Knowledge-Based Systems 228 (2021) 107290

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Dynamic selection and combination of one-class classifiers for multi-class classification

Rogério C.P. Fragoso [a,*], George D.C. Cavalcanti [a], Roberto H.W. Pinheiro [b], Luiz S. Oliveira [c]

[a] Centro de Informática, Universidade Federal de Pernambuco, Recife, PE, Brazil
[b] Universidade Federal do Cariri, Juazeiro do Norte, CE, Brazil
[c] Departamento de Informática, Universidade Federal do Paraná, Curitiba, PR, Brazil

ARTICLE INFO

Article history:
Received 5 April 2020
Received in revised form 14 May 2021
Accepted 5 July 2021
Available online 7 July 2021

Keywords:
One-class classification
One-class decomposition
Multiple classifier system
Dynamic ensemble selection

ABSTRACT

A natural solution to tackle multi-class problems is employing multi-class classifiers. However, in specific situations, such as imbalanced data or high number of classes, it is more effective to decompose the multi-class problem into several and easier to solve problems. One-class decomposition is an alternative, where one-class classifiers (OCCs) are trained for each class separately. However, fitting the data optimally is a challenge for OCCs, especially when it presents a complex intra-class distribution. The literature shows that multiple classifier systems are inherently robust in such cases. Thus, the adoption of multiple OCCs for each class can lead to an improvement for one-class decomposition. With that in mind, in this work we introduce the method called One-class Classifier Dynamic Ensemble Selection for Multi-class problems (MODES, for short), which provides competent classifiers for each region of the feature space by decomposing the original multi-class problem into multiple one-class problems. So, each class is segmented using a set of cluster validity indices, and an OCC is trained for each cluster. The rationale is to reduce the complexity of the classification task by defining a region of the feature space where the classifier is supposed to be an expert. The classification of a test example is performed by dynamically selecting an ensemble of competent OCCs and the final decision is given by the reconstruction of the original multi-class problem. Experiments carried out with 25 databases, 4 OCC models, and 3 aggregation methods showed that the proposed architecture outperforms the literature. When compared with the state-of-the-art, MODES obtained better results, especially for databases with complex decision regions.

1. Introduction

Multi-class classification may be tackled in several ways. The most common approach is to use multi-class classifiers, where the objective is to separate the classes of the problem. However, some difficulties embedded in the nature of the data, for example, class imbalance, complex data distribution, class overlap, high number of classes and small data samples, may impair the performance if not carefully treated.

Deep learning [1], which is currently one of the most remarkable machine learning techniques, show good performance in a variety of multi-class problems and, in general, does not require any special procedure to deal with the aforementioned issues. However, the amount of data required for training a satisfactory model depends on the complexity of the problem, i.e., the more complex is the problem, the more data is needed. Databases presenting complex data distribution, class overlap, and high number of classes, for example, require more data [2]. Furthermore, class imbalance is still an important issue regarding deep learning methods, mainly for non-image databases [3]. Data sampling, i.e., oversampling and downsampling, may be used to diminish this issue. Oversampling techniques, such as Synthetic Minority Over-sampling TEchnique (SMOTE) [4], augments the size of the minority class, creating synthetic data points to balance the majority class. Downsampling techniques, perform data reduction on the majority class. In oversampling, the original distribution of the minority class is maintained and no data is discarded. However, since the majority class is ignored, synthetic data points may include noisy data [5] and/or may be generated over the majority class increasing class overlap [6]. On the other hand, Downsampling diminishes the computational effort for training, since less data is used, however the classification performance may be hindered due to information loss caused by the exclusion useful data. Thus, reducing the data may not be a good option if data is already scarce. Therefore, databases with such impairments may be a difficult to solve problem, even for a powerful technique such as deep learning.

* Corresponding author.
E-mail address: rcpf@cin.ufpe.br (R.C.P. Fragoso).

https://doi.org/10.1016/j.knosys.2021.107290
0950-7051/© 2021 Elsevier B.V. All rights reserved.

An alternative that is recently gaining attention is the one-class decomposition, where each class is treated as a separate problem [7,8]. The data from each class of the original multi-class problem is fed to a one-class classifier (OCC) and each OCC estimates the likelihood of a test example belongs to a target class. The final classification decision, i.e., which class the test example is assigned to, is given by the aggregation of the decisions made by the OCCs. The rationale is that, following the divide and conquer principle, each one-class problem is simpler to solve than the original multi-class problem.

It is important to remark that one-class classifiers are not superior to standard classifiers for most of the multi-class problems. OCCs do not access counter-examples in the training phase, instead, it uses the target data to define a decision border that describes the data and maintains the generalization power. In this case, inter-class information is lost. However, the nature of OCCs make of one-class decomposition an interesting approach for specific cases, where the data contain the aforementioned issues [7]. For example, because OCCs treat each class individually, class imbalance is not an issue. Furthermore, for the same reason, a database with class overlapping may be better described by one-class classifiers than with standard classifiers. Even fuzzy classification does not allow extremely overlapped decision regions [9]. One-class decomposition permits to generate overlapped decision borders in the training phase and the aggregation on the test phase is responsible for deciding which class a test example belongs to based on the proximity or importance to each class, for example. One-class decomposition is also beneficial for problems in which the number of classes may vary, such as incremental learning or open-set problems [10].

The most intuitive approach for one-class decomposition is to use an OCC for each class of the original problem and aggregate their outputs. However, a single one-class classifier, as well as a single standard classifier, hardly ever fits the data distribution optimally [11]. Recently, Multiple Classifiers Systems (MCS) have been adopted in the context of one-class problems aiming to address this issue. MCS may perform better and be more robust than a single classifier by combining the outputs of distinct classifiers [12,13]. For this reason, MCS are an increasingly adopted approach [14]. The most common way of working with MCS is to generate a pool of classifiers using the training data (generation step) and to combine the responses of these classifiers to give the final classification (aggregation step). The classifiers should be complementary, i.e., competent in different regions of the feature space, and individually accurate. Optionally, a selection step may be adopted to select the most competent classifiers instead of combining all of them. The selection may be static (a subset of the classifiers in the pool is selected to classify all test examples) or dynamic (a subset of classifiers is selected on the fly to classify each test example).

Recent studies using Multiple Classifiers Systems with one-class classifiers have shown promising results [8,10,15,16]. To the best of our knowledge, a single technique adopted dynamic ensemble selection for one-class decomposition [8]. However, this technique does not treat the intra-class complex data distribution, i.e., the presence of remote examples and multi-modal data, as a particular issue.

We propose an architecture for multi-class classification using one-class decomposition. The proposed architecture, namely One-class Classifier Dynamic Ensemble Selection for Multi-class problems (MODES, for short), decomposes the multi-class problem into one-class problems, segments the training data of each class using different numbers of clusters, and trains an OCC for each cluster. Since each OCC is trained with different subsets, we expect a high diversity among the OCCs in the generated pool, which is a desirable characteristic of an MCS. The proposed strategy aims at facilitating the classification task because each OCC deals only with part of the whole classification problem, which is the rationale behind the divide and conquer principle. Moreover, this data segmentation strategy is also interesting in dealing with complex data distribution (as shown in the problem statement section) because in such situations, locally specialized OCCs may fit better the data [10]. It is important to remark that the determination of the ideal number of clusters in a database is still an open problem [17], so, in this work, we use cluster validity indices [18] as an alternative to defining the number of clusters in each class. During the generalization phase, an ensemble (a subset of the whole pool of OCCs) is selected per test example. This ensemble should be composed of the most competent OCCs to classify the test example. Afterwards, these OCCs in the ensemble are combined to predict the class of the test example.

This research aims at improving the classification performance in one-class decomposition tackling the issue of complex intra-class data distribution using multiple one-class classifiers for each class. The main contributions of this work are: (1) an OCC decomposition architecture capable of dealing with complex intra-class data distribution; (2) a dynamic ensemble selection strategy that selects the most competent OCCs per test example; (3) a performance evaluation of state-of-the-art one-class decomposition techniques.

This text is organized as follows. Section 2 presents some problems faced by OCC when dealing with complex data distribution. Section 3 presents the background of one-class decomposition, multiple classifiers systems, dynamic ensemble selection and other techniques for a better understanding of the following sections. Section 4 details the proposed architecture, named MODES. Section 5 shows the experimental configurations and results obtained by the proposed architecture. The final remarks are presented in Section 6.

2. Problem statement

One-class classifiers (OCCs) try to define a closed boundary around the examples belonging to only one class in the training. In the test, the class used for training, called the target class, has to be distinguished from all other possible examples. The class of non-target examples is known as outlier class.

A single OCC may not be capable of capturing well the characteristics of the target class. For example, if the training data is multi-modal or contains remote examples, empty regions may appear within the decision boundary due to an over estimation in training. That is, in order to encompass the remote examples or the multiple modes present in the training data, the decision boundary may include regions that are not covered by any training example. In the test, both target and outlier examples may appear in the empty regions [8].

Even successful OCCs, such as Support Vector Data Descriptor (SVDD) suffer from this issue. SVDD defines a spherically shaped decision boundary around the target data [19]. However, a single sphere may not be able to best describe the data if there are some distinctive distributions in it [20]. Kernel functions help to optimize the decision region in such a way that most of the target examples are inside it while minimizing its size to diminish the probability of including outliers. The training data is mapped from the input space into a higher dimensional feature space which is easier to distinguish from other distributions. However, SVDD assumes that all training examples (in the target class) come from a single distribution, which is not true in many cases. Hence, in case of distinctive data distributions in the target class, outliers may be inside the decision region, as well, if only one spherically shaped decision boundary, i.e., a single SVDD, is used [21].

Consider a target class formed by more than one distribution, i.e., the data is multi-modal. If only one OCC is trained for the target class, two scenarios may occur: (a) the OCC will define a broad (overestimated) decision region to contemplate all target examples, or (b) the OCC will define a tight decision region to diminish the proportion of examples from the outlier class inside it. A broad decision region may lead to a high false positive rate while a reduced decision region may lead to a high false negative rate.

Fig. 1 shows a hypothetical database where the data distribution presents a complex form and the classes present multi-modal data. The points in red represent the target class and the points in green represent the outlier class. Scenarios (a) and (b), mentioned above, are depicted using a single Gaussian one-class classifier, for the sake of simplicity, however, the concepts may be extended for more complex OCCs.

In scenario (a), the decision region is wide enough to embrace all examples from the target class. Only examples from the target class are used in the training process, i.e., outliers are not seen in this step. Thus, a wide decision border may lead to a high false positive rate, since, in the test, a number of outliers may be located inside it, such as shown in Fig. 1(a) and be, therefore, erroneously labeled as target class.

In order to avoid that situation, one may configure a reduced decision border, allowing a percentage of the target training examples outside the decision region, as shown in scenario (b). In the example, besides not being able to achieve such objective, many examples from a specific distribution in the target class are not embraced by the decision region. Thus, examples from this distribution in may be erroneously labeled as outliers.

To avoid the problems explained in scenarios (a) and (b), a possible solution is to identify the different data distributions, i.e. modes, present in the target class and train an OCC for each mode. With this approach, the decision region for each OCC tends to be simpler than using the original data, leading to more precise classification.

Fig. 2 shows the decision regions for the same problem exposed in Fig. 1 using (a) one, (b) two, (c) four and (d) five Gaussian Data Descriptors. The scenario depicted in Fig. 2(a) is the same of Fig. 1(a). In Fig. 2(b), two OCCs are used. In this case, it can be noted that fewer outlier examples are in the decision regions of both OCCs than in Fig. 2(a). However, some target examples are not within the decision region of none of the OCCs. The scenario shown in Fig. 2(c) uses four OCCs. In this case, the target class is almost perfectly separated from the outlier class. Fig. 2(d) presents the scenario using five OCCs, where some target class examples are not embraced by none of the OCCs.

It is important to remark that using a high number of OCCs, i.e., more OCCs than the number of modes present in the data, may lead to inaccurate results, since training an OCC with data excessively segmented leads to a reduced decision region and high false negative rate. So an important aspect is to define the number of modes present in the data to avoid such impairments.

3. Background

This section presents the concepts about one-class classification and Multiple Classifiers Systems that are necessary for better understanding the rest of this paper. Additionally, we present important works related with Multiple Classifiers Systems in the context of one-class classification and one-class decomposition.

3.1. One-class classification

One-class classification is a kind of problem where it is difficult or expensive to obtain counter-examples, such as transaction fraud recognition, machine fault detection, anomaly detection, etc. Thus, only examples belonging to one class are available. Objects from this class are called target objects and all other objects are called outliers. An OCC tries to describe the set of objects from the target class and to identify which (new) objects resemble this training set [19].

One-class classifiers are especially categorized into density methods, boundary methods, and reconstruction methods [22]. Density methods compute the probability density function (PDF) of the target class data. In the test, the PDF value for the test example is compared with a threshold. This technique requires a large amount of data to overcome the curse of dimensionality [23]. Gaussian and Parzen OCCs are examples of density methods [23]. Boundary methods build a model by optimizing a closed boundary around the target data and the acceptance of a test example is given by the distance to the fitted model. Boundary methods require less training data than density methods. Among the boundary methods, SVDD [19], One-class Support Vector Machine (OCSVM) [24] and Nearest Neighbor [25] can be remarked. Reconstruction methods assume a model of data generation process and estimate the parameters of this model in the training. Self-organizing Maps and Learning Vector Quantization are classified as reconstruction methods [26].

One-class classifiers have been successfully applied to a variety of applications such as real-time activity error detection [27], text classification [28], authorship verification [29]. Recently, one-class decomposition has gained attention from researchers. This technique separates the training data by class and trains one-class classifiers for each class. Then, the classification results are combined. Krawczyk et al. [7] showed the usefulness of one-class decomposition in some specific cases, such as, complex data distribution, imbalanced data, presence of noise, high number of classes. For these cases, the research shows that using an OCC for each class of the original problem and aggregate them is more effective than using classical binary decomposition strategies. Other important application of one-class decomposition is when the number of classes is not known a priori, since one-class classifiers are trained for each class separately [10].

3.2. Multiple classifiers systems

The main rationale behind the use of Multiple Classifiers Systems (MCS) is that, following the no free lunch theorem [30], there is not a single classifier that is competent to deal with test examples from all regions of the input space. Thus, MCS combine a set of classifiers expecting to outperform any individual classifier.

An MCS is commonly divided into 3 steps: generation, selection and fusion/aggregation. In the generation step, a pool of classifiers is generated, seeking for accuracy and diversity among the classifiers. The purpose of the generation is to form sets of classifiers that complement each other, that is, to generate a pool that contains competent classifiers for different regions of the feature space. The most effective generation techniques train the base classifiers with different feature sets or different training sets or, yet, use different classifier models (heterogeneous pools) [14].

The goal of the selection step is to select from the pool of classifiers the most competent classifiers for the problem. Since this is an optional step, it is omitted in some techniques. It is possible to build an MCS skipping this step and performing a static combination of all classifiers in the pool. However, the advantages of applying the selection step are widely known [12,14].

The selection task can be static or dynamic [31]. In static selection, a subset of classifiers is formed in the training phase. Then, in the test phase, the selected ensemble is used to classify all the test examples. Dynamic selection is performed during the classification of test examples. For each test example x, a region of competence is computed, usually based on the k-Nearest

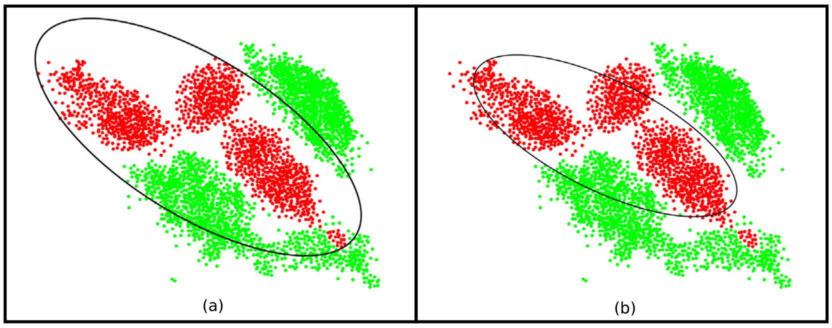

Fig. 1. Decision region of a single Gaussian OCC. Scenario (a) shows a broad decision region, leading to high false positive rate. Scenario (b) shows a reduced decision region leading to high false negative rate.

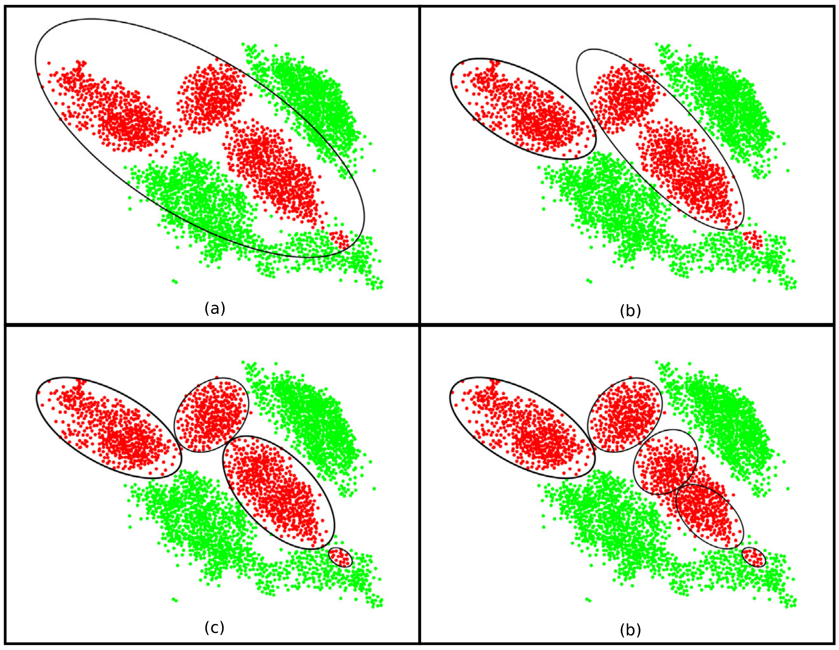

Fig. 2. Decision regions using (a) one, (b) two, (c) four and (d) five Gaussian one-class classifiers.

Neighbors of *x*. The best performing classifiers in the region of competence are selected to classify *x*. Dynamic selection can be divided into two approaches [14]. In Dynamic Classifier Selection (DCS), for each test example *x*, the most competent classifier is chosen. In Dynamic Ensemble Selection (DES), a set composed of the most competent classifiers in the pool is selected.

The fusion, or aggregation, step is responsible for combining the output of each individual classifier to provide the final output of the MCS.

3.3. Related works

Multiple Classifiers Systems have recently been adopted in the context of one-class problems. In [10], the authors used a clustering-based approach to generate ensembles of One-class Support Vector Machines. The data from the one-class problem is segmented and, for each cluster, an OCC is trained. This MCS does not include selection step, i.e., all the classifiers in the pool are used to classify all test examples. A classifier trained with examples from a specific region of the feature space may not be competent to classify test examples from other regions. Hence, the combination of all classifiers in the pool may hinder the performance. Dynamic selection techniques aim to mitigate this problem.

A DCS method for OCC was proposed in [15]. DCS techniques select a single classifier in execution time to classify each test example. This work showed that dynamic selection techniques work well in the context of OCC because during the training phase it was possible to generate a diverse pool of classifiers. However, in DCS, the performance depends on the quality of the algorithm that selects the classifier from the pool [32]. Since only one classifier is responsible for the classification of a test example, the performance may be impaired if this selected classifier is not competent for the classification of the current test example. Complex data distribution or the presence of noise may worsen this problem since the chances of selecting a non-competent classifier are higher.

This problem may be alleviated using DES, which selects a subset of classifiers from the pool to compose the ensemble responsible for the classification decision for a test example. To the best of our knowledge, the first DES for one-class decomposition was proposed by Krawczyk et al. [8].

In this work, an OCC is trained for each class of the original problem. To classify a test example, the method selects an ensemble composed of the OCCs trained with the data from the classes in the neighborhood of that test example. Then, the original multi-class problem is recomposed by aggregating the decisions of the selected one-class classifiers. Details of intra-class data

Table 1
Comparison involving MODES and other techniques that use one-class decomposition, OCC ensemble, for each one-class problem, and/or dynamic selection (DS).

Ref	Title	Decomp.	Ensemble	DS
[10]	Clustering-based ensembles for one-class classification	Yes	Yes	No
[7]	On the usefulness of one-class classifier ensembles for decomposition of multi-class problems	Yes	No	No
[15]	Dynamic classifier selection for one-class classification	No	No	DCS
[8]	Dynamic ensemble selection for multi-class classification with one-class classifiers	Yes	No	DES
[16]	Modular ensembles for one-class classification based on density analysis	No	Yes	No
[33]	Support Vector Data Descriptions and k-Means Clustering: One Class?	No	Yes	No
	One-class Dynamic Ensemble Selection for Multi-class problems (MODES)	Yes	Yes	DES

distribution are not taken into account when using only one OCC for each class, what may impair the classification performance. However, to the best of our knowledge, state-of-the-art dynamic selection techniques for one-class decomposition use only one OCC for each class of the original problem.

Other methods [16,33] successfully use multiple one-class classifiers, however they were not used for one-class decomposition. Even so, they served as inspiration to the proposed architecture, since they represent important advances in this research field.

Table 1 summarizes the characteristics of the related works in comparison with the proposed approach (MODES). We evaluate three aspects: (1) Can the technique handle multi-class problems, i.e., can it be used for one-class decomposition? (2) Does the technique adopt OCC ensembles for the one-class problem? (3) Does the technique adopt dynamic selection (DS) for the one-class problem? MODES is the only technique that exhibits all these characteristics.

4. Proposal

The proposed architecture, named One-class Dynamic Ensemble Selection for Multi-class problems (MODES), aims to tackle multi-class classification by decomposing the original problem into one-class problems, generating pools of classifiers for each one-class problem, and using dynamically selected OCC ensembles to classify each test example. It deals with complex intra-class data distribution by segmenting the training data of each original class, using a clustering algorithm such as k-Means, and training an OCC for each cluster.

MODES is composed of two main phases: (i) training phase and (ii) test phase. The training phase consists of separating the training data by class and generating a pool of one-class classifiers for each class. The data is segmented into different numbers of clusters and a one-class classifier is trained for each cluster.

To tackle the problem of defining the ideal number of clusters, a set of 13 cluster validity indices [18,34] are computed for the data from each class. Each index evaluates the clustering using from 2 to 10 clusters and outputs the number of clusters that it considers ideal for the data. This range was chosen in experiments, where we identified that, in general, using more than 10 clusters worsens the classification accuracy of MODES. The goal is to minimize the dependency on the determination of the ideal number of clusters and, thus, to approximate the number of partitions to the number of modes present in the data. Moreover, training the pool of OCCs with different input data (different partitions) should increase the diversity, which is an important aspect for ensemble methods.

Algorithm 1 describes the training phase, which follows these steps:

Algorithm 1: Training phase

```
input  : $\Gamma$: a training set
output: $M$: an array with pools of OCCs for each class
1  $M = \emptyset$
2  for $c \in C$ do
3      $\Gamma_c = examples(\Gamma, c)$            // get examples from class $c$
4      $D = g = \emptyset$
5      for $index \in \{Silhouette, Hartigan, C\text{-}index, ...\}$ do
6          $D = D \cup computeIndex(\Gamma_c, index)$
7      for $i = 1 : |D|$ do
8          $g_i = segment(\Gamma_c, d_i)$
           // $\{k_1, ..., k_j, ..., k_{d_i}\}$
9          $p_i = \emptyset$
10         for $j = 1 : d_i$ do
11             $\lambda_j = trainOCC(k_j)$
12             $p_i = p_i \cup \lambda_j$
13         $M_c = M_c \cup < p_i, g_i >$
14 return $M$
```

1. Separate the training examples by class (lines 2 and 3): let C be the set of classes, the examples in the training set Γ are separated by class resulting in $|C|$ training sets: $\{\Gamma_1, \Gamma_2, \ldots, \Gamma_{|C|}\}$.
2. Compute the cluster validity indices for Γ_c (lines 5 and 6): for each class $c \in C$, a set of cluster validity indices is computed. Each index assesses the data segmentation with 2 to 10 clusters and the best number of clusters indicated by the index is added to the set D. After the computation of all indices, D contains at least one integer (when all indices indicate the same number of clusters).
3. Segment Γ_c using the numbers of clusters in D (lines 7 and 8): Γ_c is segmented using a clustering algorithm and each $d_i \in D$. This results in $|D|$ different partitions $\{g_1, \ldots, g_i, \ldots, g_{|D|}\}$, where each g_i contains d_i clusters $\{k_1, \ldots, k_j, \ldots, k_{d_i}\}$.
4. Train an OCC for each cluster (lines 10 to 12): for each cluster k_j in g_i, a one-class classifier λ_j is trained and added to the pool p_i. Clusters containing only one example are discarded, since it may represent an outlier.
5. Repeat items 2 to 4 for each class $c \in C$, adding the pairs $\langle p_i, g_i \rangle$ to M_c.

The output of the training phase is the array M, which contains c entries M_c, one for each class. Each M_c is an array containing $|D|$ pairs $\langle p_i, g_i \rangle$, where p_i is a pool composed of d_i OCCs $\{\lambda_1, \ldots, \lambda_j, \ldots, \lambda_{d_i}\}$ and g_i is Γ_c segmented into d_i clusters $\{k_1, \ldots, k_j, \ldots, k_{d_i}\}$. M_c binds each OCC λ_j to the cluster k_j used to train it.

In the test phase, for each test example x, MODES uses only the classifiers trained with data belonging to classes present in the neighborhood of x. The neighborhood Ψ is defined as the classes of the k nearest neighbors of x, where k is configured with 3 x $|C|$. Additionally, a threshold is applied so that classes with less than 10% of the examples in the neighborhood are discarded. This approach is based on that proposed in [8] and aims to remove non-competent classifiers from the ensemble. For each class c present in the neighborhood of x, an ensemble E_c is dynamically selected to classify x. The classification of x in the cth one-class problem (i.e, if x belongs or not to class c) is given by the aggregation of the OCCs in E_c. The final classification of x, i.e., the final decision that assigns a label to x, is given by the aggregation of the decisions made by the ensembles $E_1, \ldots, E_c, \ldots, E_C$ generated for each class c present in the neighborhood of x.

Algorithm 2: Test phase

```
input  : $x$: a test example
         $M$: an array with pools of OCCs for each class
         $D_{SEL}$: a validation set
output: $\omega$: the predicted class for $x$
1  $nn = kNN(x, D_{SEL})$
2  $\Psi$ = classes $\in nn$                // get the set of classes of examples in $nn$
3  $R = \emptyset$
4  for $c \in \Psi$ do
5      $E_c = \emptyset$
6      for $< p_i, g_i > \in M_c$ do
7          // $p_i = \{\lambda_1, ..., \lambda_j, ..., \lambda_{d_i}\}$ and $g_i = \{k_1, ..., k_j, ..., k_{d_i}\}$
8          $n = argmin\{distance(x, g_i)\}$        // index to the nearest cluster
9          $E_c = E_c \cup \lambda_n$
10     $\omega_c = mean(E_c, x)$
       // mean aggregation
11     $R = R \cup \omega_c$
12 $\omega = agg(R, x)$                        // DTs, ECOC or MAX
13 return $\omega$
```

Algorithm 2 describes the test phase, which follows these steps to classify each test example x:

1. Compute the region of competence Ψ (lines 1 and 2): the region of competence is defined as the classes of the k nearest neighbors of x in the validation set D_{SEL}. k is defined as $3 \times |C|$. A threshold is applied so that classes with less than 10% of the examples in the neighborhood are discarded [8].
2. Select each class c present in the region of competence Ψ (line 4). If Ψ contains only one class, x is assigned to this class.
3. Select the ensemble E_c (5 to 9): for each pair $\langle p_i, g_i \rangle \in M_c$, identify in g_i the closest cluster to x and select in p_i the OCC trained with data from that cluster to the ensemble E_c.
4. Aggregate E_c (lines 10 and 11): the mean of probabilities is used to aggregate the predictions for the one-class problem. This process is repeated for all the classes in Ψ and the partial decisions are added to the array R.
5. Aggregate the decisions in R (line 12): Use an aggregation strategy to recompose the original multi-class problem from the one-class decisions.

MODES has three hyperparameters, however, it is not necessary to tune or optimize values for them, since previous experiments encountered efficient values that lead to enhanced classification performance. Table 2 details the values used for these hyperparameters.

It is important to note that the proposed architecture presents an increased computational complexity which may lead to increased memory and processing requirements. However, since the focus of applications for MODES does not include large databases nor data streams, we focused our efforts on enhancing the classification accuracy rather than computational requirements.

Table 2
Default values for MODES hyperparameters.

Hyperparameter	Value
Number of clusters	2 to 10
Ψ (region of competence)	3 x $\|C\|$, where C is the set of classes
Threshold for Ψ	10%

4.1. Toy example

We present a toy example in order to demonstrate how MODES works in a visual manner, which helps to understand it. Figs. 3 and 4 describe the training and test phases, respectively, for a specific class $c \in C$ of a hypothetical database. The target class is represented by the red dots while the green dots represent the outliers, i.e., the data from other classes.

The first step of the training phase is the computation of the cluster validity indices for the target data, in order to define a set of numbers of clusters to segment the data. Suppose that, in this toy example, the computation of the cluster validity indices outputs the set $D = \{2, 4, 5\}$. Thus, for each $d_i \in D$, MODES segments the target class data into d_i clusters $g_i = \{k_1, \ldots, k_{d_i}\}$. Then, an OCC λ_j is trained for each cluster $k_j \in g_i$, making up the pool p_i. For instance, for $d_1 = 2$, the target data is segmented into $g_1 = \{k_1, k_2\}$ and the pool of OCCs $p_1 = \{\lambda_1, \lambda_2\}$ is trained. The pairs $\langle p_i, g_i \rangle$ are added to the array M_c. In this toy example, $M_c = \{\langle p_1, g_1 \rangle, \langle p_2, g_2 \rangle, \langle p_3, g_3 \rangle\}$. Each of these pairs bind a pool of OCCs to the clustered data which it was trained with. The output of the training phase is the array M, composed of the arrays M_c for $c \in C$.

The test phase starts with a test instance x. The region of competence Ψ is defined, based on the neighborhood of x. For each class $c \in \Psi$, MODES retrieves from the array M (composed in the training phase) the array M_c. MODES then computes, for each g_i, the distance from x to the centers of the clusters. Then, the OCC in p_i trained with the data from the closest cluster is selected to compose the ensemble E_c. In this toy example, since the closest clusters to x are k_2 for d_1, k_2 for d_2 and k_3 for d_3, $E_c = \{\lambda_{12}, \lambda_{22}, \lambda_{33}\}$, where λ_{d_ij} is the jth OCC from the pool p_{d_i}. The one class classification for the class c is given by the mean of probabilities for E_c. Suppose that E_c outputs $\{0.75, 0.85, 0.8\}$, then $\omega_c = 0.8$, where ω_c is the mean probability computed by E_c. The final classification, is given by the aggregation of the decisions for each E_c, with $c \in E$. Using MAX aggregation, for instance, the test example would be assigned to the class c with higher support ω_c.

5. Experiments

In this section, we evaluate the performance of the proposed architecture. The experiments compare MODES with two different approaches of one-class decomposition: (1) Dynamic Ensemble Selection with THReshold based neighborhood pruning (DES_{THR}) [8], a dynamic ensemble selection method with neighborhood pruning based on threshold, which is the state-of-the-art technique regarding multi-class classification using one-class decomposition; (2) the aggregation, without selection, of a pool of one-class classifiers composed of one OCC for each class [7]. In (2), the training phase consists in dividing the data by class and training an OCC for each class. In the test phase, a test example is submitted to all trained OCC and an aggregation technique is used to give the final classification based on the outputs of each OCC. Technique (2) is further referred in this work as static combination of OCCs. The experimental protocol is described in Section 5.1 and the results are discussed in Section 5.2.

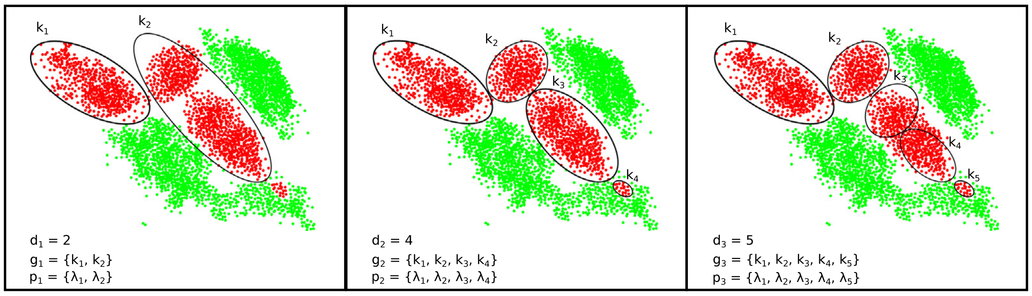

Fig. 3. Toy example of the training phase of MODES for a class c. The cluster validity indices output a set $D = \{2, 4, 5\}$ containing the numbers of clusters to segment the target data. A one-class classifier λ_j is trained for each cluster k_j.

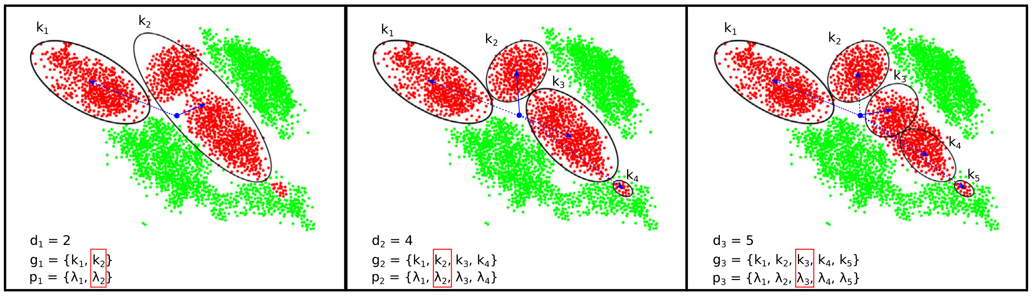

Fig. 4. Toy example of the test phase of MODES for a class c. MODES computes the euclidean distance between the test example x (represented by ●) and the centroid of each cluster. The OCC trained with data from the closest cluster is dynamically selected to the ensemble E_c.

5.1. Experimental protocol

Twenty five databases from Keel datasets repository[1] were used. Most of these databases were used in [8] which also aims to evaluate the decomposition of multi-class problem into one-class problems. Some of the selected databases are modified versions of other databases. For example, Glass1 and Glass6 are binary versions of the database Glass, where the positive examples belong to the classes 1 and 6, respectively, and the negative examples belong to the other classes. Furthermore, the databases Movement Libras, Optidigits and Texture were modified. Some classes were joined so that the modified databases have 7, 4 and 5 classes respectively. The selected databases present variable number of features (from 6 to 90), classes (from 2 to 26) and examples (from 148 to 12,960). Furthermore, there are balanced and imbalanced databases. Table 3 describes the characteristics of each database.

As the proposed architecture relies on the distance among the training examples (for clustering), in a pre-processing step, the data was scaled using z-score [35]. Non-numerical attributes were transformed using simple label encoder, where each value is bound to an integer value (for example, a is transformed to 1, b is transformed to 2 and so on).

The determination of the numbers of clusters for the training data of each class was performed using the 13 fastest indexes present in the *R* package *NbClust* [34]: Calinski and Harabasz [36], PtBiserial [37], Hartigan [38], Ball [39], Mcclain [40], KL [41], Silhouette [42], Gap [43], Dunn [44], SDindex [45], SDbw [46], C-index [47], Davies and Bouldin [48]. The evaluation of the numbers of clusters was performed from 2 to 10 clusters. Since the average number of clusters returned by the cluster validity indexes was 5.5 (with standard deviation 3), we did not evaluate other ranges. The clustering algorithm adopted in the experiments was k-Means due to its simplicity and good results [49]. We used the default hyperparameters values given by *Scikit Learn*[2] implementation, except for the number of clusters, which was chosen by clustering validity indices, as explained in Section 4.

Four one-class classifiers were adopted in this work, all of them implemented in the Matlab package *dd_tools* [50]: Gaussian Data Descriptor (GaussianDD), Parzen Data Descriptor

Table 3
Databases description. Imbalance Ratio is computed as the division of the cardinality of the largest class by the cardinality of the smallest class.

Name	Examples	Features	Numeric	Nominal	Classes	Imbalance ratio
Automobile	159	25	15	10	6	16.00
Car	1,728	6	0	6	4	18.62
Cleveland	297	13	13	0	5	12.31
Dermatology	358	34	34	0	6	5.55
Ecoli	336	7	7	0	8	71.5
Flare	1,066	11	9	2	6	7.70
Glass	214	9	9	0	6	8.44
Glass1	214	9	9	0	2	1.82
Glass6	214	9	9	0	2	6.38
Led7digit	500	7	7	0	10	1.54
Letter	2,000	16	16	0	26	1.11
Lymphography	148	18	3	15	4	40.50
Movement Libras	360	90	90	0	7	1.50
Nursery	12,960	8	0	8	5	2,160
Optdigits	5,620	64	64	0	4	1.53
Page-blocks	5,472	10	10	0	5	175.5
Penbased	10,992	16	16	0	10	1.08
Satimage	6,435	36	36	0	6	2.45
Segment	2,310	19	19	0	7	1.00
Shuttle	5,780	9	9	0	7	4,558
Texture	5,500	40	40	0	5	5.00
Vehicle	846	18	18	0	4	1.10
Vehicle2	846	18	18	0	2	2.88
Vowel	990	13	13	0	11	1.00
Yeast	1,484	8	8	0	10	92.6

[1] Available in: http://www.keel.es/dataset.

[2] https://scikit-learn.org/.

Table 4
OCCs hyperparameters. Matlab *dd_tools* library implementation was used in the experiments.

GaussianDD	Regularization parameter = 0.001 Fraction rejected = 0.05
ParzenDD	Kernel type = normal Width parameter optimization = Max. likelihood Fraction rejected = 0.05
SVDD	Kernel type = RBF $C = 5.0$ $\gamma = 0.0045$ Fraction rejected = 0.05
MSTDD	Max. path = none Fraction rejected = 0.05

(ParzenDD), Support Vector Data Descriptor (SVDD) and Minimum Spanning Tree Data Descriptor (MSTDD). SVDD, ParzenDD and MSTDD are used in [8], which is the state-of-the-art one-class decomposition technique. Our technique uses centroid-based clustering to generate the chunks of data used for training. Thus, we expect that GaussianDD performs well with MODES. For this reason, we also evaluate GaussianDD one-class classifier. We used the implementations of the OCCs available in the Matlab library *dd_tools* [51] and the hyperparameters used in the experiments are detailed in Table 4.

MODES aggregates the prediction in two levels. The first is performed in the one-class problem. The output of the one-class classification, i.e., whether the example resembles the target class or not, is given by the aggregation of the selected classifiers. For this level of aggregation, the mean of probabilities [12] is employed. The second level is in the re-composition of the multi-class problem, where the predictions of the one-class problems are aggregated to give the final prediction, i.e., which class the example belongs to. In the second level, we adopt Decision Templates (DTs) [52], Error Correcting Output Codes (ECOC) [53] or Maximum Support (MAX) [12].

The experiments were performed using 5-fold cross-validation and the final performance is given by the average of the 5 folds. The performance was evaluated using accuracy [12] and Kappa statistic [8,54], both metrics used in [8]. Kappa statistics, or Cohen's Kappa Coefficient, was adopted as a complementary metric to accuracy because it provides a different evaluation for the results, mainly for minority classes. Eqs. (1) and (2) describe the computation of such metrics.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \quad (1)$$

where TP, TN, FP and FN mean true positive, true negative, false positive and false negative, respectively.

$$\text{Kappa} = \frac{p_o - p_e}{1 - p_e}, \quad (2)$$

where p_o is the empirical probability of agreement on the label assigned to any sample (the observed agreement ratio between the predicted class and the actual class), and p_e is the hypothetical probability of chance agreement.

5.2. Experimental results

Two experiments were executed in this work: (1) different configurations of the proposed architecture (using a set of one-class classifier models and aggregation techniques) were evaluated to determine the most effective one; (2) the best configuration for MODES is compared with the state-of-the-art technique for one-class decomposition, DES_{THR} [8]. The results are presented in Sections 5.2.1 and 5.2.2, respectively.

5.2.1. Experiment 1

This experiment aims to identify the most effective configuration for MODES. Thus, we evaluate the proposed architecture using 4 OCC models and 3 aggregation techniques, making up 12 different configurations for MODES.

Tables 5 and 6 present the accuracy and Kappa statistics scores, respectively, for all configurations of MODES. The best result for each database is in bold and the last rows present the mean performance for all databases, the number of wins, ties and losses and the average rankings. The average rankings are computed using all the 12 configurations.

MODES achieved its best performance using MSTDD and Decision Templates, for both accuracy and Kappa Statistic. This configuration presented the best average accuracy and Kappa statistics scores and the higher number of wins, i.e., the number of databases in which it achieved the best performance among all configurations, for both metrics. The best results were achieved with DTs and the best performing one-class classifiers were MSTDD and GaussianDD.

Friedman test was employed to determine if the different configurations present significantly different performance. The test output $p - value = 1.11^{-16}$ for both accuracy and Kappa Statistics, meaning that the difference of performance among the configurations is significant. Then, Nemenyi *post hoc* test was used to identify which of the configurations differ from each other. Fig. 5 shows the results for Nemenyi *post hoc test*.

Again, MSTDD with DTs was identified as the best performing configuration, for both accuracy and Kappa Statistics. It is also interesting to note that the 7 best performing configurations (all base OCCs (except for ParzenDD) with DTs and MAX aggregation) present quite similar performance (mean accuracy between 81.12 and 84.05). The performance of the other 5 configurations (all base OCCs with ECOC and ParzenDD with DTs) is in a level below (mean accuracy between 73.43 and 79.28). This can be interpreted as an evidence that MODES is robust to the choice of the base one-class classifier, since, except for ParzenDD with DTs, the base OCCs performed similarly when using the same aggregation technique.

A quantitative analysis of the generation and selection phases of the proposed architecture was carried out. In the training phase, MODES segments the data of each class d using k-Means clustering algorithm with a set N of numbers of clusters $\{n_1, n_2, \ldots\}$ given by the cluster validity indices. For each $n_i \in D$, MODES trains a pool of OCCs, one OCC for each cluster. The average number of OCCs generated by class ranges from 13.60 to 33.74 (average 25.55) using DTs or MAX and from 16.17 to 35.10 (average 28.48) using ECOC.

In the selection phase, MODES selects one OCC for each number $n_i \in N$ to compose the ensemble. Thus, the cardinality of N represents the number of OCCs selected. The distribution for the frequency of selection of the OCCs is presented in Fig. 6: never selected, selected for up to 1%, 2%, 3%, 4%, 5% of the test examples and selected for more than 5% of the test examples. The frequencies are equal for DTs and MAX, since these aggregation strategies use the original classes. ECOC creates meta-binary problems which use meta-classes, thus, the frequencies for ECOC are showed separately (right side of Fig. 6). For the majority of the databases, the highest frequency of selection is between 0% and 1%, that is, most of the OCCs are selected to classify up to 1% of the test examples. It is worth noting that the majority of the OCCs is selected at least once. Using ECOC, less than 10% of the OCCs is never used while for DTs and MAX, less than 20% of the OCCs are never used. Both figures indicate that the strategy for pool generation (i.e., segment the training data and train an OCC for each cluster) is effective, since few OCC are never selected and most of the OCCs are used for classifying few examples, what means that they are used for specific regions of the feature space.

Table 5
Accuracy performance of MODES using DTs, ECOC and MAX aggregation methods for four one-class classifiers. Best result for each database in bold. The last row represents the number of wins, ties and losses achieved by each technique.

Dataset	Decision templates				Error correcting output codes				MAX			
	GaussianDD	ParzenDD	SVDD	MSTDD	GaussianDD	ParzenDD	SVDD	MSTDD	GaussianDD	ParzenDD	SVDD	MSTDD
Automobile	66.44	57.12	72.46	**78.69**	66.56	62.10	60.79	63.11	65.04	72.95	69.19	71.09
Car	89.12	**90.80**	90.68	84.84	90.05	77.49	85.48	82.00	83.85	89.70	81.71	84.32
Cleveland	58.24	47.43	57.24	53.88	46.46	37.41	48.77	54.18	57.18	55.86	**60.28**	53.85
Dermatology	93.02	84.89	92.72	92.74	**93.28**	91.05	92.75	93.03	91.56	92.20	92.41	91.89
Ecoli	75.90	76.80	78.28	77.69	**78.61**	77.69	77.71	72.93	75.00	76.50	75.30	75.91
Flare	68.76	69.51	66.60	69.61	62.94	68.39	63.41	64.63	64.07	63.60	**70.08**	69.89
Glass	62.18	64.95	59.41	**71.54**	57.50	63.11	61.24	56.57	57.48	65.42	64.98	68.69
Glass1	71.01	75.70	69.15	81.33	62.64	71.54	65.91	70.09	75.27	78.98	75.24	**81.34**
Glass6	94.85	94.85	**96.26**	95.32	93.44	93.44	93.91	95.78	**96.26**	95.78	95.79	95.32
Led7digit	70.80	71.60	71.20	**72.20**	68.00	67.60	69.20	55.00	70.00	71.00	72.00	60.80
Letter	92.48	83.81	68.17	90.52	65.50	44.90	51.55	70.41	**95.34**	94.22	84.65	93.88
Lymphography	74.37	66.25	76.97	74.32	68.94	71.68	77.72	71.68	66.90	68.97	**79.03**	70.92
Movement Libras	77.22	52.78	77.78	**86.67**	79.17	67.22	71.94	70.56	78.06	83.33	76.94	85.56
Nursery	92.23	80.05	83.56	74.58	83.70	54.55	62.82	63.73	**92.56**	79.61	89.00	72.69
Optdigits	98.02	95.23	97.38	98.02	96.69	90.11	96.30	96.80	97.85	**98.11**	97.33	98.01
Page-blocks	**95.18**	94.76	94.32	95.01	93.93	91.25	93.21	92.68	95.14	94.76	94.92	94.96
Penbased	99.36	97.19	97.40	99.27	96.53	81.96	94.13	97.42	**99.42**	99.19	98.45	99.02
Satimage	87.57	86.03	86.62	**90.35**	82.50	82.50	81.29	83.26	88.02	88.56	87.80	88.39
Segment	93.77	91.34	90.82	**95.37**	91.95	94.42	90.43	94.07	94.42	94.72	91.99	95.24
Shuttle	99.23	99.36	99.17	**99.42**	99.33	99.21	99.17	99.33	99.24	99.31	99.24	99.42
Texture	**99.80**	93.35	94.76	98.53	98.60	95.84	94.53	96.49	99.78	98.67	96.80	98.95
Vehicle	**78.72**	62.06	70.44	70.45	77.07	63.24	63.24	63.71	78.60	68.80	69.26	68.44
Vehicle2	**99.17**	94.21	94.45	97.64	94.33	86.88	92.91	93.50	97.05	96.93	94.44	97.64
Vowel	97.68	80.30	90.10	**98.48**	89.39	47.17	66.77	89.60	97.58	97.07	88.99	97.88
Yeast	53.98	56.06	52.16	54.78	44.81	54.92	44.20	44.40	49.93	**56.40**	47.30	51.01
Mean	83.56	78.66	81.12	**84.05**	79.28	73.43	75.98	77.40	82.62	83.23	82.12	82.60
Win/tie/loss	4/0/21	0/1/24	1/0/24	8/0/17	2/0/23	0/0/25	0/0/25	0/0/25	3/1/21	2/0/23	3/0/22	1/0/24
Avg ranks	4.74	7.73	6.43	3.94	7.44	9.12	9.07	8.51	5.24	4.82	5.94	5.02

Table 6
Kappa performance of MODES using DTs, ECOC and MAX aggregation methods for four one-class classifiers. Best result for each database in bold. The last row represents the number of wins, ties and losses achieved by each technique.

Dataset	Decision templates				Error correcting output codes				MAX			
	GaussianDD	ParzenDD	SVDD	MSTDD	GaussianDD	ParzenDD	SVDD	MSTDD	GaussianDD	ParzenDD	SVDD	MSTDD
Automobile	0.5788	0.5477	0.6346	**0.7199**	0.5453	0.5012	0.4729	0.5283	0.5290	0.6493	0.5899	0.6176
Car	0.7822	**0.8029**	0.8028	0.6808	0.7753	0.4328	0.6855	0.6355	0.5905	0.7746	0.5350	0.6526
Cleveland	0.2735	0.2537	0.3054	0.2996	0.2358	0.1781	0.2208	0.1688	0.1955	**0.3387**	0.2918	0.2984
Dermatology	0.9125	0.8110	0.9082	0.9086	**0.9164**	0.8887	0.9098	0.9125	0.8934	0.9017	0.9040	0.8977
Ecoli	0.6667	0.6815	0.6979	0.6903	0.6943	**0.6980**	0.6782	0.6027	0.6419	0.6697	0.6488	0.6615
Flare	0.6039	0.6144	0.5777	0.6146	0.5350	0.5972	0.5362	0.5531	0.5419	0.5377	**0.6178**	0.6156
Glass	0.4855	0.5314	0.4632	**0.6128**	0.4095	0.4878	0.4626	0.3970	0.3763	0.5250	0.4996	0.5654
Glass1	0.3649	0.4840	0.3626	**0.5810**	0.1601	0.4553	0.2114	0.2395	0.3799	0.5284	0.4473	0.5626
Glass6	0.7705	0.7755	**0.8226**	0.7855	0.7281	0.7281	0.7437	0.8092	0.8134	0.8092	0.7955	0.7855
Led7digit	0.6752	0.6842	0.6797	**0.6909**	0.6447	0.6403	0.6578	0.4996	0.6666	0.6774	0.6884	0.5642
Letter	0.9218	0.8316	0.6689	0.9014	0.6412	0.4267	0.4962	0.6922	**0.9516**	0.9399	0.8404	0.9363
Lymphography	0.5038	0.3845	0.5509	0.4977	0.3910	0.4681	0.5620	0.4483	0.3784	0.4059	**0.5885**	0.4245
Movement Libras	0.7331	0.4518	0.7388	**0.8434**	0.7542	0.6153	0.6690	0.6559	0.7402	0.8043	0.7280	0.8302
Nursery	0.8875	0.7087	0.7616	0.6315	0.7593	0.3380	0.4487	0.4686	**0.8902**	0.7018	0.8373	0.6031
Optdigits	0.9733	0.9357	0.9646	0.9733	0.9552	0.8671	0.9500	0.9567	0.9709	**0.9745**	0.9639	0.9731
Page-blocks	**0.7700**	0.7550	0.6952	0.7564	0.6860	0.6137	0.6337	0.5711	0.7665	0.7585	0.7131	0.7615
Penbased	0.9929	0.9688	0.9711	0.9919	0.9615	0.7994	0.9348	0.9714	**0.9935**	0.9910	0.9828	0.9891
Satimage	0.8464	0.8253	0.8357	**0.8812**	0.7824	0.7856	0.7670	0.7946	0.8511	0.8589	0.8491	0.8570
Segment	0.9273	0.8990	0.8929	**0.9460**	0.9061	0.9348	0.8884	0.9308	0.9348	0.9384	0.9066	0.9444
Shuttle	0.9785	0.9823	0.9771	**0.9837**	0.9813	0.9780	0.9769	0.9813	0.9789	0.9807	0.9788	**0.9837**
Texture	**0.9971**	0.9048	0.9253	0.9789	0.9798	0.9401	0.9213	0.9496	0.9969	0.9809	0.9538	0.9848
Vehicle	**0.7162**	0.4935	0.6066	0.6060	0.6943	0.5103	0.5090	0.5151	0.7145	0.5836	0.5901	0.5789
Vehicle2	**0.9781**	0.8365	0.8593	0.9372	0.8521	0.7071	0.8034	0.8194	0.9197	0.9206	0.8467	0.9370
Vowel	0.9744	0.7833	0.8911	**0.9833**	0.8833	0.4189	0.6344	0.8856	0.9733	0.9678	0.8789	0.9767
Yeast	0.4064	**0.4360**	0.3818	0.4221	0.2471	0.4082	0.2379	0.2396	0.3435	0.4337	0.3065	0.3668
Mean	0.7488	0.6953	0.7190	**0.7552**	0.6848	0.6168	0.6405	0.6491	0.7213	0.7461	0.7193	0.7347
Win/tie/loss	4/0/21	2/0/23	1/0/24	8/1/16	1/0/24	1/0/24	0/0/25	0/0/25	3/0/22	2/0/23	2/0/23	0/1/24
Avg ranks	4.78	7.53	6.48	4.08	7.39	8.65	9.04	8.64	5.58	4.58	6.32	4.93

5.2.2. Experiment 2

In this experiment, we compare MODES and DES_{THR} [8], which is the state-of-the-art technique for one-class decomposition. The comparison involves the most effective configuration for each technique: MSTDD with DTs for MODES and SVDD with DTs for DES_{THR}. We also evaluate DES_{THR} with GaussianDD, using DTs, ECOC and MAX, since this OCC model was not evaluated in [8].

Additionally, we compare these techniques with the static combination of one OCC for each class [7]. For this technique, we carried out a preliminary evaluation involving four OCC models

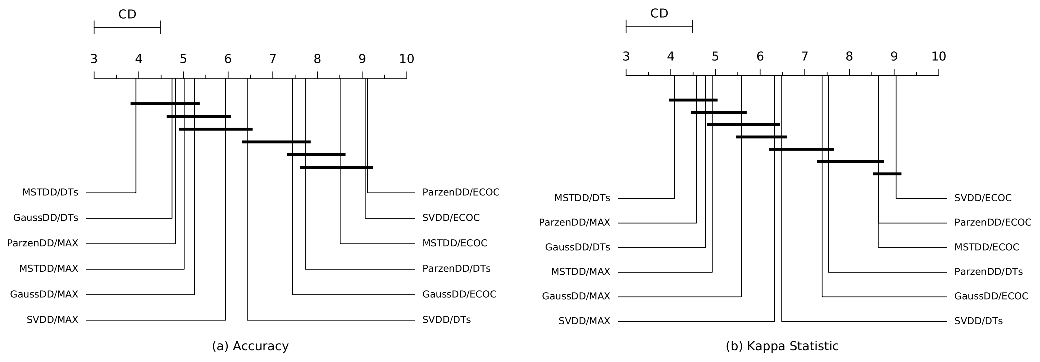

Fig. 5. Result for Nemenyi post hoc test for (a) accuracy and (b) Kappa Statistic.

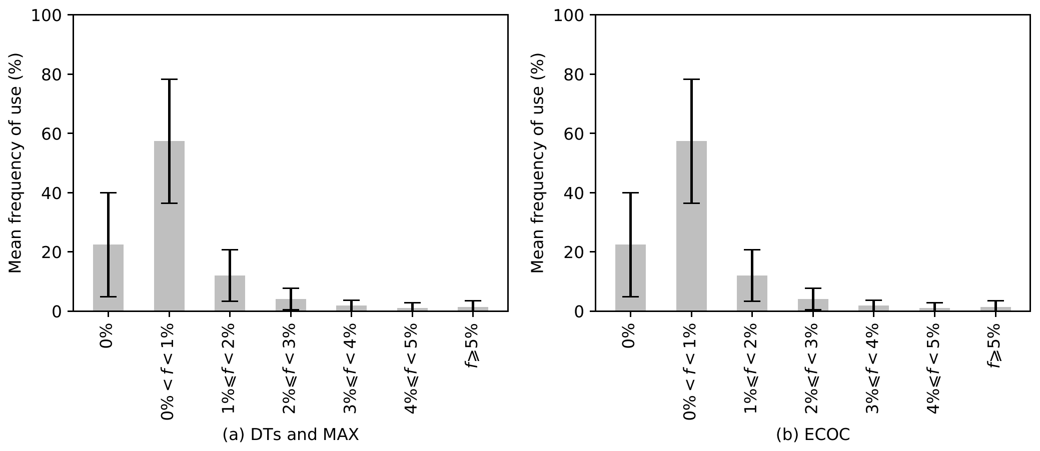

Fig. 6. Frequency of use of base OCCs with MODES architecture for DTs and MAX (a) and ECOC (b).

(GaussianDD, ParzenDD, SVDD and MSTDD) and three aggregation techniques (DTs, ECOC and MAX). The preliminary evaluation aimed at selecting to Experiment 2 the best performing configuration for each aggregation technique. Considering both Accuracy and Kappa statistic, the evaluation identified that GaussianDD with DTs, ParzenDD with MAX, SVDD with MAX and MSTDD with DTs present the top performances.

Tables 7 and 8 show the results obtained in Experiment 2. The analysis of the mean performances and the win/tie/loss (number of databases in which a technique performs better than, equals to or worse than another technique) indicate that MODES is superior than all configurations of DES_{THR} and the static combination of OCCs.

MODES outperformed the other techniques for 10 and 9 databases, regarding accuracy and Kappa statistic, respectively. The second best performing technique was DES_{THR} with GaussianDD and MAX, which presented the best performance for 6 databases, for both accuracy and Kappa statistic.

MODES also presented the higher mean scores, for both accuracy and Kappa statistic (84.05 and 0.7552, respectively). DES_{THR} with GaussianDD and MAX presented the second best accuracy performance (81.71) while DES_{THR} with GaussianDD and DTs presented the second best Kappa statistic performance (0.7208).

Wilcoxon signed-rank test [55] was executed to verify if the differences between the average performances obtained by MODES and the other techniques are significant. The test was performed in a pairwise manner, verifying the null hypothesis that each of the methods present similar performance and the alternative hypothesis that MODES presents better performance than the other techniques. The results of Wilcoxon Signed Rank Test indicate a strong evidence that MODES performance is better than all the other techniques ($p - value < 0.002$ for all of the cases).

In order to evaluate the applicability of the proposed architecture, an analysis of the region of competence, i.e., the neighborhood of the test examples was performed. We identified that, for some databases, a considerable proportion of the examples are in regions where the neighbors belong to only one class. We say that such examples are in a homogeneous neighborhood. When an example is in a homogeneous neighborhood, the classification is directly given by the class of the examples in the neighborhood, without using the DES of OCCs. Fig. 7 shows the accuracy for MODES and DES_{THR}. The figures are divided into databases where up to 25% of the examples are in homogeneous regions (a) and databases where more than 25% of the examples are in homogeneous regions (b).

It is noticeable that, for databases with homogeneous neighborhoods, the performances of the OCCs are more stable. For databases with heterogeneous neighborhoods the variation of performance among OCCs is higher. The performance in databases with homogeneous neighborhoods are higher than in databases with heterogeneous neighborhoods. This is expected since it is known that examples in complex neighborhoods are hard to classify.

It is also important to notice that, for databases with homogeneous neighborhoods, MODES and DES_{THR} perform very similarly. This happens because, in such databases, more examples are classified in the same way by both methods (assigning the test example to the single class in the neighborhood). For databases with heterogeneous neighborhood, the difference between MODES and DES_{THR} is bigger. It is worth noting that MODES performs mostly better than DES_{THR} for both databases with homogeneous or heterogeneous neighborhood.

Table 7
Accuracy performance of MODES, DES_{THR} and Static aggregation of OCCs. Best result for each database in bold. The last rows represent the mean performance across all databases, the number of wins, ties and losses achieved by each technique, the average rankings and the $p-value$ for Wilcoxon Signed Ranking Test.

Dataset	MODES	DES_{THR}				Static			
	MSTDD/DTs	SVDD/DTs	GaussDD/DTs	GaussDD/ECOC	GaussDD/MAX	GaussDD/DTs	ParzenDD/MAX	SVDD/MAX	MSTDD/DTs
Automobile	78.69	57.82	67.80	68.15	58.54	64.77	75.48	60.93	**79.39**
Car	84.84	78.42	82.58	78.94	76.56	73.38	**89.58**	69.16	82.06
Cleveland	53.88	54.83	56.22	53.56	**58.90**	55.90	42.69	58.13	47.76
Dermatology	92.74	93.55	**96.36**	96.12	95.20	93.02	92.18	92.99	92.47
Ecoli	**77.69**	76.49	75.31	69.96	74.42	68.18	41.38	61.60	64.93
Flare	69.61	64.35	**69.98**	61.44	65.01	69.32	61.54	62.10	64.54
Glass	**71.54**	50.93	53.73	51.40	59.36	51.86	54.20	51.88	64.51
Glass1	**81.33**	66.38	71.98	69.16	73.39	62.16	72.46	58.41	74.33
Glass6	95.32	**96.26**	94.85	94.39	**96.26**	94.85	93.47	84.06	93.00
Led7digit	72.20	**74.00**	70.80	68.00	68.80	69.40	72.60	70.20	70.40
Letter	90.52	63.73	76.92	48.15	89.64	70.87	78.96	56.61	**92.75**
Lymphography	74.32	**77.77**	76.34	73.68	70.34	73.01	44.92	77.06	43.56
Movement Libras	**86.67**	78.06	75.28	65.28	74.44	69.44	85.00	75.28	86.11
Nursery	74.58	76.05	74.85	73.75	**86.38**	64.30	46.16	72.66	51.59
Optdigits	**98.02**	96.81	97.21	96.01	96.94	93.61	97.92	94.34	97.31
Page-blocks	**95.01**	93.99	94.99	93.21	94.19	87.43	33.76	92.98	77.50
Penbased	**99.27**	96.12	98.07	91.38	98.60	95.71	99.14	93.26	99.15
Satimage	**90.35**	85.72	85.33	83.10	87.58	80.25	88.83	83.37	89.42
Segment	**95.37**	86.80	92.42	88.70	91.60	91.21	89.22	72.55	93.03
Shuttle	**99.42**	99.28	99.14	99.19	99.28	95.26	88.28	90.13	93.62
Texture	98.53	93.31	99.24	97.25	**99.71**	98.84	97.78	83.75	95.47
Vehicle	70.45	64.54	81.21	72.10	**82.04**	81.32	71.99	57.69	69.98
Vehicle2	97.64	92.79	98.11	94.56	**98.47**	97.52	96.57	85.22	94.92
Vowel	98.48	67.58	90.71	50.51	90.71	91.21	98.28	58.08	**99.60**
Yeast	54.78	50.74	54.38	45.08	**56.47**	42.99	51.62	38.48	47.44
Mean	**84.05**	77.45	81.35	75.32	81.71	77.43	74.56	72.04	78.59
Win/tie/loss	10/0/15	2/1/22	2/0/23	0/0/25	6/1/18	0/0/25	1/0/24	0/0/25	3/0/22
Avg ranks	2.68	5.42	3.88	6.28	3.96	6.00	5.00	7.02	4.76
Wilcoxon	–	0.0000	0.0000	0.0000	0.0001	0.0000	0.0000	0.0000	0.0000

Table 8
Kappa statistic performance of MODES, DES_{THR} and Static aggregation of OCCs. Best result for each database in bold. The last rows represent the mean performance across all databases, the number of wins, ties and losses achieved by each technique, the average rankings and the $p-value$ for Wilcoxon Signed Ranking Test.

Dataset	MODES	DES_{THR}				Static			
	MSTDD/DTs	SVDD/DTs	GaussDD/DTs	GaussDD/ECOC	GaussDD/MAX	GaussDD/DTs	ParzenDD/MAX	SVDD/MAX	MSTDD/DTs
Automobile	0.7199	0.4412	0.5775	0.5727	0.4220	0.5380	0.6910	0.4797	**0.7344**
Car	0.6808	0.5287	0.6157	0.5728	0.4104	0.4697	**0.7704**	0.1914	0.6362
Cleveland	0.2996	0.2946	0.3091	0.1840	**0.3313**	0.3126	0.2080	0.3250	0.2581
Dermatology	0.9086	0.9190	**0.9543**	0.9514	0.9395	0.9130	0.9013	0.9110	0.9050
Ecoli	**0.6903**	0.6770	0.6630	0.5573	0.6521	0.5782	0.3230	0.4979	0.5507
Flare	0.6146	0.5503	**0.6183**	0.5190	0.5610	0.6107	0.5236	0.5140	0.5563
Glass	**0.6128**	0.3657	0.3740	0.3202	0.4096	0.3580	0.3860	0.3742	0.5429
Glass1	**0.5810**	0.3371	0.4189	0.2253	0.3586	0.2697	0.3883	0.0784	0.4567
Glass6	0.7855	**0.8197**	0.7699	0.7549	0.8134	0.7699	0.7746	0.5548	0.7514
Led7digit	0.6909	**0.7110**	0.6752	0.6450	0.6528	0.6594	0.6953	0.6688	0.6709
Letter	0.9014	0.6228	0.7599	0.4608	0.8923	0.6970	0.7812	0.5488	**0.9245**
Lymphography	0.4977	**0.5711**	0.5375	0.4681	0.4152	0.5183	0.2882	0.5550	0.2508
Movement Libras	**0.8434**	0.7426	0.7090	0.5922	0.6959	0.6388	0.8241	0.7066	0.8373
Nursery	0.6315	0.6542	0.6391	0.6129	**0.8000**	0.5050	0.3985	0.5999	0.4247
Optdigits	**0.9733**	0.9569	0.9622	0.9460	0.9586	0.9137	0.9719	0.9231	0.9637
Page-blocks	**0.7564**	0.6675	0.7557	0.6070	0.7375	0.5319	0.1078	0.5776	0.3521
Penbased	**0.9919**	0.9568	0.9786	0.9042	0.9844	0.9523	0.9905	0.9251	0.9906
Satimage	**0.8812**	0.8248	0.8202	0.7893	0.8478	0.7577	0.8631	0.7924	0.8703
Segment	**0.9460**	0.8460	0.9116	0.8682	0.9020	0.8975	0.8742	0.6798	0.9187
Shuttle	0.9460	**0.9798**	0.9761	0.9773	**0.9798**	0.8773	0.7112	0.6952	0.8403
Texture	0.9789	0.9046	0.9890	0.9606	**0.9958**	0.9833	0.9683	0.7511	0.9361
Vehicle	0.6060	0.5284	0.7496	0.6278	**0.7607**	0.7512	0.6263	0.4350	0.5997
Vehicle2	0.9372	0.8225	0.9510	0.8528	**0.9591**	0.9360	0.9128	0.5339	0.8736
Vowel	0.9833	0.6433	0.8978	0.4556	0.8978	0.9033	0.9811	0.5389	**0.9956**
Yeast	0.4221	0.3707	0.4075	0.2501	**0.4386**	0.3065	0.3937	0.2699	0.3600
Mean	**0.7552**	0.6695	0.7208	0.6270	0.7126	0.6660	0.6542	0.5651	0.6880
Win/tie/loss	9/0/16	3/1/21	2/0/23	0/0/25	6/1/18	0/0/25	1/0/24	0/0/25	3/0/22
Avg ranks	2.85	5.38	3.76	6.52	4.08	5.76	4.95	7.12	4.59
Wilcoxon	–	0.0000	0.0001	0.0000	0.0002	0.0000	0.0000	0.0000	0.0000

We also carried out a performance analysis regarding the Imbalance Ratio (IR). The databases were separated in two groups: (a) low IR databases, composed of 16 databases; and (b) high IR databases, composed of 9 databases. Fig. 8 and Table 9 show the accuracy performance obtained by MODES and DES_{THR} in the two groups of databases. The boxplots show that the performances of both methods are higher and more stable in databases with low IR than in databases with high IR. For both groups, MODES present higher mean accuracy than DES_{THR} for all, except one, configuration. Furthermore, from Table 9, we notice that the

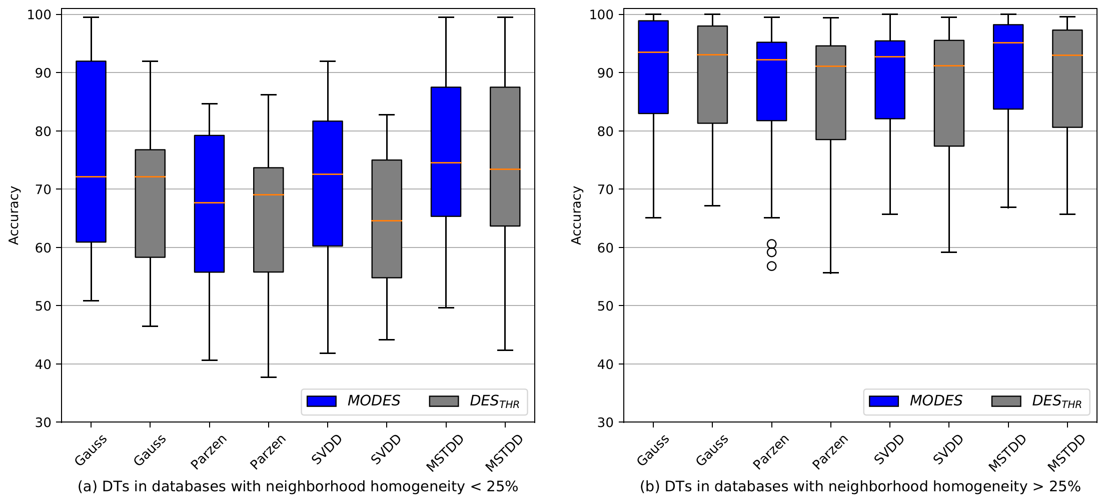

Fig. 7. Accuracy for MODES and DES_{THR} using Decision Templates according to the homogeneity of the neighborhood. Databases where more than 25% of the examples present more than one class in the neighborhood are showed in the left side of the figure. The right side shows databases where less than 25% of the examples present more than one class in the neighborhood.

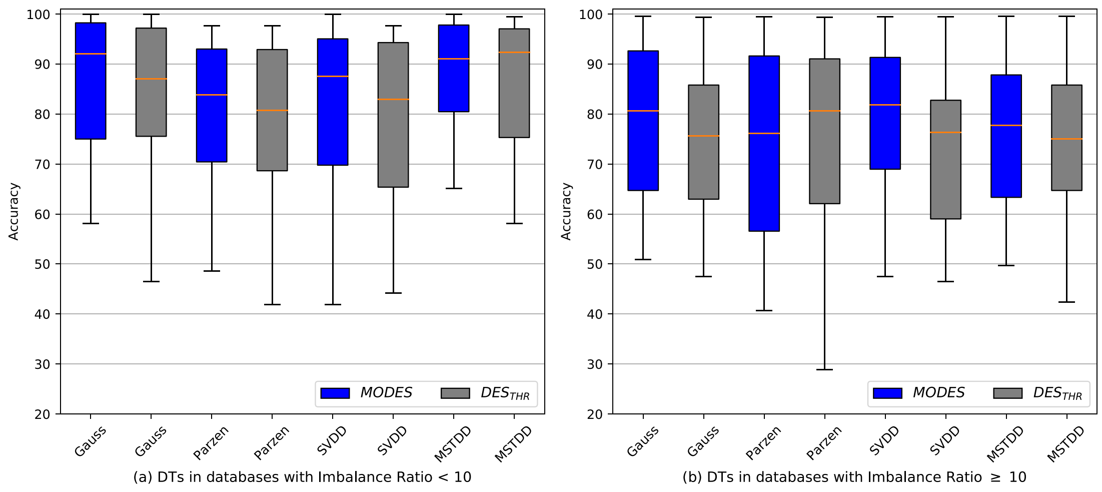

Fig. 8. Accuracy for MODES and DES_{THR} using Decision Templates according to the Imbalance Ratio. Databases with IR $<$ 10 are showed in the left side of the figure. The right side shows databases with IR $\geq$ 10.

Table 9
Accuracy for MODES and DES_{THR} using decision templates according to the imbalance ratio.

	IR $<$ 10		IR $\geq$ 10	
	MODES	DES_{THR}	MODES	DES_{THR}
GaussDD	86.53	84.52	78.30	75.74
ParzenDD	81.11	78.47	74.29	75.75
SVDD	82.79	79.43	78.32	73.93
MSTDD	88.00	87.00	77.02	76.24

difference of performance is higher for databases with high IR. For instance, with SVDD, MODES accuracy is 3.36 p.p. higher than DES_{THR} for databases with low IR while this difference is 4.39 p.p. for databases with high IR. The only exception is observed with ParzenDD, where DES_{THR} presents a slightly higher accuracy than MODES for databases with high IR.

6. Final remarks

This research proposed an architecture for multi-class classification, named MODES. MODES aims to improve the robustness and classification performance in problems with complex intra-class data distribution. It decomposes the original multi-class problem into multiple one-class problems and employs a clustering-based approach to generate pools of OCC. Dynamic Ensemble Selection is applied to classify each test example.

Experiments were carried out in a comprehensive experimental setting. The first experiment identified the best configuration for MODES. The best performance was achieved using the OCC model Minimum Spanning Tree Data Descriptor and the aggregation technique Decision Templates. In the second experiment, this configuration was compared with state-of-the-art techniques, using a variety of configurations. The results showed that MODES average performance is superior than all the configurations of the other methods. Additionally, we identified that MODES performs better for databases containing both complex and simple neighborhood (high or low presence of examples of other classes in the neighborhood, respectively) and for both balanced and imbalanced databases.

MODES does not require any parameter tuning and can be used along with any OCC. MODES architecture is composed of two levels: the first is responsible for one-class classification, i.e., whether the test example belongs or not to the target class;

and the second is responsible for the re-composition of the multi-class problem, i.e., aggregating the outputs of the one-class problems. Hence, MODES can, optionally, be easily adapted to be used in one-class problems.

For future works, we aim to analyze other clustering techniques, such as Density Based Spatial Clustering of Application with Noise (DBSCAN) [56], along with MODES, since this technique does not require parametrization for the number of clusters. Furthermore, we aim to explore the use of deep neural networks for feature extraction. These networks have reported in the literature good results [57].

CRediT authorship contribution statement

Rogério C.P. Fragoso: Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing - original draft. **George D.C. Cavalcanti:** Conceptualization, Methodology, Supervision, Writing - review & editing. **Roberto H.W. Pinheiro:** Conceptualization, Writing - review & editing. **Luiz S. Oliveira:** Conceptualization, Writing - review & editing.

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

The authors would like to thank Brazilian agencies: CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) and FACEPE (Fundação de Amparo à Ciência e Tecnologia de Pernambuco).

References

[1] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M.P. Reyes, M.-L. Shyu, S.-C. Chen, S. Iyengar, A survey on deep learning: Algorithms, techniques, and applications, ACM Comput. Surv. 51 (5) (2018) 1–36.
[2] P. Cao, S. Zhang, J. Tang, Preprocessing-free gear fault diagnosis using small datasets with deep convolutional neural network-based transfer learning, Ieee Access 6 (2018) 26241–26253.
[3] J.M. Johnson, T.M. Khoshgoftaar, Survey on deep learning with class imbalance, J. Big Data 6 (1) (2019) 1–54.
[4] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, J. Artificial Intelligence Res. 16 (2002) 321–357.
[5] H.-J. Kim, N.-O. Jo, K.-S. Shin, Optimization of cluster-based evolutionary undersampling for the artificial neural networks in corporate bankruptcy prediction, Expert Syst. Appl. 59 (2016) 226–234.
[6] H. He, E.A. Garcia, Learning from imbalanced data, IEEE Trans. Knowl. Data Eng. 21 (9) (2009) 1263–1284.
[7] B. Krawczyk, M. Woźniak, F. Herrera, On the usefulness of one-class classifier ensembles for decomposition of multi-class problems, Pattern Recognit. 48 (12) (2015) 3969–3982.
[8] B. Krawczyk, M. Galar, M. Woźniak, H. Bustince, F. Herrera, Dynamic ensemble selection for multi-class classification with one-class classifiers, Pattern Recognit. 83 (2018) 34–51.
[9] T.H. Cupertino, L. Zhao, M.G. Carneiro, Network-based supervised data classification by using an heuristic of ease of access, Neurocomputing 149 (2015) 86–92.
[10] B. Krawczyk, M. Woźniak, B. Cyganek, Clustering-based ensembles for one-class classification, Inform. Sci. 264 (2014) 182–195.
[11] D.M. Tax, R.P. Duin, Combining one-class classifiers, in: International Workshop on Multiple Classifier Systems, Springer, 2001, pp. 299–308.
[12] L. Kuncheva, Combining pattern classifiers. Hoboken, 2014.
[13] Z.-H. Zhou, Ensemble Methods: Foundations and Algorithms, Chapman and Hall/CRC, 2012.
[14] R.M. Cruz, R. Sabourin, G.D. Cavalcanti, Dynamic classifier selection: Recent advances and perspectives, Inf. Fusion 41 (2018) 195–216.
[15] B. Krawczyk, M. Woźniak, Dynamic classifier selection for one-class classification, Knowl.-Based Syst. 107 (2016) 43–53.
[16] J. Liu, Q. Miao, Y. Sun, J. Song, Y. Quan, Modular ensembles for one-class classification based on density analysis, Neurocomputing 171 (2016) 262–276.
[17] A. Kolesnikov, E. Trichina, T. Kauranne, Estimating the number of clusters in a numerical data set via quantization error modeling, Pattern Recognit. 48 (3) (2015) 941–952.
[18] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J.M. PéRez, I. Perona, An extensive comparative study of cluster validity indices, Pattern Recognit. 46 (1) (2013) 243–256.
[19] D.M. Tax, R.P. Duin, Support vector data description, Mach. Learn. 54 (1) (2004) 45–66.
[20] T. Le, D. Tran, W. Ma, D. Sharma, A theoretical framework for multi-sphere support vector data description, in: International Conference on Neural Information Processing, Springer, 2010, pp. 132–142.
[21] Y. Xiao, B. Liu, L. Cao, X. Wu, C. Zhang, Z. Hao, F. Yang, J. Cao, Multi-sphere support vector data description for outliers detection on multi-distribution data, in: 2009 IEEE International Conference on Data Mining Workshops, IEEE, 2009, pp. 82–87.
[22] S.S. Khan, M.G. Madden, A survey of recent trends in one class classification, in: Irish Conference on Artificial Intelligence and Cognitive Science, Springer, 2009, pp. 188–197.
[23] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, John Wiley & Sons, 2012.
[24] B. Schölkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, R.C. Williamson, Estimating the support of a high-dimensional distribution, Neural Comput. 13 (7) (2001) 1443–1471.
[25] D.M. Tax, R.P. Duin, Data description in subspaces, in: Proceedings 15th International Conference on Pattern Recognition. ICPR-2000, 2, IEEE, 2000, pp. 672–675.
[26] T. Kohonen, M. Schroeder, T. Huang, S.-O. Maps, Springer-Verlag New York, Inc., Secaucus, NJ, 43, (2).
[27] B. Das, D.J. Cook, N.C. Krishnan, M. Schmitter-Edgecombe, One-class classification-based real-time activity error detection in smart homes, IEEE J. Sel. Top. Sign. Proces. 10 (5) (2016) 914–923.
[28] L.M. Manevitz, M. Yousef, One-class SVMs for document classification, J. Mach. Learn. Res. 2 (Dec) (2001) 139–154.
[29] M. Koppel, J. Schler, Authorship verification as a one-class classification problem, in: Proceedings of the Twenty-First International Conference on Machine Learning, ACM, 2004, p. 62.
[30] D.H. Wolpert, The lack of a priori distinctions between learning algorithms, Neural Comput. 8 (7) (1996) 1341–1390.
[31] M. Woźniak, M. Graña, E. Corchado, A survey of multiple classifier systems as hybrid systems, Inf. Fusion 16 (2014) 3–17.
[32] A.H. Ko, R. Sabourin, A.S. Britto Jr, From dynamic classifier selection to dynamic ensemble selection, Pattern Recognit. 41 (5) (2008) 1718–1731.
[33] N. Görnitz, L.A. Lima, K.-R. Müller, M. Kloft, S. Nakajima, Support vector data descriptions and *k*-means clustering: One class? IEEE Trans. Neural Netw. Learn. Syst. 29 (9) (2017) 3994–4006.
[34] M. Charrad, N. Ghazzali, V. Boiteau, A. Niknafs, M.M. Charrad, Package 'nbclust', J. Stat. Softw. 61 (2014) 1–36.
[35] A. Jain, K. Nandakumar, A. Ross, Score normalization in multimodal biometric systems, Pattern Recognit. 38 (12) (2005) 2270–2285.
[36] T. Caliński, J. Harabasz, A dendrite method for cluster analysis, Comm. Statist. Theory Methods 3 (1) (1974) 1–27.
[37] G.W. Milligan, A Monte Carlo study of thirty internal criterion measures for cluster analysis, Psychometrika 46 (2) (1981) 187–199.
[38] J.A. Hartigan, Clustering Algorithms, Wiley, 1975.
[39] G.H. Ball, D.J. Hall, ISODATA, a novel method of data analysis and pattern classification, Tech. rep., Stanford research inst Menlo Park CA, 1965.
[40] J.O. McClain, V.R. Rao, Clustisz: A program to test for the quality of clustering of a set of objects, J. Mar. Res. (1975) 456–460.
[41] W.J. Krzanowski, Y. Lai, A criterion for determining the number of groups in a data set using sum-of-squares clustering, Biometrics (1988) 23–34.
[42] L. Kaufman, P.J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, Vol. 344, John Wiley & Sons, 2009.
[43] R. Tibshirani, G. Walther, T. Hastie, Estimating the number of clusters in a data set via the gap statistic, J. R. Stat. Soc. Ser. B Stat. Methodol. 63 (2) (2001) 411–423.
[44] J.C. Dunn, Well-separated clusters and optimal fuzzy partitions, J. Cybern. 4 (1) (1974) 95–104.
[45] M. Halkidi, M. Vazirgiannis, Y. Batistakis, Quality scheme assessment in the clustering process, in: European Conference on Principles of Data Mining and Knowledge Discovery, Springer, 2000, pp. 265–276.
[46] M. Halkidi, Y. Batistakis, M. Vazirgiannis, On clustering validation techniques, J. Intell. Inf. Syst. 17 (2–3) (2001) 107–145.
[47] L.J. Hubert, J.R. Levin, A general statistical framework for assessing categorical clustering in free recall., Psychol. Bull. 83 (6) (1976) 1072.
[48] D.L. Davies, D.W. Bouldin, A cluster separation measure, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-1 (2) (1979) 224–227.
[49] A.K. Jain, Data clustering: 50 years beyond K-means, Pattern Recognit. Lett. 31 (8) (2010) 651–666.

[50] D. Tax, Ddtools, the data description toolbox for matlab, Delft University of Technology Ed, 2005.

[51] D. Tax, Ddtools, the data description toolbox for matlab, 2018, version 2.1.3.

[52] L.I. Kuncheva, J.C. Bezdek, R.P. Duin, Decision templates for multiple classifier fusion: an experimental comparison, Pattern Recognit. 34 (2) (2001) 299–314.

[53] O. Pujol, P. Radeva, J. Vitria, Discriminant ECOC: A heuristic method for application dependent design of error correcting output codes, IEEE Trans. Pattern Anal. Mach. Intell. 28 (6) (2006) 1007–1012.

[54] J. Cohen, A coefficient of agreement for nominal scales, Educ. Psychol. Meas. 20 (1) (1960) 37–46.

[55] A. Benavoli, G. Corani, F. Mangili, Should we really use post-hoc tests based on mean-ranks, J. Mach. Learn. Res. 17 (5) (2016) 1–10.

[56] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise, in: Kdd, 96, (34) 1996, pp. 226–231.

[57] T. Wiatowski, H. Bölcskei, A mathematical theory of deep convolutional neural networks for feature extraction, IEEE Trans. Inform. Theory 64 (3) (2017) 1845–1866.