 117–133.

[AFT18] Alessia Angeli, Massimo Ferri, and Ivan Tomba. “Symmetric functions for fast image retrieval with persistent homology”. In: *Mathematical Methods in the Applied Sciences* 41.18 (2018), pp. 9567–9577.

[TSB18] Christopher Tralie, Nathaniel Saul, and Rann Bar-On. “Ripser.py: A Lean Persistent Homology Library for Python”. In: *The Journal of Open Source Software* 3.29 (Sept. 2018), p. 925. DOI: `10.21105/joss.00925`. URL: `https://doi.org/10.21105/joss.00925`.

[Rob+19] Antonio Robles-Guerrero et al. “Analysis of a multiclass classification problem by Lasso Logistic Regression and Singular Value Decomposition to identify sound patterns in queenless bee colonies”. In: *Computers and Electronics in Agriculture* 159 (2019), pp. 69–74. ISSN: 0168-1699. DOI: `https://doi.org/10.1016/j.compag.2019.02.024`. URL: `https://www.sciencedirect.com/science/article/pii/S0168169918313267`.

[BFT20] Mattia G Bergomi, Massimo Ferri, and Antonella Tavaglione. “Steady and ranging sets in graph persistence”. In: *arXiv preprint arXiv:2009.06897* (2020).

[Lav20] Vito Lavecchia. *Mel Frequency Cepstral Coefficient (MFCC) – Guidebook*. https://vitolavecchia.altervista.org/mel-frequency-cepstral-coefficient-mfcc-guidebook/. 2020.

[MHM20] Leland McInnes, John Healy, and James Melville. “UMAP: uniform manifold approximation and projection for dimension reduction”. In: (2020).

[CDL21] Rodrigo Castellon, Chris Donahue, and Percy Liang. “Codified audio language modeling learns useful representations for music information retrieval”. In: *arXiv preprint arXiv:2107.05677* (2021).

[McF+21] Brian McFee et al. *librosa/librosa: 0.8.1rc2*. Version 0.8.1rc2. May 2021. DOI: 10.5281/zenodo.4792298. URL: https://doi.org/10.5281/zenodo.4792298.

[Man22] Marco Manetti. *Istituzioni di Algebra e Geometria 2021-22*. https://www1.mat.uniroma1.it/people/manetti/Geosup2018/DispenseIAG.pdf. La Sapienza, Rome. Jan. 2022.

[Ang] Alessia Angeli. “Recupero di immagini dermatologiche mediante persistenza e formule di Viète”. PhD thesis. URL: http://amslaurea.unibo.it/14677/.

[Gur21] Davide Gurrieri. *Un nuovo metodo per la selezione di punti in un diagramma di persistenza*. Bachelor Thesis. Advisors: Prof. Massimo Ferri, Dr. Mattia G. Bergomi. 2021.

Ringraziamenti

Vorrei ringraziare il Prof. Marco Manetti per aver accettato di essere il relatore di questa tesi e per avermi dato la possibilità di conoscere il Prof. Massimo Ferri, a cui sono grato per aver fin da subito assecondato con entusiasmo la mia proposta di lavorare sotto la sua supervisione, dimostrandosi sempre pronto a fornire preziosi consigli e nuovi spunti. Ringrazio il Dr. Mattia Bergomi, che con grande pazienza e competenza mi ha costantemente seguito e indirizzato in questi mesi, sia negli aspetti più teorici, che nell'implementazione in linguaggio Python di tutti i metodi analizzati, oltre ad aver puntualmente rivisto le bozze di questo lavoro.

Non sarò mai sufficientemente grato a mia Madre Dora per avermi sempre supportato in tutto, rendendo possibili i miei studi, e a mia Sorella Lilia, per avermi sempre sostenuto, sul piano mentale e materiale. Devo riservare una particolare menzione anche a suo marito Manuel, che, soprattutto nell'ultimo periodo, non ha mai mancato di dimostrarmi vicinanza nel momento del bisogno.

Ringrazio mia Zia Rosella e Zio Peppe, che da sempre sono come genitori per me, mio Zio Pierluigi, su cui ho sempre potuto contare come un padre, e Zia Rosalba, Zio Guido e Zia Vincenzina, Zio Vincenzo e Zia Carla, per avermi sempre accolto in questi anni con il calore di una vera famiglia, Zia Maria, Zio Pacifico, tutti loro hanno contribuito in modi diversi affinché potessi arrivare a questo punto.

Un pensiero di particolare gratitudine va a Mia Cugina Stefania, per esserci sempre stata, a mio Cugino Dante, sua moglie Jana e il piccolo Guido, e alle mie Cugine Letizia e Rita.

Devo ricordare poi due persone che se ne sono andate troppo presto e che avrei voluto potessero essere state presenti in questo momento, mio Padre Antonio e mio Zio Gianni. Un pensiero particolare va a loro e a mia Nonna Rita.

Determinante in questi anni per me è stata l'amicizia fraterna di Nello, al quale devo molto, non essendosi mai risparmiato nel fornirmi supporto di ogni tipo, dandomi anche preziosi consigli durante la preparazione di questo lavoro, e di Gianmarco, anche lui sempre presente nel momento del bisogno e non. Un ringraziamento particolare va anche agli altri amici di vecchia data, sui quali, in vario modo, ho sempre potuto contare prima e durante gli anni universitari: Alessandro, Silvia, Ermal, Marco e tutto il gruppo del Newton a cui sono e sarò sempre profondamente legato.

Un ruolo molto importante hanno rivestito negli ultimi anni tutti i miei amici musicisti, tra cui Francesco, Marta, Jacopo, Paolo e Guglielmo, il quale, soprattutto nel periodo di lavoro su questa tesi, con le sue competenze informatiche, non si è mai risparmiato nel fornirmi spiegazioni e consigli ogni volta che glieli abbia richiesti.

Infine, non posso non menzionare Gabriele, una delle persone che più di tutte ha rivoluzionato positivamente il mio modo di vedere la vita negli ultimi anni, e sicuramente la ricerca che mi ha spinto verso l'argomento di questa tesi è dipesa in qualche modo anche da lui.