Preliminary results on classification based on Topological Data Analysis(1)

Rolando Kindelan
Computer Science Department, Faculty of Mathematical and Physical Sciences, University of Chile, 851 Beauchef Av. Santiago de Chile, Chile
Center of Medical Biophysics, University of Oriente, Santiago de Cuba, Cuba
rolan2kn@gmail.com

Mauricio Cerda
Laboratory for Scientific Image Analysis (SCIAN-Lab), Biomedical Neuroscience Institute (BNI), Faculty of Medicine, University of Chile, 1027 Independencia Av. Santiago, Chile.
mauriciocerda@med.uchile.cl

Nancy Hitschfeld
Computer Science Department, Faculty of Mathematical and Physical Sciences, University of Chile, 851 Beauchef Av. Santiago, Chile.
nancy@dcc.uchile.cl

Abstract

In recent years, algebraic topology techniques have been applied to data analysis, giving rise to an emerging research field called Topological Data Analysis (TDA). TDA tools are useful for inferring the topology underlying a dataset. On the other hand, it is known that most Machine Learning (ML) techniques cannot capture topological information. Consequently, TDA has been used to create filters and topological descriptors to improve the results of ML methods. However, and despite the good results, these hybrid methods do not to fully exploit the potentials of TDA and hence implicit duplications of computations occur. This paper proposes a TDA-based method to solve a 3-class classification problem. The method constructs a filtered simplicial complex K from a training set S and testing set X. This construction produces a large size $O(2^{|S\cup X|})$ searching space to classify each $x \in X$. We use persistent homology to downsize the searching space by detecting persistent intervals of desired topological features. In consequence, the searching space is reduced to $O(2^{q+1} \cdot |X|)$, with $q \ll |S \cup X|$ the constant dimension of the sub-complex $K_i \subset K$ resulting from selected persistent interval. In the last stage, the algorithm labels each $x \in X$, by using the label-contribution of every 0-simplex $\in K_i$ which shares at least a simplex with x. We develop an experimental proof of concept by comparing our TDA-based classifier with a k-NN classifier on the Iris dataset, using 10 evaluation metrics in a repeated-cross validation process. Our method gets 96% accuracy versus 97% for k-NN. Our work shows that it is possible to classify only with TDA, with no additional machine learning algorithms. For future work we plan to apply the proposed method to datasets with highly dimensional and noisy samples.

2012 ACM Subject Classification Mathematics of Computing → Point-set Topology; Mathematics of Computing → Algebraic Topology; Theory of computation → Computational geometry; Theory of computation → Machine learning theory; Theory of computation → Semi-supervised learning

Keywords and phrases TDA, TDA-based classifier, simplicial complexes, persistent homology, machine learning

Funding *Rolando Kindelan*: CONICYT 2018/BECA DOCTORADO NACIONAL-21181978

Lines 0

(Submitted to 36th SoCG - Zürich, Switzerland - June 23-26, 2020)

1 Introduction

The processing and extraction of information from large and noisy data sets is a challenging problem in Computer Science. The techniques of algebraic topology have gained the attention of scientists for years, giving rise to an emerging research field called Topological Data Analysis (TDA) [24, 5, 8, 9, 30, 13, 12]. TDA is an approach to infer the topology underlying a dataset by using combinatorial algebraic structures known as simplicial complexes. TDA also involves the computation of invariant properties from continuous transformations of these simplicial complexes: a process known as persistent homology [8, 9, 30, 13].

Over several decades the high-dimensionality of datasets and the combinatorial and continuous character of Topology have been issues, making computing persistent homology a challenge. Regarding persistent homology, Edelsbrunner et al. [8] present an efficient algorithm and its visualization as a persistence diagram [8, 30]. Carlson et al. [5] strengthened the mathematical foundations and also proposed another visualization tool called persistence Barcodes [13, 5]. Further developments of the TDA field are derived from those initial works.

The construction and representation of simplicial complexes also represent a challenge, as a consequence of their combinatorial nature. Many works have dealt with efficient construction, representation and filtration of simplicial complexes. As a consequence, efficient data structures and algorithms were developed [7, 1, 29, 2, 3, 17, 18], mainly focusing on efficient construction of *Čech*, Rips and other kinds of simplicial complexes not used here. Theoretical and practical results have been organized in the form of TDA libraries like GUDHI [3, 19], Dionysus, Ripser, Dipha, Perseus and JavaPlex. A complete benchmark of those libraries can be found in Otter et al. [21].

A TDA-based method was used in [11] for classifying high-resolution diabetic retinopathy images. They use a preprocessing stage for computing persistent homology to detect topological features encoded into persistence diagrams. A support vector machine (SVM) was used to classify the images according to the persistence descriptors which was used to discriminate between diabetic and healthy patients. They recommend exploring their TDA+SVM method further in larger datasets of high-resolution images.

Also TDA has been applied to time-series analyses [6]. One common pipeline is to consider the time-series as a dynamic system and compute the attractors or time-variant of the signal which creates a manifold around the attractors, turning the signal into phase-domain [28, 27]. Persistent Homology or another TDA-tool is applied on these phase-space manifold to create topological descriptors [23] and as a final step, a machine learning method is applied like k-NN, CNN, and also SVM. There are other applications of TDA presented in [6] like scientific visualization, bioinformatics, atmospheric and climate data analysis, cosmology and combustion simulations. Furthermore, TDA has been applied in neurosciences [25] and recently in human motion understanding [16, 14, 28].

All those examples of TDA-applications have one thing in common: TDA has been used as a preprocessing stage of conventional Machine Learning (ML) algorithms [26]. However, to the best of our knowledge, there are no references to using TDA directly as a classification method.

In this paper, we propose a classification method that takes advantage of the TDA information and topological properties. We compare our method to k-NN as a baseline, one of the most used supervised classification methods. This document has been organized as follows. Section 2 exposes the mathematical foundations that we use in this work. Section 3 explains the concepts, algorithms, and methodology of our proposed classification method. Next, Section 4 presents our experimental design, evaluation criteria, selected metrics, and

our results. Section 5 is where we explain our preliminary results and development decisions of our method. Section 7 presents several research lines that have arisen from this work. We present the main conclusions of our work in the Section 6.

2 Mathematical foundations

In this section, we introduce the mathematical definition of simplices, simplicial complex, the *Čech* and Rips complexes, the star, and link concepts. We also define persistent homology, filtration, sub-complex, and filtration levels.

2.1 Simplicial Complexes

Simplicial complexes are combinatorial and algebraic objects which represent a discrete space homotopically equivalent to data space. There are some related concepts to understand. In a nutshell, a q-simplex is the convex hull of $q+1$ affinely independent points $\{s_0, s_1, \cdots, s_q\} \in S, S \subset \mathbb{R}^n$. A q-simplex σ has dimension $dim(\sigma) = q$ and cardinality $|\sigma| = card(\sigma) = dim(\sigma)+1$. A simplex τ defined by $S' \subseteq S$ is a $face$ of σ and has σ as a $coface$. A q-simplex has $\binom{q+1}{d+1}$ faces of dimension d and $\sum_{d=-1}^{q}\binom{q+1}{d+1} = 2^{q+1}$ faces in total. We symbolize the $face$ and $coface$ relationships with $\sigma \geq \tau$ and $\tau \leq \sigma$. So, a simplicial complex K is a finite collection of simplices such that:

- $\sigma \in K$ and $\tau \leq \sigma \implies \tau \in K$.
- $\sigma_1, \sigma_2 \in K \implies \sigma_1 \cap \sigma_2 \leq \sigma_1, \sigma_2$.

The dimension of K is $dim(K) = max\{dim(\sigma) \mid \sigma \in K\}$.

There are many known simplicial complexes, though the most popular are the *Čech* and Vietoris-Rips complexes. The *Čech* complex is built by all non-empty intersections of closed balls $\mathbb{B}_s(\varepsilon)$, with ε radius and centered on each point s from the dataset [13, 8]:

$$\check{C}ech(\varepsilon) = \{\sigma \subseteq S \mid \bigcap_{\tau \in \sigma} \mathbb{B}_\tau(\varepsilon) \neq \emptyset\}.$$

The Vietoris-Rips (VR) complex from a point set S and ε value is built with all subsets, for which each minimum enclosing ball has a diameter up to 2ε [8]:

$$VR(\varepsilon) = \{\sigma \subseteq S \mid diam(\sigma) \leq 2\varepsilon\}.$$

This implies $\check{C}ech(\varepsilon) \subseteq VR(\varepsilon) \subseteq \check{C}ech(\sqrt{2}\varepsilon)$ a proof is given in [8], this relationship is shown in Figure 1. The *Čech* complex is intrinsically a high dimensional simplicial complex. From a computational sense, VR complex is more feasible (i.e. lower storage and time complexity) than *Čech*, even when VR complex has more simplices in general. Compared to *Čech*, VR complex does not need to be completely stored, it can be stored like a graph and reconstituted combinatorially [13].

▶ **Definition 1** (Star and Link). *Let K be a simplicial complex, and $\tau \in K$ a q-simplex. The star of τ defined by $St(\tau) = \{\sigma \in K \mid \tau \leq \sigma\}$ is the set of all cofaces of τ [18, 8] (see Figure 2). The $St(\tau)$ is not a simplicial complex because of the missing faces. If those faces are added to $St(\tau)$, we get the closed star of τ denoted by $\overline{St}(\tau)$, which is the smallest simplicial complex that contains the star. The link of τ is a set of simplices in the closed star that does not share any face with τ, $Lk(\tau) = \{\nu \in \overline{St}(\tau) \mid \nu \cap \tau = \emptyset\}$ [18, 8]. If τ is a 0-simplex, then $Lk(\tau) = \overline{St}(\tau) - St(\tau)$ (see Figure 2).*

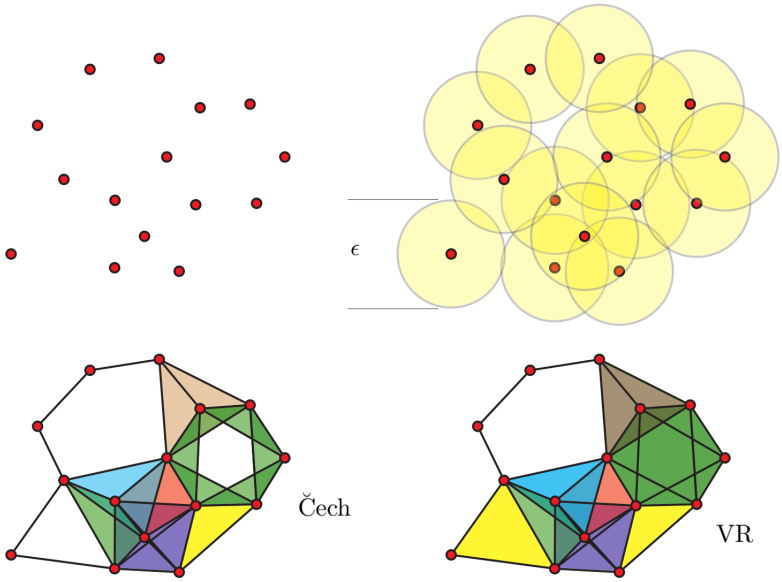


Figure 1 From a point set [upper left] a proximity parameter *varepsilon* is applied [upper right] and two complexes were built: a *Čech* complex [lower left] and a VR complex [lower right]. There are differences between $VR(\varepsilon)$ and $\check{C}ech(\frac{\varepsilon}{2})$. VR has more simplices than $\check{C}ech$ as expected, note all colored 2-simplices. *This picture was taken from [13].*

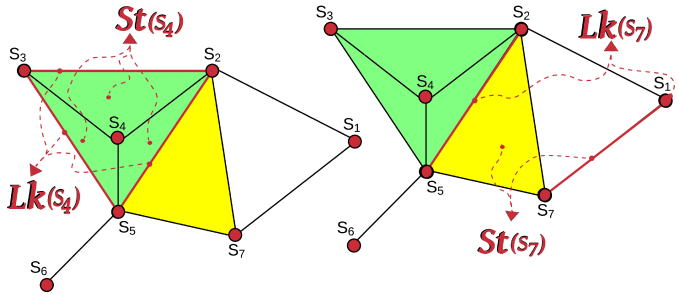


Figure 2 Example of $Lk(S_4), St(S_4)$ and $Lk(S_7), St(S_7)$ on a given simplicial complex K.

In Figure 2 we present two examples of the St and Lk from points s_4, s_7 from a point set S. If we expand the solutions we get the following:

$$St(s_4) = \{\{s_4\}, \{s_4, s_2\}, \{s_5, s_4\}, \{s_4, s_3\}, \{s_3, s_4, s_2\}, \{s_3, s_5, s_4\}, \{s_4, s_5, s_2\}, \{s_3, s_5, s_2, s_4\}\},$$
$$Lk(s_4) = \{\{s_3\}, \{s_2\}, \{s_5\}, \{s_2, s_3\}, \{s_3, s_5\}, \{s_5, s_2\}, \{s_3, s_5, s_2\}\},$$
$$St(s_7) = \{\{s_7\}, \{s_7, s_1\}, \{s_5, s_7, s_2\}\},$$
$$Lk(s_7) = \{\{s_1\}, \{s_2\}, \{s_5\}, \{s_5, s_2\}\}.$$

2.2 Persistent Homology

Persistent homology is a tool to find topological features in a metric space [8, 13, 5]. As a general rule, the objective of persistent homology is to track how topological features

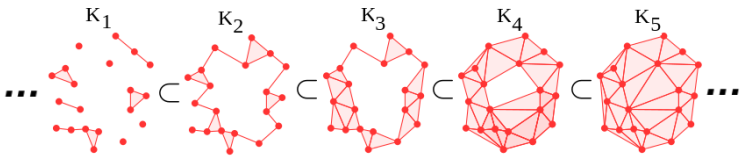


Figure 3 A fragment example of a simplicial complex filtration..

on a topological space appear and disappear when a scale value (usually a radius) varies incrementally, a process known as filtration [9, 29, 30].

▶ **Definition 2** (Sub-complex). *Let K be a simplicial complex. K' is a sub-complex of K if $K' \subseteq K$ and besides K' is by itself a simplicial complex.*

▶ **Definition 3** (Filtration). *Let K be a simplicial complex. A filtration f is a succession of increasing sub-complexes of K:*

$$\emptyset = K_0 \subseteq K_1 \subseteq K_2 \subseteq K_3 \subseteq \cdots \subseteq K_n = K.$$

We can understand a filtration as a method to build the whole simplicial complex K from a "family" of sub-complexes incrementally sorted according to some criteria, where each level i corresponds to the "birth" or "death" of a q-simplex set as described in the following definition.

▶ **Definition 4** (Birth and Death). *birth is a metaphorical concept to describe the filtration level when a set of simplices are created. Similarly, death refers to the filtration level when a set of simplices disappeared. Thus, a persistence interval (birth, death) is the "lifetime" of a given set of simplices Q [8, 9, 5, 29, 30]. Note Q is not a sub-complex because of missing faces with other intervals.*

3 Proposed Classification Method

Let P be a metric space, with every $p \in P$ a data feature value vector in R^n. We split the dataset P in two sets S, X in which $S \neq \emptyset; X \neq \emptyset; S \cap X = \emptyset$ and $S \cup X = P$. Let S be the training set and X the testing set to be classified. Let L be a label set. The 2-tuple set $T = \{(p, l) \mid p \in P; l \in L\}$ relates each element of P with its associated label from L. The incomplete association set T' is a 2-tuple set with $T' = \{(p, l) \mid (p \in S; l \in L) \vee (p \in X; l = \emptyset)\}$ where each element of S has an associated label from L, but each element of X does not have any label.

3.1 Definitions

In order to understand and clarify the proposed method, we define several mathematical concepts.

▶ **Definition 5** (Useful-simplex and Non-useful-simplex). *Let K be a simplicial complex built from $S \cup X$. A q-simplex $\sigma \in K$. Let α, β be two sets where $\alpha \subset S, \beta \subset X, \alpha \subseteq \sigma, \beta \subseteq \sigma$ and $\alpha \cup \beta = \sigma, \alpha \cap \beta = \emptyset$. We say σ is a useful-simplex if $|\alpha| > |\beta|$. In another case, if $|\alpha| \leq |\beta|$, then σ is a non-useful-simplex.*

▶ **Definition 6** (Association function, Ψ). *Let K be a simplicial complex, σ a q-simplex $\sigma \in K$. Let $\mathcal{P}(L)$ be the power-set of L. Let $\mathbb{T}$ be a 2-tuple set of associations $\mathbb{T} \in \{T, T'\}$. We define the association function $\Psi : K \times K \to \mathcal{P}(L)$ such as:*

$$\Psi(\sigma, K) = \begin{cases} l \in \mathcal{P}(L) \vee (\sigma, l) \in \mathbb{T} & \text{iff } card(\sigma) = 1; l \neq \emptyset, \\ \bigcup_{\alpha \in Lk(\sigma))} \Psi(\alpha, K) & \text{iff } card(\sigma) = 1; l = \emptyset, \\ \bigcup_{\tau \in \sigma} \Psi(\tau, K) & \text{iff } card(\sigma) > 1 \end{cases} \tag{1}$$

When $card(\sigma) = 1$, σ is a 0-simplex with, $\sigma \in S$, or (exclusive or) $\sigma \in X$. In the first case it is obvious that $l \neq \emptyset$ because the labels are always known for all elements in S. In the second case, $l \neq \emptyset$ if l was previously computed or $\mathbb{T} = T$. In any other case, as $l = \emptyset$, we found l value by using the union of all associations of each element of $Lk(\sigma)$.

By definition 1, for a simplicial complex K, the intersection of Lk of any element of a simplex $\sigma \in K$, is necessarily non-empty, therefore all solutions of $\Psi(\sigma, K)$ are overlapping and it is possible to optimize its computation by using dynamic programming.

▶ **Definition 7** (Indicator function, $\mathcal{I}$). *Let c be a logical statement. The indicator function $\mathcal{I}$ is defined by:*

$$\mathcal{I}(c) = \begin{cases} \text{if } c = True & 1, \\ \text{if } c = False & 0. \end{cases} \tag{2}$$

Function $\Gamma(\sigma, K)$ returns a vector $V \in \mathbb{R}^{|L|}$ for which each element $v_i \in V$ represents the amount of apparitions (votes) achieved by label $l_i \in L$ during $\Psi(\sigma, K)$ computation. According to definition 6, we can also make optimizations on computing $\Gamma(\sigma, K)$ by using dynamic programming.

▶ **Definition 8** (Labeling function, Υ). *Let K be a simplicial complex, and σ a q-simplex $\sigma \in K$. Let $\mathcal{M} : \mathbb{R}^{|L|} \to \mathbb{N} \cup \{0\}$ a function, which by taking a vector $V \in \mathbb{R}^{|L|}$ returns an integer $0 \leq i < |L|$ where i is the position of maximum value in V. We define the labeling function of a q-simplex as follows:*

$$\Upsilon(\sigma, K) = \mathcal{G}(i); i = \mathcal{M}(\Gamma(\sigma, K)). \tag{3}$$

Function $\Upsilon(\sigma, K)$ assigns to σ the most voting label $l \in L$ during computation of $\Psi(\sigma, K)$.

▶ **Definition 9** (Real label list, Y). *Let assume a given lexicographic order in X and another one in L. Y is a list of labels assigned to each element of X defined by:*

$$Y = (l_i \mid (x_i, l_i) \in T; x_i \in X; l_i \in L).$$

we call Y the real label list of X.

▶ **Definition 10** (Predicted label list, $\hat{Y}$). *Let K be a simplicial complex, T an association set, T' an uncompleted association set and L a label set. Assuming a lexicographic order in X, we define the predicted label list $\hat{Y}$ as:*

$$\hat{Y} = (\hat{y}_i \mid \hat{y}_i = \Upsilon(x_i, K); x_i \in X).$$

Note, when we assume $\mathbb{T} = T$ as known associations in basic cases of computation of Ψ, then $\hat{Y} = Y$, with Y the real labels list from definition 9. In another case, when using $\mathbb{T} = T'$ as associations then $\hat{Y}$ is a prediction of labels of any value in X.

▶ **Lemma 11.** *Let P be a point set and S a non-empty subset of P. Let C be a simplicial complex built from P. If K is a simplicial complex built on S with the same construction rules of C, then $K \subseteq C$. In other words: K is a sub-complex of C.*

Proof. Suppose by contradiction $K \not\subseteq C$. According to section 2.1: we have $\forall \sigma \in K; \tau \in \sigma \implies \tau \in K$. Now, if C is formed by all points in P, then C is a set of subsets of P. Since C and K have the same formation rules, moreover $S \subseteq P$, then $\forall \sigma \in K \implies \sigma \in C$. As well as $\forall \sigma \in K; \sigma \in C$ also $K \in C$, which implies $K \subseteq C$. That makes K in a sub-complex of C. But this contradicts the premise of $K \not\subseteq C$. ◀

▶ **Definition 12** (Simplicial complexes union operator, $\uplus$)**.** *Let S, X be two point sets. Let K, A be two simplicial complexes where K was built from S and A built from X respectively. The union operator $\uplus$ is defined by $K \uplus A = C$ where C is a simplicial complex built from $\{S \cup X\}$.*

▶ **Definition 13** (Filtration level access function, ψ)**.** *Let $\mathcal{F}$ be the set of all possible filtrations on a point set S. Let K be a simplicial complex built from S. The level access function $\psi : \mathcal{F} \times \mathbb{N} \to K$ is defined by $\psi(f, i) = K_i$. In other words, for a filtration $f \in \mathcal{F}$ and a level i, ψ returns the sub-complex in the i^{th} level of f. If $i \geq n$ with n the maximum level of f, then $\psi(f, n) = K_n$ is invoked.*

Let C be a simplicial complex built from P, let f' be a filtration of C (see Figure 4) with $n + 1$ levels:

$$\emptyset = C_0 \subseteq C_1 \subseteq C_2 \subseteq \cdots \subseteq C_n = C,$$

and let K be a simplicial complex built from S, with $S \subseteq P$, and let f be a filtration of K (see Figure 3) with $m + 1$ levels and $m \leq n$:

$$\emptyset = K_0 \subseteq K_1 \subseteq K_2 \subseteq \cdots \subseteq K_m = K.$$

Thus, for each level value i, we can get $C_i = \psi(f', i)$ and $K_i = \psi(f, i)$ according to Definition 13. As we said in Lemma 11 and definitions 1, 3 and 12 we can affirm $\forall i \leq n$:

$$K_i \leq C_i, \tag{4}$$

$$C_i - \{St(x, C_i) \mid x \in C_0 \wedge x \in X\} = K_i, \tag{5}$$

$$K_i \uplus \{St(x, C_i) \mid x \in C_0 \wedge x \in X\} = C_i. \tag{6}$$

That is, if we remove from C_i those simplices which contain elements of X, then we can get K_i. At the same time, by using the union operator of definition 12 and equation 6, if we include all elements of X in K_i step by step, we can get some sub-complex of C_i.

Now we define the algorithm.

3.2 Classification by using simplicial complexes and persistent homology

The proposed method computes the predicted label list $\hat{Y}$ corresponding to X according to definition 10 assuming T' is an incomplete association set. The entire process is summarized in Algorithm 1. We show a simplified example in Figure 4. The following section explains each step in detail.

Algorithm 1 TDABC: TDA-Based Classification Algorithm

Require: A training set $S \neq \emptyset$.
A testing set $X \neq \emptyset$ to be classified.
The incomplete association set T'.
Ensure: A prediction list $\hat{Y}$ of X by using T'.
1: Obtain the filtration f' (see Figure 4) of the simplicial complex C constructed by using S and X (Algorithms 2 and 3)
2: Obtaining the prediction list $\hat{Y} = (l_0, l_1, \cdots, l_{|X|})$ where each $l_i \in \hat{Y}$ is the most reliable label corresponding to $x_i \in X$, with $0 \leq i < |X|$ by using f' filtration (Algorithm 4)
3: **return** the prediction list $\hat{Y}$.

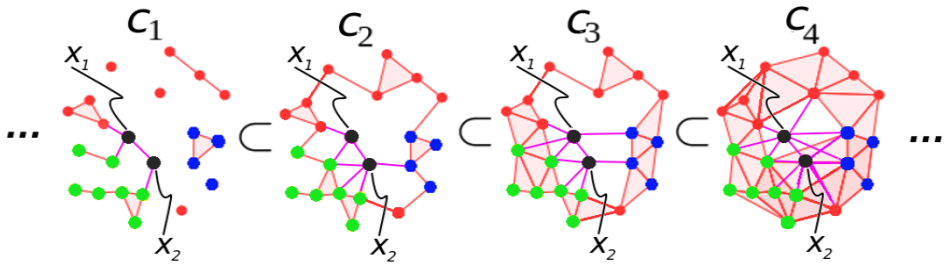


Figure 4 TDABC (see algorithm 1) using a 4-level fragment of a filtration f' from a C simplicial complex. Two black points $x_1, x_2 \in X$ must be classified. The colored points are elements in S, where each color represents different labels. By applying the Algorithm 1 in C_1, C_2, C_3, C_4 we get different labels for x_1 and x_2 depending on a selected C_i simplicial complex with $1 \leq i \leq 4$.

3.2.1 Building the simplicial complex C and its filtration f'.

We can build the simplicial complex C in two ways: using an incremental algorithm (Algorithm 2) or using a direct one (Algorithm 3). From Lemma 11 we know both algorithms are equivalents.

If we have a complex K from set S, we can use the incremental algorithm to insert in K every point $x \in X$. Each time we insert a point, we need to recompute or update the entire simplicial complex. In comparison to the direct algorithm, this strategy has the advantage of controlling resource consumption because it builds the simplicial complex step by step according to Equation 6. The direct strategy is useful when we need to classify a set and not just one point. In this case, it is better to build the complete simplicial complex once, in order to avoid updating the simplicial complex every time we insert a point.

3.2.2 Obtaining the most reliable label

Once we have a filtering f' from a simplicial complex C, we need to label all the elements of X, by using $\forall x \in X; \Upsilon(\{x\}, C)$ to assign the most voted label $l \in L$ during computation of $\Gamma(\{x\}, C)$ and $\Psi(\{x\}, C)$ according to definitions 6, **??** and 8.

As we know from Definition 13, C is the infinite level of the filtration f'. Theoretically the dimension of C can be $N-1; N = |P|$ in case all the vertices are connected with all. We need to check $\forall x \in X, Lk(x)$ which contains the other elements of C. But, if $\exists x' \in X; x' \in Lk(x)$ we need to analyze it recursively and so on until some $s \in S$ is found, which must contribute with its label. Resulting in a method with $O(|X| \cdot 2^N)$ is the worst case. Even with a table of dynamic programming to avoid recalculating the labels of the values twice, the computational

Algorithm 2 Incremental Construction of simplicial complex C

Require: A pre-existent filtration f from a non-empty simplicial complex K. A non-empty testing set X. An ξ increment value used to change the filtration level.
Ensure: f' a filtration of C simplicial complex (see Figure 4).

Let $n \leftarrow |f|$ be the maximum level of filtration f
Split X in n disjoint partitions $\{X_0, X_1, \cdots, X_n\}$
$f' \leftarrow f$
$i \leftarrow 0$
while $i \leq n$ **do**
    $K_i \leftarrow \psi(f, i)$, see definition 13
    $C_i \leftarrow K_i \uplus X_i$
    update f' with $\psi(f', i) \leftarrow C_i$
    $i \leftarrow i + \xi$
end while
return f'

Algorithm 3 Direct Construction of simplicial complex C

Require: A non-empty training set S. A non-empty testing set X. An ξ increment value used to change the filtration level. Let n be the maximum desired level of resulting filtration f'.
Ensure: f' a filtration of C simplicial complex (see Figure 4).

$f' \leftarrow \{\emptyset\}$
Unify S and X by $P \leftarrow S \cup X$
$i \leftarrow 0$
while $i \leq n$ **do**
    Build a simplicial complex C_i from P by using an i value [8, 1, 3, 2]
    Update i^{th} level of filtration f' with $\psi(f', i) \leftarrow C_i$
    $i = i + \xi$
end while
return f'

complexity would still be at least $O(2^N)$ that is required to initialize the table. If we could compute functions $\Psi(x, C)$ and $\Gamma(x, C)$ from definitions 6 y **??** without any change, it is very likely that the point $x \in X$ it would have all possible labels, due to the high degree of the analyzed simplices dimensions.

Two questions arise that address the calculation:

(a) How do we know which simplices of C that contain some $x \in X$ are reliable and which are noise?

(b) How can we make sure to assign a reliable label $\forall x \in X$?

We can use persistent homology to downsize the dimension of the problem. With persistence, we can get all topological invariants for all simplices dimension in f'. But we are just interested in working with q dimension; with $2 \leq q \ll N - 1$, which represents a complexity reduction of $O(|X| \cdot 2^{q+1})$ with q a constant value. Thus, we can say our method is linear in $|X|$. Those invariants are determined by persistence intervals with values $(birth, death)$ (see definition 4).

For long life invariants (high $death - birth$) we are in the presence of a topological feature, but for short life, we have noise [13, 8, 29]. We need to find the level of filtration C_i that

Algorithm 4 Labeling a testing set X set

Require: A filtration f' of a simplicial complex C.
A non-empty testing set X to classify.
The incomplete association set T' .
Ensure: A predicted labels list $\hat{Y}$ of X according to definitions 10.

```
D ← ComputePersistentHomology(f') [24, 9] where: D = {d_i | d_i = (birth, death)}
Get a desired persistent interval d ∈ D with d ∈ {MaxInt(D), RandInt(D), AvgInt(D)}

i = d[birth]
C_i = ψ(f', i) see definition 13
Ŷ ← {∅}
while X ≠ ∅ do
    x ∈ X
    l ← ϒ({x}, C_i) see definitions 6, ??, 8
    Ŷ ← Ŷ ∪ {l}
    X ← X/{x}
end while
return Ŷ
```

maximizes the number of topological features associated to each $x \in X$. At the same time, we would like sub-complex C_i to have as many useful-simplices (see definition 5) as possible. As a result, it is highly likely that we get a reliable label $\forall x \in X$.

As we have already described in Definition 4, a persistent interval $d = (birth, death)$ represents the life time of a simplex set Q, in which every q-simplex was created and destroyed on filtration levels $d[birth]$ and $d[death]$, respectively. Let D be the set of persistence interval of q-simplices, and $d \in D$, then $int(d) = d[death] - d[birth]$. We define three ways to find the desired persistent interval:

(a) The maximum persistent interval:

$$d_m = MaxInt(D) = Max(int(d)); \forall d \in D. \tag{7}$$

(b) A persistent interval selected in a random way:

$$d_r = RandInt(D) = random(D). \tag{8}$$

(c) The ceiling of persistent interval average:

$$d_a = AvgInt(D) = \lceil Avg(int(d)) \rceil; \forall d \in D. \tag{9}$$

Algorithm 4 executes the labeling process.

4 Results

We use the GUDHI library [19, 2, 1] to implement our TDA Based Classifier (TDABC), see Algorithm 1. GUDHI is one of the most complete libraries for building simplicial complexes and computing persistent homology [2, 21, 18, 1, 3].

To evaluate our proposed TDABC algorithm we use the Iris dataset [10]. The dataset contains 3 classes of 50 instances each, where each class refers to a type of Iris plant. One class is linearly separable from the other two; the latter are NOT linearly separable from each

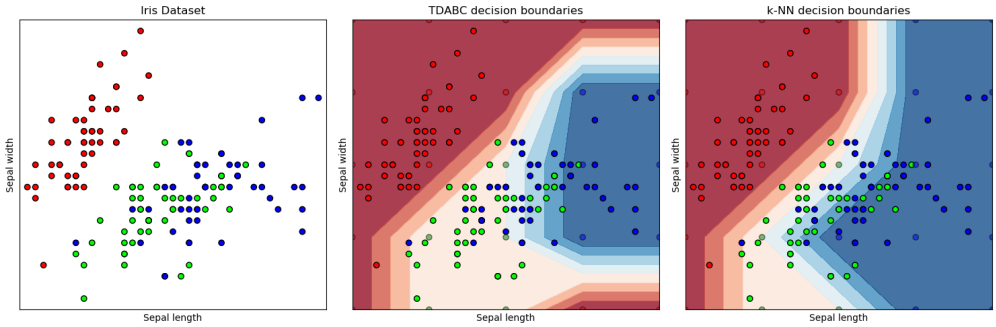


Figure 5 Iris dataset (first two components)[left], decision boundaries from TDABC by using RandInt-function [center], and decision boundaries from k-NN [right].

other [10] (see Figure 5). L is the label set where $L = \{\text{“}Setosa\text{”}, \text{“}Versicolor\text{”}, \text{“}Virginica\text{”}\}$. Each sample in the Iris dataset is a 5-tuple, defined by:

$$\text{5-tuple} = (sepal\_length, sepal\_width, petal\_length, petal\_width, label)$$

We use the class of Iris plant as a predicted attribute. The left side of Figure 5 shows the distribution of the dataset (we choose the first two components for plotting purposes). We build a point set P using the first four components of every Iris dataset Sample.

4.1 Classifier Evaluation

We create three different versions on our TDABC method depending on the applied selection function (see equations 7, 8 and 9): TDABC+MaxInt (TDABC-M), TDABC+RandInt (TDABC-R) and TDABC+AvgInt (TDABC-A). As a baseline algorithm, we choose the sci-kit learn k-Nearest Neighbors (k-NN) Classifier [22] to compare our results.

To make our evaluation more robust and avoid overfitting and underfitting we use Cross-Validation [22, 15, 20] method. The idea of Cross-Validation is to divide the data set P into equal pieces or folds. One piece of them is taken as the testing set X, and remaining folds as the set S. When we repeat this process, and folds of different sizes are generated, the method is called Repeated Cross-Validation.

Let R be the fold number in our Cross-Validation approach to avoid confusion with our use of k in k-NN. We decide to execute the repeated R-FOLD 5 times (n=5) and R will be one of (5, 10, 15, 20, 25) values, depending on the value of n, i.e.: n=1, R=5; n=2, R=10 and so on. For any value of R, we use (R-1)-fold for a training set as S and the remaining fold as testing data as X in each iteration. The overall evaluation process is shown in Figure 6.

In machine learning algorithms it is common to use parameters whose values are set before the learning process begins. Those parameters are called hyper-parameters [22, 15, 20]. By contrast, the values of other parameters are derived via training. In our evaluation, we need to find the best values for hyper-parameters k, and q (Param Estimator process in Figure 6). For k-NN we found k=15 as a good number of neighbors; we get it by using the hyper-parameter estimators from scikit-learn [22].

For the TDABC algorithm, we need to know the max simplex dimension q to control the VR-complex construction process. We fix $q = 3$ because the hardware limitation:16 GB RAM, Intel® Core™ i5 CPU 750 a 2.67GHz.

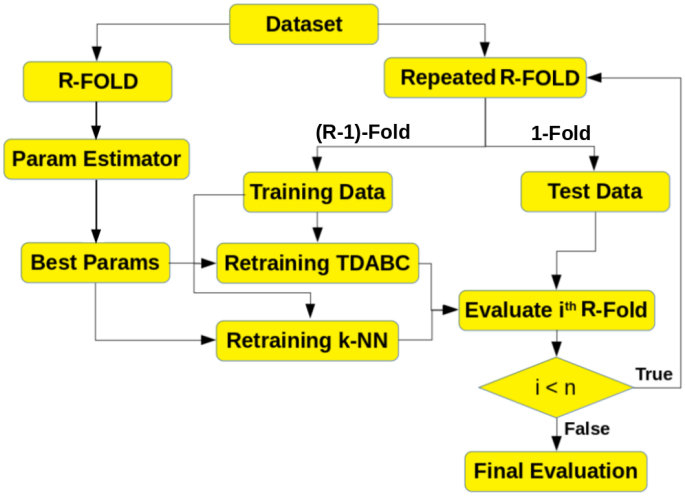


Figure 6 Repeated Cross Validation overall process to compare TDABC variants and k-NN.

4.2 Metrics for Classifiers Evaluation

In the interest of evaluating the performance of the proposed and baseline classifiers, we need to compute several metrics like: Accuracy (Acc), Precision (P), Recall (R), False Positive Rate (FPR), F1 measure ($F1$ or Armonic Measure of P and R), Mean Squared Error (MSE). As a graphical and general evaluation of a classifier we use the Confusion Matrix. With the exception of MSE, aforementioned metrics are defined by using the True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN). Since the Iris dataset has multiple classes we consider TP_l, FP_l, TN_l, FN_l for each class $l \in L$. Let Y be the real label set $\forall x \in X$ (see definition 9). Let $\hat{Y}$ be the predicted label set $\forall x \in X$ computed by Algorithm 1, $n = |Y| = |\hat{Y}| = |X|$.

Therefore, we get each metric as the average of metric computation by each class $l \in L$,

according to equations from 10 to 19:

$$TP_l = \sum_{i=1}^{n} \mathcal{I}(l = \hat{y}_i) \cdot \mathcal{I}(\hat{y}_i = y_i), \tag{10}$$

$$FP_l = \sum_{i=1}^{n} \mathcal{I}(l = \hat{y}_i) \cdot \mathcal{I}(\hat{y}_i \neq y_i), \tag{11}$$

$$TN_l = \sum_{i=1}^{n} \mathcal{I}(l \neq \hat{y}_i) \cdot \mathcal{I}(\hat{y}_i = y_i), \tag{12}$$

$$FN_l = \sum_{i=1}^{n} \mathcal{I}(l \neq \hat{y}_i) \cdot \mathcal{I}(\hat{y}_i \neq y_i), \tag{13}$$

$$Acc = \frac{1}{|L|} \cdot \sum_{l \in L} \frac{TP_l + TN_l}{TP_l + TN_l + FP_l + FN_l}, \tag{14}$$

$$P = \frac{1}{|L|} \cdot \sum_{l \in L} \frac{TP_l}{TP_l + FP_l}, \tag{15}$$

$$R = \frac{1}{|L|} \cdot \sum_{l \in L} \frac{TP_l}{TP_l + FN_l}, \tag{16}$$

$$PR = \frac{1}{|L|} \cdot \sum_{l \in L} \frac{FP_l}{TN_l + FP_l}, \tag{17}$$

$$F1 = 2 \cdot \frac{P \cdot R}{P + R}, \tag{18}$$

$$MSE = \frac{1}{n} \cdot \sum_{i=1}^{n} (\hat{y}_i - y_i)^2. \tag{19}$$

We compute these 10 metrics [15, 4, 22] for any iteration of our repeated R-Fold strategy.

4.3 Comparison

Since $k \in (5, 10, 15, 20, 25)$ we execute each classifier a total number of $\sum_{i=1}^{5} k_i = 75$ times. We then compute the average of each metric and show the results in Table 1.

Name	Acc	P	R	FPR	MSE	F1
k-NN	0.97	0.92	0.89	0.04	0.10	0.91
TDABC-R	0.96	0.93	0.90	0.06	0.14	0.92
TDABC-A	0.95	0.91	0.88	0.05	0.29	0.90
TDABC-M	0.93	0.88	0.83	0.10	0.57	0.85

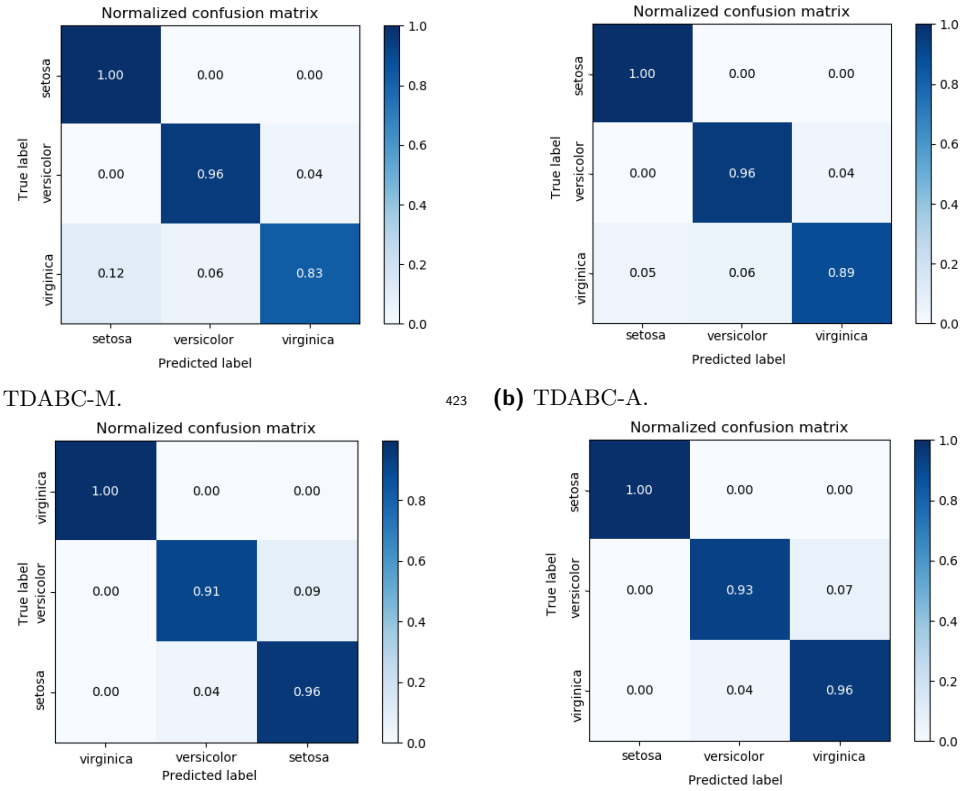
Table 1 Comparison of General Metrics by Classifier.

On the other hand, we compute the Overall Confusion Matrix (CM) for each classifier concatenating all sets into one, resulting in two big sets of predicted labels and real labels:

$$\hat{Y} = (\hat{Y}_1, \hat{Y}_2, \cdots, \hat{Y}_{75}), \tag{20}$$

$$Y = (Y_1, Y_2, \cdots, Y_{75}). \tag{21}$$

where $\hat{Y}_i$ is the predicted labels set and Y_i the real labels set, both resulting from i^{th} execution of Repeated K-Fold. As we respect the relations between each predicted and real labels we can easily calculate the CM for each classifier. The result is shown in the Figure 7.



(a) TDABC-M. **(b)** TDABC-A.

(c) TDABC-R. **(d)** k-NN.

Figure 7 Confusion Matrices.

5 Discussion

In section 3.2.1 we presented two algorithms to construct the simplicial complex using testing X and training S point sets. One is an incremental algorithm (Algorithm 2) and another is a direct one (Algorithm 3). Both are equivalents in their final result, as we have shown in Lemma 11. However, the incremental algorithm is recommended when a previous filtered simplicial complex was computed. Thus, a few values are classified if needed and you avoid recomputing all every time. This approach gives us the possibility of storing the filtered simplicial complex as a sort of knowledge representation (or topological database) for a specific dataset. The Direct algorithm is useful for training the TDABC with a new dataset or for classifying an entire point set at the same time.

The proposed voting system of assigning the most reliable label (see Definition 8), also offers the possibility of dealing with elements belonging to multiple classes at the same time. Perhaps in a more complex dataset, this possibility can be exploited in order to eliminate intra-class and extra-class variability.

The persistent homology is vital in Algorithm 4 to reduce the search space by taking advantage of topological features that are encoded inside selected persistent intervals.

Another interesting aspect is that the proposed method does not necessarily generate same-sized "neighborhoods" for each point to be classified as k-NN does. To get multi-sizes neighborhoods, no previous training is required, but k-NN needs a tuning process before inferring the best k hyper-param in order to get better results.

For the evaluation process, we use 10 metrics to cover as much as possible of analyzed classification methods. In the end, we compute the overall confusion matrix for our three classifiers (TDABC-R, TDABC-A, TDABC-M) and also for the k-NN classifier (see Figure 7). Table 1 shows metric results in general terms, and our three TDABC have good classification rates. However, the TDABC-R method is better than other TDBC analyzed methods, being close to the k-NN classifier. On the other hand, TDABC-M is the worst, showing us that more persistent topological features are not necessarily the best. Thus, more persistent topological features could contain simplices with elements from several classes, resulting in several errors during the labeling process with $\Psi(\sigma)$ function.

A more detailed analysis of the confusion matrices shows that for some classes the quality of the proposed classification algorithms is equal to or surpasses k-NN. The TDABC-R gets better results at classifying the Virginica class, see decision boundaries in Figure 5. TDABC-A and TDABC-M are better at classifying Versicolor class and they are similar to k-NN classifying Setosa with 100% accuracy. Therefore, in a detailed analysis of classification per-class, our method is better than k-NN by detecting non-isolated classes.

6 Conclusions

In this work, we develop a new classification method exclusively using Topological Data Analysis tools. In particular, simplicial complexes and persistent homology. We also create algorithms to build filtered simplicial complexes combining test and training sets. We also create functions to label simplicial complexes and show how to use them in practice. The use of persistent homology was determinant to reduce the complexity of the search space, giving us a robust framework to understand data shapes and use it for classification. We compare our proposed TDABC and its variants with k-NN obtaining similar rates. But, in non-isolated classes, our methods were better. We think our method can be applied to solve at least the same problems that k-NN solves. In this preliminary study, we experimentally prove the possibility of classifying data by using TDA directly, with good accuracy rates. Hopefully, this will open further research opportunities and of understanding data.

7 Future Works

We have shown preliminary results about TDA-based classification. To our knowledge, this is the first time TDA is directly used for classification. We still need to further explore new data structures and algorithms to reveal and take advantage of the topological properties of data. Furthermore, we want to process more complex datasets and evaluate the behavior of our methods in the presence of noisy and high-dimensional data.

A new challenge is to find the lower simplicial complex dimension q needed to capture all topological features from an arbitrary and high dimensional point set P. A lower q value is useful for reducing the size of the searching space as much as possible and maintaining our TDABC method in a linear or quasi-linear computation worst case. We do not know if there is any relation between q and dataset sample dimensions. Finding this relation, we will be able to understand how the optimum simplicial dimension q' can be found for any complex dataset.

We want to explore the possibilities of our method in Feature Engineering as a data imputation method in Missing Data Analysis (MDA). Finally, we want to better understand the sort of problems and datasets in which our method could outperforms other methods.

8 References

[1] Jean-Daniel Boissonnat and Karthik C. S. 'An Efficient Representation for Filtrations of Simplicial Complexes'. In: *ACM Trans. Algorithms* 14.4 (2018), 44:1–44:21. URL: https://doi.org/10.1145/3229146.

[2] Jean-Daniel Boissonnat, Karthik C. S. and Sébastien Tavenas. 'Building Efficient and Compact Data Structures for Simplicial Complexes'. In: *Algorithmica* 79.2 (2017), pp. 530–567. DOI: 10.1007/s00453-016-0207-y. URL: https://doi.org/10.1007/s00453-016-0207-y.

[3] Jean-Daniel Boissonnat and Clément Maria. 'The Simplex Tree: An Efficient Data Structure for General Simplicial Complexes'. In: *Algorithmica* 70.3 (Nov. 2014), pp. 406–427. ISSN: 1432-0541. DOI: 10.1007/s00453-014-9887-3. URL: https://doi.org/10.1007/s00453-014-9887-3.

[4] Michael W Browne. 'Cross-Validation Methods'. In: *Journal of Mathematical Psychology* 44.1 (2000), pp. 108–132. ISSN: 0022-2496. DOI: https://doi.org/10.1006/jmps.1999.1279. URL: http://www.sciencedirect.com/science/article/pii/S0022249699912798.

[5] Gunnar Carlsson. 'Topology and data'. In: *Bulletin of the American Mathematical Society* 46.2 (Jan. 2009), pp. 255–308.

[6] Hamish Carr, Chistoph Garth and Tino Weinkauf. *Topological Methods in Data Analysis and Visualization IV. Theory, Algorithms, and Applications*. Mathematics and Visualization. Gewerbestrasse, Switzerland: Springer International Publishing, 2017. ISBN: 9783319446844. DOI: 10.1007/978-3-319-44684-4.

[7] Milka Doktorova and Afra Zomorodian. 'Constructing Simplicial Complexes over Topological Spaces'. MsC. Thesis. MA thesis. New Hampshire, USA, 2012.

[8] Herbert Edelsbrunner and John Harer. *Computational Topology - an Introduction*. Michigan, USA: American Mathematical Society, 2010. ISBN: 978-0-8218-4925-5. URL: http://www.ams.org/bookstore-getitem/item=MBK-69.

[9] Herbert Edelsbrunner and Dmitriy Morozov. 'Persistent homology: theory and practice'. In: *European Congress of Mathematics* (Jan. 2014), pp. 31–50. DOI: 10.4171/120-1/3.

[10] Ronald A. Fisher. 'The Use of Multiple Measurements in Taxonomic Problems'. In: *Annals of Eugenics* 7.7 (1936), pp. 179–188.

[11] Kathryn Garside et al. 'Topological data analysis of high resolution diabetic retinopathy images'. In: *PLOS ONE* 14.5 (May 2019), pp. 1–10. DOI: 10.1371/journal.pone.0217413. URL: https://doi.org/10.1371/journal.pone.0217413.

[12] William Gasarch, Brittany Terese Fasy and Bei Wang. 'Open Problems in Computational Topology'. In: *SIGACT News* 48.3 (Sept. 2017), pp. 32–36. ISSN: 0163-5700. DOI: 10.1145/3138860.3138867. URL: http://doi.acm.org/10.1145/3138860.3138867.

[13] Robert Ghrist. 'Barcodes: The persistent topology of data'. In: *BULLETIN (New Series) OF THE AMERICAN MATHEMATICAL SOCIETY* 45 (Feb. 2008), pp. 61–75. DOI: 10.1090/S0273-0979-07-01191-3.

[14] Mostafa Hossny et al. 'Driving behaviour analysis using topological features'. In: *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. Budapest, Hungary: IEEE, Oct. 2016, pp. 003258–003263. DOI: 10.1109/SMC.2016.7844736.

[15] Nathalie Japkowicz and Mohak Shah. *Evaluating Learning Algorithms: A Classification Perspective*. New York, NY, USA: Cambridge University Press, 2011. ISBN: 9780521196000.

[16] Javier Lamar-Leon et al. 'Persistent homology-based gait recognition robust to upper body variations'. In: *2016 23rd International Conference on Pattern Recognition (ICPR)*. Dec. 2016, pp. 1083–1088. DOI: 10.1109/ICPR.2016.7899780.

[17] Ngoc-Khuyen Le et al. 'Construction of the Generalized Czech Complex'. In: *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*. May 2015, pp. 1–5. DOI: 10.1109/VTCSpring.2015.7145759.

[18] Clément Maria. 'Algorithms and data structures in computational topology. (Algorithmes et structures de données en topologie algorithmique)'. PhD thesis. University of Nice Sophia Antipolis, France, 2014. URL: https://tel.archives-ouvertes.fr/tel-01123744.

[19] Clément Maria et al. 'The Gudhi Library: Simplicial Complexes and Persistent Homology'. In: *Mathematical Software – ICMS 2014*. Ed. by Hoon Hong and Chee Yap. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.

[20] Tom M. Mitchell. *Machine learning, International Edition*. McGraw-Hill Series in Computer Science. McGraw-Hill, 1997. ISBN: 0071154671. URL: http://www.worldcat.org/oclc/61321007.

[21] Nina Otter et al. 'A roadmap for the computation of persistent homology'. In: *EPJ Data Science* 6.1 (Aug. 2017), p. 17. ISSN: 2193-1127. DOI: 10.1140/epjds/s13688-017-0109-5. URL: https://doi.org/10.1140/epjds/s13688-017-0109-5.

[22] Fabian Pedregosa et al. 'Scikit-learn: Machine Learning in Python'. In: *Journal of Machine Learning Research* 12 (Jan. 2012).

[23] Lee M. Seversky, Shelvi Davis and Matthew Berger. 'On Time-Series Topological Data Analysis: New Data and Opportunities'. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. June 2016, pp. 1014–1022. DOI: 10.1109/CVPRW.2016.131.

[24] Vin de Silva, Dmitriy Morozov and Mikael Vejdemo-Johansson. 'Persistent Cohomology and Circular Coordinates'. In: *Discrete & Computational Geometry* 45.4 (June 2011), pp. 737–759. ISSN: 1432-0444. DOI: 10.1007/s00454-011-9344-x. URL: https://doi.org/10.1007/s00454-011-9344-x.

[25] Ann E. Sizemore et al. 'The importance of the whole: Topological data analysis for the network neuroscientist'. In: *Network Neuroscience* 3.3 (2019), pp. 656–673. DOI: 10.1162/netn_a_00073. eprint: https://doi.org/10.1162/netn_a_00073. URL: https://doi.org/10.1162/netn_a_00073.

[26] Primoz Ŝkraba. 'Persistent Homology and Machine Learning'. In: *Informatica. An International Journal of computing and Informatics* 42.2 (Jan. 2018), pp. 253–258.

[27] Yuhei Umeda. 'Time Series Classification via Topological Data Analysis'. In: *Transactions of the Japanese Society for Artificial Intelligence* 32 (May 2017), D–G72$_1$. DOI: 10.1527/tjsai.D-G72.

[28] Vinay Venkataraman, Karthikeyan Natesan Ramamurthy and Pavan K. Turaga. 'Persistent homology of attractors for action recognition'. In: *2016 IEEE International Conference on Image Processing, ICIP 2016, Phoenix, AZ, USA, September 25-28, 2016*. 2016, pp. 4150–4154. DOI: 10.1109/ICIP.2016.7533141. URL: https://doi.org/10.1109/ICIP.2016.7533141.

[29] Afra Zomorodian. 'Fast construction of the Vietoris-Rips complex'. In: *Computers & Graphics* 34 (3 June 2010), pp. 263–271.

[30] Afra Zomorodian. *Topology for Computing*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2009. ISBN: 0521136091.