Information Fusion 89 (2023) 1–15

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Full length article

A One-Class Classification method based on Expanded Non-Convex Hulls

David Novoa-Paradela [*], Oscar Fontenla-Romero, Bertha Guijarro-Berdiñas

Universidade da Coruña, CITIC, Campus de Elviña s/n, 15008, A Coruña, Spain

ARTICLE INFO

Keywords:
Machine learning
One-Class Classification
Convex Hull
Delaunay triangulation
Random projections
Ensemble learning

ABSTRACT

This paper presents an intuitive, robust and efficient One-Class Classification algorithm. The method developed is called OCENCH (One-class Classification via Expanded Non-Convex Hulls) and bases its operation on the construction of subdivisible and expandable non-convex hulls to represent the target class. The method begins by reducing the dimensionality of the data to two-dimensional spaces using random projections. After that, an iterative process based on Delaunay triangulations is applied to these spaces to obtain simple polygons that characterizes the non-convex shape of the normal class data. In addition, the method subdivides the non-convex hulls to represent separate regions in space if necessary. The method has been evaluated and compared to several main algorithms of the field using real data sets. In contrast to other methods, OCENCH can deal with non-convex and disjointed shapes. Finally, its execution can be carried out in a parallel way, which is interesting to reduce the execution time.

1. Introduction

Anomaly detection (AD) is the branch of machine learning that builds models capable of differentiating between normal and anomalous data. A priori, this turns anomaly detection into a classification problem with only two classes. However, since anomalies tend to occur sporadically, normal data are the ones that prevail in these scenarios, so it is common that models are required that can be trained with only data of the normal class. Assuming all training instances have only one class label, in AD is common to use One-Class Classification (OCC) techniques. OCC is a special case of binary classification, where data observed during training is from a single positive class. The objective is to represent the boundary of the known as positive or normal class with high precision in order to be able to classify new data by comparing them with these limits. The ideal boundary is often imprecise, and can even evolve over time, which is a great challenge. The availability of labeled data to train and validate models along with the presence of noise in the training set can be another challenge. For this reason, and although some successful applications can be found in fields such as medical anomaly detection [1], fraud detection [2], intrusion detection [3] or maintenance of industrial systems [4], this is still a very active line of research.

In AD, the distribution presented by the data sets can be decisive for the application of certain types of techniques. AD systems play a vital role in a wide range of real-world applications where the shape of the normal class will not always be convex. A convex representation can never provide good characterization of a non-convex distribution, so it is necessary to develop specific methods capable of operating on non-convex data sets. In addition, the distribution of the data can be composed of several separate regions in space that are difficult to represent by means of a single hull. This paper presents OCENCH (One-class Classification via Expanded Non-Convex Hulls), an OCC and AD method based on the use of random projections [5] of the original data space to reduce their complexity, followed by a process based on Delaunay triangulation [6] to geometrically represent the normal class in these low-dimensional spaces through subdivisible and expandable non-convex hulls. The limits of the normal class are iteratively adapted during the training phase. This process is carried out based on a normalized parameter that controls the adjustment level and can be easily tuned by the user for each scenario. Furthermore, if in a low-dimensional space the normal class cannot be accurately represented by a single non-convex hull, it will be subdivided as many times as necessary to fit the shape of the data. The developed OCENCH algorithm allows working with non-convex data sets in a novel way, offering a robust behavior and remarkable performance, positioning itself as an alternative for both convex and non-convex problems.

2. Related work

This section summarizes the main types of anomaly detection techniques available in the state-of-the-art [7,8] and focuses on methods based on non-convex hull, as they are the most related. In anomaly detection, depending on the way in which the methods approximate the solution of the problem, we can classify them as follows:

[*] Corresponding author.
E-mail addresses: david.novoa@udc.es (D. Novoa-Paradela), oscar.fontenla@udc.es (O. Fontenla-Romero), berta.guijarro@udc.es (B. Guijarro-Berdiñas).

https://doi.org/10.1016/j.inffus.2022.07.023
Received 5 November 2021; Received in revised form 20 April 2022; Accepted 28 July 2022
Available online 6 August 2022

- Probabilistic methods: these methods assume that the normal data are generated by a probability distribution function. Their objective is to estimate this density function from the normal data to define the boundaries of normality in the data space and test whether a new sample comes from the same distribution or not, with the hypothesis that normal data will occur in regions with high probabilities whilst anomalies in regions with lower probabilities [9,10]. If the assumptions regarding the distribution of data are true, in the case of the parametric ones, these techniques provide a statistically justifiable solution. However, this is not always the case, especially when working with large data sets with a high number of variables. In addition, the presence of abnormalities during training may prevent them from obtaining good results.
- Distance-based methods: These methods rely on well-defined distance metrics to compute the similarity between two data points. They assume that normal data are in the form of dense neighborhoods, while anomalies occur in regions far from these neighborhoods [11–14]. They can work in a unsupervised way. The main disadvantages of these methods are that anomalies can also form significant groupings with each other and, furthermore, the computational complexity to group the data is usually a bottleneck.
- Reconstruction based methods: neural networks can be trained to capture the shape of normal data if it is used as training data. The most common type of neural network in AD is the Autoencoder Network [15], which make this decision by calculating the reconstruction error, defined to be the distance between the test vector and the output of the system. They can operate in a supervised and unsupervised way, without requiring previous information of the data. However, they are usually computationally expensive and can present many hyperparameters to tune.
- Methods based on information theory: these methods assume that anomalies significantly alter the information contained in the normal data set and, therefore, they process the content of data sets through measures, such as entropy or relative entropy, to check whether there are differences in these measures when potential abnormal data is included [16,17]. These techniques can work in a unsupervised way and do not assume information about the distribution of data, however, the confidence in their predictions is highly dependent on the metric used. In addition, if the number of anomalies is very low, certain metrics are unable to detect them due to their low impact.
- Boundary-based methods: they generate a limit based on the structure of training data that separates the classes in a space. These classifiers become insensitive to the size and density of each class, since the classification of the new data is determined only by its location with respect to the boundary, and not by its density [18,19]. Some of these techniques need data from both classes to train (normal and abnormal), which is difficult to achieve in problems where there are almost no signs of anomalies. In addition, some methods are sensitive to noise, which can cause over-adjustments.

This work is focused on boundary-based anomaly detection methods. A important group of boundary-based techniques that have successfully solved OCC problems are those based on convex hulls [20,21]. A convex hull (CH) is the smallest convex polyhedron that contains a set of data points. The usefulness of this geometric shape in OCC and AD is to serve as a limit that contains the normal data set and allows the classification of new data based on whether they are inside the convex hull (normal data) or outside (anomaly). However, these systems play a vital role in a wide range of real-world applications where the shape of the normal class will not always be convex. As a convex representation can never provide good characterization of a non-convex distribution, a whole line of research has emerged to work with non-convex shapes using non-convex hulls. The objective of these methods is the same, to build a non-convex hull around the normal data as a decision boundary. We can distinguish between two types of approximations depending on whether they try to calculate the non-convex hull in the original space or in a dimensionally reduced space:

- Computing the boundary in the original space. Calculating the limits of the normal class in the original data space despite the high cost, is a strategy followed by some methods. A common way to solve the problem is calculating the convex hull and digging to obtain the non-convex version. For example, DINA algorithm produces a non-convex hull by removing the hollow spaces from the n-dimensional convex hull [22]. To do this, it uses the normal vectors of the hyperplanes that form the facets of the structure, giving rise to what is called oriented non-convex hulls. The normal vector of a facet in a non-convex hull is the normal vector of the corresponding hyperplane, where this facet is located, in the direction from this hyperplane towards the space outside the non-convex hull. Given the facets of an oriented non-convex hull, the algorithm can determine if a new point is inside or outside this non-convex hull. Another approach is the one developed by Park et al. [23], a method that follows the same philosophy but carries out the digging stage using the closest data from inside the hull as support. An edge will be pruned if its size in relation to the distance to its support point exceeds a certain threshold defined by the user. This threshold influences the smoothness of the non-convex hulls. Although they can be appropriate for low dimensional problems, the calculation of non-convex hulls in high dimensional spaces is computationally very expensive and challenging, so there are not too many methods of this type.
- Computing the boundary in a reduced space. Working with high-dimensional data sets makes it difficult to calculate these structures due to their high computational cost. One solution is to reduce the dimensionality of the data set to make these operations manageable. The most common methods to simplify the representation of the problem by reducing the dimensionality of data are Principal Component Analysis [24], Linear Discriminant Analysis [25] or Autoencoder Networks [15]. In the best of cases, a two-dimensional representation will be obtained in which the calculation of both convex and non-convex hulls is a simple operation. However, this remarkable reduction is not always possible, in most cases it is necessary to use more than two dimensions in order not to lose information. To achieve a good two-dimensional representation, some methods opt for the use of random projections. This random projection technique is based on the idea that high-dimensional data spaces can be projected into a lower dimensional space without significantly losing the data structure if multiple projections are used [26]. An example of this is NCBoP [21] which, based on the Approximate Polytope Ensemble (APE) algorithm [20], takes advantage of these reduced spaces to build non-convex hulls in a simple way.

3. Background

This section summarizes the main ideas of the two methods taken as the basis for the development of the proposed method: the random projections used in the APE algorithm developed by Casale et al. to reduce the complexity of the data sets [20], and the process for the generation of simple polygons developed by Duckham et al. based on Delaunay triangulation [6].

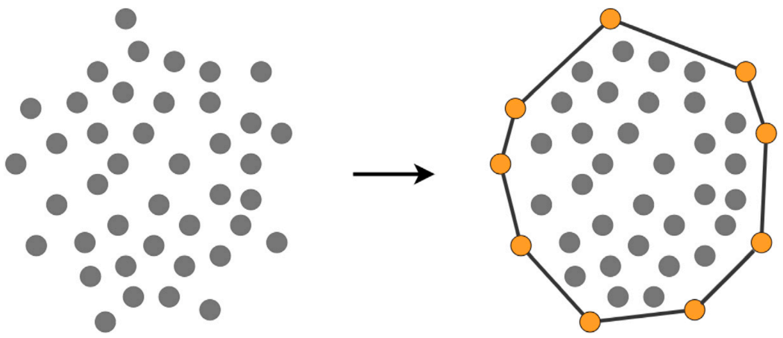

Fig. 1. Construction of a convex hull from a cloud of points.

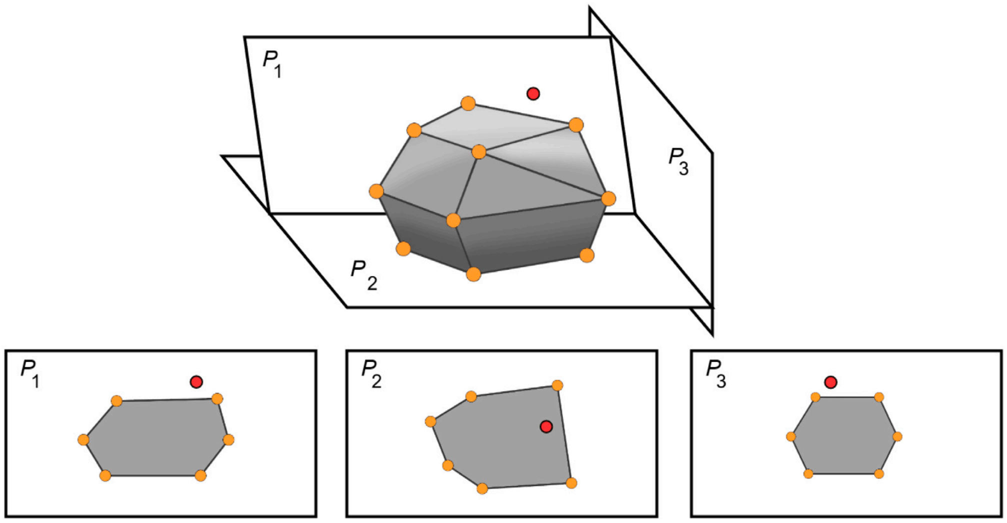

Fig. 2. Three 2-D projections of a three-dimensional boundary and a new point (red) that will be classified as an anomaly.

3.1. *Approximate polytope ensemble*

A convex hull (CH) is the smallest convex polyhedron that contains a set of data points. Fig. 1 shows the CH of a two-dimensional data set, in this case the CH is a polygon. The usefulness of this geometric structure in anomaly detection problems is to represent the normal class, so that classifying a new data consists of checking whether it is inside the CH (normal data) or outside (anomaly).

Calculating the CH in high-dimensional spaces is a very expensive task. Due to this, as some other anomaly detection methods, APE [20] chooses to project the data into two-dimensional spaces where it will calculate the limits in a more computationally affordable way.

In the training phase, the APE algorithm projects the normal data set using random projections and calculates the convex hulls in each of these two-dimensional spaces. To classify a new data point, first it is projected in the two-dimensional spaces. Afterwards, for each of the two-dimensional spaces, given the set of vertices of its CH, it is possible to check if the point is within the normal class or not. Finally, a point will be classified as normal only if it is within the CH for every projection. If in any of the projections the point is outside, it will be considered anomalous. Fig. 2 shows this idea.

Building a CH in two-dimensional spaces and check if a point falls inside are common tasks in geometric computing for which there are very efficient solutions [27]. However the convex nature of this kind of method makes it unsuitable for non-convex data sets. Furthermore, the use of a single CH per projection space makes it difficult to represent disjointed regions.

3.2. *Generation of simple polygons to characterize the shape of a set of points*

The procedure developed by Duckham et al. [6] makes it possible to efficiently and flexibly construct the non-convex simple polygon that characterizes the shape of a set of input points in a plane. This technique is based on Delaunay triangulation, which allows to model a data set using a polygonal surface with the following properties:

- All the points are connected to each other and form as many triangles as possible without their edges crossing.
- Triangles are defined so that the closest points are connected to each other by an edge.
- The triangles formed are as regular as possible, that is, their minor angles are maximized and the length of their edges is minimized.

An example of Delaunay triangulation for a half-moon data set can be seen in Fig. 3.

The properties of triangulation allow to apply a pruning process to adapt the boundaries to non-convex shapes. This pruning process iteratively removes those exterior edges of the closure that exceed a size defined through a parameter l. The exterior edges are sorted from largest to smallest, after which it is checked if they exceed the threshold. If an edge is greater than l, then it is eliminated, and the two edges with which it formed a triangle will become part of the provisional closure. Fig. 4 shows this procedure. The pruning process will be repeated until no edge of the closure exceeds the size defined by the l parameter. Starting from the triangulation in Figs. 3, 5 shows an example of how the parameter l can influence the adjustment level

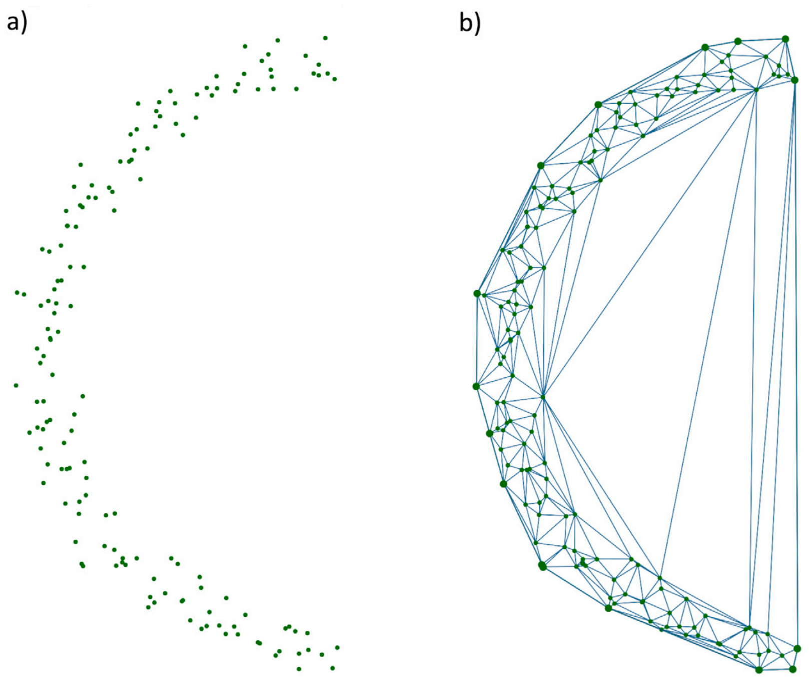

Fig. 3. Delaunay triangulation (b) of a cloud of points (a).

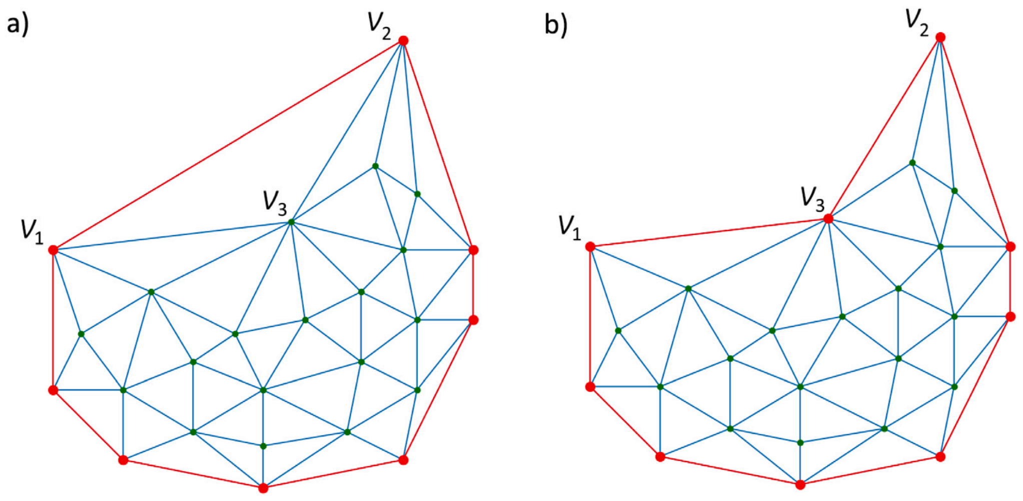

Fig. 4. Pruning process of an edge: (a) The edge formed by vertices $\mathbf{v}_1$ and $\mathbf{v}_2$ is the longest of the closure (red lines) and exceeds the threshold established by parameter l, therefore it must be pruned; (b) The two edges that were part of the triangle in which the pruned edge was located (edge between $\mathbf{v}_1$ and $\mathbf{v}_3$ and between $\mathbf{v}_2$ and $\mathbf{v}_3$) become part of the new provisional closure.

of a boundary. As can be seen, the data set is non-convex, so the edges of the interior of the half-moon (the non-convex area) will be the first to be pruned because they are the largest. A too large l value (1.8) results in insufficiently pruned closures, while too small a value (0.1) can generate very steep boundaries.

The method achieves excellent results, however, it cannot be precisely adapted in cases where the data is separated in different areas of the space (Fig. 6a). These situations, in which it is necessary to represent more than one isolated region in space, are common in real problems of AD or OCC. In these scenarios, it would be convenient to have more than one non-convex hull to be able to represent each region independently. As seen in Fig. 6c, the results obtained by the method of Duckham et al. are not the most accurate in these circumstances, as a single non-convex hull is used to represent the whole set of points.

For all this, in this work we propose an improvement of the Duckam et al. algorithm for its application in disjointed regions which, together with the idea of random projections described in Section 3.1, has allowed us to design a new one-class classification algorithm. The algorithm will drastically reduce the dimensionality of any input data set using random projections. On these reduced spaces, it will calculate the closures that best adapt to the data in a flexible way, since they can be recursively subdivided as many times as necessary to adapt to the shape of the data. This will allow working not only with both convex and non-convex data sets, but also with data sets that present connected and unconnected regions.

4. The proposed method

The main objective of the proposed OCENCH (One-class Classification via Expanded Non-Convex Hulls) method is to fit robustly to the non-convex shape of the data of the normal class during the training phase to achieve a good performance when the system is classifying

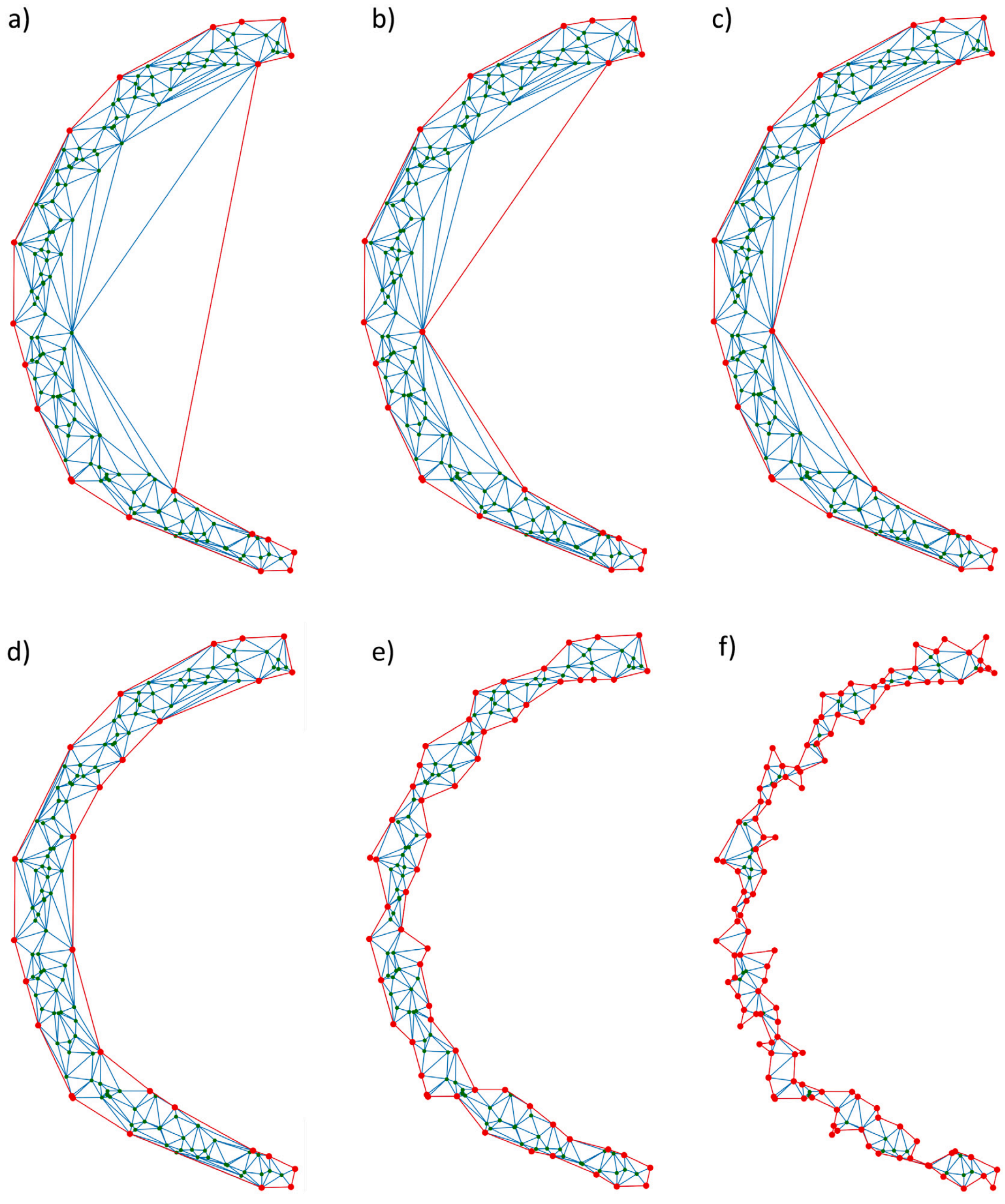

Fig. 5. Final non-convex hulls (red) result of applying the pruning process with different values of l over a non-convex data set (green) previously triangulated (blue): (a) l = 1.8; (b) l = 1.6; (c) l = 1.2; (d) l = 0.6; (e) l = 0.2; (f) l = 0.1.

new data. This section presents the theoretical foundations of the steps followed by the method:

- Dimensionality reduction: reduce the complexity of the normal data set by projecting the normal class data into two-dimensional spaces.
- Calculation of the non-convex hulls: construction of non-convex hulls in these two-dimensional spaces to represent the limits of the normal class.
- Subdivision of non-convex hulls: subdivision of non-convex hulls to more accurately represent the shape of the data.
- Calculation of the expanded non-convex hulls: expansion of the final non-convex hulls to smooth the classification of new instances.

4.1. Dimensionality reduction

Given a training data set $\mathbf{X} \in \mathbb{R}^{d\times n}$ (d variables $\times$ n samples) of normal data, the first step of the algorithm is to reduce its dimensionality carrying out projections into 2-D spaces. To project the data, a certain number of $[2 \times d]$ projection matrices are randomly generated. The number of projections τ is the first parameter that must be defined by the user, taking into account that more projections implies a representation of the class with greater precision in exchange for higher computational cost. To determine the appropriate number of projections for a scenario, it is advisable to experiment with different values to find a balance between precision and time. As a guide, there are experimental studies [28–30] based in the Johnson–Lindenstrauss lemma that allow estimating the appropriate number of projections based on characteristics such as the size of the data set. Once the projection matrices have been generated, the training set will be projected

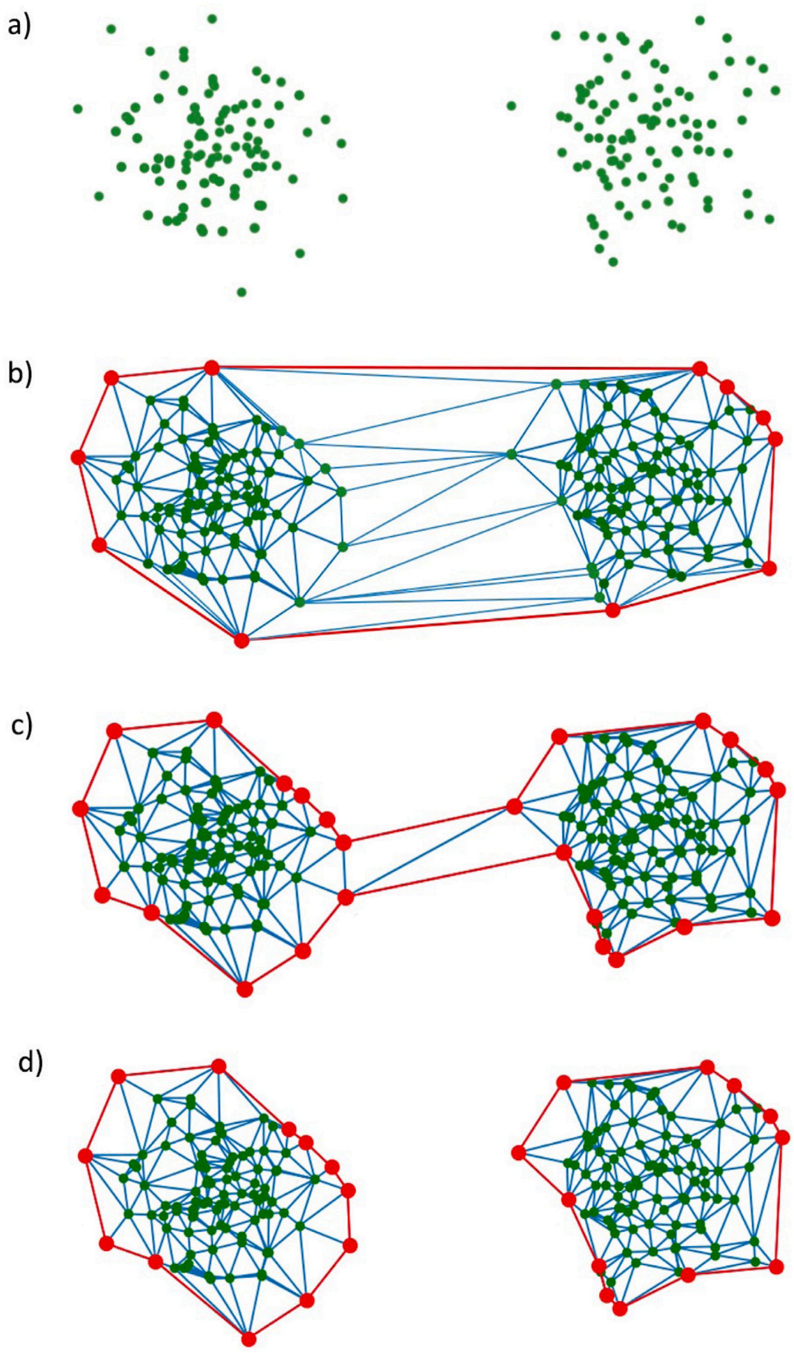

Fig. 6. Characterization of two separate point regions using OCENCH: (a) Projected data set; (b) Initial convex hull; (c) Pruned non-convex hull; (d) Subdivided non-convex hulls.

using each of these projection matrices. The steps described below are applied independently to the convex hull at each of the projections.

4.2. *Calculation of the non-convex hulls*

Once the training data are projected in two-dimensional spaces (Fig. 6a), the convex hull that surrounds the data is calculated for each projection using the algorithm proposed by Duckam et al. [6] (Fig. 6b). Due to the characteristics of this process, the result is a convex hull, so following the Duckam et al. [6] method, an edge pruning process is carried out to adjust the limits to non-convex shapes (Fig. 6c). As commented in Section 3.2, this pruning process iteratively removes those edges of the closure that exceed a size defined by the user through a parameter l. Since projecting the data causes that the size of the hulls is not similar in all the projections, we propose that to facilitate the choice of the parameter l the data of each projection be normalized independently using a standard scaler with zero mean and unit variance. That process allows to choose a value of l that consistently fits the data across all projections.

4.3. *Subdivision of non-convex hulls*

As shown in Fig. 6c, the problem of the method presented by Duckam et al. is its inability to accurately represent separate regions in space. However, the use of a single closure is not enough in general and, therefore, a subdivision process has been implemented that results in the coexistence of multiple non-convex hulls.

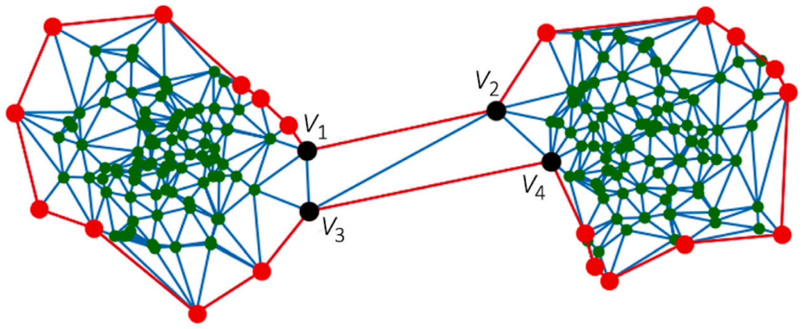

Fig. 7. Typical morphology of a Duckham et al. non-convex hull that envelops two separate regions in space generating a region of union ($\mathbf{v}_1 - \mathbf{v}_2 - \mathbf{v}_3 - \mathbf{v}_4$).

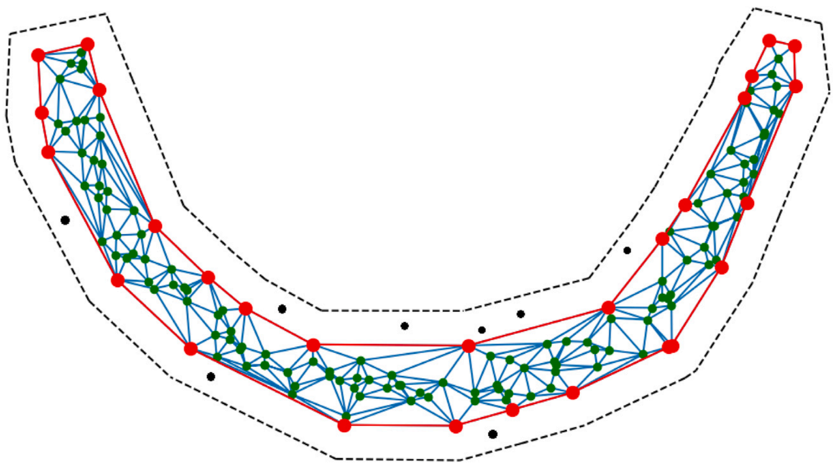

Fig. 8. Expanded non-convex hull (black lines) obtained from an initial non-convex hull (red lines). The normal data (black dots) will be correctly classified thanks to the use of the expanded margin.

To design the process of subdivision of closures, the detailed behavior of the Duckham et al. algorithm when applied to data sets with separate regions in space has been analyzed. As can be seen in Fig. 7, if there is more than one region, the method produces a single closure that wraps these separate regions in space, keeping them connected by a union region. This union region is characterized by the lack of data points in its interior, being formed by only two triangles whose edges were not pruned in the stage described in Section 4.2, since their elimination would alter the properties of the triangulation. We can define these unions as a region of the triangulation composed of two triangles that make up a quadrilateral, which has two opposite external edges and has no data points inside.

In this case, the union region is formed by triangles $\mathbf{v}_1 - \mathbf{v}_2 - \mathbf{v}_3$ and $\mathbf{v}_2 - \mathbf{v}_3 - \mathbf{v}_4$. An intuitive way to subdivide this non-convex hull in two would be to eliminate the two external edges $\mathbf{v}_1 - \mathbf{v}_2$ and $\mathbf{v}_3 - \mathbf{v}_4$ and the common edge of both triangles $\mathbf{v}_2 - \mathbf{v}_3$.

Since the existence of these union regions is very frequent after the application of the pruning process, it has been decided to implement a process of subdivision of closures that tries to locate these structures to treat them in a specific way. To do this, the edges of the closure will be iteratively traversed to verify the existence of this type of union regions. If the existence of a union like the one described in Fig. 7 is determined, it will be eliminated, giving rise to two independent non-convex hulls. These two new non-convex hulls will be readjusted, if needed, through the pruning process described in Section 4.2. The subdivision and pruning processes are repeated iteratively until there are no union regions and the non-convex hulls are fully adjusted. In Fig. 6d it can be seen the result of applying this subdivision process. The identification of several closures will allow a more precise characterization of the regions.

Although the subdivision process is recommended, a *subdivision* hyperparameter has been included in the final algorithm to enable or disable this behavior. Given the existence of scenarios in which it is known with certainty that there are no disjointed regions, omitting this subdivision stage could be useful to reduce the algorithm training time.

4.4. Calculation of the expanded non-convex hulls

In some cases, there may be normal data that is wrongly classified as anomalies because they are very close to the training data but outside the limits of the non-convex hulls (NCH). To avoid the effect of over-adjustment of the training data, the method uses a third parameter, the λ expansion factor. This parameter allows to generate expandable non-convex hulls (ENCH), whose purpose is to soften the classification decision of new data. Fig. 8 shows an example of an ENCH. Each vertex of the closing boundary will be expanded based on the incenter of the triangle that it forms with the other two boundary vertices with which it is connected as will be explained below.

Equations used to calculate the incenters and the ENCH are the following. Given a list of vertex $V \subseteq \mathbb{R}^2$ of a NCH and one vertex $\mathbf{v}_i \in V$ connected with vertices $\mathbf{v}_{i-1}$ and $\mathbf{v}_{i+1}$, its expanded version $\mathbf{v}_i^\lambda$ is defined with respect to their incenter point $\mathbf{c}_i$ as:

$$\mathbf{v}_i^\lambda = \mathbf{v}_i + s\lambda \frac{(\mathbf{v}_i - \mathbf{c}_i)}{\|\mathbf{v}_i - \mathbf{c}_i\|}, \forall\ \mathbf{v}_i \in V \tag{1}$$

where $\|\mathbf{v}_i - \mathbf{c}_i\|$ is the distance between the vertex $\mathbf{v}_i$ and $\mathbf{c}_i$, s is the sign of the expansion and the expansion parameter $\lambda \in [0, +\infty)$. The incenter $\mathbf{c}_i$ is defined as:

$$\mathbf{c}_i = \frac{\mathbf{v}_{i-1}\|\mathbf{v}_i - \mathbf{v}_{i+1}\| + \mathbf{v}_i\|\mathbf{v}_{i-1} - \mathbf{v}_{i+1}\| + \mathbf{v}_{i+1}\|\mathbf{v}_{i-1} - \mathbf{v}_i\|}{\|\mathbf{v}_i - \mathbf{v}_{i+1}\| + \|\mathbf{v}_{i-1} - \mathbf{v}_{i+1}\| + \|\mathbf{v}_{i-1} - \mathbf{v}_i\|}, \forall\ \mathbf{v}_i \in V \tag{2}$$

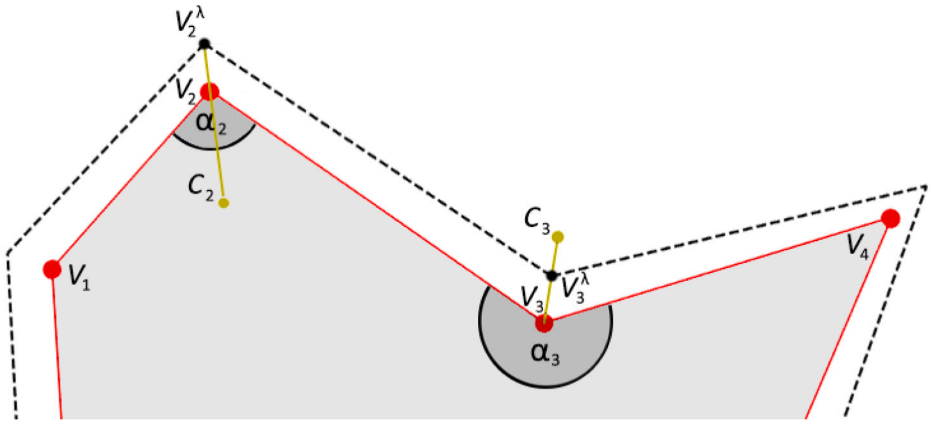

Fig. 9. Expansion process of two vertices ($\mathbf{v}_2$ and $\mathbf{v}_3$) of a NCH (red lines). The interior angle formed by vertex $\mathbf{v}_2$ with vertices $\mathbf{v}_1$ and $\mathbf{v}_3$ is convex ($\alpha_2 < 180$), so the incenter of the triangle is inside the NCH ($\mathbf{c}_2$). However, the interior angle formed by vertex $\mathbf{v}_3$ with vertices $\mathbf{v}_2$ and $\mathbf{v}_4$ is non-convex ($\alpha_3 > 180$), so the incenter of this triangle is outside the NCH ($\mathbf{c}_3$). The incenters allow calculating the direction (yellow lines) in which the vertices should be expanded ($\mathbf{v}_2^\lambda$ and $\mathbf{v}_3^\lambda$), thus obtaining the ENCH (black lines).

The sign of the expansion s of a vertex $\mathbf{v}_i$ is defined based on the internal angle α that it forms with the vertices $\mathbf{v}_{i-1}$ and $\mathbf{v}_{i+1}$ as:

$$s = \begin{cases} 1, & \text{if } \alpha \leq 180 \\ -1, & \text{if } \alpha > 180 \end{cases} \tag{3}$$

Depending on the interior angle α that the vertex $\mathbf{v}_i$ forms with the other two ($\mathbf{v}_{i-1}$ and $\mathbf{v}_{i+1}$), we can determine if the area where $\mathbf{v}_i$ is located is convex ($\alpha < 180$) or non-convex ($\alpha > 180$). This will influence the expansion of the vertex since the incenter of a non-convex zone is always located outside the NCH, which will invert the sign of the expansion operation (Eq. (1)). Fig. 9 exemplifies this behavior.

The parameter λ specifies a constant contraction ($0 < \lambda < 1$) or extension ($\lambda > 1$) of the NCH with respect to $\mathbf{c}_i$. We will always use $\lambda > 1$ since we are not interested in reducing the NCH. Using a too high λ value to expand a non-convex vertex could cause the ENCH to intersect itself, resulting in a complex polygon. To avoid this behavior, after each expansion, the algorithm checks whether the generated ENCH is a simple or complex polygon. If the defined value of λ causes a complex hull in some projection, the NCH will not be expanded.

4.5. Pseudocode

Algorithm 1 contains the pseudocode for the OCENCH training phase where the training data will be projected, the NCHs of each projection will be calculated and, finally, expanded to be able, later on, to classify new data. In line 4, a matrix is randomly generated to, in line 5, project the training data into a two-dimensional space. In line 6, a standardization model is used to normalize the projected data, and in line 7 the non-convex hulls (NCH) that surrounds these data are calculated. In line 8, the expanded version of the non-convex hulls (ENCH) are calculated from the vertices of the NCH and their incenters. In line 9, the system checks if the chosen λ value is valid by verifying if the expanded polygons are simple or complex.

The method to compute the NCH is shown in Algorithm 2, based on the one developed by Duckham et al. to geometrically represent the normal class in low-dimensional spaces through non-convex hulls but improved in this work to be able to perform subdivisions. In line 3, a Delaunay triangulation is made to get the initial convex hull. In lines 4 to 6, this initial convex hull is pruned (Algorithm 3) to fit the shape of the data. If necessary, in line 7 the closure will be subdivided (Algorithm 4) if union regions are detected. If this happens, in lines 8 to 13 the subdivision and pruning processes will alternate until all the non-convex hulls are fully adjusted.

Algorithm 3, used by Algorithm 2, describes the pruning process of a NCH based on the l parameter. This algorithm modifies that developed by Duckham et al. by adding the detection of union regions. This is done while iterating over the edges to prune them, marking those that could form union regions. On line 16, those edges whose removal would alter the properties of the triangulation will be added to a list of candidate edges. This operation does not increase the complexity of the pruning algorithm, and it greatly benefits the subdivision algorithm since this list of candidate edges considerably reduces the number of necessary checks.

Algorithm 4, also used by Algorithm 2, describes the subdivision process of a NCH based on the l parameter. The goal is to use the list of candidate edges generated by Algorithm 3 to determine if union regions exist. If so, in lines 12 to 19, the method takes care of subdividing the NCH in two NCHs, as presented in Section 4.3.

Algorithm 5 describes the expansion process of the NCHs based on the λ parameter used in Algorithm 1. For each vertex of the NCH, in lines 5 to 8, the incenter and the interior angle is calculated, whilst in lines 9 to 15 the NCH is expanded.

Finally, Algorithm 6 describes the classification phase of the OCENCH to determine if a new point is normal or is an anomaly. In

Algorithm 1 : OCENCH training phase

```
Inputs:
  $\mathbf{X} \in \mathbb{R}^{d \times n}$          ▷ Training data set ($d$ variables × $n$ samples)
  $\tau \in \mathbb{N}^+$                 ▷ Number of random 2D-projections
  $l \in \mathbb{R}^+$                ▷ Maximum edge length allowed in the NCH
  $\lambda \in \mathbb{R}^+$                      ▷ Expansion parameter of the NCH
Output:
  $M$     ▷ Model composed of $\tau$ $ENCHs$, projection matrices and normalizers
1:  function TRAIN
2:      $M = \emptyset$
3:      for $t = 1 \ldots \tau$ do
4:          $\mathbf{P}_t \sim N(0,1)$               ▷ Random projection matrix $[2 \times d]$
5:          $\bar{\mathbf{X}}_t = \mathbf{P}_t\mathbf{X}$                           ▷ Project the data
6:          $\mathbf{N}_t, normalizer_t = normalize(\mathbf{X}_t)$      ▷ Data normalization
7:          $NCH_t = computeNCH(\mathbf{N}_t, l)$               ▷ NCH calculation
8:          $ENCH_t = expandNCH(NCH_t, \lambda)$             ▷ ENCH calculation
9:          if isSimple($ENCH_t$) then    ▷ Check if the ENCHs are simple
10:             $M = M \cup (ENCH_t, \mathbf{P}_t, normalizer_t)$
11:         else
12:             return Error                 ▷ Lambda value is not feasible
13:         end if
14:     end for
15:     return $M$                              ▷ Returns the final model
16: end function
```

Algorithm 2 : Function used during training to construct, prune and subdivide a 2-D non-convex hull from a set of points based on a parameter l

```
Inputs:
   $N \in \mathbb{R}^{2\times n}$          ▷ Projected data (2 variables × $n$ samples)
   $l \in \mathbb{R}^+$          ▷ Maximum edge length allowed in the NCH
Output:
   $NCH$          ▷ 2-D non-convex hull
1:  function computeNCH
2:      $\Delta, E, V, S = \emptyset$          ▷ Auxiliary lists
3:      $\Delta[1]$ = Delaunay triangulation of N
4:      E[1] = Boundary edges of $\Delta$
5:      V[1] = Boundary vertices of $\Delta$
6:      E[1], V[1], $\Delta[1]$, S[1] = Pruning(E[1], V[1], $\Delta[1]$, $l$)
7:      E, $\Delta$, $continue$ = Subdivision(E, $\Delta$, S, $l$)
8:      while $continue == True$ do
9:          for $i = 1..len(E)$ do          ▷ For each NCH
10:             E[$i$], V[$i$], $\Delta[i]$, S[$i$] = Pruning(E[$i$], V[$i$], $\Delta[i]$, $l$)
11:         end for
12:         E, $\Delta$, $continue$ = Subdivision(E, $\Delta$, S, $l$)
13:     end while
14:     return E
15: end function
```

Algorithm 3 : Function used during training to prune the edges of a 2-D non-convex hull.

```
Inputs:
   $E_i$          ▷ Non-convex hull boundary edges
   $V_i$          ▷ Boundary vertices
   $\Delta_i$          ▷ Triangulation
   $l \in \mathbb{R}^+$          ▷ Maximum edge length allowed in the NCH
Output:
   FB          ▷ New boundary edges
   $V_i$          ▷ New boundary vertices
   $\Delta_i$          ▷ New triangulation
   $C_i$          ▷ Candidate triangles that could form union regions
1:  function Pruning
2:      $FB = \emptyset$          ▷ List to store the final boundaries edges
3:      $C_i = \emptyset$          ▷ List to store potential union regions
4:      Sort the list $E_i$ in descending order by length
5:      Sort the list $V_i$ of vertices to form a chain of connected vertices
6:      while $E_i$ is not empty do
7:          e ← head($E_i$)
8:          Remove e from $E_i$
9:          if $\|e\| > l$ then
10:             Find the triangle $t$ of $\Delta_i$ that contains the two vertices of e and find its third vertex v
11:             if not (v in $V_i$) then
12:                 Remove e from the triangulation $\Delta_i$
13:                 Insert the other two edges of $t$ into $E_i$ in order
14:                 Insert v into $V_i$ in order
15:             else
16:                 Insert $t$ into $C_i$
17:             end if
18:         else
19:             Insert e into FB
20:         end if
21:     end while
22:     return FB, $V_i$, $\Delta_i$, $C_i$
23: end function
```

lines 7–8, if the new point is outside all ENCHs for some projection, it will be classified as an anomaly and it will be not necessary to continue checking the remaining projections.

Algorithm 4 : Function used during training to detect and subdivide 2-D non-convex hulls.

```
Inputs:
   E          ▷ NCH boundary edges, one list per NCH
   $\Delta$          ▷ List of triangulations, one per NCH
   C  ▷ Candidate edges that could form union regions, one list per NCH
   $l \in \mathbb{R}^+$          ▷ Maximum edge length allowed in the NCH
Output:
   E          ▷ Updated list of NCH boundary edges
   $\Delta$          ▷ Updated list of triangulations
   $continue$  ▷ $True$ if there has been any subdivision or $False$ if not
1:  function Subdivision
2:      $continue = False$
3:      for $i = 1 \dots card(E)$ do          ▷ For each NCH
4:          for $\mathbf{e_1}$ in $C_i$ do          ▷ For each candidate edge of the NCH
5:              $t_1$ = triangle of $\Delta_i$ that contains the vertices of $\mathbf{e_1}$
6:              $\mathbf{v_1}$ = third vertex of $t_1$
7:              $e\_list$ = list of edges of $C_i$ that contain vertex $\mathbf{v_1}$
8:              for $\mathbf{e_2}$ in $e\_list$ do
9:                  $t_2$ = triangle of $\Delta_i$ that contains the vertices of $\mathbf{e_2}$
10:                 $\mathbf{v_2}$ = third vertex of $t_2$
11:                 if $\mathbf{v_2} \in \mathbf{e_1}$ then
12:                     Remove the edge $\mathbf{e_1}$ from $E_i$ and $\Delta_i$
13:                     Remove the edge $\mathbf{e_2}$ from $E_i$ and $\Delta_i$
14:                     Remove the edge $\mathbf{v_1}$-$\mathbf{v_2}$ from $\Delta_i$
15:                     Add new boundary edges to E
16:                     Divide $E_i$ structure in two and update E
17:                     Divide $\Delta_i$ structure in two and update $\Delta$
18:                     $continue = True$
19:                     $Break$          ▷ Finish subdivision
20:                 end if
21:             end for
22:         end for
23:     end for
24:     return E, $\Delta$, $continue$
25: end function
```

Because the main operations of the algorithm, such as the construction of the NCH, are executed independently for each projection, the implementation of the method (training and classification phases) has been parallelized through the use of multi-threaded libraries.

5. Results

In this section, several experiments are presented to show the behavior of the proposed algorithm in real scenarios. In this study we tested the proposed method against the Non-Convex Boundary over Projections (NCBoP) method [21], a similar method based on the use of random projections, as well as against other known machine learning methods for anomaly detection: Autoencoder (AE) [15], Isolation Forest (IF) [16], Local Outlier Factor (LOF) [11], One-Class Support Vector Machine (OCSVM) [19], Robust Covariance (RC) [12] and Support Vector Data Description (SVDD) [31].

These algorithms have been evaluated over 10 real data sets available in the UCI Machine Learning Repository [32] and in the ODDS benchmark [33]. The characteristics of these data sets are summarized in Table 1. The data have been normalized using standard scalers with zero mean and unit variance. To assess the performance of each algorithm, the data has been divided into training and test sets using a 10-fold cross validation. All algorithms have been trained using only normal data, while the test phase included data from both classes. The

Algorithm 5 : Function used during training to expand a NCH

```
Inputs:
    NCH                    ▷ List of NCHs, output from $computeNCH$
    $\lambda \in \mathbb{R}^+$            ▷ Expansion parameter
Output:
    ENCH                   ▷ Expanded NCHs
1:  function EXPANDNCH
2:      $ENCH = \emptyset$                ▷ List to store the final ENCHs
3:      for c in NCH do                ▷ For each NCH
4:          for $i = 1 \ldots card(\mathbf{c}.V)$ do     ▷ For each vertex of the NCH
5:              $ENCH\_aux = \emptyset$      ▷ Auxiliary list to store one ENCH
6:              Get vertices $\mathbf{v}_i$, $\mathbf{v}_{i-1}$ and $\mathbf{v}_{i+1}$ of $\mathbf{c}.V$
7:              Calculate the incenter $\mathbf{c}_i$ for the vertex $\mathbf{v}_i$ using Eq. (2)
8:              Calculate the interior angle ($\alpha$) for the vertex $\mathbf{v}_i$
9:              if $\alpha > 180°$ then
10:                 sign = -1             ▷ Non-convex angle
11:             else
12:                 sign = 1              ▷ Convex angle
13:             end if
14:             $\mathbf{v}_{expanded} = \mathbf{v}_i + sign * \lambda * (\mathbf{v}_i - \mathbf{c}_i)/||\mathbf{v}_i - \mathbf{c}_i||$
15:             $ENCH\_aux = ENCH\_aux \cup (\mathbf{v}_{expanded})$
16:         end for
17:         Add ENCH_aux to the ENCH list
18:     end for
19:     return ENCH
20: end function
```

Algorithm 6 : OCENCH classification phase

```
Input:
    $\mathbf{x} \in \mathbb{R}^d$                 ▷ Datum to be classified
    M                      ▷ Trained model
Output:
    $Result \in \{NORMAL, ANOMALY\}$
1:  function CLASSIFY
2:      $Result = NORMAL$
3:      for $t = 1 \ldots length(M.P)$ do        ▷ For each projection
4:          $\bar{\mathbf{x}}_t = \mathbf{P}_t\mathbf{x}$                ▷ Project the new point
5:          $\mathbf{n}_t = normalize(\bar{\mathbf{x}}_t, normalizer_t)$   ▷ Normalize the data
6:          if not $isIn(\mathbf{n}_t, ENCH_t).any$ then
7:              $Result = ANOMALY$        ▷ The point is an anomaly
8:              Break
9:          end if
10:     end for
11:     return Result
12: end function
```

choice of the parameters of the different methods has been carried out using grid search, selecting those configurations that produce the best results. The combinations of parameters that reported the best results are available in Appendix.

The metric used to measure the performance of the algorithms was the F1-score that relates precision and recall. In order to obtain more reliable results the experiments were repeated twice. Table 2 summarizes the mean test results. The chosen statistical test was Nemenyi, a non-parametric test which makes a pairwise comparison between models [34,35]. Using a significance level of 5% ($\alpha = 0.05$) and the F1-scores obtained for each data set independently, the best values in Table 2 have been highlighted in bold. As can be seen, the OCENCH algorithm presents a very robust behavior, achieving good performance for most data sets. In five of the ten data sets, the subdivision of the NCH has been beneficial, increasing the performance obtained by the method (Annthyroid, Shuttle and Optdigit) or achieving similar results in a significantly shorter time (Telescope and Miniboone).

Table 1
Characteristics of the data sets used.

Dataset	Size	Anomalies	Dimension
Annthyroid	7200	534 (7.4%)	6
Thyroid	3772	93 (2.5%)	6
Shuttle	49 097	3511 (7.2%)	9
Telescope	19 020	6573 (34.5%)	10
Pendigits	6870	156 (2.3%)	16
Cardio	1831	176 (9.6%)	21
Ionosphere	351	126 (35.9%)	33
Miniboone	130 065	36 499 (28.1%)	50
Optdigit	5216	64 (2.9%)	64
MNIST	7603	700 (9.2%)	100

To validate this statement, a statistical test was carried out to compare the global performance of the algorithms. The chosen test was again Nemenyi. Using a significance level of 5% and the F1-scores of the algorithms for the different data sets, OCENCH ranks within the group of best methods, represented graphically by Fig. 10. As can be seen, the null hypothesis of algorithms having similar performance is accepted for OCENCH, LOF, OCSVM, RC, IF, NCBoP and AE, so we can affirm that in these tests our method was found among the best models. In addition, we can affirm than OCENCH performance was significantly better than SVDD.

Table 3 shows the ranking obtained by each algorithm for each data set according to average test F1-score. To calculate this ranking, when two algorithms obtained the same average F1-score, they were assigned the same ranking. For example, in Ionosphere, OCENCH and AE rank 3rd and 4th based on their F1-score. Since their F1-score has been the same (85.9), they were both assigned the same position in the table (3.5). The final average ranking is shown in the last row. As can be observed, OCENCH was the algorithm that obtained the best average ranking (2.8) with a considerable difference between the average ranking of the first three methods (OCENCH, LOF and OCSVM) and the rest.

Table 4 shows the mean training time of each algorithm (lower values than 0.05 have been represented as 0.0). As can be seen, the training time of the methods that use random projections such as OCENCH or NCBoP is higher than that of most algorithms, however they are still reasonable times for their use. Test times have not been included in this work because they are very low for all the algorithms.

6. Ablation study

In order to understand the impact of the different parameters of the method on its performance, an ablation study has been carried out. As has been seen, the OCENCH method has four parameters to adapt the algorithm to the scenario on which it is going to be used:

- Number of projections (τ): number of random projections used to represent the target data set. A very low number of projections may imply a too simple representation, while a very high number carries a higher computational cost that in some cases may not be worth it. The goal is to use a sufficient number of projections to represent the class accurately while maintaining a feasible computational cost.
- Subdivision (boolean value): by default it is recommended to allow the subdivision of closures to represent the data set. If a priori it is known that the data set does not present disjointed regions, this subdivision process could be avoided to reduce the computational cost of the algorithm. In any case, if subdivision is allowed, it will be carried out by OCENCH automatically only when necessary.
- Maximum edge size (l): value that regulates the edge pruning process and, if it is enabled, the subdivision process. Since the projected data is normalized in each 2-D subspace independently,

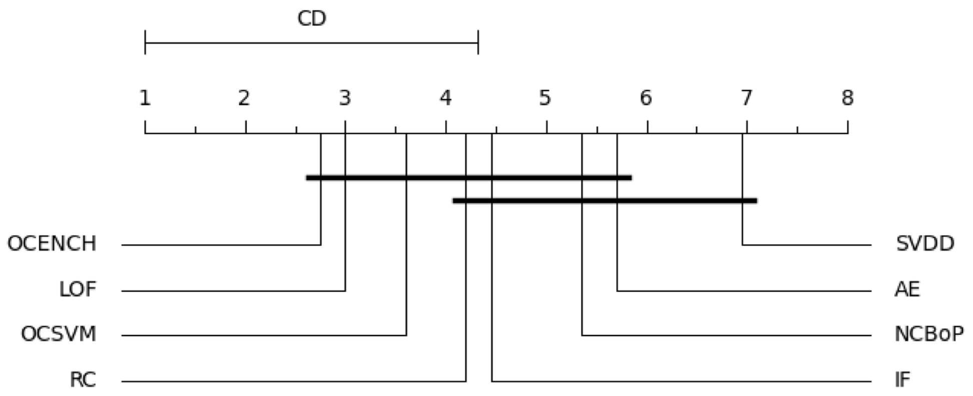

Fig. 10. Graphical representation of Nemenyi test with $\alpha = 0.05$. The critical distance (CD) obtained was 3.32.

Table 2
Average test F1-score ± standard deviation for the different data sets.

Dataset	OCENCH	NCBoP	AE	IF	LOF	OCSVM	RC	SVDD
Annthyroid	**77.5 ± 0.9**	51.0 ± 1.7	**78.9 ± 2.6**	**84.2 ± 1.7**	72.4 ± 0.7	66.8 ± 0.4	**85.9 ± 0.5**	63.5 ± 4.4
Thyroid	**93.4 ± 1.1**	89.9 ± 1.0	87.5 ± 1.7	**94.9 ± 1.4**	**92.7 ± 1.4**	**92.4 ± 1.4**	**95.0 ± 1.1**	83.9 ± 3.3
Shuttle	**99.2 ± 0.2**	97.5 ± 0.1	86.3 ± 0.4	**98.1 ± 0.2**	95.2 ± 0.6	96.9 ± 0.4	96.3 ± 0.3	85.0 ± 10.7
Telescope	**70.1 ± 0.9**	**69.7 ± 0.7**	66.7 ± 1.1	63.5 ± 2.7	**71.0 ± 0.4**	66.6 ± 0.4	**70.7 ± 0.4**	66.8 ± 0.1
Pendigits	**93.73 ± 1.1**	**92.9 ± 1.1**	82.4 ± 1.3	**93.6 ± 1.4**	**95.2 ± 0.9**	**94.2 ± 0.7**	84.1 ± 1.3	**96.4 ± 0.6**
Cardio	**94.3 ± 1.4**	**92.0 ± 1.9**	86.7 ± 1.5	**90.7 ± 1.1**	89.7 ± 1.6	**92.3 ± 0.7**	79.5 ± 1.7	80.5 ± 3.1
Ionosphere	**85.9 ± 2.6**	**84.3 ± 4.3**	**85.9 ± 3.5**	82.8 ± 3.5	**85.2 ± 3.6**	**87.9 ± 2.4**	80.1 ± 2.7	**89.8 ± 2.3**
Miniboone	**76.6 ± 0.5**	49.9 ± 2.3	**71.9 ± 0.3**	64.0 ± 1.9	**72.2 ± 0.6**	67.9 ± 0.2	68.5 ± 0.3	69.6 ± 1.8
Optdigit	**84.3 ± 1.4**	74.7 ± 1.9	71.5 ± 0.6	72.2 ± 6.4	**95.8 ± 0.8**	**85.2 ± 0.6**	75.7 ± 1.6	66.7 ± 0.2
MNIST	80.2 ± 0.9	73.7 ± 1.6	72.5 ± 0.3	78.0 ± 1.1	**88.9 ± 0.4**	**93.7 ± 0.6**	**82.7 ± 2.2**	73.7 ± 5.4

Table 3
Ranking of the algorithms for the different data sets.

Dataset	OCENCH	NCBoP	AE	IF	LOF	OCSVM	RC	SVDD
Annthyroid	4	8	3	2	5	6	1	7
Thyroid	3	6	7	2	4	5	1	8
Shuttle	1	3	7	2	6	4	5	8
Telescope	3	4	6	8	1	7	2	5
Pendigits	4	6	8	5	2	3	7	1
Cardio	1	3	6	4	5	2	8	7
Ionosphere	3.5	6	3.5	6	5	2	8	1
Miniboone	1	8	3	7	2	6	5	4
Optdigit	3	5	7.5	6	1	2	4	7.5
MNIST	4	6.5	8	5	2	1	3	6.5
Avg. Rank	**2.8**	5.6	5.9	4.7	3.3	3.8	4.4	5.5

it is only necessary to define a single value for this parameter that will be used universally in all these subspaces. A too small value will cause a very extreme fit to the shape of the data, which can cause an excessive number of subdivisions with the consequent computational cost, while a too large value could cause a poorly adjusted representation, making it difficult to subdivide the closures. The objective is to use a maximum edge size that allows the class to be represented accurately while maintaining a feasible computational cost.

- Expansion parameter (λ): to avoid the effect of over-adjustment of the training data for the anomaly detection problem, it is possible to expand the final closures in a controlled way. This process is computationally inexpensive and in certain scenarios it can be beneficial, such as in cases where there are not a large number of instances of the class to be modeled.

To show the influence of the parameter selection on the performance of the OCENCH algorithm (F1-score), several experiments have been carried out using the data sets tested in this work and the best parameter combinations for each of them Appendix. These experiments are aimed to study the influence of each of the parameters on the performance of the algorithm when the other three parameters take a fixed value. The parameters that have been studied are the number of projections, the maximum edge size l and the λ expansion factor. The boolean parameter Subdivision is always on, it is not modified.

Since the data are normalized, to study the behavior of the parameter l, it has taken values between 0.1 and 1.7, with increments of 0.1. As can be seen in Fig. 11, the range of values with which the best results are obtained for these 10 scenarios is between 0.1 and 1.1. However, there is a particular subrange of values of l for each data set in which the best performance of the algorithm is obtained, for example $l \in [0.3, 1.1]$ in the case of Thyroid. Using a grid search with the proposed generic values is an easy way to estimate the best operating range for this parameter that controls the level of adjustment of the closures and their subdivision.

To study the influence of the number of projections on the performance of the algorithm, between 5 and 2000 projections have been used. Fig. 12, whose horizontal axis is on a logarithmic scale, shows the results obtained. As can be seen, the number of projections required to characterize the data set varies considerably depending on the characteristics of each scenario (Table 1). In any case, as with the l parameter, there is a subrange of values for each data set in which the best performance of the algorithm is obtained. Using an initial grid search with a number of projections in the interval [5, 2000] is an easy way to estimate the best operating range for this parameter.

To study the influence of the expansion factor on the performance of the algorithm, values between 0.0001 (0.01%) and 2 (200%) have been used. Fig. 13, whose horizontal axis is on a logarithmic scale, shows the results obtained. In all but three of the data sets (Pendigits, Miniboone and Annthyroid), the expansion has improved the performance of the algorithm compared to using the original NCHs. Using an initial grid search with values in the interval [0, 2] is an easy way to estimate if expansion is beneficial to algorithm performance and if so, determine the best value for λ.

7. Conclusion

The developed OCENCH algorithm allows working with non-convex and disjointed data sets in a novel way, offering a robust behavior in the

Table 4
Average training time ± standard deviation for the different data sets.

Dataset	OCENCH	NCBoP	AE	IF	LOF	OCSVM	RC	SVDD
Annthyroid	329.5 ± 9.1	622.9 ± 14.7	2.6 ± 0.7	0.2 ± 0.0	0.1 ± 0.0	0.5 ± 0.0	1.1 ± 0.2	50.7 ± 1.3
Thyroid	1.7 ± 0.1	16.8 ± 2.7	1.1 ± 0.1	0.4 ± 0.0	0.0 ± 0.0	0.1 ± 0.0	0.8 ± 0.1	8.9 ± 1.6
Shuttle	7.6 ± 1.4	0.7 ± 0.0	1.5 ± 0.8	0.6 ± 0.0	0.7 ± 0.0	0.2 ± 0.0	3.5 ± 0.7	125.8 ± 9.6
Telescope	75.8 ± 3.2	193.2 ± 8.7	1.51 ± 0.2	0.0 ± 0.0	0.2 ± 0.0	0.5 ± 0.0	3.2 ± 0.9	264.8 ± 43.9
Pendigits	0.9 ± 0.0	0.5 ± 0.0	2.6 ± 0.6	0.5 ± 0.0	0.3 ± 0.0	0.2 ± 0.0	1.8 ± 0.0	53.5 ± 3.2
Cardio	0.5 ± 0.0	0.8 ± 0.0	1.1 ± 0.1	0.3 ± 0.0	0.1 ± 0.0	0.0 ± 0.0	0.6 ± 0.1	1.2 ± 0.1
Ionosphere	0.1 ± 0.0	0.0 ± 0.0	5.8 ± 1.9	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.1 ± 0.0	0.0 ± 0.0
Miniboone	139.4 ± 4.2	494.2 ± 19.1	3.4 ± 0.5	0.7 ± 0.0	1.4 ± 0.0	55.1 ± 4.2	17.2 ± 2.6	76.1 ± 3.6
Optdigit	11.6 ± 2.7	252.7 ± 9.8	2.1 ± 0.4	0.2 ± 0.0	0.4 ± 0.0	0.4 ± 0.0	2.7 ± 0.0	13.6 ± 0.9
MNIST	1.7 ± 0.0	133.3 ± 1.2	10.5 ± 2.3	1.4 ± 0.0	0.7 ± 0.0	0.2 ± 0.0	9.1 ± 0.2	30.0 ± 1.4
Avg. Rank	5.2	5.3	5.4	1.6	1.2	1.7	4.4	6.5

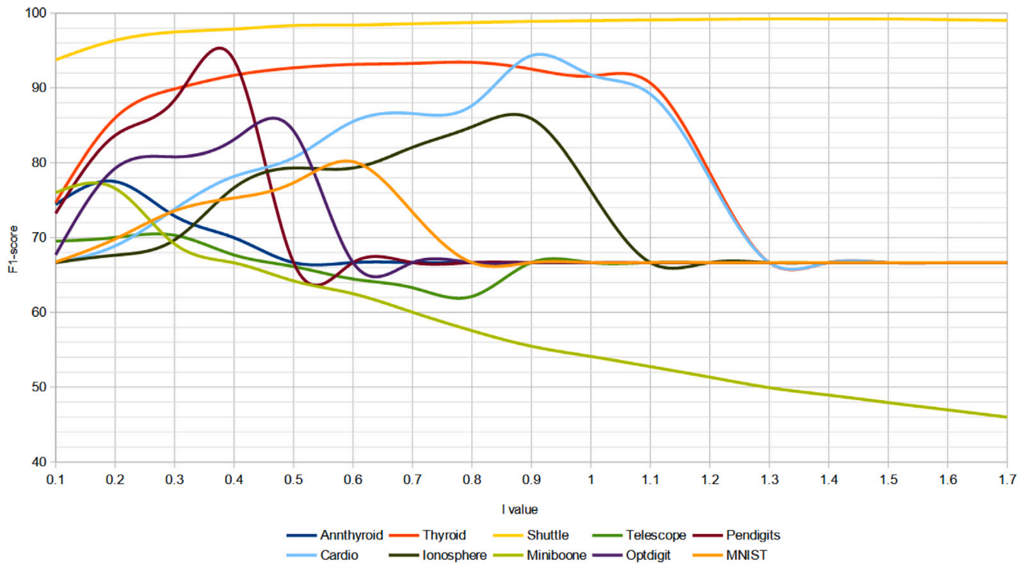

Fig. 11. Influence of l on the F1-score when the number of projections and the λ expansion factor are fixed.

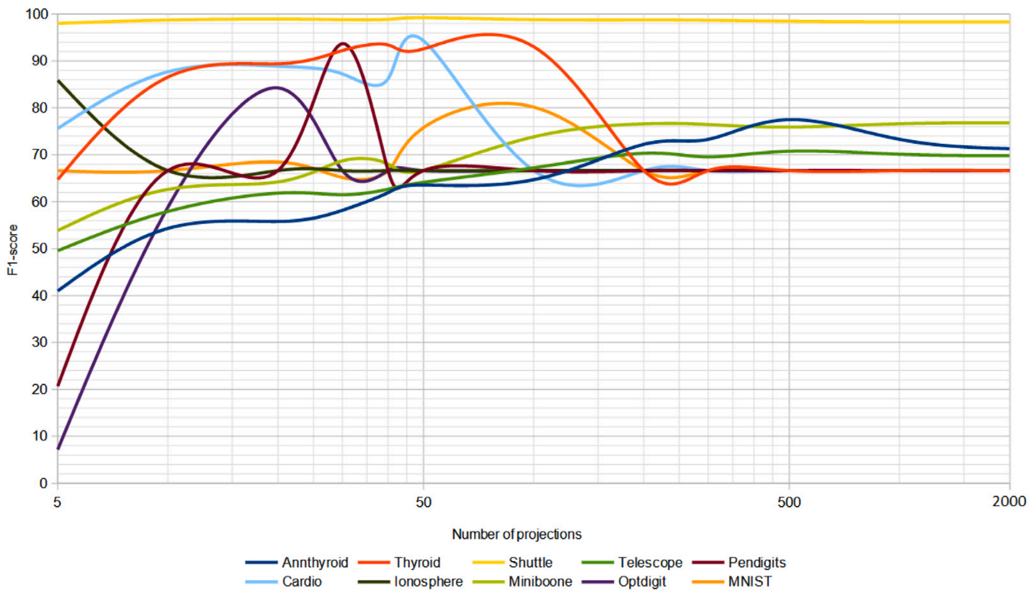

Fig. 12. Influence of the number of projections on the F1-score when l and the λ expansion factor are fixed.

different tests carried out. Its performance is remarkable, positioning itself as an alternative for both convex and non-convex problems. In addition, OCENCH is easily configurable through its parameters, it only needs normal data to be trained and is not necessary to know a priori the percentage of anomalies in the data set. The models generated with this algorithm are light on memory requirements, since it is not

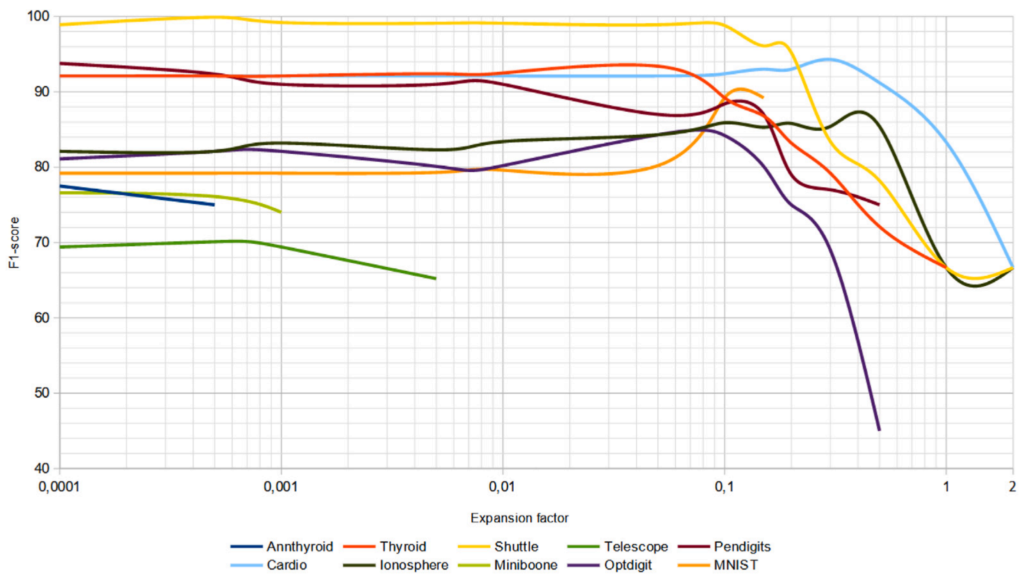

Fig. 13. Influence of the expansion factor on the F1-score when the number of projections and l are fixed. The lines are drawn up to the maximum possible value without generating complex polygons.

necessary to store all the data used for the creation of the non-convex hulls. The model only needs the information of the vertices of each non-convex hull to determine if a new data is inside or outside the normal class. Finally, the algorithm has been implemented using specialized libraries so that it can be executed in parallel. The calculations of each projection are carried out independently, so its parallelization allows drastically reducing the execution time of the training and classification phases.

As future work, it would be interesting to develop and test a version of the algorithm that could detect the existence of empty regions inside dense regions to be able to fit more precisely in these scenarios, such as rings or concentric rings. Furthermore, it could also be interesting to test the algorithm in edge computing environments.

CRediT authorship contribution statement

David Novoa-Paradela: Conceptualization, Methodology, Investigation, Software, Writing. **Oscar Fontenla-Romero:** Conceptualization, Methodology, Investigation, Software, Supervision, Writing. **Bertha Guijarro-Berdiñas:** Conceptualization, Methodology, Investigation, Supervision, Writing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been supported by the grant *Machine Learning on the Edge - Ayudas Fundación BBVA a Equipos de Investigación Científica 2019*; the National Plan for Scientific and Technical Research and Innovation of the Spanish Government (Grant PID2019-109238GB-C2); and by the Xunta de Galicia, Spain (Grant ED431C 2018/34) with the European Union ERDF funds. CITIC, as a Research Center of the University System of Galicia, is funded by Consellería de Educación, Universidade e Formación Profesional of the Xunta de Galicia, Spain through the European Regional Development Fund (ERDF) and the Secretaría Xeral de Universidades (Ref. ED431G 2019/01). Funding for open access charge: Universidade da Coruña/CISUG.

Appendix. Parameters used during training

This appendix contains the values of the parameters used in the grid search and those finally chosen as the best for each method and data set in the experiments carried out in this study (Tables 1–3). The combinations of parameters that obtained the best results are listed below in Table A.5.

- OCENCH
  - Number of projections (τ).
  - Maximum edge size (l).
  - Expansion parameter (λ).
  - Subdivision (boolean value)
- Non-Convex Boundary over Projections (NCBoP) [21].
  - Number of projections.
- Autoencoder (AE) [15].
  - Network architecture.
  - Epochs.
  - Contamination of the data set (c).
- Isolation Forest (IF) [16].
  - The number of base estimators in the ensemble.
  - Contamination of the data set (c).
- Local Outlier Factor (LOF) [11].
  - Number of neighbors.
  - Contamination of the data set (c).
- One-Class Support Vector Machine (OCSVM) [19].
  - An upper bound on the fraction of training errors and a lower bound of the fraction of support vectors (ν).
  - Kernel type: Linear, Polynomial or RBF.
  - Kernel coefficient γ (in the case of polynomial and RBF kernels).
  - Degree (in the case of polynomial kernel).
- Robust Covariance (RC) [12].

Table A.5
Parameters used for training.

Dataset	OCENCH	NCBoP	AE	IF	LOF	OCSVM	RC	SVDD
Annthyroid	Proj: 500, l: 0.2, Extend: 0, Subdivision: True	Proj: 10000	Layers: [10, 10, 8, 10, 10], Epochs: 100, c: 0.4	Estimators: 100, c: 0.2	Neighbors: 15, c: 0.4	ν: 0.2, γ: 0.17, Kernel: Polynomial, Degree: 5	c: 0.2	Pos.: 0.1, Neg.: 0.1, Kernel: Gaussian, Width: 8
Thyroid	Proj: 200, l: 1, Extend: 0.05, Subdivision: False	Proj: 500	Layers: [5, 2, 5], Epochs: 100 , c: 0.3	Estimators: 200, c: 0.05	Neighbors: 15, c: 0.1	ν: 0.5, γ: 8, Kernel: RBF	c: 0.05	Pos.: 0.1, Neg.: 0.1, Kernel: Gaussian, Width: 8
Shuttle	Proj: 50, l: 1.5, Extend: 0.001, Subdivision: True	Proj: 50	Layers: [6, 4, 6], Epochs: 100, c: 0.3	Estimators: 200, c: 0.01	Neighbors: 100, c: 0.1	ν: 0.05, γ: 2^{-4}, Kernel: RBF	c: 0.05	Pos.: 0.1, Neg.: 0.1, Kernel: Gaussian Width: 8
Telescope	Proj: 200, l: 0.3, Extend: 0.0005, Subdivision: True	Proj: 2000	Layers: [10, 5, 10], Epochs: 100, c: 1	Estimators: 5, c: 0.2	Neighbors: 15, c: 0.2	ν: 0.1, γ: 4, Kernel: poly, Degree: 5	c: 0.6	Pos.: 0.1, Neg.: 0.1, Kernel: Gaussian, Width: 8
Pendigits	Proj: 100, l: 20, Extend: 0, Subdivision: False	Proj: 50	Layers: [10, 8, 6, 8, 10] Epochs: 100, c: 0.4	Estimators: 200, c: 0.1	Neighbors: 15, c: 0.1	ν: 0.05, γ: 1, Kernel: RBF	c: 0.35	Pos.: 0.05, Neg.: 0.05, Kernel: Gaussian, Width: 4
Cardio	Proj: 200, l: 20, Extend: 0.3, Subdivision: False	Proj: 50	Layers: [5, 2, 5], Epochs: 100 c: 0.3	Estimators: 200, c: 0.2	Neighbors: 100, c: 0.2	ν: 0.05, γ: 0.06, Kernel: RBF	c: 0.35	Pos.: 0.1, Neg.: 0.1, Kernel: Gaussian Width: 2^{-4}
Ionosphere	Proj: 5, l: 0.9, Extend: 0.1, Subdivision: False	Proj: 10	Layers: [10, 5, 10], Epochs: 100, c: 0.3	Estimators: 15, c: 0.2	Neighbors: 15, c: 0.35	ν: 0.05, γ: 2^{-3}, Kernel: RBF	c: 0.2	Pos.: 0.05, Neg.: 0.05, Kernel: Gaussian Width: 1
Miniboone	Proj: 200, l: 0.2, Extend: 0, Subdivision: True	Proj: 5000	Layers: [10, 5, 10], Epochs: 100, c: 0.3	Estimators: 100, c: 0.2	Neighbors: 100, c: 0.5	ν: 0.05, γ: 2^{-10}, Kernel: RBF	c: 0.8	Pos.: 0.1, Neg.: 0.1, Kernel: Linear
Optdigit	Proj: 20, l: 0.5, Extend: 0.05, Subdivision: True	Proj: 5000	Layers: [10, 7, 5, 7, 10], Epochs: 100, c: 1	Estimators: 50, c:0.2	Neighbors: 50, c: 0.1	ν: 0.1, γ: 2^{-9}, Kernel: RBF	c: 0.5	Pos.: 0.1, Neg.: 0.1, Kernel: Linear
MNIST	Proj: 200, l: 1.6, Extend: 0.05, Subdivision: False	Proj: 2000	Layers: [10, 7, 5, 7, 10], Epochs: 100, c: 1	Estimators: 200, c: 0.2	Neighbors: 15, c: 0.1	ν: 0.1, γ: 2^{-4}, Kernel: RBF	c: 0.2	Pos.: 0.4, Neg.: 0.4, Kernel: Linear

  - Contamination of the data set (c).

- Support Vector Data Description (SVDD) [31].
  - Positive penalty.
  - Negative penalty.
  - Kernel type: Linear, Polynomial or Gaussian.
  - Width (in the case of gaussian kernel).
  - Degree (in the case of polynomial kernel).

References

[1] S. Khan, T. Yairi, A review on the application of deep learning in system health management, Mech. Syst. Signal Process. 107 (2018) 241–265.

[2] S. Sorournejad, Z. Zojaji, R.E. Atani, A.H. Monadjemi, A survey of credit card fraud detection techniques: data and technique oriented perspective, 2016, arXiv:1611.06439.

[3] G. Vigna, C. Kruegel, Host-based intrusion detection, 2005.

[4] D. Ramotsoela, A. Abu-Mahfouz, G. Hancke, A survey of anomaly detection in industrial wireless sensor networks with critical water system infrastructure as a case study, Sensors 18 (8) (2018).

[5] S. Vempala, The Random Projection Method, American Mathematical Society, 2004.

[6] M. Duckham, L. Kulik, M. Worboys, A. Galton, Efficient generation of simple polygons for characterizing the shape of a set of points in the plane, Pattern Recognit. 41 (10) (2008) 3224–3236.

[7] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: a survey, ACM Comput. Surv. 41 (3) (2009) 15:1–15:58.

[8] S.S. Khan, M.G. Madden, One-class classification: Taxonomy of study and review of techniques, Knowl. Eng. Rev. (2013) abs/1312.0049. arXiv:1312.0049.

[9] G. Thatte, U. Mitra, J. Heidemann, Parametric methods for anomaly detection in aggregate traffic (Extended version), IEEE/ACM Trans. Netw. 19 (2011).

[10] Dit-Yan Yeung, C. Chow, Parzen-window network intrusion detectors, in: Object Recognition Supported By User Interaction for Service Robots, Vol. 4, 2002, pp. 385–388, vol.4.

[11] M. Breunig, H.-P. Kriegel, R.T. Ng, J. Sander, LOF: identifying density-based local outliers, in: ACM Sigmoid International Conference on Management of Data, ACM SIGMOD Record, 2000, pp. 93–104.

[12] D. Peña, F.J. Prieto, Multivariate outlier detection and robust covariance matrix estimation, Technometrics 43 (3) (2001) 286–310.

[13] P. Soucy, G.W. Mineau, A simple KNN algorithm for text categorization, in: IEEE International Conference on Data Mining, 2001, pp. 647–648.

[14] K. Khan, S.U. Rehman, K. Aziz, S. Fong, S. Sarasvady, DBSCAN: past, present and future, in: 5th IEEE International Conference on the Applications of Digital Information and Web Technologies, 2014, pp. 232–238.

[15] M. Ma, C. Sun, X. Chen, Deep coupling autoencoder for fault diagnosis with multimodal sensory data, IEEE Trans. Ind. Inf. 14 (2018) 1137–1145.

[16] F.T. Liu, K.M. Ting, Z. Zhou, Isolation forest, in: 2008 Eighth IEEE International Conference on Data Mining, 2008, pp. 413–422.

[17] E. Keogh, J. Lin, A. Fu, HOT SAX: efficiently finding the most unusual time series subsequence, in: 5th IEEE International Conference on Data Mining (ICDM'05), 2005, pp. 1–8.

[18] N. Cristianini, J. Shawe-Taylor, An introduction to support vector machines: and other Kernel-based learning methods, Cambridge University Press, New York, NY, USA, 2000.

[19] G. Ratsch, S. Mika, B. Scholkopf, K. Muller, Constructing boosting algorithms from SVMs: an application to one-class classification, IEEE Trans. Pattern Anal. Mach. Intell. 24 (9) (2002) 1184–1199.

[20] P. Casale, O. Pujol, P. Radeva, Approximate polytope ensemble for one-class classification, Pattern Recognit. 47 (2) (2014) 854–864.

[21] E. Jove, J.-L. Casteleiro-Roca, H. Quintián, J.-A. Méndez-Pérez, J.L. Calvo-Rolle, A new method for anomaly detection based on non-convex boundaries with random two-dimensional projections, Inf. Fusion 65 (2021) 50–57.

[22] P. Li, O. Niggemann, Non-convex hull based anomaly detection in CPPS, Eng. Appl. Artif. Intell. 87 (2020) 103301.

[23] J.-S. Park, S. Oh, A new concave hull algorithm and concaveness measure for n-dimensional datasets, J. Inf. Sci. Eng. 29 (2012) 379–392.

[24] H. Hotelling, Analysis of a complex of statistical variables into principal components., J. Educ. Psychol. 24 (6) (1933) 417–441.

[25] S. Ioffe, Probabilistic linear discriminant analysis, in: A. Leonardis, H. Bischof, A. Pinz (Eds.), Computer Vision - European Conference on Computer Vision 2006, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 531–542.

[26] W.B. Johnson, J. Lindenstrauss, Extensions of Lipschitz mappings into a Hilbert space, Contemp. Math. 26 (1984) 189–206.

[27] C.B. Barber, D.P. Dobkin, D.P. Dobkin, H. Huhdanpaa, The quickhull algorithm for convex hulls, ACM Trans. Math. Software 22 (4) (1996) 469–483.

[28] S. Dasgupta, A. Gupta, An elementary proof of a theorem of Johnson and Lindenstrauss, Random Struct. Algorithms 22 (1) (2003) 60–65.

[29] S. Dasgupta, Experiments with random projection, in: Conference on Uncertainty in Artificial Intelligence, 2013, abs/1301.3849. arXiv:1301.3849.

[30] E. Bingham, H. Mannila, Random projection in dimensionality reduction: applications to image and text data, in: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '01, Association for Computing Machinery, New York, NY, USA, 2001, pp. 245–250.

[31] D.M. Tax, R.P. Duin, Support vector data description, Mach. Learn. 54 (2004) 45–66.

[32] A. Asuncion, D. Newman, Uci machine learning repository, 2007, https://archive.ics.uci.edu/ml/index.php.

[33] S. Rayana, Odds library, 2016, http://odds.cs.stonybrook.edu/, Revised at 23/03/2021.

[34] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (1) (2006) 1–30.

[35] S. García, F. Herrera, An extension on "Statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons, J. Mach. Learn. Res. 9 (89) (2008) 2677–2694.