Advances in Data Analysis and Classification
https://doi.org/10.1007/s11634-023-00548-4

REGULAR ARTICLE

A topological data analysis based classifier

Rolando Kindelan[1,3] · José Frías[4] · Mauricio Cerda[2] · Nancy Hitschfeld[1]

Received: 4 February 2022 / Revised: 15 May 2023 / Accepted: 12 June 2023

Abstract
Topological Data Analysis (TDA) is an emerging field that aims to discover a dataset's underlying topological information. TDA tools have been commonly used to create filters and topological descriptors to improve Machine Learning (ML) methods. This paper proposes a different TDA pipeline to classify balanced and imbalanced multi-class datasets without additional ML methods. Our proposed method was designed to solve multi-class and imbalanced classification problems with no data resampling preprocessing stage. The proposed TDA-based classifier (TDABC) builds a filtered simplicial complex on the dataset representing high-order data relationships. Following the assumption that a meaningful sub-complex exists in the filtration that approximates the data topology, we apply Persistent Homology (PH) to guide the selection of that sub-complex by considering detected topological features. We use each unlabeled point's link and star operators to provide different-sized and multi-dimensional neighborhoods to propagate labels from labeled to unlabeled points. The

José Frías, Mauricio Cerda and Nancy Hitschfeld have contributed equally to this work.

✉ Rolando Kindelan
rkindela@dcc.uchile.cl

José Frías
frias4@cimat.mx

Mauricio Cerda
mauricio.cerda@uchile.cl

Nancy Hitschfeld
nancy@dcc.uchile.cl

1 Computer Sciences Department, Faculty of Mathematical and Physical Sciences, University of Chile, 851 Beauchef Av., 8370456 Santiago, Metropolitan Region, Chile

2 Integrative Biology Program, Institute of Biomedical Sciences, Center for Medical Informatics and Telemedicine, Faculty of Medicine, Universidad de Chile, 1027 Independencia Av., Santiago, Metropolitan Region, Chile

3 Medical and Biophysics Center, University of Oriente, Patricio Lumumba S/N, Santiago, Cuba

4 Center for Research in Mathematics, Jalisco S/N, Col. Valenciana, 63023 Guanajuato, Guanajuato, Mexico

Published online: 01 July 2023

labeling function depends on the filtration's entire history of the filtered simplicial complex and it is encoded within the persistence diagrams at various dimensions. We select eight datasets with different dimensions, degrees of class overlap, and imbalanced samples per class to validate our method. The TDABC outperforms all baseline methods classifying multi-class imbalanced data with high imbalanced ratios and data with overlapped classes. Also, on average, the proposed method was better than K Nearest Neighbors (KNN) and weighted KNN and behaved competitively with Support Vector Machine and Random Forest baseline classifiers in balanced datasets.

Keywords Topological data analysis · Persistent homology · Simplicial complex · Supervised learning · Classification · Machine learning

Mathematics Subject Classification 55N31 · 62R40 · 68T09 · 62R07 · 68T10

1 Introduction

Classification is a Machine Learning (ML) task that uses known data labels to label data with an unknown category. Classification faces challenges such as high dimensionality, noise, and imbalanced data distributions. Topological Data Analysis (TDA) has been successful in reducing dimensionality (Luo et al. 2021), and it has demonstrated robustness to noise by inferring a dataset's underlying topology (Edelsbrunner and Harer 2010). However, how TDA can classify imbalanced datasets has not been explored.

Persistent Homology (PH) is a powerful tool in TDA that captures the topological features in a nested family of simplicial complexes built on data, according to an incremental scale value (Edelsbrunner and Harer 2010). These topological features are encoded, considering scale values at which they appear (*born*) and disappear or merge (*die*). The numerical difference between birth and death scales is the topological feature's persistence, also known as the *lifetime* or *lifespan*. The evolution of the simplicial structure is encoded using high-level representations called barcodes and persistence diagrams (Edelsbrunner and Harer 2010).

Regarding the relation between PH and classification problems, existing approaches consider hybrid TDA+ML methods, which combine topological information with conventional ML classifiers. Topological information is extracted from persistence diagrams and barcodes, for instance, by vectorizing (Adams et al. 2017; Bubenik and Dłotko 2017), computing distances, summarizing persistence diagrams via topological curves (Chung and Lawson 2022), or kernel approaches (Carrière et al. 2017; Hofer et al. 2017; Atienza et al. 2020; Carriere et al. 2020). A comprehensive Topological ML (TDA+ML) survey was presented in Hensel et al. (2021). More recently, in Ali et al. (2022), the authors study thirteen methods of vectorizing persistence homology barcodes and persistence diagrams, discovering that the best-performing method is simple (naive) vectorization obtained by collecting basic statistical quantities associated with (the multiset of) intervals in a given barcode.

Examples of these hybrid TDA+ML methods are TDA+SVM for high-resolution diabetic retinopathy images classification (Garside et al. 2019), and TDA+KNN,

TDA+Convolutional Neural Network, and TDA+SVM for time series classification (Venkataraman et al. 2016; Seversky et al. 2016; Umeda 2017). TDA+RF and TDA+SVM for image, triangular meshes, and cloud classification in Ali et al. (2022). Self-Organized Maps were combined with PH tools to cluster and classify time series in the financial domain (Majumdar and Laha 2020). There are also TDA applications in Deep Learning to address the interpretability problem (Carlsson and Gabrielsson 2020), to regularize loss functions (Gabrielsson et al. 2020), to build a persistence layer to consider topological information during learning (Gabrielsson et al. 2020).

Regarding imbalanced data distributions, approaches to address the imbalanced classification problem can be mainly classified into data level (also known as resampling), and algorithmic level approaches (Fernández et al. 2018). Data resampling methods balance data by augmenting or removing samples from minority or majority classes. The Synthetic Minority Oversampling Technique (SMOTE) is the conventional geometric approach to balance classes by oversampling the minority class (Chawla et al. 2002). Multiple variations of SMOTE have been developed (Goyal et al. 2021), including novel approaches such as the SMOTE-LOF, which considers the Local Outlier Factor (Asniar and Surendro 2022) to identify noisy synthetic samples. Furthermore, overlapping samples from different classes have been a big issue in imbalance problems. Neighborhood under-sampling from the majority class on the overlapped region has been applied to achieve better results (Vuttipittayamongkol and Elyan 2020). These heuristics are simple and can be combined with any classifier as they modify the training set, although they assume data points can always be discarded or generated. In contrast, SVM, RF, or neural network adaptations modify their objective function to give higher relative importance to minority class samples (Ibrahim et al. 2021). More related to our work, Zhang et al. (2017) proposes the Rare-class Nearest Neighbour (KRNN), which defines a dynamic neighborhood based on the inclusion of at least k positive samples (Zhang et al. 2017).

Undersampling and oversampling techniques can be applied as a preprocessing stage of any classifier; however, it could be challenging to devise a winning resampling approach to apply in a multi-class imbalanced data classification problem. Having classes A, B, and C, class A can be a majority class regarding class B, and B can be, at the same time, a majority class concerning class C, making it hard to resample B. This scenario is known as the problem of multi-minority and multi-majority classes (Fernández et al. 2018). A typical approach is to split the problem into multiple binary-imbalanced data classification subproblems and perform resampling per subproblem, but other issues may arise. Addressing the multi-class imbalanced classification problem has the advantage of considering class relationships. On the contrary, we can lose information due to binarization (Fernández et al. 2018).

According to Fernández et al. (2018), the imbalance ratio is not the only cause of performance degradation on imbalance problems. Another big concern is the intrinsic data complexity. When samples from different classes are linearly separable standard classifiers behave well. Yet, even state-of-the-art methods may fail where noise and overlapped classes complicate the classification. Gaining knowledge concerning data complexity may help creating successful approaches to deal with imbalanced data classification problems. In this context, TDA has proven to be a well-established

tool for understanding the data topology that can be applied to unravel the intrinsic complexities of data.

This work proposes a method entirely based on TDA to classify imbalanced and noisy datasets without resampling. The main idea is to provide multidimensional and multi-size neighborhoods around each unlabeled point. We use topological invariants computed through PH to detect appropriate neighborhoods. These neighborhoods are then used to propagate labels from labeled to unlabeled points. A preliminary version of this work is available in Kindelan et al. (2021). The same method is applied to balanced, binary- and multi-imbalanced data. We conduct experiments in eight datasets covering different aspects like class overlapping (non-easily separable classes), multiple imbalanced ratios, and high dimensions (more dimensions than samples). We compare the proposed method with KNN, weighted-KNN, Linear SVM, and RF baseline classifiers. The KNN algorithm is one of the most popular supervised classification methods used in the backbone of SMOTE techniques. The second baseline method is an enhanced version of KNN, the weighted KNN (WKNN) especially suited for imbalanced datasets. Linear SVM and RF, two popular classifiers, are also applied as algorithmic approaches to deal with imbalanced data.

This document is organized as follows. The fundamental concepts and mathematical foundations used in this work are presented in Sect. 2. Section 3 explains the concepts, algorithms, and methodology related to the proposed classification method. Next, Sect. 4 describes the experimental protocol to assess the proposed and baseline algorithms. Section 5 includes a discussion of the results and the proposed solution. Conclusions are presented in Sect. 6.

2 Fundamental concepts

This section introduces mathematical definitions to explain our proposed method; for a complete theoretical basis, see Edelsbrunner and Harer (2010) and Rabadan and Blumberg (2019).

2.1 Simplicial complexes

Simplicial complexes are combinatorial and algebraic objects that can represent a discrete space encoding topological features of the data space.

Definition 1 (*Simplicial complex and simplices*) An *abstract simplicial complex* $\mathcal{K}$, is a set of non-empty sets such that if σ is an element of $\mathcal{K}$, then every subset $\tau \subset \sigma$ is also in $\mathcal{K}$ (Rabadan and Blumberg 2019). Each element σ of $\mathcal{K}$ is called a simplex (simplices in plural), as we show in Fig. 1. Every subset τ of σ is a *face* of σ and has σ as a *coface*. We represent the face and coface relationships as $\tau \leq \sigma$ and $\sigma \geq \tau$, respectively. When a simplex σ does not have proper cofaces, it is known as a *maximal simplex*.

The *dimension* of a simplex σ is defined as $dim(\sigma) = |\sigma| - 1$, and the dimension of $\mathcal{K}$ is $dim(\mathcal{K}) = \max\{dim(\sigma) \mid \sigma \in \mathcal{K}\}$. We denote by $\mathcal{K}^{(0)}$ the vertex set of $\mathcal{K}$.

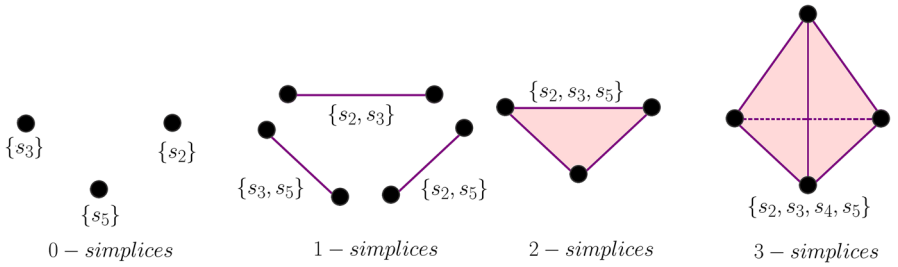

Fig. 1 Given a point set $\{s_2, s_3, s_4, s_5\}$, we show examples of low dimensional simplices, with 0-simplices as black points, 1-simplices as purple line segments, 2-simplices and the 3-simplex as red triangles

There are many possible ways to build abstract simplicial complexes on point sets, though two of the most popular are the Čech and Vietoris-Rips (*VR*) complexes, see (Ghrist 2008; Edelsbrunner and Harer 2010). From a computational perspective, *VR* complex is more practical than Čech during the construction phase since it only requires computing distances between points instead of checking all possible intersections of balls centered at the points for all radii. Besides, the Rips complex is a flag complex, meaning its 0 and 1-dimensional simplices can fully characterize it. Consequently, there is no need to store it explicitly, resulting in significant savings in memory resources.

Definition 2 (*Star and Link*) Let $\mathcal{K}$ be a simplicial complex, and $\sigma \in \mathcal{K}$ be a q-simplex. The *star* of σ in $\mathcal{K}$ is the set of all cofaces of σ in $\mathcal{K}$ (Edelsbrunner and Harer 2010):

$$St_{\mathcal{K}}(\sigma) = \{\tau \in \mathcal{K} \mid \sigma \leq \tau\}. \tag{1}$$

The *link* of σ is the set of simplices with cofaces in $St_{\mathcal{K}}(\sigma)$ that do not share any face with σ (Edelsbrunner and Harer 2010):

$$Lk_{\mathcal{K}}(\sigma) = \{\tau \in \mathcal{K} \mid \tau \cup \sigma \in \mathcal{K} \text{ and } \tau \cap \sigma = \emptyset\}, \tag{2}$$

or equivalently:

$$Lk_{\mathcal{K}}(\sigma) = \bigcup_{\mu \in St_{\mathcal{K}}(\sigma)} \{\mu \setminus \sigma\}. \tag{3}$$

Figure 2 presents an example of the star and link of the 0-simplex $\{s_4\}$ in a given simplicial complex $\mathcal{K}$ built on a point set $S = \{s_2, s_3, s_4, s_5\}$.

2.2 Persistent homology

The objective of PH is to track how topological features on a topological space appear and disappear when a scale value (usually a radius) varies incrementally, in a process known as filtration (Edelsbrunner and Harer 2010).

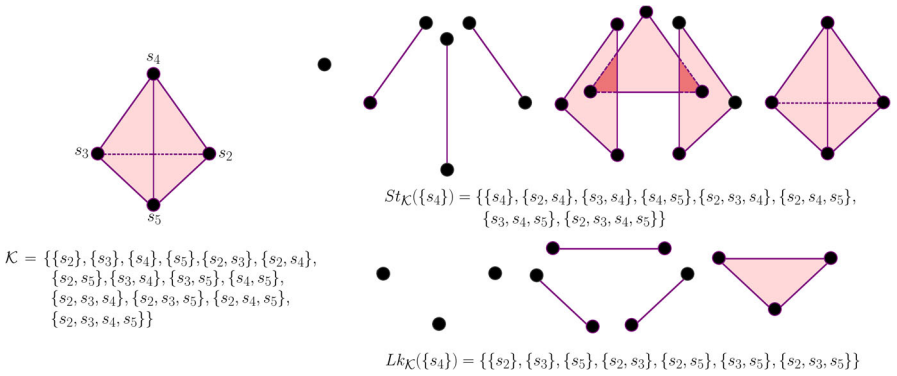

Fig. 2 Example of $St_{\mathcal{K}}(\{s_4\})$ and $Lk_{\mathcal{K}}(\{s_4\})$ on a given simplicial complex (a tetrahedron and all its faces) $\mathcal{K}$

Definition 3 (*Filtration value of a q-simplex*) Let $\mathcal{K}$ be a simplicial complex. Let $\sigma \in \mathcal{K}$ be a q-simplex. A function $f : \mathcal{K} \to \mathbb{R}^+$ satisfying $f(\tau) \leq f(\sigma)$, whenever $\tau \leq \sigma$ is called a level-set function. When $\mathcal{K}$ is endowed with such a level-set function f, we call it a *filtered simplicial complex*, and therefore we can get a sub-complex $\mathcal{K}_i = \{\sigma \in \mathcal{K};\, f(\sigma) \leq \varepsilon_i\}$ for any level value $\varepsilon_i > 0$, namely $\mathcal{K}_i = f^{-1}([0, \varepsilon_i])$. For a simplex $\sigma \in \mathcal{K}$, we call $f(\sigma)$ its filtration value.

Definition 4 (*Filtration*) Let $\mathcal{K}$ be a filtered simplicial complex. A *filtration* of $\mathcal{K}$ is a succession of increasing sub-complexes of $\mathcal{K}$:

$$\emptyset \subseteq \mathcal{K}_0 \subseteq \mathcal{K}_1 \subseteq \mathcal{K}_2 \subseteq \mathcal{K}_3 \subseteq \cdots \subseteq \mathcal{K}_n = \mathcal{K}.$$

We consider $\mathcal{K}_i = \{\sigma \in \mathcal{K};\, f(\sigma) \leq \varepsilon_i\}$. These sub-complexes form a filtration of $\mathcal{K}$. Note that the condition $\tau \leq \sigma \implies f(\tau) \leq f(\sigma)$ means that in a filtered simplicial complex $\mathcal{K}$ with the filtration associated to $f(\cdot)$, every simplex $\tau \in \mathcal{K}$ appears before all its cofaces.

Different types (or dimensions) of topological features exist in a filtered simplicial complex. The topological features of dimension 0 are the connected components. The topological features in dimension 1 are 1-cycles (closed chains of 1-simplices), which appear as holes. In dimension 2, closed chains of 2-simplices connected by 1-simplices create voids. In a general setting, a topological feature of dimension j is a closed chain of j-simplices, which are not the boundary of a chain of $(j+1)$-simplices. The simplicial homology with coefficients in $\mathbb{Z}/2\mathbb{Z} = \{0, 1\}$ captures the collection of j-cycles which are not the boundary of a chain of $(j+1)$-simplices and creates an equivalence relation between them, the j-th homology group H_j. An associated invariant to each homology group H_j is the j-th Betti number that denotes the number of j-dimensional topological features or $rank(H_j)$.

PH captures the behavior and evolution of homology across the filtration levels, detecting when a topological feature is created ("birth") and when it disappears or is merged with a previous one ("death"). The topological information captured via PH

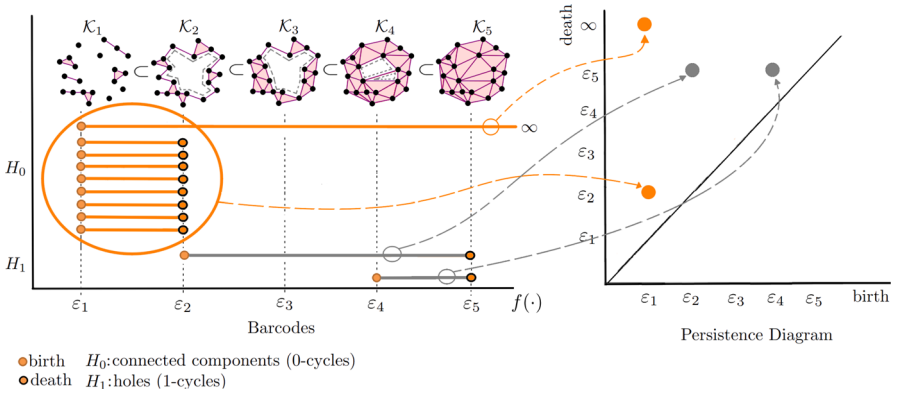

Fig. 3 A fragment of a simplicial complex filtration is presented with some selected topological features. Initially, nine connected components in $\mathcal{K}_1$ were merged into one in $\mathcal{K}_2$. As the filtration value increases, the topological features of higher dimensions are born and die at the end. The complete topological information is then summarized in barcodes and persistence diagrams

can be summarized in persistence intervals. Each persistence interval (*birth*, *death*) determines the *lifetime* or *lifespan* of a given topological feature. The collection of persistence intervals in all dimensions is then represented as a set of sorted intervals (barcodes) or as a multiset of points in the plane (the persistence diagram) as presented in Fig. 3. We invite the reader to review (Hatcher 2002; Edelsbrunner and Harer 2010) for a detailed and formal exposition about PH and Algebraic Topology.

In Fig. 3, we show in a toy filtration, how PH tracks the evolution of topological information as ε_i increases from ε_1 to ε_5. Initially, there are nine connected components (H_0 in orange) in $\mathcal{K}_1 = f^{-1}([0, \varepsilon_1])$. Next, these components were merged into a single connected component in $\mathcal{K}_2 = f^{-1}([0, \varepsilon_2])$, as shown encircled in the Barcodes plot. Equivalently, the eight persistence intervals born in ε_1 and died in ε_2 are represented with an orange-filled dot in the Persistence Diagram plot. At the same time, there are two 1-cycles (H_1 in gray), illustrating the evolution of H_1 along the filtration. The first 1-cycle was born in ε_2, and the second 1-cycle appears when the first one was broken down into two 1-cycles in ε_4.

2.3 Classification problem

Let $\mathbb{F}$ be a finite feature space endowed with a proximity function $h(\cdot, \cdot)$, and $P \subset \mathbb{F}$ be a feature subspace. Let L be a label set. Suppose P is divided into two subspaces $P = X_l \cup X_u$, X_l the set of labeled points, and X_u is the unlabeled point set, which can be empty. Thus, the classification problem could be defined as predicting a suitable label $l \in L$ for every $x \in X_u$. Consequently, the predicted label list, $\hat{Y} \subset L^{|X_u|}$, will be the collection of labels resulting from the classification method.

3 Proposed classification method

We are interested in classifying a collection of unlabeled points X_u by considering the labels on X_l. A common way to achieve this aim is by using the Nearest Neighbor (NN) rule, which is based on the assumption that elements with the same label should be close. In the case of the K-Nearest Neighbors (KNN) rule, the label of $x \in X_u$ is calculated as

$$l = \hat{y}_x = \arg\max_{y \in Y} \sum_{i=1}^{k} I(y_{p_i} = y),$$

where p_i is the i-th nearest neighbor of x, Y is the label predictions list, $\hat{y}_x$ is the predicted label for x, and $I(y_{p_i} = y)$ is the indicator function and equals 1 when $y_{p_i} = y$ and 0 otherwise. The challenge in KNN is to choose the right value of k: a low value of k increases the noise sensibility, and a high value of k could include a large neighborhood making the label l not representative of x. A typical solution to find the right value of k for each dataset is to perform cross-validation (Jiang and Wang 2017). The KNN rule creates the KNN Graph by considering the point set (0-simplices) as a vertex set, and an edge (1-simplex) $e = (x, v) = (v, x)$ exists if v is a k-nearest neighbor of x.

KNN classifier is an instance-based learning method. These methods store the labeled samples, and generalizations beyond these samples are generated when a new instance must be classified (Mitchell 1997). In contrast, model-based learning methods infer a target function with a training set, and generalizations are made based on the learned function. Our proposed method is a KNN generalization extending the neighbors from 0-simplices to q-simplices and neighborhoods from k edges to collections of simplices of variable sizes and dimensions. The main challenge is to select an optimal simplicial complex that accurately represents our data, considering that multiple simplicial complexes can be constructed from the same point cloud.

The proposed method is divided into four steps summarized in Fig. 4. The steps are: to build a filtered simplicial complex (Sect. 3.1), to compute persistence intervals (Sect. 3.2), to choose a meaningful sub-complex (Sect. 3.3), and to classify unlabeled points (Sect. 3.4). Design choices are presented in Sect. 3.5, and described in more details in the Discussion (Sect. 5).

3.1 Step 1: Building the filtered simplicial complex

A filtered simplicial complex $\mathcal{K}$ is built on the dataset $P = X_l \cup X_u$ using a distance or proximity function $h(\cdot, \cdot)$ which is problem-specific (e.g., Euclidean, Manhattan, or Cosine). In our approach, X_u can be empty, and the unlabeled points $x \in X_u$ can be included in the classification stage, leading to the special cases explained in Sect. 3.4.2. In Fig. 4, X_l are the red, blue, and green points; $X_u = \{x_1, x_2\}$ is illustrated by the two unlabeled black points. $\mathcal{K}_1$, $\mathcal{K}_2$, $\mathcal{K}_3$, and $\mathcal{K}_4$ are examples of possible sub-complexes.

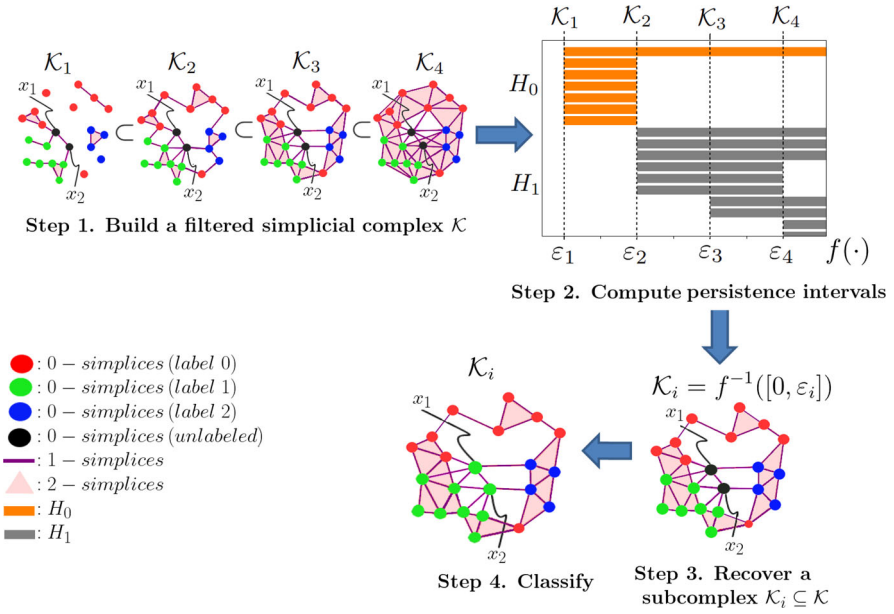

Fig. 4 The proposed algorithm involves several steps. The **Step 1** constructs a filtered simplicial complex $\mathcal{K}$, which may or may not include the unlabeled points. In **Step 2**, PH is computed to extract persistence intervals, revealing the hidden topological information within the data. Building upon this, **Step 3** utilizes selection functions based on the recovered topological information to identify a meaningful sub-complex. Finally, the classification phase begins in **Step 4**. It involves determining the simplicial neighborhood for each unlabeled point and calculating the label contributions using a weighted sum approach. The unlabeled point is then assigned the label that receives the highest number of votes

The proposed method can be applied to several simplicial complexes such as Rips, Čech, Witness, or Alpha. We choose Rips complexes due to their properties and extensive use in the field. Yet, we use three existing scale-up strategies to apply our proposed method to real datasets. (i) Limiting simplicial complexes maximal dimension q, $2 \leq q \ll |P|$. (ii) Limiting maximal edge length E to the average of the distance between all points in P. (iii) Applying the edge collapse method proposed by Boissonnat and Pritam (2020) to obtain a simplicial complex with fewer simplices than the original one but maintaining the same PH. The implementation details are available in Appendix 1.

Recall that to address scalability issues, we can build the complex by using the selected filtration value in Sect. 3.3 as maximal edge length as explained in Sect. 5.3.2, and the resultant complex will be the same as the desired sub-complex $\mathcal{K}_i$.

3.2 Step 2: Compute persistence intervals

A filtered simplicial complex $\mathcal{K}$ provides numerous multiscale data relationships. However, focusing on all of them can make it challenging to extract useful information. To overcome this, we aim to choose a sub-complex $\mathcal{K}_i$ from the filtered simplicial

complex $\mathcal{K}$ to approximate the actual structure of the dataset. In this vein, we exploit the ability of PH to detect topological features. For a filtered simplicial complex $\mathcal{K}$ of dimension q, PH will reveal up to q-dimensional homology groups H_q. In this paper, we use public implementations of several algorithms to compute PH through its dual (Persistent Cohomology), such as the ones proposed by de Silva et al. (2011), and Dey et al. (2014), which leads to more efficient computations than the classical PH computation through the reduction matrix algorithm proposed by Edelsbrunner et al. (2002).

When we build a filtered simplicial complex, an arbitrary level-set function $f(\cdot)$ will, with a high likelihood, impose a total order on the simplices, giving rise to a non-decreasing sequence of simplices. A partial order may occur as we work with quantized data defined by the proximity function $h(\cdot, \cdot)$. Thus, quantization should be defined accordingly. Following the reduction matrix algorithm for computing PH proposed by Edelsbrunner et al. (2002), we can classify each simplex as "positive" or "negative". A positive simplex (a row of the reduced matrix) creates a topological feature when added to the filtration. A negative simplex (a column of the reduced matrix) ends a topological feature when it is included in the filtration. Thus, a topological feature (j-cycle) can be identified entirely by the positive and negative simplices $\alpha, \beta \in \mathcal{K}$ that create and end that topological feature. Recall that the negative simplex is optional, meaning the topological feature never dies. The persistence interval of the cycle is given by the filtration values of α, β, i.e., $(f(\alpha), f(\beta))$. If there is no related negative simplex up to the maximal edge length E (essential j-cycle), the corresponding persistence interval is $(f(\alpha), \infty)$.

Let D^j be the set of the persistence intervals of j-cycles along the filtration of $\mathcal{K}$. We then collect all persistence intervals as the disjoint union of multisets $D = \bigsqcup_{j>0} D^j = \bigcup_{j>0}\{(d_i, j) \mid d_i \in D^j\}$, with $d_i = (birth_i, death_i)$ is the i-th persistence interval of D^j, where multiple copies of a given persistence interval with dimensions higher than zero are allowed. The 0-dimensional homology group is excluded because we aim to minimize the number of connected components while looking at homology groups in higher dimensions. This minimization will always happen if $dim(\mathcal{K}) > 0$. We should minimize the number of connected components to increase the likelihood that every unlabeled point $x \in X_u$ (that is not an outlier) belongs to a connected component and can be classified if there are labeled points in that connected component. If x is an outlier, it will take more effort (and high $f(\cdot)$ values) to connect it with any connected component. Figure 5 shows an isolated blue point that remains disconnected in different chosen sub-complexes.

3.3 Step 3: Recover a meaningful sub-complex

In this section, we provide four naive selection functions to address the Sub-complex Selection problem, and in Sect. 4.3, we present an experimental-based explanation of how the functions behave in practice. We hope this information can help the reader decide which function to use for his/her own data classification problem. In addition, in Sect. 5.4, a discussion is made about other approaches to solving the same problem to help the reader further to make an informed decision.

If $d \in D$ is a persistence interval, then the lifetime of the topological feature associated to d is measured by:

$$\text{lifetime}(d) = \min\{d[death], E\} - d[birth].$$

Essential persistence intervals (infinite death values) are truncated to the maximum value E in the collection of filtration values. A desired persistence interval $d \in D$ is selected by using the following (naive) selection functions:

(a) The persistence interval with maximal lifetime:

$$d_{max} = \text{MaxInterval}(D) = \arg\max_{d \in D}(\text{lifetime}(d)). \tag{4}$$

(b) The persistence interval with the closest persistence to the median lifetime:

$$d_{med} = \text{MedInterval}(D) = \arg\min_{d \in D}(|\text{lifetime}(d) - \text{median}(D)|). \tag{5}$$

where median(D) is computed over the persistence set $\{\text{lifetime}(d)\}_{d \in D}$.

(c) The persistence interval with the closest persistence to the average lifetime:

$$d_{avg} = \text{AvgInterval}(D) = \arg\min_{d \in D}(|\text{lifetime}(d) - \text{average}(D)|), \tag{6}$$

where $\text{average}(D) = \frac{1}{|D|} \cdot \sum_{d_i \in D} \text{lifetime}(d_i)$.

(d) Let $D' = \{d \mid \text{average}(D) < \text{lifetime}(d) < E\}$ be the collection of persistence intervals with persistence higher than the average lifetime. We select one of them uniformly at random:

$$d_{random} = \text{RandomInterval}(D) = \text{random}(D'). \tag{7}$$

Persistence intervals with significant persistence are commonly considered genuine topological features, and intervals with brief persistence are considered noise or local geometric information, depending on the application. For this reason, we propose the MaxInterval selection function. Selection functions MedInterval and AvgInterval use central tendency measures on the collection of lifetimes to capture a persistence interval with an expected lifetime. The preferred function depends on the skewness of the lifetime distribution. We can take the persistence interval with persistence closest to the maximum between average(D) and median(D). Finally, the RandomInterval function is based on the properties of randomized algorithms (Motwani and Raghavan 1995), which distributes the chances of taking a good choice between all possible intervals with persistence on the interval $(\text{average}(D), E)$.

When working with abstract simplicial complexes, the presence and arrangement of holes (j-cycles) convey both topological and geometrical information. While our primary focus is on analyzing the topological aspects, there are also hidden geometrical details. For example, during a persistence interval lifespan, certain connections among simplices may emerge, but they might not necessarily signify an observable

topological characteristic at a PH level. As a result, a higher filtration value may contain the same topological information as a previous one but with greater or equal geometrical information. Therefore, in the event of a tie during the computation of MaxInterval, MedInterval, and AvgInterval, the persistence interval with the later birth time (the youngest) should be selected in order to increase geometrical differentiation. This choice also has topological implications because we aim to minimize the number of connected components, which increases the likelihood that an unlabeled point belongs to a connected component or gets at least a 0-simplex as a neighbor, thus avoiding an empty link.

For the selected persistence interval $d \in \{d_{max}, d_{med}, d_{avg}, d_{random}\}$ we must obtain a sub-complex $\mathcal{K}_i$ from $\{\mathcal{K}_t = f^{-1}([0, \varepsilon_t])\}_{\varepsilon_t \in [d[birth], d[death]]}$. Multiple approaches exist to select a level value $\varepsilon_i \in [d[birth], d[death]]$ such as maximizing the area under a topological curve like Betti Curves (Saadat-Yazdi et al. 2021), Euler characteristic curve (Curry et al. 2018), or Persistence Entropy Summary (Atienza et al. 2020). We instead apply a straightforward approach, taking the half lifetime of the chosen interval $\frac{d[death]-d[birth]}{2}$ as a measure of required persistence concerning d. Thus, "relevant" features will be intercepted[1] at the birth, middle, and death times of d. The middle time captures those intervals intercepted by the birth time and with enough lifetime, but also those intervals d' such that $d[birth] \leq d'[birth] \leq d[middle]$ with $d[middle] = \frac{d[birth]+d[death]}{2}$. The death time captures those intervals born before $d[death]$ with enough lifetime to reach $d[death]$; this will capture some of the persistence intervals captured in the middle but also new ones. Among these three milestones on a persistence interval, we have experimentally tested that taking the sub-complex $\mathcal{K}_i$ on the $d[middle]$ and $d[death]$ filtration levels behaves better than the sub-complex taken on the $d[birth]$ time. However, taking the sub-complex at the death time works better than other filtration values on d in the tested datasets, giving similar results to the previously mentioned approaches, and it is simpler to obtain. Therefore, we set $\varepsilon_i = d[death] - \delta$, with $0 \leq \delta \leq 1$ to capture a minimal instant before the death of d; then, we obtain the respective sub-complex $\mathcal{K}_i = f^{-1}([0, \varepsilon_i])$. Results of exhaustive experiments across all filtration values are reviewed in Sect. 4.3.

In Fig. 5, a filtered simplicial complex $\mathcal{K}$ was built on the Circles dataset (described in Table 1, Sect. 4) with noise = 10. We show how PH is applied to perform an informed selection of a sub-complex $\mathcal{K}_i \subseteq \mathcal{K}$ by following the procedure explained in Sects. 3.2 and 3.3. Two persistence intervals (b_1, d_1) and (b_2, d_2) were captured by applying the MaxInterval$(\cdot)$, and RandomInterval$(\cdot)$ selection functions (see Eqs. 4 and 7), respectively. Then, we retrieve two sub-complexes $\mathcal{K}_{d_1} = f^{-1}([0, d_1 - \delta])$, $\mathcal{K}_{d_2} = f^{-1}([0, d_2 - \delta])$ by taking the dead times as we suggest in Sect. 3.3. Note that close to the death time d_1, there are three connected components H_0 and three 1-cycles H_1 captured in $\mathcal{K}_{d_1}$. We observe that $d_2 > d_1$, and just before d_2, only two of these three connected components remain, and two 1-cycles were filled; this situation is seamlessly captured by $\mathcal{K}_{d_2}$. From Fig. 5, we realize that by taking the sub-complexes from the birth time, the resulting sub-complex $\mathcal{K}_{b_1}$ has only two 1-cycles, but four connected components and $\mathcal{K}_{b_2}$ has three 1-cycles but also four connected components.

[1] By intervals intercepted at level value ε_i, we mean the homology of $f^{-1}([0, \varepsilon_i])$, i.e, the j-cycles alive at time ε_i. See Fig. 5, how the selected purple and yellow intervals are "intercepting" barcodes.

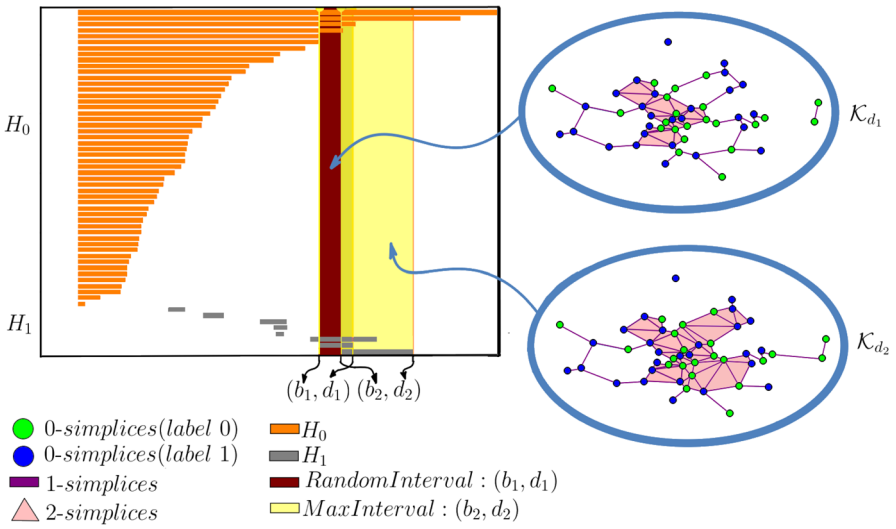

Fig. 5 A barcode representation of PH in $\mathcal{K}$ is shown, which have two homology groups H_0 (connected components) and H_1 (1-cycles). Two persistence intervals (b_1, d_1) (purple), and (b_2, d_2) (yellow) were selected according to the RandomInterval$(\cdot)$ and MaxInterval$(\cdot)$ selection functions, respectively. The sub-complexes $\mathcal{K}_{d_1} = f^{-1}([0, d_1 - \delta])$ and $\mathcal{K}_{d_2} = f^{-1}([0, d_2 - \delta])$ are shown. In order for outliers to be connected, a large filtration value is required, as demonstrated in the example. The chosen sub-complexes $(\mathcal{K}_{d_1}, \mathcal{K}_{d_2})$ still leave the same blue point isolated

A similar analysis can be made using the middle time, where the sub-complex $\mathcal{K}_{m_1}$ will contain the same topological info as $\mathcal{K}_{b_1}$. However, the sub-complex K_{m_2} will have missing one 1-cycle by maintaining the same number of connected components as $\mathcal{K}_{d_2}$.

3.4 Step 4: Classify

The simplicial neighborhood of a q-simplex $\sigma \in \mathcal{K}$ will be recovered by using the link and star (Definition 2). A key component of the proposed method is the label propagation over a filtered simplicial complex detailed in this section.

3.4.1 General case

Let us assume that $\mathcal{K}_i \subset \mathcal{K}$ was chosen according to the selection functions presented in Sect. 3.3. Let A be the real vector space with generators $\hat{l}_1, \hat{l}_2, \ldots, \hat{l}_N$, where $L = \{l_1, l_2, \ldots, l_N\}$ is the set of labels. We consider $0 \in A$ to represent no value. The generator $\hat{l}_j$ will be associated to the label l_j according to Definition 5.

Definition 5 (*Association function*) Let $\Phi_i : \mathcal{K}_i \to A$ be the *association function* defined on a 0-simplex $v \in \mathcal{K}_i$ as $\Phi_i(v) = \hat{l}_j$ if $v \in X_l$ and l_j is the label of v and $\Phi_i(v) = 0$ in any other case. The association function can be extended to a q-simplex $\sigma \in \mathcal{K}_i$ by setting $\Phi_i(\sigma) = \sum_{v \in \sigma} \Phi_i(v)$.

Note that the simplicial neighborhood of $x \in X_u$ includes simplices added to the filtration at different filtration values (distances) and related to simplices of different dimensions (for instance, in 2D, segments, and triangles). We are interested in assigning weights to quantify the relevance of each simplex belonging to the simplicial neighborhood of x, considering the vanishing effect of weights conforming filtration values among x and its neighbors increases. A simple way to define such a decay function is by adapting the so-called Shepard's Method (Shepard 1968), also known as Inverse Distance Weighting (IDW), to simplicial neighborhoods.

In a simplicial complex, we can use the link and star operations to propagate labels from labeled points in X_l to unlabeled points in X_u. We use a modification of the conventional IDW, suitable for simplicial neighborhoods. We use the filtration values as a proximity function among simplex elements. As a result, we define the Inverse Squared Filtration Value Weighting (ISFVW) to assign the following weight to a simplex σ regarding x, whenever $\sigma \cup \{x\} \in \mathcal{K}_i$ (recall that $f : \mathcal{K} \to [0, E]$ is the filtration function as in Definition 3):

$$w_{x,\sigma} = \frac{\left(\frac{1}{f(\sigma \cup \{x\})^2}\right)}{\left(\sum_{\mu \in St_{\mathcal{K}_i}(\{x\})} \frac{1}{f(\mu)^2}\right)}, \tag{8}$$

We then define the extension function as follows,

Definition 6 (*Extension function*) Let $\Psi_i : X \to A$ be the function defined on a point $x \in X_u$ by

$$\Psi_i(x) = \sum_{\sigma \in Lk_{\mathcal{K}_i}(\{x\})} w_{x,\sigma} \Phi_i(\sigma) = \sum_{\mu \in St_{\mathcal{K}_i}(\{x\})} w_{x,\mu \setminus \{x\}} \Phi_i(\mu \setminus \{x\}). \tag{9}$$

In Eq. 9, the normalizing factor (i.e., the denominator in Eq. 8) is the same throughout the whole sum. Hence, the normalization will not affect the ratio between components of the resulting vector element $(a_j)_{j=1}^N$. Therefore the normalization (in Eq. 8) can be avoided, giving rise to the following simplified equation:

$$\Psi_i(x) = \sum_{\sigma \in Lk_{\mathcal{K}_i}(\{x\})} \frac{\Phi_i(\sigma)}{f(\sigma \cup \{x\})^2} = \sum_{\mu \in St_{\mathcal{K}_i}(\{x\})} \frac{\Phi_i(\mu \setminus \{x\})}{f(\mu)^2}. \tag{10}$$

In Eq. 10 we obtain the cofaces of x such that $\mu \in St_{\mathcal{K}_i}(\{x\})$, and $\mu = \sigma \cup \{x\}$ for some $\sigma \in Lk_{\mathcal{K}_i}(\{x\})$, according to Eq. 3. We then compute an ISFVW to the label contributions on the simplicial neighborhood of x to prioritize the influence of σ to label x. Let $\alpha, \beta \in St_{\mathcal{K}_i}(\{x\})$ be two simplices, such that $f(\alpha) < f(\beta)$. This condition implies that α was clustered around x earlier than β was since α appears before β in the filtration. Consequently, α label contributions should be more important than β label contributions. The classification becomes more aware of the filtration history by using filtration values as inverse weights. This approach provides various benefits, including distance encoding operators, implicit local outlier detection, and density estimators.

Small values are associated with dense regions and small-volume q-simplices, meaning simplicial elements are more strongly related. High filtration values mean it takes longer to create the simplex; consequently, the relationship among their elements is weaker.

According to the previous definitions, given a point $x \in X_u$, the evaluation of the extension function at x would be $\Psi_i(x) = \sum_{j=1}^{N} a_j \cdot \hat{l}_j$, where $a_j \in \mathbb{R}_+ \cup \{0\}$, $j = 1, \ldots, N$.

Definition 7 (*Labeling function*) Let x be a point in X_u such that

$$\Psi_i(x) = \sum_{j=1}^{N} a_j \cdot \hat{l}_j.$$

Let $\tilde{a}$ be the maximum value in $(a_j)_{j=1}^{N}$, and $\tilde{A} = \{j \mid a_j = \tilde{a}\}$ be the set of maximum value indexes. We define the *labeling function* Υ_i at x as $\Upsilon_i(x) = l_k$ where k is uniformly selected at random from $\tilde{A}$. If $\tilde{a} = 0$ then $\Upsilon_i(x) = \emptyset$.

If there is a unique maximum value in the set $(a_j)_{j=1}^{N}$ introduced in the previous definition, the labeling function is uniquely defined at x. In all tested datasets, the label assignment of each point in X_u was uniquely defined because the factor $\frac{1}{f(\cdot)^2}$ acts as a tie-breaker. The classification process is summarized in Algorithm 1.

Algorithm 1 Labeling: Labeling a point set X_u.

Require: A filtered simplicial complex $\mathcal{K}$. A non-empty point set X_u.
Ensure: A predicted labels list $\hat{Y}$ of X_u.

1: $D \leftarrow GetPersistenceIntervalSet(\mathcal{K})$ where:
   $D = \{d_i \mid d_i = (birth, death)\}$ ▷ See Section 3.2
2: Get a desired persistence interval d where:
   $d \in \left\{ \begin{matrix} \text{MaxInterval}(D), \text{RandomInterval}(D), \\ \text{MedInterval}(D), \text{AvgInterval}(D) \end{matrix} \right\}$ ▷ See Section 3.3
3: $\varepsilon_i \leftarrow d[death]$; $\mathcal{K}_i \leftarrow f^{-1}([0, \varepsilon_i])$; $\hat{Y} \leftarrow \{\}$
4: **while** $X_u \neq \emptyset$ **do**
5: $x \in X_u$; $l \leftarrow \Upsilon_i(x)$
6: **if** $l = \emptyset$ **then**
7: $l \leftarrow handling\_special\_cases(x)$ ▷ see Section 3.4.2
8: **end if**
9: $\hat{Y} \leftarrow \hat{Y} \cup \{l\}$; $X_u \leftarrow X_u \setminus \{x\}$
10: **end while**
11: **return** $\hat{Y}$

In Fig. 6, an unlabeled point x_1 is being classified. In the simplicial neighborhood of x_1, there is a tie among the blue, red, and green-labeled points. This tie makes it difficult for the KNN algorithm to assign a label, especially for k values between 2 and 7. Even IDW-based weighted KNN approaches struggle with this problem because of

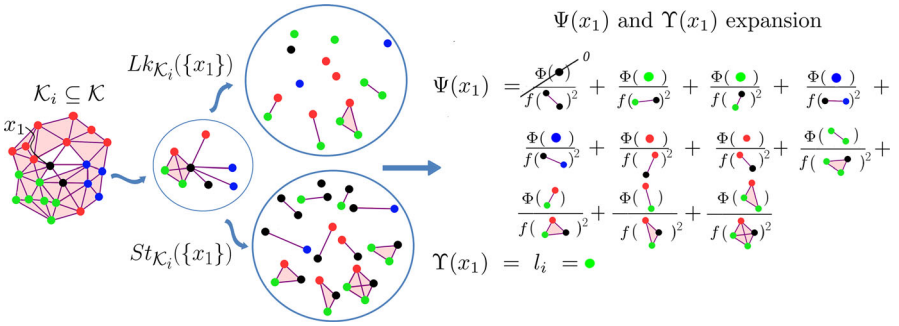

Fig. 6 Let $\mathcal{K}_i \subseteq \mathcal{K}$ be a chosen sub-complex and x_1 an unlabeled (black colored) point. The simplicial neighborhood of x_1, $St_{\mathcal{K}_i}(\{x_1\})$, and $Lk_{\mathcal{K}_i}(\{x_1\})$ are shown inside blue circles. Consequently, the extension function $\Psi_i(x_1)$ is illustrated, collecting label contributions. Finally, the labeling function $\Upsilon_i(x_1)$ assigns a green label to x_1. The inverse level-set function $\frac{1}{f(\cdot)^2}$ acts as a tie-breaker of green/red/blue label contributions, even when looking at the simplicial neighborhood of x_1 might seem there is a tie between labels

the dependence on the chosen k value and the distance similarities between the red and green-labeled points. Our approach solves this issue by using $\frac{1}{f(\cdot)^2}$ weighting to classify x_1 without concern for the tie among labels in the simplicial neighborhood. Figure 6 shows that the topology of labeled points entirely determines the labeling process of an unlabeled point since other unlabeled points do not make any contribution. Each unlabeled point will find the contribution of labeled points by exploring its link or through executing the border cases discussed in Sect. 3.4.2.

3.4.2 Special cases

Recall that $P \in \{\{X_l \cup X_u\}, \{X_l\}\}$ is the dataset applied to build the filtered simplicial complex $\mathcal{K}$ and $\mathcal{K}_i = f^{-1}([0, \varepsilon_i])$ is the selected sub-complex. There are three special cases related to an unlabeled point x, where the labeling function presented in Definition 7 produces no label contribution ($\tilde{a} = 0$):

(i) **Isolated point:** $x \in P$ and $\{x\}$ a maximal simplex, then $Lk_{\mathcal{K}_i}(\{x\}) = \emptyset$.
(ii) **External point:** $x \notin P$, thus $Lk_{\mathcal{K}_i}(\{x\}) = \emptyset$.
(iii) **Unlabeled link:** All points in $Lk_{\mathcal{K}_i}(\{x\})$ are unlabeled.

Case (i): Isolated point

In the first case, for the chosen ε_i, x is considered an outlier; therefore, no label should be assigned to it. See the example of the isolated blue point in Fig. 5, it remains isolated even for different sub-complexes.

Case (ii): External point

To handle external points, we obtain $Sn_{\mathcal{K}_i}(\{x\})$, a simplicial neighborhood of x in $\mathcal{K}_i$ computed with Algorithm 2. Let $U_x = \{u \in \mathcal{K}^{(0)} \mid h(u, x) \leq 2\varepsilon_i\}$[2] be a $(2\varepsilon_i)$-neighborhood of x in $\mathcal{K}_i$, with $h(\cdot, \cdot)$ the proximity function[3] applied to build $\mathcal{K}$. We

[2] See Appendix 3 for details about data structures for computing U_x efficiently.

[3] See Sect. 5.2 for details.

consider the selected ε_i twice, including a ε_i-radius around x and the ε_i-radius around the closest points of x. Then, for each simplex $\mu \in St_{\mathcal{K}_i}(\{u\}), u \in U_x$ we obtain a face $\mu' \leq \mu$ such that $\mu' = \mu \cap U_x$. Meaning that if x were added to the complex, then μ' would belong to the link of x.

Algorithm 2 $\mathbf{Sn}_{\mathcal{K}_i}(\mathbf{x})$: Simplicial neighborhood of an external point x.

Require: A new point x, a filtration value ε_i, the chosen sub-complex $\mathcal{K}_i$.
Ensure: A simplicial neighborhood $Sn_{\mathcal{K}_i}(\{x\})$ of x in $\mathcal{K}_i$, and F a collection of filtration values.

$U_x = \{u \in \mathcal{K}^{(0)} \mid h(u,x) \leq 2\varepsilon_i\}$ ▷ u is inside an open ball $B(x, 2\varepsilon_i)$
$Sn \leftarrow \{\}$
$F \leftarrow \{\}$
while $u \in U_X$ **do**
  for $\mu \in St_{\mathcal{K}_i}(\{u\})$ **do**
    $\mu' \leftarrow \mu \cap U_x$ ▷ to guarantee that each $\mu' \subseteq U_x$
    $Sn \leftarrow Sn \cup \{\mu'\}$
    if $dim(\mu') > 0$ **then**
      $F \leftarrow F \cup \{f(\mu')\}$
    else ▷ μ' is a 0-simplex
      $F \leftarrow F \cup \{h(u,x)\}$ ▷ Thus, we assume that $f(\mu')$ is $h(u,x)$
    end if
  end for
end while
return Sn, F

As a result, the simplicial neighborhood of x,

$$Sn_{\mathcal{K}_i}(x) = \bigcup_{u \in U_x} \{\mu' = \mu \cap U_x\}_{\mu \in St_{\mathcal{K}}(\{u\})},$$

can replace the link (or star) in Eq. 10 given rise:

$$\Psi(x) = \sum_{\mu' \in Sn_{\mathcal{K}_i}(x)} \frac{\Phi_i(\mu')}{f'(\mu')^2},$$

where $f'(\mu') \in F$. Then, $\Upsilon_i(x)$ is performed according to Definition 7. Note that when $U_x \subseteq X_u$, we reach case(iii).

Case (iii): Unlabeled link

To address the case where all points in $Lk_{\mathcal{K}_i}(\{x\})$ are unlabeled, we look for the shortest paths from x to the labeled points on the connected component containing x. We address this problem by implementing a modified version of the Dijkstra algorithm on simplicial complexes, similar to Yershov and LaValle (2011) but using the filtration values as costs to minimize. This approach can be considered "semi-supervised learning" by constraining the domain to a neighborhood around the link. Our approach applies a Min-Heap-based priority queue Q with operations:

- *enqueue*(σ, r) : σ is inserted in Q, before every simplex with priority $r' > r$, or at the end of Q if no priority higher than r is found. The priority r means the cost of reaching (following a path) σ from x, if σ is not on the same connected component than x, then $r = \infty$. A low priority is considered more critical than a higher one.
- *dequeue*() : Extract a tuple (σ, r) from Q, the first simplex σ with its priority r.
- *empty*() : Says if Q has 0 element or not.

The first step, is to mark each $\sigma \in Lk_{\mathcal{K}_i}(x)$ as visited and we do $Q.enqueue(\sigma,\ f(\sigma \cup \{x\}))$. With $f(\sigma \cup \{x\})$, we guarantee to process first those simplices that were early clustered around x on the filtration. We use a flag to avoid enqueueing a simplex twice; therefore, each element is marked as *visited* before being enqueued to Q. For each tuple $(\sigma, r) \in Q$, we find $Lk_{\mathcal{K}_i}(\sigma)$. Then, we consider each $\mu \in Lk_{\mathcal{K}_i}(\sigma)$, if it was visited, we ignore it; otherwise, we mark it as visited and check their label contributions with $\frac{\Phi_i(\mu)}{(r+f(\mu\cup\sigma))^2}$. With $f(\mu \cup \sigma)$, we follow the reasoning of using the filtration values as weights in Eq. 10 to consider simplices that were early clustered around σ on the filtration. By using $r + f(\mu \cup \sigma)$, we reach the minimum cumulative cost of reaching μ from x. We squared it to maximize the influence of label contributions with minimal cost over those with high costs to penalize even more simplicial contributors far from x. If μ has no contribution, we enqueue it with priority $r + f(\mu \cup \sigma)$ and continue this process until $Q.empty()$ returns true.

Algorithm 3 $Lp_{\mathcal{K}_i}(x)$:label contributions of x by applying link propagation.

Require: A new unlabeled point x.
Ensure: Label contributions $\sum_{j=1}^{N} a_j \cdot \hat{l}_j$ collected during link propagation in $\mathcal{K}_i$.

```
set_as_visited({x})
for σ ∈ Lk_{K_i}({x}) do
    set_as_visited(σ)
    Q.enqueue(σ, f(σ ∪ {x}))
end for
C ← 0                                   ▷ Initialized C ∈ A with no label contributions
while Q.empty() ≠ true do
    (σ, r) ← Q.dequeue()
    for μ ∈ Lk_{K_i}(σ) do
        if visited(μ) ≠ true then
            if Φ_i(μ) = (0)^N then ▷ if μ does not contribute, we propagate it
                set_as_visited(μ)
                Q.enqueue(μ, r + f(μ ∪ σ))
            else                        ▷ if μ contributes, we accumulate it
                C ← C + Φ_i(μ) / (r + f(μ ∪ σ))^2
            end if
        end if
    end for
end while
return C = (a_j)_{j=1}^N
```

Consequently, to find the label contributions, we expand the simplicial neighborhood of x by applying the Algorithm 3, obtaining:

$$\Psi(x) = (a_j)_{j=1}^{N} = Lp_{\mathcal{K}_i}(x).$$

Then, $\Upsilon_i(x)$ is performed according to Definition 7. If the entire connected component containing x is unlabeled, then $Lp_{\mathcal{K}_i}(x) = (0)^N$, and therefore no label will be assigned to x.

3.5 Design choices

Assuming the data (the feature space $\mathbb{F}$) is already in tabular form, there are several decisions the reader/user must make to get the best performance or to apply the proposed method properly. A non-exhaustive list of design choices is now presented. A more detailed discussion is given in Sect. 5.

3.5.1 Filtered simplicial complex construction (step 1)

To build the filtered simplicial complex of Sect. 3.1 at least four main design choices must be made: Distance selection, Data transformation, Simplicial complex family and construction.

Distance selection. Distance choice is a pervasive issue in data analysis; see (Francois et al. 2007; Aggarwal et al. 2001) to choose distance and transformations for high dimensional datasets. In Deza and Deza (2013), the authors broadly describe many distance functions, which is very helpful in understanding which distance is better for which type of data. When dealing with datasets with missing data, we suggest using a nan-distance proposed by Dixon (1979). See Sect. 5.2 for a detailed discussion.

Data transformation choice. From a statistical perspective, applying data transformations, including scaling, normalization, and standardization, is common practice to enable accurate comparisons among data points. In the present study, we specifically perform a log transformation on certain datasets, as detailed in Sect. 5.2.

Simplicial complex choice. The Rips complex is the common choice in TDA applications because it is easy to compute. Still, it grows exponentially on the number of points and the simplicial complex dimension. Other simplicial complexes may behave better with large datasets such as Witness complexes (Arafat et al. 2019; Silva and Carlsson 2004), in Sect. 5.3.2, a detailed exposition can be a guide.

Simplicial complex construction. Filtration values quantization can reduce the topological noise to the detriment of topological accuracy. Quantization increases the possibility of having a partial ordering among simplices on the complex; thus, any decision should consider precaution with the quantization effect. The sequence of the first three stages (Steps 1, 2, and 3) is not fixed and can be modified to address scalability issues, as discussed in Sect. 5.3.2.

3.5.2 Select the sub-complex (Step 3)

To recover the sub-complex $\mathcal{K}_i \subseteq \mathcal{K}$, as proposed in Sect. 3.3, two design choices are made relative to: the persistence interval, and the sub-complex in the persistence internal.

Persistence interval selection. We propose several selection functions to determine the sub-complex where the classification is made. The reader should decide which function to use. We give information about each proposed function and present experimental results showing how it behaves (Sect. 4.3). We consider that the AvgInterval function is more stable. Others can behave better but are more unstable. Functions based on Betti curves (Saadat-Yazdi et al. 2021) could work better, but considering that Betti 0 needs to be minimized (as we explain in Sect. 3.3). Therefore, conventional functions on Betti numbers must be modified. Section 5.4 discusses this topic.

Sub-complex selection. Once the persistence interval $(birth_i, death_i)$ is chosen we need to retrieve a filtration value $birth_i \leq \varepsilon_i \leq death_i$, then $\mathcal{K}_i = f^{-1}([0, \varepsilon_i])$. Our paper mainly uses death time as the preferred filtration value. This decision was made based on performance comparison, as death time surpassed birth and middle times more frequently than the reverse. Regardless, the reader is encouraged to pick another one if other arguments arise. It could be the case that the reader thinks about performing the method on a subset of filtration values creating an ensemble TDABC and then taking a vote to decide the label to assign, as discussed in Sect. 5.3.2.

3.5.3 Classify (step 4)

In the classification step in Sect. 3.4: how the weights influence of neighbors on labeling and the voting system were identified also as design choices.

Decay function to weight the influence of neighbors on labeling. Once the sub-complex $\mathcal{K}_i$ is chosen, we apply label propagation by using the link and star operators as simplicial neighborhoods. We should choose a decay function to determine the relevance of each label contribution. There are many valid functions (Chen 2015), but we provide one based on the inverse power law using a power value $p = 2$. The reader can choose another decay function with other properties; an analysis is presented in Sect. 5.5.

Voting system. The last decision is determining which label to assign after computing contributions to unlabeled points. Since we use the above-decaying weights successively in this work, applying a simple majority vote is enough (Lam and Suen 1997). The reader, however, can prefer another voting algorithm. In Sect. 5.5, we briefly discuss this matter.

4 Experimental results

When applying the different sub-complex selection functions from Sect. 3.3, the results of our TDA-based classifier (TDABC) may vary. Hence, we derived four variants of our TDA-based classifier (TDABC), each using a different selection function: TDABC-RANDOMIZED (RandomInterval), TDABC-MAXIMAL (MaxInterval),

TDABC-MEDIAN (MedInterval), and TDABC-AVERAGE (AvgInterval). Four baseline methods were selected to compare the proposed variants: KNN, WKNN, LSVM, and RF. All baseline classifiers were manually configured to deal with imbalanced datasets using the known class frequencies, i.e., the "class_weight" parameter in Scikit-Learn Library (Pedregosa et al. 2012). Table 1 shows the datasets and their characteristics.

4.1 Artificial datasets

The ***Circles, Swissroll, Moon, Norm,*** and ***Sphere*** datasets were artificially generated. In the case of ***Circles, Moon,*** and ***Swissroll***, a Gaussian noise factor was added to diffuse per-class boundaries and to assess classification performance for overlapped data regions.

The ***Norm***, and ***Sphere*** datasets were generated based on a Normal distribution per dimension. The ***Norm*** dataset has a high dimension ($P \subset \mathbb{R}^{350}$, $|P| < 350$). The ***Sphere*** dataset is always in three dimensions ($P \subset \mathbb{R}^3$), aiming to capture entanglement and imbalance sample distributions.

4.2 Real-world datasets

The ***Iris, Wine*** and ***Cancer*** datasets were selected as real datasets to compare the proposed classifiers and the baseline ones. The ***Iris dataset*** (Dua and Graff 2017) is a balanced dataset where one class is linearly separable, and the other two are slightly entangled with each other. The ***Wine dataset*** (Dua and Graff 2017) is an imbalanced dataset with thirteen different measurements to classify three types of wine. The ***Breast Cancer dataset*** (***Cancer***) (Dua and Graff 2017) is an imbalanced dataset with thirty features and two classes. The ***Wine*** and ***Cancer*** datasets were transformed using a logarithmic statistical transformation. Accordingly, a resulting dataset was obtained $P' = \{\ln(p + M)\}_{p \in P}$, with M the minimum component value of the dataset employed to deal with negative numbers. On the other datasets, no transformation was required.

4.3 Topological information

We assess the behavior of the selection functions by computing TDABC with cross-validation on each filtration value in the interval $[0, E]$. Figure 7 shows the plot of the $F1$ metric results on the ***Swissroll*** and ***Sphere*** datasets. ***Swissroll*** is a balanced dataset with a more complex data topology; it has multiple overlapped classes, and ***Sphere*** has multi-imbalanced classes with overlapped classes and a high imbalance ratio.

In Fig. 7a, for the ***Swissroll*** dataset, the AvgInterval function selects $\varepsilon_i = 6.57$ and gets $F1 = 0.80 \pm 0.16$, with the RandomInterval function the $\varepsilon_i = 9.73$ and $F1 = 0.72 \pm 0.16$, with the MedInterval function we obtain $\varepsilon_i = 11.41$ and $F1 = 0.78 \pm 0.19$, with the MaxInterval function we obtain $\varepsilon_i = 26.03$ and $F1 = 0.62 \pm 0.14$.

Table 1 Selected datasets to evaluate proposed and baseline classifiers

Name	Dimensions	Classes	Size	Samples per class	Noise	Mean	Stdev
Circles	2	2	50	[25,25]	3	–	–
Moon	2	2	200	[100,100]	10	–	–
Swissroll	3	6	300	[50,50,50,50,50,50,]	10	–	–
Iris	4	3	150	[50,50,50]	–	–	–
Norm	350	5	300	[60,10,50,100,80]	–	[0,0.3,0.18,0.67,0]	0.3
Sphere	3	5	653	[500,100,25,16,12]	–	0.3	0.147
Wine	13	3	178	[59, 71, 48]	–	–	–
Cancer	30	2	569	[212, 357]	–	–	–

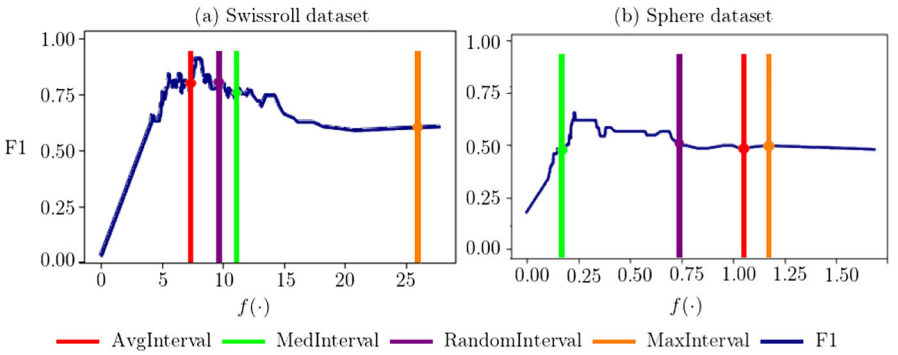

Fig. 7 TDABC with Cross Validation (10% fold) was performed on each filtration value in ***Swissroll*** and ***Sphere*** datasets. The $F1$ curve was plotted. Four vertical segments represent the filtration value selected by each selection function

In Fig. 7b the same process was applied on the ***Sphere*** dataset, and using the AvgInterval function, we obtain $\varepsilon_i = 1.07$ and $F1 = 0.48 \pm 0.15$, with the RandomInterval function we obtain $\varepsilon_i = 0.75$ and $F1 = 0.53 \pm 0.13$, with the MedInterval function we obtain $\varepsilon_i = 0.20$ and $F1 = 0.47 \pm 0.09$, with the MaxInterval function we obtain $\varepsilon_i = 1.16$ and $F1 = 0.49 \pm 0.15$.

In experimental tests, we found that metric curves depending on the filtration values, as shown in Fig. 7, are pretty different between distinct datasets. Sometimes there is a clear optimum value, and sometimes is less clear or nonexistent; thus, even cross-validation does not guarantee the best possible outcome in all datasets. Small filtration values behave poorly; the high number of connected components may lead to unlabeled points isolated at the selected filtration value. The most significant filtration values do not necessarily have the best behavior (an optimum value may exist). In our experiments, the optimum value never appears equal to the highest possible filtration value, which is the maximal edge length E. By limiting the maximum edge length, we also limit the maximal diameter of any topological feature, and therefore we control the number of elements returned by the link operator. There are stability areas (sets of consecutive filtration values, which lead to the same results); those areas can be seen in Fig. 7 as minor, moderate, and significant plateaus. Our naive selection functions reach a plateau frequently close to the optimum. We consider that these plateaus are produced because of the TDABC method concerning the local neighborhood around each unlabeled point. If this neighborhood does not change, the results remain the same, even when other regions of the complex are changing for different filtration values. Since points are fixed, at least that new edges or high dimensional simplices are created, the simplicial neighborhood will not change significantly. In Appendix 5, we present more topological information on the selected intervals in ***Swissroll*** and ***Sphere*** datasets.

4.4 Evaluation methodology

We compute the following metrics: F1 $= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$, the harmonic mean between Precision $\left(\frac{TP}{TP+FP}\right)$ and Recall $\left(\frac{TP}{TP+FN}\right)$, where *TP* are the true positives, *TN* true negatives, *FP* the false positives and *FN* the false negatives. The Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC-AUC) and the Area Under the Precision-Recall Curve (PR-AUC). The True Negative Ratio or Specificity $\left(TNR = \frac{TN}{TN+FP}\right)$, False Positive Ratio $\left(FPR = \frac{FP}{TP+FP}\right)$, and the Geometric Mean ($GMEAN = \sqrt{TNR \cdot Recall}$) between precision and recall. In each metric, we apply a one-vs-rest approach and macro average. A high value of a metric is better than a lower one, except in *FPR*, where a lower value is preferred. For an extensive exposition of classification metrics, please see (Japkowicz and Shah 2011).

We follow the experimental setting presented in Kubat et al. (1997) to assess the classifiers' behavior under imbalanced data conditions. Using the Normal distribution, we created 16 sets of data with two classes in $\mathbb{R}^2$. The positive class (label 0) was generated with $\mu = 0$ and $\sigma = 1.1$, while the negative class (label 1) was generated with $\mu = 2.0$ and $\sigma = 2.2$. Each dataset contains 100 samples, with 50 in each class. We then increased the negative class by 50 samples in each successive dataset, resulting in 16 datasets with varying levels of imbalance, ranging from 1 : 1 to 16 : 1. We perform repeated cross-validation in each dataset, computing the F1, PR-AUC, ROC-AUC, TNR, FPR, and GMEAN metric averages and their standard deviation (vertical segments). Figure 8, shows the experimental setting results per classifier on the generated datasets.

We conduct another set of experiments across the artificial and real datasets mentioned in Sects. 4.1 and 4.2 to evaluate the classifiers' performance on the selected metrics using a Repeated R-Fold Cross-Validation process (10% fold, N=5). We present the results according to the balancing condition of the datasets: imbalanced dataset results in Table 2 and balanced dataset results in Table 3. Figure 9 shows the results. We report the metric results on the imbalanced datasets on the minority class.

5 Discussion

This section will delve into our results and design choices when implementing TDABC in practical situations.

5.1 Result analysis

In Fig. 9, we have presented the results of all metrics across imbalanced and balanced datasets. Overall, we observe that TDABC is as good as baseline methods, but in the most imbalanced or entangled datasets, TDABC surpasses the others. This advantage becomes evident in ***Sphere***, where the imbalance ratio is up to 41:1 considering the maximal and minority classes (500:12). In a dataset with lower imbalanced ratios like ***Cancer*** and ***Wine***, the same pattern arises with the TDABC providing similar results

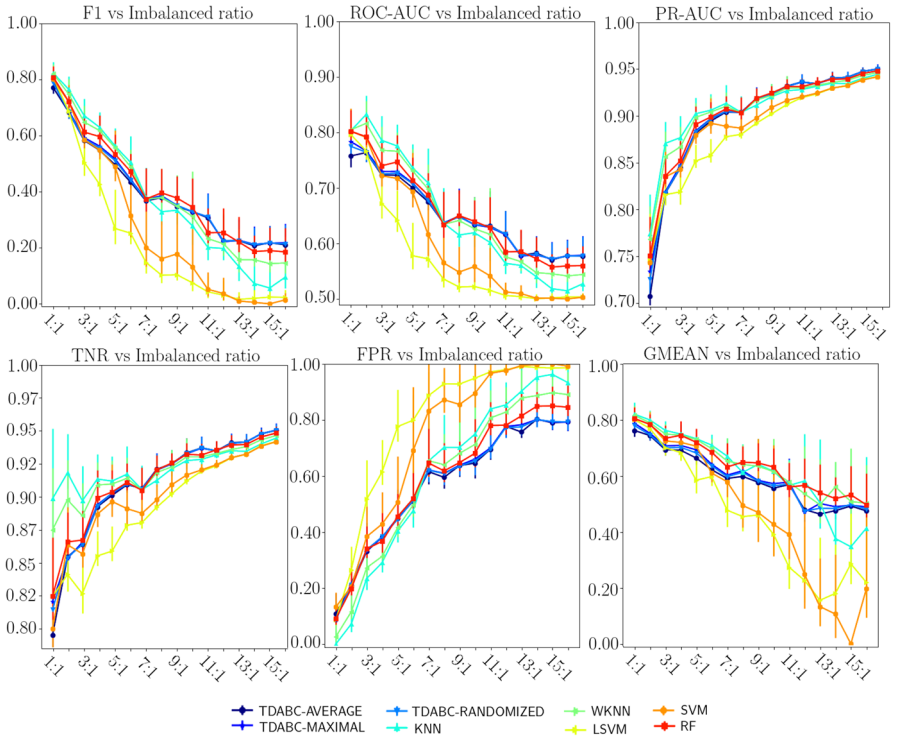

Fig. 8 Classifier's behavior under imbalanced data conditions. Results of the F1, ROC-AUC, PR-ROC, TNR, FPR, and GMEAN metrics in the minority class. From the 16 generated dataset with different imbalance ratios

as baseline classifiers. Table 2 shows detailed results for F1, PR-AUC, and ROC-AUC to confirm that TDABC behaves better in classifying minority classes when a high imbalanced ratio appears. The behavior under highly imbalanced data can be seen in Fig. 8, where we present results under 16 different imbalanced conditions. Again, we see that in F1, ROC-AUC, PR-AUC, TNR, and GMEAN metrics, the TDABC methods are similar to baseline approaches from 1:1 (50:50) to 7:1 (350:50) imbalanced ratios, respectively. From 7:1 to 16:1 (800:50), TDABC becomes better than baseline methods.

In FPR plots in Fig. 8, we show that the behavior in the minority class is better from 350:50 upwards, which means that baseline methods wrongly classify minority classes more often than TDABC does. Interestingly, the standard deviation in our proposed methods is more stable than the baseline ones.

Experiments on balanced datasets (see Table 3) show that the proposed TDABC methods behave competitively concerning baseline classifiers (better than the average), even with broadly used classifiers such as SVM and RF. Specifically, when topology becomes complex, like in the Swissroll case, where no hyperplane correctly identifies classes, TDABC shows better F1, ROC-AUC, PR-AUC, TNR, FPR, and GMEAN metrics than all reference methods. Type 1 errors or FPR is lower in our TDABC than

Table 2 Results per classifier across imbalanced datasets

Classifiers	Normal			Sphere			Wine			Cancer		
	Global	Minority class	STD	Global	Minority class	STD	Global	Minority class	STD	Global	Minority class	STD
F1												
TDABC-AVERAGE	**0.318**	0.018	0.078	**0.483**	**0.297**	0.097	**0.958**	**0.950**	0.066	0.941	0.941	0.023
TDABC-MAXIMAL	**0.307**	–	0.059	**0.483**	**0.297**	0.097	**0.958**	**0.950**	0.066	0.943	**0.943**	0.025
TDABC-MEDIAN	**0.317**	**0.034***	0.042	0.441	0.092	0.073	**0.958**	**0.950**	0.066	0.943	**0.943**	0.023
TDABC-RANDOMIZED	0.297	**0.010**	0.063	**0.537**	**0.323***	0.109	0.929	0.917	0.090	0.937	0.937	0.025
KNN	0.290	–	0.055	0.359	0.198	0.052	0.918	0.906	0.082	0.947	**0.947**	0.022
WKNN	0.294	–	0.055	0.424	0.250	0.058	0.918	0.906	0.082	0.947	**0.947**	0.022
LSVM	**0.380**	–	0.067	0.456	**0.256**	0.090	0.940	0.930	0.068	0.932	0.932	0.035
RF	0.257	–	0.053	**0.518**	**0.314**	0.106	**0.976**	**0.972***	0.045	**0.955**	**0.955***	0.025
AVERAGE	0.307	0.008	0.059	0.463	0.254	0.085	0.945	0.935	0.071	0.943	0.943	0.025
ROC-AUC												
TDABC-AVERAGE	0.817	0.678	–	**0.930**	**1.000***	–	0.992	**1.000***	–	0.937	0.937	–
TDABC-MAXIMAL	0.794	0.685	-	**0.931**	**1.000***	–	0.992	**1.000***	–	0.939	0.939	–
TDABC-MEDIAN	0.776	0.601	–	0.805	0.665	–	0.992	**1.000***	–	0.939	0.939	–
TDABC-RANDOMIZED	0.807	**0.708**	-	0.877	0.941	-	0.992	0.995	–	0.934	0.934	–
KNN	**0.859**	**0.831**	-	**0.931**	**0.997**	–	**0.998**	**1.000***	-	**0.945**	**0.945**	–
WKNN	**0.868**	**0.838**	-	**0.941**	**0.999**	–	**0.999**	**1.000***	-	**0.945**	**0.945**	–
LSVM	**0.902**	**0.905***	–	**0.913**	**0.999**	–	**0.997**	**1.000***	–	0.930	0.930	–
RF	0.717	0.472	–	**0.944**	**1.000***	-	**0.999**	**1.000***	–	**0.951**	**0.951***	–

Table 2 (continued)

Classifiers	Normal			Sphere			Wine			Cancer		
	Global	Minority class	STD	Global	Minority class	STD	Global	Minority class	STD	Global	Minority class	STD
F1												
AVERAGE	0.818	0.715	–	0.909	0.950	–	0.995	0.999	–	0.940	0.940	–
PR-AUC												
	Normal			Sphere			Wine			Cancer		
TDABC-AVERAGE	0.702	**0.241**	–	**0.961**	**0.994**	–	0.979	**1.000***	–	0.936	0.936	–
TDABC-MAXIMAL	0.640	0.082	-	**0.959**	**0.988**	–	0.979	**1.000***	–	0.937	0.937	–
TDABC-MEDIAN	0.642	0.133	–	0.914	0.329	–	0.979	**1.000***	–	0.937	0.937	–
TDABC-RANDOMIZED	0.667	**0.188**	–	0.934	**0.883**	–	0.982	0.994	–	0.934	0.934	–
KNN	**0.740**	**0.256**	–	**0.948**	**0.920**	-	**0.990**	**1.000***	–	**0.943**	**0.943**	–
WKNN	**0.766**	**0.391***	–	**0.961**	**0.969**	–	**0.993**	**1.000***	–	**0.943**	**0.943**	-
LSVM	**0.824**	0.156	–	0.932	**0.972**	–	**0.994**	**0.999**	–	0.929	0.929	–
RF	0.679	0.032	-	**0.963**	**0.998***	–	**0.999**	**0.999**	–	**0.948**	**0.948***	–
AVERAGE	0.708	0.185	–	0.946	0.882	–	0.987	0.999	–	0.938	0.938	–

The asterisk represents the highest value for the minority class on the imbalanced datasets
We show global metric results and the results for the minority (Min) class. In bold, those classifiers that were superior to the arithmetic mean

Table 3 Results per classifier across balanced datasets

Classifiers	Circles		Moon		Swissroll		Iris	
	Global	Std	Global	Std	Global	Std	Global	Std
F1								
TDABC-AVERAGE	**0.580**	0.194	0.480	0.081	0.707	0.127	0.932	0.074
TDABC-MAXIMAL	**0.599**	0.207	0.505	0.085	**0.713**	0.088	0.922	0.081
TDABC-MEDIAN	**0.599**	0.207	0.500	0.092	**0.805**	0.080	**0.951**	0.079
TDABC-RANDOMIZED	**0.580**	0.203	0.460	0.089	**0.794**	0.097	**0.941**	0.081
KNN	0.360	0.159	0.464	0.099	0.583	0.158	**0.961**	0.062
WKNN	0.498	0.220	0.450	0.095	0.708	0.086	**0.951**	0.079
LSVM	0.474	0.196	0.452	0.100	0.660	0.113	0.937	0.066
RF	**0.584**	0.170	**0.484**	0.118	**0.711**	0.104	0.930	0.075
AVERAGE	0.534	0.194	0.474	0.095	0.710	0.107	0.940	0.075
ROC-AUC								
TDABC-AVERAGE	**0.580**	–	**0.480**	–	0.983	–	0.988	–
TDABC-MAXIMAL	**0.600**	-	0.505	-	**0.994**	-	0.987	–
TDABC-MEDIAN	**0.600**	–	0.500	–	0.980	–	0.983	–
TDABC-RANDOMIZED	**0.580**	–	0.460	–	0.979	–	0.985	-
KNN	0.360	–	0.465	–	**0.992**	–	**0.997**	–
WKNN	0.500	–	0.450	–	**0.997**	–	**0.998**	–
LSVM	0.476	–	0.454	–	**0.991**	–	**0.993**	–
RF	**0.584**	–	**0.484**	–	**0.992**	–	**0.994**	–
AVERAGE	0.535	–	0.475	–	0.988	–	0.991	–
PR-AUC								
TDABC-AVERAGE	**0.547**	–	**0.490**	–	**0.976**	–	0.974	–
TDABC-MAXIMAL	**0.561**	-	**0.503**	–	0.968	-	0.973	–
TDABC-MEDIAN	**0.561**	–	**0.500**	–	**0.983**	-	0.964	–
TDABC-RANDOMIZED	**0.547**	–	0.482	–	**0.975**	–	0.971	–
KNN	0.450	–	0.484	–	0.946	–	**0.994**	–
WKNN	0.500	–	0.478	–	**0.979**	–	**0.997**	–
LSVM	0.489	–	0.479	–	0.959	–	**0.990**	–
RF	**0.549**	–	**0.492**	–	0.964	–	**0.992**	–
AVERAGE	0.525	–	0.488	–	0.969	–	0.982	–

In bold, those classifiers that were superior to the arithmetic mean

the other classifiers in all databases except in the Sphere dataset, where the TDABC-MEDIAN behaves better than KNN and WKNN but worse than SVM and RF, as shown in Fig. 9.

We also applied GMEAN to measure the combined growth of TNR and Recall. Our method shows in Fig. 9 higher values of GMEAN on datasets with a high imbalanced ratio like ***Norm*** and ***Sphere*** and in balanced datasets with non-linearly separable classes like ***Circles*** and ***Moon***. In general, in each dataset, our method surpasses

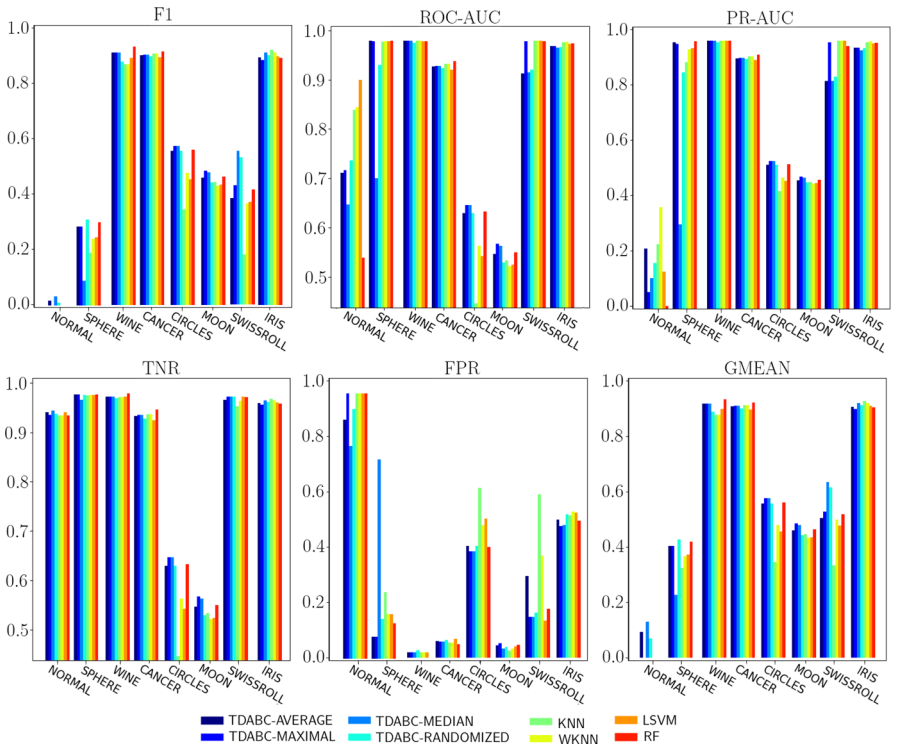

Fig. 9 Results of the F1, ROC-AUC, PR-ROC, TNR, FPR, and GMEAN metrics across datasets. In the case of imbalanced datasets, the results of minority classes were shown since it is commonly considered the more important class

at least one of the baseline methods in GMEAN. The TNR measurement results in Fig. 9 also show that our TDABC behaves well because our method differentiated negative instances similarly to RF and SVM and better than KNN and WKNN. Again, our method overcame the others in balanced datasets with overlapped classes, except for SVM in ***Circles***. However, our method was better in ***Moon***, which has more entanglement between classes and more elements.

The ***Circles*** and ***Moon*** datasets are balanced and have very entangled classes due to the noise factor, making the classification more challenging. In these datasets, KNN ($F1 = 0.449$ and $F1 = 0.445$) and WKNN ($F1 = 0.480$ and $F1 = 0.463$) behave poorly (lower than average). This behavior is related to the fixed value of k and the assumption that each data point is equally relevant. Even though WKNN imposes a local data point weight based on distances, it is not enough with highly entangled classes, as our results show. The TDABC methods can deal with the entanglement challenge through a disambiguation factor based on filtration values ($f(\cdot)$). The Iris dataset is a simple case where practically all methods perform well.

5.2 Distance choice

In Sect. 3.1, we have presented approaches based on a Rips filtration, but our methodology can be applied to other filtrations with minimal or no changes. Since we are considering distance-based simplicial complexes, a significant concern is the choice of the distance or proximity functions. It is a recurrent issue in data analysis, and the right choice depends on the data nature. Euclidean distance is the most commonly used distance; however, it becomes biased by the higher dimension components and is not recommended in high dimensional data (Aggarwal et al. 2001; Francois et al. 2007). It is advised to apply data transformations such as the standard or logarithmic transformations as we do with ***Cancer*** and ***Wine*** datasets. Those datasets have multiple measures with different meanings and scales; weighted Euclidean distances such as Mahalanobis distance are also appropriated in those cases. Cosine similarity is easier to compute and more suited for highly dimensional datasets than the Euclidean distance. The Minkowski distance is a generalization of Euclidean and Manhattan distances, and the power value p can be adapted to different situations. In Aggarwal et al. (2001), it was proved that distances with higher values of p behave poorly in high-dimensional settings. On the contrary, the Manhattan distance behaves better than Euclidean distance in almost every situation, and distances with power value $0 < p < 1$ were the most suited for high dimensional datasets (Aggarwal et al. 2001; Francois et al. 2007).

When working with nominal data types, it is recommended to leverage distance measures such as the Hamming Distance, Jaccard Index, and Sorensen-Dice to facilitate analysis and computations. When the dataset is incomplete (having missing values), a common choice is to use the so-called nan-Euclidean distance, proposed by Dixon (1979), which is a weighted Euclidean distance where the weight mitigates the impact of missing components. In some situations, data are conformed with mixed-type attributes (numerical and nominal data). In those cases, we need to build an ensemble distance. See (Bishnoi and Hooda 2020) for a survey of different mixed distances. The better results were obtained with the Heterogeneous Euclidean-Overlap Metric (HEOM) proposed in Wilson and Martinez (1997), which is a component-wise distance that combines specific distances for each component data type.

This paper uses the Euclidean distance to build the complex, but it is a problem-specific decision. For a detailed exposition of distances and to make an informed choice, we suggest reviewing (Deza and Deza 2013).

5.3 Filtration choice

To choose the proper filtration, we must consider which problem we want to solve besides choosing the suitable distance function. The choice is mainly motivated by our computational resources and the size and quality of our data.

5.3.1 Noise and outlier treatment

For example, we can tolerate noise and outliers (perform robust TDA) by using one of the filtrations' distances to measure (DTM) families. The Distance To Measure family (DTM) of filtrations proposed by Anai et al. (2020) provides a framework for TDA building filtrations robust over noise and outliers. However, DTM filtration introduces hyperparameters m and p that can impact the classification in the TDABC setting; we need to explore these implications further. Since different values of m highlight various dataset areas, we must explore how to take advantage of this to pay more attention to overlapped areas. We perform experiments on ***Circles*** and ***Norm*** datasets. In ***Circles***, the behavior was better than the reported results, but in Norm, it was worst. Modifying p tends to simplify the persistence diagram. The properties of DTM to exclude noise from topological features can provide a robust approach to applying TDABC on highly noisy data, such as other weighted PH approaches (Ren et al. 2021).

5.3.2 Scalability

Our TDABC method is limited by scalability challenges in constructing the simplicial complex and topological challenges to apply PH to recover the desired sub-complex successfully. Gudhi (Maria et al. 2014) has the main advantage of having an explicit filtered simplicial complex to perform our classification stage by finding appropriate simplicial neighborhoods. However, having an explicit complex has a scalability disadvantage and memory constraints. To reduce the impact of those issues, we can interchange the step sequence from **Step 1** → **Step 2** → **Step 3** to be **Step 2** → **Step 3** → **Step 1**. Meaning that we should compute PH directly with other frameworks such as Giotto (Pérez et al. 2021), and Ripser (Bauer 2021; Zhang et al. 2020) that do not build the simplicial complex explicitly. Then we suggest choosing an appropriate filtration value according to Sect. 3.3 and building the simplicial complex with Gudhi by fixing the chosen filtration value as the maximal edge length. As a result, we obtain a filtered simplicial complex that is the same as the sub-complex we would obtain by following the original step sequence presented in Fig. 4. The classification stage of Sect. 3.4 is always the last one.

Changing the complex Certain simplicial complexes, such as the Witness complex, are more suitable for effectively handling large amounts of data. We suggest using a Witness Complex developed according to the methodology of Arafat et al. (2019) through ϵ-covers, while following the advice of Silva and Carlsson (2004) in selecting landmark points. Careful selection of landmark points is essential, and we recommend prioritizing labeled points when selecting landmarks at random. Using the maxmin approach of Silva and Carlsson (2004), we can include labeled and unlabeled points since the approach ensures uniform distribution of landmark points. Doing this guarantees that witness points will be among labeled and unlabeled points, thus maintaining the classification error bounded by the witness complex's topological estimation error and ensuring convergence of Algorithms 2 and 3.

Another approach is to build non-flag simplicial complexes based on m-dissimilarities and m-metrics (it compares m+1 points simultaneously instead of only

two as typical distances do) as proposed in Wagner and Dłotko (2014), to produce simplicial complexes with a lower number of simplices at expenses of more computational time. Nevertheless, by finding appropriate permutations of the point set by using space-filling curves (Z-order, Hilbert-order), we can build the complex faster, see Sects. 3, and 4. These constructions can be done using Gudhi's implementation of skeleton blockers (Attali et al. 2011), which prevents some simplices from being expanded to higher dimensions if a criterion is not reached.

Localized TDABC (LTDABC)

The TDABC proposed method is affected the exponential growth of filtered simplicial complexes. However, those methods classify each unlabeled point in a localized region of the selected simplicial complex, and therefore it is not strictly necessary to build the whole filtered simplicial complex. Following this intuition, it is straightforward to develop a localized approach to TDABC, which we call the Localized TDABC.

The main idea is to infer a localized region from the original filtered simplicial complex, to perform the conventional TDABC. This region is associated with each unlabeled point x, and in normal circumstances, it is limited by $N_x = Lk_{\mathcal{K}}(x)^{(0)}$, the vertex set of $Lk_{\mathcal{K}}(x)$. Nevertheless, in the case of $N_x \subseteq X_u$, we cannot get any label contribution to classify x. A propagation schema will be necessary until a suitable label is reached, as explained in Sect. 3.4.2. Accordingly, we need to be sure that $N_x = Lk_{\mathcal{K}}(x)^{(0)} \cup \mathcal{L}^{(0)}$, with $\mathcal{L}$ is the collection of visiting simplices during the propagation stage. We need to infer this information from the dataset because we do not yet have any filtered simplicial complex. We recommend collecting N_x with the $\lceil\sqrt{|P|}\rceil$-nearest neighbors of x. In this way, the localized region N_x always exists and becomes scalable. Once we have the localized region N_x, we perform the standard TDABC process presented in Fig. 4 by constructing the filtered simplicial complex considering N_x instead of P on the Sect. 3.1.

Ensemble TDABC

We can employ ensemble learning techniques (Dietterich 2000; Fernández et al. 2018) in combination with our TDABC algorithm to address scalability concerns. By creating partial datasets through subsampling with replacement of samples or features, we effectively reduce the complexity of the original dataset. For each sub-dataset, we apply TDABC using one or a mixture of selection functions. Subsequently, a voting scheme is utilized to aggregate the results of each TDABC instance, enabling us to determine the final label assignment. This ensemble approach not only enhances scalability by reducing the computational burden but also maintains the effectiveness and accuracy of the classification framework.

5.4 Sub-complex selection

The Sub-complex selection is one of the most challenging problems in this paper. The reader can solve this by considering the sub-complex to choose as a hyper-parameter to optimize by using Cross-Validation based approaches (Jiang and Wang 2017); this approach always reaches an optimal sub-complex but is impractical. Our TDABC is designed in such a manner that the sub-complex $\mathcal{K}_i \subseteq \mathcal{K}$ need to be chosen once or

a minimal number of times, e.g., when too many new points need to be added to the dataset. Even in these cases, this approach is computationally expensive.

5.4.1 Model selection

In Caillerie and Michel (2011), the Sub-complex selection problem is addressed as a model selection problem in the context of density estimation. A slope heuristic combined with a least-squared penalized criterion is proposed to choose a "convenient scale" at which the topological features will be studied. However, the method is defined for simplicial complexes where all its simplices are k-simplices or a face of a k-simplex (all maximal simplices are k-simplices). The Caillerie and Michel (2011) approach applies (in theory) to general simplicial complexes, but the penalty function can be highly complex to devise. The complexities of Caillerie and Michel (2011) methodology to general complexes (inferring actual density distribution and obtaining a penalization function) can be boiled down by applying PH, and utilizing statistics on the persistence intervals' lifespan.

5.4.2 Optimal stopping

The Sub-complex selection problem can also be considered a variant of the classical Secretary problem (Freeman 1983), where its classical solution applies to select the appropriate persistence interval (close to optimal). One potential issue with this selection method is that it can be costly regarding computing power; the classifier's effectiveness must be evaluated beforehand for the first $\frac{N}{e}$ intervals, where N is the total number of intervals and e is the Euler number. Some extensions of the Secretary problem may not be suitable for selecting a persistence interval due to assumptions about data distribution. The Hiring Problem using Lake Wobegon approaches (Broder et al. 2010), for example, assumes a uniform distribution in evaluating candidates and still requires premature evaluation of the classifier's performance, making it unsuitable for our purposes.

5.4.3 Proposed approach

In this paper, we propose a naive approach with several selection functions that achieve competitive classification results. We use PH in Sect. 3.2 to gain knowledge about data topology and perform an informed selection of a sub-complex in Sect. 3.3. We propose four naive selection functions MaxInterval (Eq. 4), MedInterval (Eq. 5), AvgInterval (Eq. 6), and RandomInterval (Eq. 7). The classical solution to the Secretary problem has similarities to our proposed method in the sense that avoiding the first $\frac{N}{e}$ candidates resembles our decision to exclude 0-dimensional homology groups, where we only consider filtration values such that $\varepsilon \geq \hat{e}_0$ with $\hat{e}_0 = \min\{d[birth]\}_{d \in D}$ the first filtration value on the persistence intervals of homology groups of dimension higher than zero. In both scenarios, $\frac{N}{e} > \hat{e}_0$ or $\frac{N}{e} \leq \hat{e}_0$, our method will provide a filtration value ε higher than $\hat{e}_0$ except in the cases where all persistence intervals have equal birth (everyone is the youngest) and same lifespan that is rarely seen in real-life data. The

Lake Wobegon hiring approach over the mean and median is similar to our proposed selection functions AvgInterval, MedInterval. Let d_m be the chosen persistence interval with $d_m = \arg\min_{d \in D}\{|\text{lifetime}(d) - m|\}$, $m \in \{\text{average}(D), \text{median}(D)\}$, whenever the inequality $|\text{lifetime}(d_m) - m| > d_m[birth]$ holds, the selected filtration value $d_m[death]$ will be over m the central tendency measure of lifespans. We can force the condition to be strictly greater than m instead of taking the minimal distance to guarantee that every time we take a persistence interval in which lifespan is over m, the classifiers can behave better. Still, we leave these experiments for future work. Consequently, it can be considered an approximated Lake Wobegon strategy. Our approach is less exact due to the avoidance of considering the evaluation of each sub-complex to determine the best, and it is also faster for the same reason.

In Sect. 4.3, we show how the selection functions behave by considering the whole filtration numbers. There are stability regions, and the proposed naive functions can identify them. Besides, we can see in these results that it is possible to obtain better results for TDABC than provided in the Sect. 4 by proposing more sophisticated selection functions or approaches. More research needs to be done to find theoretical guarantees. In future work, we will explore other selection functions based on Betti numbers, topological curves, and signatures; we are also interested in mixing model selection techniques described in Baudry et al. (2012) and PH to address the Sub-complex selection problem.

5.5 Classification stage

We propose a simple majority vote to assign a reliable label (see Definition 7) to an unlabeled point. The decision rests on our proposed ISFVW, which applies the filtration value as an inverse squared weighting to decide which contributions are more influential than others. The inverse weighted function should belong, in fact, to a family of decay functions following, for instance, negative-exponential ($\exp(-pf(x))$) or inverse power ($f(x)^{-p}$) laws. We choose an inverse power law function, as negative-exponential functions vanish faster. This decision is also supported by the analysis made by Chen (2015), concluding that spatial models based on power-law decay are more suitable than negative exponential for analyzing large, complex, and scale-free models in spatial analysis. The exponent p in inverse power law changes the ratio between long-range and short-range weights. If a point is adjacent to a data point, its value is added to the value of that data point. When p is high, the result is similar to a Voronoi plot where the sites are the known values. When p comes down, the result looks like a bed on a stick (known value). Usually, $p = 2$, although other p values higher than 2 could also work seamlessly depending on the dataset.

When all label contributions are collected, a simple choice is made, and the label with a majority vote is taken. In Lam and Suen (1997), the effectiveness of several voting schemas was investigated, and the majority voting system was by far the simplest. It was found to be as equal and efficient as more complicated schemas.

Since our ISFVW performs a weighted sum of contributions, it could also be applied for regression problems, yet we need to normalize the weights in this case. The result of Eq. 10 is an element x of a vector space; by performing the L_1 normalization $\frac{V}{\|V\|_1}$,

we obtain the probability vector where each component i has the probability of the unlabeled point x get labeled with l_i. We can identify and correct mislabeled points in future work by applying this approach to labeled points.

6 Conclusion

This paper introduces a new TDA-based classifier (TDABC) for classifying multi-class datasets with imbalanced and overlapped classes. It uses data topology to identify the simplicial neighborhood of an unlabeled point and calculates a weighted sum based on inverse squared filtration values to determine label relevance. Our method can classify multi-class data without needing multiple binary classification problems. Unlike the common usage of TDA as a topological descriptor provider for ML, our proposed classifier is a novel application of TDA for classification tasks without any additional ML method. We present four methods, TDABC-AVERAGE, TDABC-MAXIMAL, TDABC-MEDIAN, and TDABC-RANDOMIZED, to address binary and multi-imbalance data classification problems without any re-sampling preprocessing. Overall, our methods perform better on average than the baseline methods in overlapped and minority classes. Despite the simplicity of the proposed selection functions, TDABC behaves well in challenging conditions such as overlapped and multi-imbalanced classes.

In our work, PH is essential in selecting a sub-complex that approximates well enough data topology through the MaxInterval, MedInterval, AvgInterval, or RandomInterval selection functions. Moreover, simplicial neighborhoods provide dynamic neighborhoods for classification. This work presents an application for the filtration values as inverse squared weighting to measure each simplex label contribution. These weights have several properties to deal with overlapped classes, such as distance encoding operators, indirect local outlier factors, and density estimators. As a result, the labeling function depends on the entire filtration history of the filtered simplicial complex and is encapsulated within the persistence diagrams at various dimensions.

Summarizing, the main contributions of this paper are the following:

- A link-based label propagation method for filtered simplicial complexes.
- A labeling function that depends on the whole filtration history encapsulated within the persistence diagrams at various dimensions.
- The application of PH to perform an informed selection of a sub-complex a from a filtered simplicial complex.
- The design, implementation, and evaluation of four naive selection functions to take advantage of the topological information to select a sub-complex from filtration.
- A TDA approach to solving classification problems that shows advantages for imbalanced datasets without further ML stages, which is permutation invariant.

Future work includes applying metric learning techniques for distance choice, exploring theoretical guarantees for sub-complex selection, analyzing the impact of the DTM filtration hyperparameters in TDABC results, and investigating the TDABC potential

in exploratory data analysis for feature selection, missing data analysis, and mislabeled classification.

Acknowledgements This research work was supported by the National Agency for Research and Development of Chile (ANID), with grants ANID 2018/BECA DOCTORADO NACIONAL-21181978, FONDECYT 1211484, 1221696, ICN09 015, and PIA ACT192015. Beca postdoctoral CONACYT (Mexico) also supports this work. The first author would like to thank professor José Carlos Gómez-Larrañaga from CIMAT, Mexico, for his insightful discussions.

Declarations

Conflict of interest The authors are unaware of any affiliations, memberships, funding, or financial holdings that might be perceived as affecting the objectivity of this paper.

Appendix 1: Implementation details

This Section provides some details about implementing our TDA-based classifier (TDABC). The TDABC was implemented on top of the Gudhi Library (Maria et al. 2014), Giotto Library (Pérez et al. 2021), Ripser (Bauer 2021; Zhang et al. 2020) to solve the computational topology aspects such as Simplicial Complexes and Persistent Homology (PH). Sci-kit learn (Pedregosa et al. 2012) for the Machine Learning algorithms such as baseline classifiers and PCA and TSNE dimensionality reduction methods. Numpy (Harris et al. 2020) for multi-dimensional arrays manipulation. UMAP-learn for UMAP-based dimensionality reduction (McInnes et al. 2020). HDF5 (The HDF Group 1997-2022) to handle large data in primary and secondary memory. Matplotlib (Hunter 2007) for visualization purposes. The source code of our proposed TDABC is available on https://github.com/rolan2kn/TDABC-4-ADAC. The following sections cover different aspects of the TDABC implementation.

Build simplicial complexes

Maximal edge length

A p-cycle can be born at any time and live unaltered up to the maximum edge length, in which case the p-cycle will die or be divided into two topological features. By controlling the maximal edge length, we control the topological feature-length and the size of the simplicial complex, combinatorial on the number of points and the simplex dimension. Thus, we recommend using the mean distance of the distance matrix as the maximal edge length. Consequently, noise points will only affect those cycles with a diameter twice the mean distance and make the filtration robust.

Edge collapse

Edge collapsing in Gudhi must be performed on the 1-skeleton of the simplicial complex and then expand from 1-skeleton to build all high dimensional simplices up to a maximal dimension $q \ll |P|$. The Algorithm 4 computes a simplex tree using the edge

collapsing method (Boissonnat and Pritam 2020). A collapsing coefficient is defined to be dependent on the maximal dimension q. However, it could be enhanced by repeatedly calling the *collapse_edges* method until the simplex tree no longer changes. We recommend applying a collapsing factor (obtained experimentally) computed as a function of the point cloud's ambient dimension and the simplicial complex's maximal dimension. Edge collapse in Gudhi and Giotto supported only flag complexes like the Vietoris Rips. Our method can be used in any simplicial complex with minimal variations, but each complex has its intricacies to optimize the consumption of time and space resources.

Algorithm 4 Build a simplex tree with edge collapsing

Require: A non-empty point set $P \in \{\{X_l \cup X_u\}, \{X_l\}\}$.
Ensure: An updated simplex tree $\mathcal{S}$.

$M \leftarrow distance\_matrix(P)$
$\hat{m} \leftarrow avg(M)$ ▷ The mean distance
$\mathcal{K} \leftarrow create\_rips\_complex(M, \hat{m})$
$\mathcal{S}_\mathcal{K} \leftarrow create\_simplex\_tree(\mathcal{K}, max\_dim = 1)$
$c \leftarrow \left\lceil \frac{\max\{q,d\}}{\min\{q,d\}} \right\rceil$ ▷ a simple coefficient of edge collapsing
$\mathcal{S}_\mathcal{K} \leftarrow collapse\_edges(\mathcal{S}_\mathcal{K}, c)$
$\mathcal{S}_\mathcal{K} \leftarrow expansion(\mathcal{S}_\mathcal{K}, max\_dim = q)$
return $\mathcal{S}_\mathcal{K}$

Computing link and filtration values

The association function, Ψ_i from Definition 5 depends on the $Lk_\mathcal{K}$ operation. However, up to now, the Python interface of Gudhi Library (v.3.6.0) (Rouvreau 2022) does not have an implementation of the simplex link operation. Regardless, it can be derived from the star and co-face operators according to Definition 2.

In Gudhi, each q-simplex $\sigma \in \mathcal{K}$ is stored with its filtration value $f(\sigma)$. Thus, the $star(\mathcal{S}_\mathcal{K}, \sigma)$ is a function in $\mathcal{S}_\mathcal{K}$ which returns a 2-tuple set

$$\{(\mu, f(\mu)) \mid \mu \in St_\mathcal{K}(\sigma)\}.$$

This data structure makes it easy to recover the filtration values required to implement Eq. 10, Algorithm 2, and Algorithm 3.

Finding neighborhoods of external points

The Algorithm 2 has a lot of room for optimizations. The search for the closest points to x can be drastically enhanced by applying computational geometry algorithms and spatial/metric data structures. As a few examples: a pivot point could be selected from P and then built as a Vantage Point Tree (VP-tree), Ball tree, or M-tree, among others, to perform multidimensional indexing of all elements by their distance (or proximity)

to the pivot (Samet 2006). Other metric space searching data structures like the Spatial Approximation Tree (a-tree) (Navarro 2002) could be applied by considering the same complex as support instead a Delaunay triangulation. It could be more efficient than other space partitioning approaches to solve neighboring queries in datasets with more than 20 dimensions. These data structures could help to find points likely to share a simplex with x reducing the computational cost in time complexity to $O(|P| \log |P|)$ in the worst case.

Permutation invariance

Given a proximity function $h(\cdot, \cdot)$, labeled (X_l) and unlabeled (X_u) point sets both subsets of P. Our TDABC is invariant to permutations of X_l and permutations of X_u building every time the same type of filtered simplicial complex. Let $\mathcal{K}(M_P)$ be a filtered simplicial complex constructed using a distance matrix M_P over a point set P. For any permutation $\pi(P)$ we obtain a distance matrix $M_{\pi(P)}$ and a filtered simplicial complex $\mathcal{K}(M_{\pi(P)})$. We know that M_p and $M_{\pi(P)}$ are equivalent matrices because both have the same number of rows and cols. We can turn one into another by elemental row and column transpositions since they were constructed with the same $h(\cdot, \cdot)$ function over the same point set P. Thence, $\mathcal{K}(M_{\pi(P)})$ and $\mathcal{K}(M_P)$ are the same complexes up to labeling permutations because we can always use a simplicial map that applies the inverse permutation $\pi^{-1}(P)$ to each element of $\pi(P)$ to obtain the corresponding element in P this is equivalent to perform the column and row transpositions to rename simplices in $\mathcal{K}(M_{\pi(P)})$ to obtain $\mathcal{K}(M_P)$. The case of permutations in unlabeled points X_u is straightforward since the unlabeled points do not contribute to labeling other unlabeled points. Therefore, no matter which permutation is applied, TDABC will label the same point (among permutations) with the same label.

Although the results of the TDABC remain consistent across different permutations, there is room for improvement in terms of time execution. Certain permutations result in faster computations compared to others, contributing to the observed difference in performance. One of the reasons for this discrepancy is the locality property, where elements with small distances between them are clustered together in the element collection. In such cases, finding similar elements requires less time as they are located in close proximity. On the other hand, when similar elements are scattered in arbitrary positions, their retrieval becomes more time-consuming, leading to the wastage of computational resources. To address this issue, we propose using space-filling curves and related data structures (refer to Sect. 3), such as Z-order and Hilbert-order. These techniques, similar to those employed in Gudhi (Maria et al. 2014), have been shown to accelerate the construction of complexes and streamline simplicial queries.

Topological information

We complement Sect. 4.3 by presenting more information regarding the selected filtration value on the Swissroll and Sphere Datasets. See Figs. 10, 11 and Table 4.

Swissroll topological information

See Fig. 10.

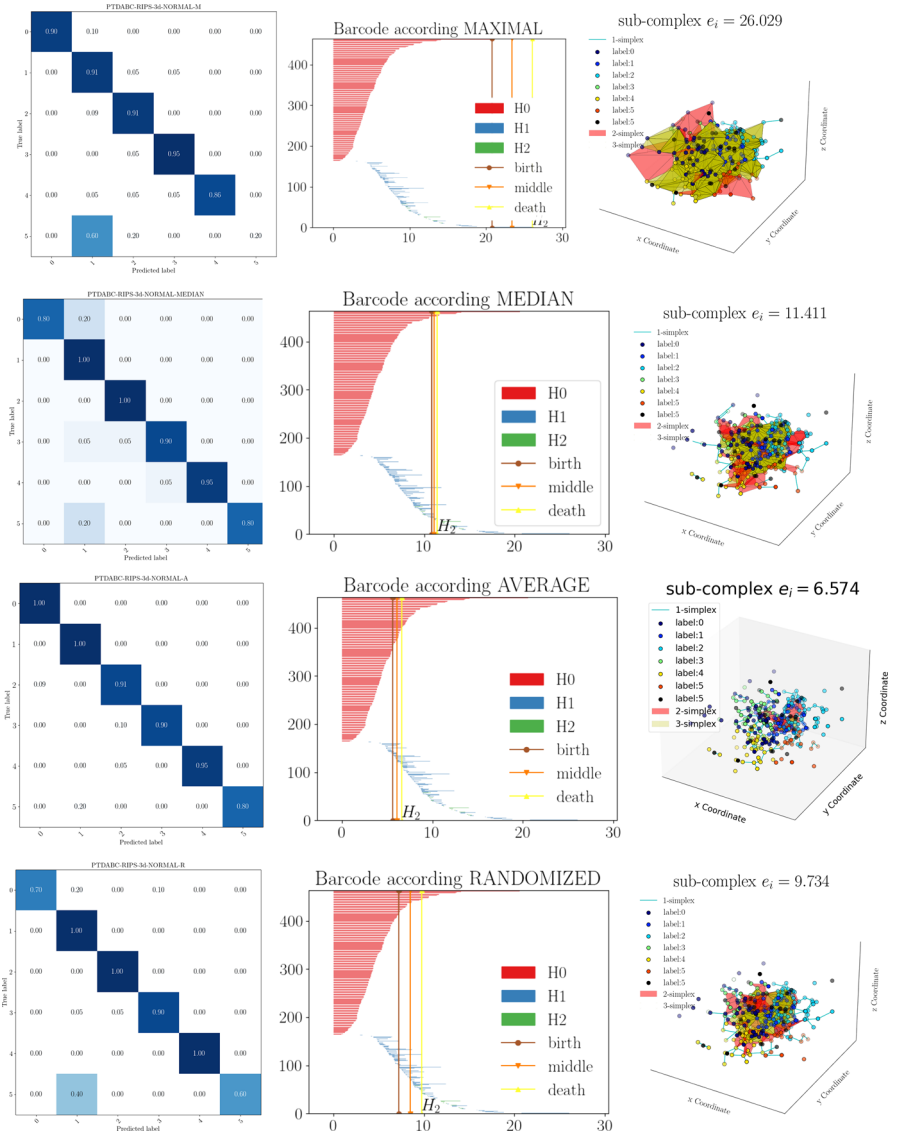

Fig. 10 Results of applying the selection function on the Swissroll Dataset: the confusion matrices (first column), barcodes with the chosen interval (second column), and the chosen sub-complex $\mathcal{K}_i$ (third column), each 0-simplex has a color representing its label, unlabeled points in black

Sphere topological information

See Fig. 11.

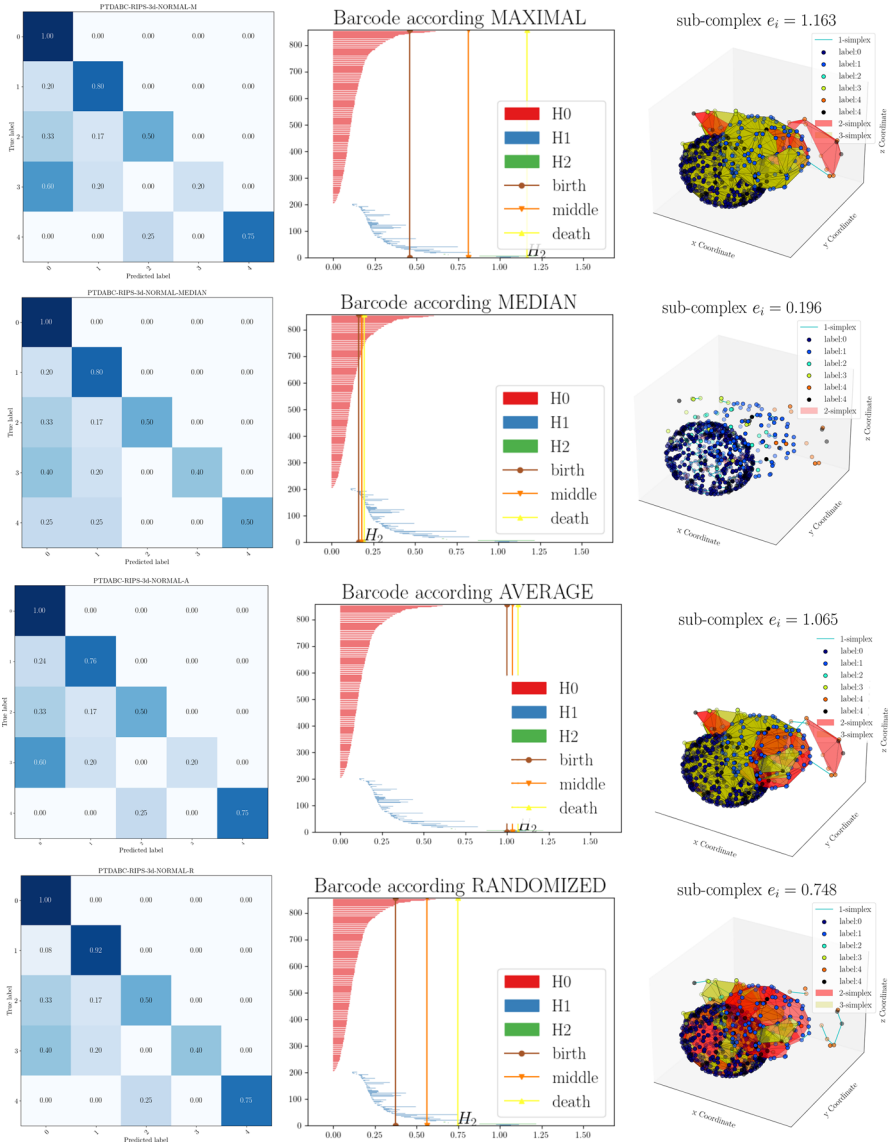

Fig. 11 Results of applying the selection function on the Sphere Dataset: the confusion matrices (first column), barcodes with the chosen interval (second column), and the chosen sub-complex $\mathcal{K}_i$ (third column), each 0-simplex has a color representing its label, unlabeled points in black

Extended results

We also conduct experiments increasing the number of unlabeled points. We perform TDABC with three different cross-validation configurations by taking the fold size to be NORMAL (10%), EXTREME (60%), and HYPER EXTREME (90%) as fold size. We present the results of the F1 metric in Swissroll and Sphere Datasets in Table 4.

Complexity analysis

We utilize the simplex tree from Gudhi as the chosen data structure in our paper because of its capability to represent any type of simplicial complex. It is worth mentioning that Gudhi offers various other data structures, as explained in the paragraph, which are specifically designed for maximal simplices (simplices without cofaces) (Boissonnat et al. 2017; Boissonnat and Karthik 2018). However, for our purposes, we focus on the simplex tree due to its versatility and ability to handle a broad range of complex types.

The simplex tree data structure was presented by Clement Maria and Jean-Daniel Boisonnat (Boissonnat and Maria 2014). Let $\mathcal{S}_\mathcal{K}$ be a simplex tree representation of a simplicial complex $\mathcal{K}$. Let σ be a q-simplex where $\sigma \in \mathcal{K}$. The operations $insert(\mathcal{S}_\mathcal{K}, \sigma)$, and $search(\mathcal{S}_\mathcal{K}, \sigma)$ have a time complexity $O(|\sigma| \cdot log|P|)$ by using red-black trees to represent sibling nodes (nodes sharing its father node, see Fig. 12). When using hashing functions to represent sibling nodes, the time complexity is reduced to $O(|\sigma|)$. Insertion of a q-simplex σ with all its faces has a complexity of $O(2^{|\sigma|} \cdot |\sigma| \cdot log|P|)$. See (Boissonnat and Maria 2014) for a more detailed explanation.

Let $\tau = \{\tau_0, \tau_1, \cdots, \tau_j\}$ be a j-simplex, with $\tau \in \mathcal{K}$. Compute $St_\mathcal{K}(\tau)$ in a simplex tree $\mathcal{S}_\mathcal{K}$ is performed by the operation $Locate\_cofaces(\mathcal{S}_\mathcal{K}, \tau)$ (Boissonnat and Maria 2014). To locate all cofaces of τ in $\mathcal{S}_\mathcal{K}$, it is needed to find all occurrences of τ_j in nodes whose depth is greater than j, and navigate upwards on $\mathcal{S}_\mathcal{K}$ looking for remaining elements of τ. Those paths where τ was completely found, will contain the cofaces of τ. Traversing a path in a simplex tree has a worst-case time complexity of $O(q+1)$ with $q = dim(\mathcal{K})$. Lets $O(\mathcal{T}_{\tau_j}^{>j})$ be the time complexity to locate all nodes at a depth greater than j, which contains τ_j. Accordingly, the worst case time complexity of $St_\mathcal{K}(\tau)$ is $O((q+1) \cdot \mathcal{T}_{\tau_j}^{>j})$. In Gudhi, every path from the root to any leaf defines a maximal simplex.

A few algorithms remain missing from this section like the Algorithm 1 to label a test point set. This algorithm could be implemented considering the explanations mentioned above. The computation of PH is done using the method provided by Gudhi. In Sect. 3.3, we define the selection of persistence interval after obtaining the PH. The Algorithm 4 builds a simplicial complex with distance matrix and edge collapses. Computing the distance matrix can be done in time $O(d \cdot |P|^2)$ with the brute force method, but it could be at most $O(|P|)$ in a massive parallelism platform (Ji and Wang 2022). The edge collapse method runs in time $O(n \cdot n_c \cdot k^2)$, with n, n_c the number of edges on the input and output graphs, k is the maximal degree of a vertex.

Table 4 Results varying the number of unlabeled points

F1 Classifiers	Sphere			Swissroll		
	Global	Minority class	Fold standard deviation	Global	Minority class	Fold standard deviation
Normal fold 10%						
TDABC-AVERAGE	0.557	0.146	0.107	0.474	0.364	0.143
TDABC-MAXIMAL	0.528	0.143	0.105	0.479	0.377	0.140
TDABC-MEDIAN	0.502	0.125	0.096	0.414	0.235	0.133
TDABC-RANDOMIZED	0.554	0.147	0.101	0.472	0.364	0.134
KNN	0.372	–	0.047	0.721	0.559	0.171
WKNN	0.468	–	0.064	0.802	0.697	0.155
LSVM	0.494	–	0.090	0.747	0.642	0.053
RF	0.527	0.073	0.096	0.720	0.560	0.114
AVERAGE	0.500	0.079	0.088	0.604	0.475	0.131
Extreme fold 60%						
TDABC-AVERAGE	0.452	0.066	0.025	0.465	0.345	0.019
TDABC-MAXIMAL	0.441	0.060	0.006	0.463	0.368	0.021
TDABC-MEDIAN	0.379	0.047	0.034	0.374	0.217	0.033
TDABC-RANDOMIZED	0.472	0.076	0.050	0.468	0.360	0.014
KNN	0.327	–	0.008	0.493	0.172	0.031
WKNN	0.373	–	0.025	0.629	0.447	0.064
LSVM	0.419	–	0.031	0.695	0.567	0.013
RF	0.453	–	0.032	0.646	0.484	0.039
AVERAGE	0.415	0.031	0.026	0.529	0.370	0.029
Hyper extreme 90%						
F1						
	Sphere			Swissroll		
Classifiers	Global	Minority class	Fold standard deviation	Global	Minority class	Fold standard deviation
TDABC-AVERAGE	0.305	–	0.013	0.320	0.156	0.028
TDABC-MAXIMAL	0.339	–	0.004	0.314	0.207	0.006
TDABC-MEDIAN	0.257	–	0.010	0.267	0.172	0.005
TDABC-RANDOMIZED	0.329	–	0.031	0.318	0.213	0.026
KNN	0.295	–	0.011	0.155	0.006	0.057
WKNN	0.315	–	0.006	0.287	0.128	0.021
LSVM	0.321	–	0.021	0.370	0.195	0.045
RF	0.391	–	0.016	0.376	0.253	0.046
AVERAGE	0.319	–	0.014	0.301	0.166	0.029

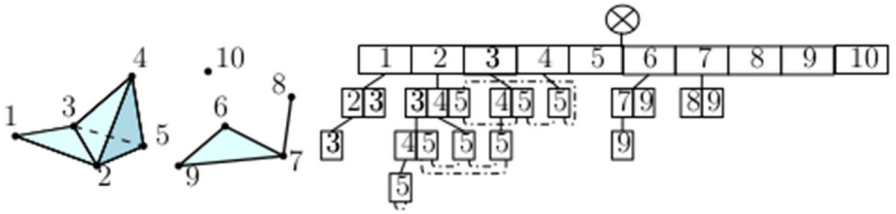

Fig. 12 A simplicial complex on ten vertices and its simplex tree. The deepest node represents the tetrahedron of the complex. Every label position at a given depth is linked in a list, as illustrated in the case of label 5. Picture and caption were taken from (Boissonnat and Maria 2014)

Algorithm 1 includes computing PH, which has a worst-case time complexity of $O(|\mathcal{K}|^3)$, but in practice, it has a time complexity near linear. The selection function computation is linear on the number of persistence intervals $O(|D|)$. The inverse level-set function $f^{-1}(\cdot)$ that has a worst-case time complexity of $O((q+1)\cdot \log|P|)$ the same time required to find a simplex in the simplex tree, since we need to locate the simplex to ask for its filtration value. Algorithm 2 has a time complexity of $O(|P|\cdot \log|P|\cdot|U|\cdot(q+1)\cdot \mathcal{T}_{\tau_j}^{>j})$; it is an output-sensitive algorithm which depends of the number of points inside the (2ε)-ball and for each point the complexity of the star. There is much room for optimizations by applying dynamic programming techniques since multiple star queries on the same dense regions have many non-disjoint solutions. Algorithm 3 finds label contributions to label an unlabeled point x by building an implicit minimal spanning tree on the connected component containing $Lk_{\mathcal{K}_i}(\{x\})$. By the time the tree is finished, we have visited $O(M)$ nodes performing a link operation per node, where M can be, at most, the number of simplices on the connected component. If the entire complex is connected, $M = |K|$. Enqueue and dequeue operations on the priority queue Q have a time complexity of $O(\log M)$ in Q. Therefore the time complexity is $O(q\cdot \mathcal{T}_{\tau_j}^{>j}\cdot M\cdot \log M)$. Each node in this implicit tree has space complexity $O((q+2)\cdot w)$ bits, $(q+1)$ the maximal q-simplex cardinality plus one because of the priority. Since we have $O(M)$ nodes, the total space complexity is $O((q+2)\cdot w\cdot M)$ bits.

References

Adams H, Emerson T, Kirby M et al (2017) Persistence images: a stable vector representation of persistent homology. J Mach Learn Res 18(8):1–35

Aggarwal CC, Hinneburg A, Keim DA (2001) On the surprising behavior of distance metrics in high dimensional spaces. In: Proceedings of the 8th international conference on database theory. Springer, Berlin, Heidelberg, ICDT '01, pp 420–434

Ali D, Asaad A, Jimenez MJ et al (2022) A survey of vectorization methods in topological data analysis. https://doi.org/10.48550/ARXIV.2212.09703

Anai H, Chazal F, Glisse M et al (2020) Dtm-based filtrations. In: Baas NA, Carlsson GE, Quick G et al (eds) Topological data analysis. Springer, Cham, pp 33–66

Arafat NA, Basu D, Bressan S (2019) Topological data analysis with ϵ-net induced lazy witness complex. In: Hartmann S, Küng J, Chakravarthy S et al (eds) Database and expert systems applications. Springer, Cham, pp 376–392

Asniar MNU, Surendro K (2022) Smote-lof for noise identification in imbalanced data classification. J King Saud Univ Comput Inf Sci 34(6, Part B):3413–3423. https://doi.org/10.1016/j.jksuci.2021.01.014
Atienza N, Gonzalez-Díaz R, Soriano-Trigueros M (2020) On the stability of persistent entropy and new summary functions for topological data analysis. Pattern Recogn 107:107–509. https://doi.org/10.1016/j.patcog.2020.107509
Attali D, Lieutier A, Salinas D (2011) Efficient data structure for representing and simplifying simplicial complexes in high dimensions. In: Proceedings of the twenty-seventh annual symposium on computational geometry. Association for Computing Machinery, New York, SoCG '11, pp 501–509. https://doi.org/10.1145/1998196.1998277
Baudry JP, Maugis C, Michel B (2012) Slope heuristics: overview and implementation. Stat Comput. https://doi.org/10.1007/s11222-011-9236-1
Bauer U (2021) Ripser: efficient computation of vietoris-rips persistence barcodes. J Appli Comput Topol. https://doi.org/10.1007/s41468-021-00071-5
Bishnoi S, Hooda BK (2020) A survey of distance measures for mixed variables. Int J Chem Stud 8:338–343. https://doi.org/10.22271/chemi.2020.v8.i4f.10087
Boissonnat J, Karthik CS (2018) An efficient representation for filtrations of simplicial complexes. ACM Trans Algorithms 14(4):44:1-44:21
Boissonnat J, Maria C (2014) The simplex tree: an efficient data structure for general simplicial complexes. Algorithmica 70(3):406–427
Boissonnat JD, Pritam S (2020) Edge collapse and persistence of flag complexes. In: Cabello S, Chen DZ (eds) 36th International symposium on computational geometry (SoCG 2020), Leibniz international proceedings in informatics (LIPIcs), vol 164. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp 19:1–19:15. https://doi.org/10.4230/LIPIcs.SoCG.2020.19
Boissonnat J, Karthik CS, Tavenas S (2017) Building efficient and compact data structures for simplicial complexes. Algorithmica 79(2):530–567
Broder AZ, Kirsch A, Kumar R et al (2010) The hiring problem and lake Wobegon strategies. SIAM J Comput 39(4):1233–1255. https://doi.org/10.1137/07070629X
Bubenik P, Dłotko P (2017) A persistence landscapes toolbox for topological statistics. J Symb Comput 78:91–114. https://doi.org/10.1016/j.jsc.2016.03.009
Caillerie C, Michel B (2011) Model selection for simplicial approximation. Found Comput Math 11(6):707–731
Carlsson G, Gabrielsson RB (2020) Topological approaches to deep learning. In: Topological data analysis. Springer, pp 119–146
Carrière M, Cuturi M, Oudot S (2017) Sliced wasserstein kernel for persistence diagrams. In: Proceedings of the 34th international conference on machine learning, vol 70. JMLR.org, ICML'17, pp 664–673
Carriere M, Chazal F, Ike Y, et al (2020) Perslay: a neural network layer for persistence diagrams and new graph topological signatures. In: Chiappa S, Calandra R (eds) Proceedings of the twenty third international conference on artificial intelligence and statistics, proceedings of machine learning research, vol 108. PMLR, pp 2786–2796
Chawla N, Bowyer K, Hall L et al (2002) Smote: synthetic minority over-sampling technique. J Artif Intell Res (JAIR) 16:321–357. https://doi.org/10.1613/jair.953
Chen Y (2015) The distance-decay function of geographical gravity model: Power law or exponential law? Chaos, Solitons Fractals 77:174–189. https://doi.org/10.1016/j.chaos.2015.05.022
Chung YM, Lawson A (2022) Persistence curves: A canonical framework for summarizing persistence diagrams. Adv Comput Math 48(1):6. https://doi.org/10.1007/s10444-021-09893-4
Curry J, Mukherjee S, Turner K (2018) How many directions determine a shape and other sufficiency results for two topological transforms. arXiv: Algebraic Topology
de Silva V, Morozov D, Vejdemo-Johansson M (2011) Persistent cohomology and circular coordinates. Discrete Comput Geom 45(4):737–759
de Silva V, Carlsson G (2004) Topological estimation using witness complexes. In: Gross M, Pfister H, Alexa M, et al (eds) SPBG'04 symposium on point-based graphics 2004. The Eurographics Association. https://doi.org/10.2312/SPBG/SPBG04/157-166
Dey TK, Fan F, Wang Y (2014) Computing topological persistence for simplicial maps. SOCG'14, Association for Computing Machinery, New York
Deza MM, Deza E (2013) Generalizations of metric spaces. Springer, Berlin, Heidelberg, pp 67–78. https://doi.org/10.1007/978-3-642-30958-8_3

Dietterich TG (2000) Ensemble methods in machine learning. Multiple classifier systems. Springer, Berlin, Heidelberg, pp 1–15
Dixon JK (1979) Pattern recognition with partly missing data. IEEE Trans Syst Man Cybern 9(10):617–621. https://doi.org/10.1109/TSMC.1979.4310090
Dua D, Graff C (2017) UCI machine learning repository. http://archive.ics.uci.edu/ml
Edelsbrunner H, Harer J (2010) Computational Topology—an Introduction. American Mathematical Society, Michigan. https://doi.org/10.1007/978-3-540-33259-6_7
Edelsbrunner, Letscher, Zomorodian (2002) Topological persistence and simplification. Discrete Comput Geom 28(4):511–533. https://doi.org/10.1007/s00454-002-2885-2
Fernández A, García S, Galar M et al (2018) Foundations on imbalanced classification. Springer, Cham, pp 19–46. https://doi.org/10.1007/978-3-319-98074-4_2
Francois D, Wertz V, Verleysen M (2007) The concentration of fractional distances. IEEE Trans on Knowl and Data Eng 19(7):873–886. https://doi.org/10.1109/TKDE.2007.1037
Freeman PR (1983) The secretary problem and its extensions: a review. Int Stat Rev Revue Internationale de Statistique 51(2):189–206
Gabrielsson RB, Nelson BJ, Dwaraknath A, et al (2020) A topology layer for machine learning. In: PMLR, pp 1553–1563
Garside K, Henderson R, Makarenko I et al (2019) Topological data analysis of high resolution diabetic retinopathy images. PLoS ONE 14(5):e0217,413-e0217,413. https://doi.org/10.1371/journal.pone.0217413
Ghrist R (2008) Barcodes: the persistent topology of data. Bull (New Series) Am Math Soc 45:61–75
Goyal A, Rathore L, Kumar S (2021) A survey on solution of imbalanced data classification problem using smote and extreme learning machine. In: Sharma H, Gupta MK, Tomar GS et al (eds) Communication and intelligent systems. Springer, Singapore, pp 31–44
Harris CR, Millman KJ, van der Walt SJ et al (2020) Array programming with NumPy. Nature 585(7825):357–362. https://doi.org/10.1038/s41586-020-2649-2
Hatcher A (2002) Algebraic Topology. Cambridge University Press, Cambridge
Hensel F, Moor M, Rieck B (2021) A survey of topological machine learning methods. Front Artif Intell 4:123. https://doi.org/10.3389/frai.2021.681108
Hofer C, Kwitt R, Niethammer M, et al (2017) Deep learning with topological signatures. In: Proceedings of the 31st international conference on neural information processing systems. Curran Associates Inc., Red Hook, NIPS'17, pp 1633–1643
Hunter JD (2007) Matplotlib: a 2d graphics environment. Comput Sci Eng 9(3):90–95. https://doi.org/10.1109/MCSE.2007.55
Ibrahim H, Anwar SA (1878) Classification of imbalanced data using support vector machine and rough set theory: a review. J Phys Conf Ser 1:012054. https://doi.org/10.1088/1742-6596/1878/1/012054
Japkowicz N, Shah M (2011) Evaluating learning algorithms: a classification perspective. Cambridge University Press, New York
Ji Z, Wang CL (2022) Efficient exact k-nearest neighbor graph construction for billion-scale datasets using gpus with tensor cores. In: Proceedings of the 36th ACM international conference on supercomputing. Association for Computing Machinery, New York, ICS '22. https://doi.org/10.1145/3524059.3532368
Jiang G, Wang W (2017) Error estimation based on variance analysis of k-fold cross-validation. Pattern Recogn 69:94–106
Kindelan R, Frías J, Cerda M, et al (2021) Classification based on topological data analysis. 2102.03709
Kubat M, Holte R, Matwin S (1997) Learning when negative examples abound. In: van Someren M, Widmer G (eds) Machine learning: ECML-97. Springer, Berlin, Heidelberg, pp 146–153
Lam L, Suen CY (1997) Application of majority voting to pattern recognition: an analysis of its behavior and performance. IEEE Trans Syst Man Cybern Part A 27:553–568
Luo H, Patania A, Kim J et al (2021) Generalized penalty for circular coordinate representation. Found Data Sci 3(4):729–767
Majumdar S, Laha AK (2020) Clustering and classification of time series using topological data analysis with applications to finance. Expert Syst Appl 162(113):868. https://doi.org/10.1016/j.eswa.2020.113868
Maria C, Boissonnat J, Glisse M et al (2014) The gudhi library: simplicial complexes and persistent homology. In: Hong H, Yap C (eds) Mathematical software-ICMS 2014. Springer, Berlin, Heidelberg
McInnes L, Healy J, Melville J (2020) Umap: uniform manifold approximation and projection for dimension reduction. 1802.03426

Mitchell TM (1997) Machine learning, international edition. McGraw-Hill Series in Computer Science, McGraw-Hill
Motwani R, Raghavan P (1995) Randomized algorithms. Cambridge University Press, Cambridge. https://doi.org/10.1017/CBO9780511814075
Navarro G (2002) Searching in metric spaces by spatial approximation. VLDB J
Pedregosa F, et al (2012) Scikit-learn: machine learning in python. J Mach Learn Res 12
Pérez JB, Hauke S, Lupo U, et al (2021) giotto-ph: a python library for high-performance computation of persistent homology of vietoris–rips filtrations. 2107.05412
Rabadan R, Blumberg AJ (2019) Topological data analysis for genomics and evolution: topology in biology. Cambridge University Press, Cambridge. https://doi.org/10.1017/9781316671665
Ren S, Wu C, Wu J (2021) Computational tools in weighted persistent homology. Chin Ann Math Ser B 42(2):237–258. https://doi.org/10.1007/s11401-021-0255-8
Rouvreau V (2022) Cython interface. In: GUDHI user and reference manual, 3.6.0 edn. GUDHI Editorial Board. https://gudhi.inria.fr/python/3.6.0/
Saadat-Yazdi A, Andreeva R, Sarkar R (2021) Topological detection of Alzheimer's disease using Betti curves. In: Reyes M, Henriques Abreu P, Cardoso J et al (eds) Interpretability of machine intelligence in medical image computing, and topological data analysis and its applications for medical data. Springer, Cham, pp 119–128
Samet H (2006) Foundations of multidimensional and metric data structures. Morgan Kaufman, San Francisco
Seversky LM, Davis S, Berger M (2016) On time-series topological data analysis: new data and opportunities. In: CVPRW, pp 1014–1022. https://doi.org/10.1109/CVPRW.2016.131
Shepard D (1968) A two-dimensional interpolation function for irregularly-spaced data. In: Proceedings of the 1968 23rd ACM national conference. Association for Computing Machinery, New York, ACM '68, pp 517–524. https://doi.org/10.1145/800186.810616
The HDF Group (1997–2022) Hierarchical data format, version 5. https://www.hdfgroup.org/HDF5/
Umeda Y (2017) Time series classification via topological data analysis. Trans Jpn Soc Artif Intell 32:D–G72_1. https://doi.org/10.1527/tjsai.D-G72
Venkataraman V, Ramamurthy K, Turaga P (2016) Persistent homology of attractors for action recognition. In: 2016 IEEE international conference on image processing, ICIP 2016-proceedings. IEEE Computer Society, pp 4150–4154. https://doi.org/10.1109/ICIP.2016.7533141
Vuttipittayamongkol P, Elyan E (2020) Neighbourhood-based undersampling approach for handling imbalanced and overlapped data. Inf Sci 509:47–70. https://doi.org/10.1016/j.ins.2019.08.062
Wagner H, Dłotko P (2014) Towards topological analysis of high-dimensional feature spaces. Comput Vis Image Underst 121:21–26. https://doi.org/10.1016/j.cviu.2014.01.005
Wilson DR, Martinez TR (1997) Improved heterogeneous distance functions. J Artif Int Res 6(1):1–34
Yershov DS, LaValle SM (2011) Simplicial dijkstra and a* algorithms for optimal feedback planning. In: 2011 IEEE/RSJ international conference on intelligent robots and systems, pp 3862–3867. https://doi.org/10.1109/IROS.2011.6095032
Zhang S, Xiao M, Wang H (2020) Gpu-accelerated computation of vietoris-rips persistence barcodes. arXiv:2003.07989
Zhang X, Li Y, Kotagiri R et al (2017) Krnn: k rare-class nearest neighbour classification. Pattern Recognit 62:33–44. https://doi.org/10.1016/j.patcog.2016.08.023