Research Article

Balanced binary search tree multiclass decomposition method with possible non-outliers

Rahul Kumar Sevakula[1,2] · Nishchal Kumar Verma[1]

Received: 7 July 2019 / Accepted: 29 April 2020 / Published online: 28 May 2020

Abstract
Multiclass decomposition algorithms are the means by which binary classification algorithms like support vector machine are used for multiclass classification problems. The popular multiclass decomposition algorithms like one against one (OAO), one against all (OAA), etc., perform the decomposition in a naive manner. This paper presents a novel heuristic-based decomposition algorithm that takes the Hausdorff distance between two classes to decide the decomposition. During the decomposition, rules are made to ensure a balanced binary search tree structure. To model the uncertainty and class noise present in the data, an unsupervised outlier detection technique has been used so that only possible non-outliers take part in the decomposition process. The presented algorithm has been evaluated and compared against OAO and OAA methods across 6 datasets. While evaluating the decomposition algorithms, fuzzy support vector machine has been used to model the class noise during each binary classification. The comparison shows that presented method not only provides comparable performance, but also in all cases, can classify the test samples with fewer average number of support vectors, thus leading to faster test performance. The paper further observes that the proposed approach can provide statistically better performance when the decomposition structure is learned only using the possible non-outliers, as compared to the scenario where the decomposition structure is learned using all samples.

Keywords Classification · Support vector machine · Multi-class classification · Decomposition approaches · Hausdorff distance · Class noise

1 Introduction

Binary classifiers have a very special place in the history of classification problems. The first classifiers were prominently binary classifiers, e.g., linear discriminant analysis. The classifiers having maximum ability to exhibit low variance are also binary classifiers, e.g., SVM and MVP classifiers [29, 34]. When we look back in history, many of the classifiers were first designed for binary classification and later extended to multiclass classification [7, 14]. Primarily, there have been two ways by which binary classifiers are made capable of solving multiclass classification problems: (1) extend the learning algorithm to a multiclass version and (2) decompose the multiclass problem into binary sub-problems. The first method can lead to computationally costly algorithms [14] or sometimes may also be impractical to formulate. The second method, on the other hand, is easy to implement, and it provides the facility of parallel processing, where the binary sub-problems could be independently solved using different processors. This paper shall focus on the second class of methods.

Lorena et al. [20] present an excellent literature survey on decomposition algorithms. Decomposition algorithms involve two steps: (1) dividing the multiclass problem into

R. K. Sevakula: Research was performed during stay at Indian Institute of Technology Kanpur, Kanpur, India.

✉ Rahul Kumar Sevakula, rahul.sevakula@gmail.com | [1]Indian Institute of Technology Kanpur, Kanpur, India. [2]Present Address: Massachusetts General Hospital, Harvard Medical School, Boston, USA.

SN Applied Sciences (2020) 2:1130 | https://doi.org/10.1007/s42452-020-2853-6

binary sub-problems and (2) combining the results of the binary classifiers to assign a class to the sample. Allwein et al. [1] proposed a code-matrix framework, represented by code-matrix $\mathbf{M}$ for obtaining insights on how multiclass problems are/could be decomposed to sub-problems. Consider a multiclass classification problem having k classes, which is decomposed to l binary sub-problems. The columns define the labeling that the k classes (groups of data samples) assume while training individual binary classifiers. Thus, $\mathbf{M}$ has the dimensions of $k \times l$. Element $m_{i,j}$ gives the expected value of jth classifier w.r.t. ith class and may assume the values of $-1, 0, +1$. '+1' indicates positive label, '−1' indicates negative label, and '0' indicates non-participation from class i. Figure 1 shows a sample code matrix. One can observe that given a value of k, the codes for individual classes can be either made with lot of redundancy or can be made compact, i.e., with little redundancy. The most compact decomposition possible for a multiclass classification problem of k classes would be $l = \log_2(k)$.

The most popular of all decomposition methods is one against one (OAO) and one against all (OAA). Sometimes these methods are also referred to as one versus one (OVO) and one versus all (OVA), respectively. OAO decomposes the multiclass classification problem to a situation where a binary classifier is trained to classify each possible pair of individual classes. Therefore, the number of binary classifiers trained would equal $\frac{k(k-1)}{2}$, where k is the number of classes. Final assignment of class is typically done by taking a majority vote across decisions taken by the individual classifiers. OAA, on the other hand, solves the problem by training a binary classifier for each class versus all remaining classes together, leading to a total of k classifiers. The individual classifiers then classify the test samples with a probability describing the belongingness of the sample to the individual classifier. Final assignment of class is typically done by finding the class assigned maximum probability. A major disadvantage of both OAO and OAA is the presence of shadow region [18], which is a region where final class assignment may turn out to be a tie. Another disadvantage of OAA approach is that each of the binary classifiers (in most cases) is trained on class imbalanced data. Accordingly, class imbalance methods are preferably employed while training each of the classifiers. Figure 2a–d presents the code matrix and illustrations for OAO and OAA, respectively.

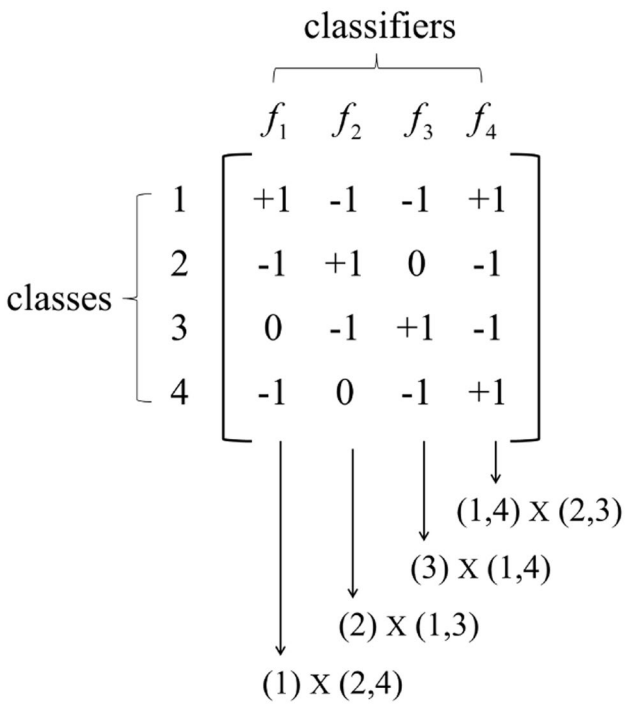

Fig. 1 Code-matrix framework to describe multiclass decomposition algorithms

Dietterich et al. [9] proposed the use of error-correcting output codes (ECOC). The idea behind ECOC is to have more number of classifiers so that some redundancy is created. This redundancy provides the facility to correct errors. Allwein et al. [1] pointed out that although the codes generated by the ECOC decomposition have good error-correcting properties, many of the binary sub-problems generated can be difficult to learn. For this reason, case studies showed that simpler decompositions like OAA and OAO often provide comparable and sometimes superior results as compared to ECOC. Further, ECOC is computationally not efficient as well [22, 24].

Another popular set of strategies are hierarchical strategies [27]. Hierarchical strategies develop a graph-like structure for the binary classifiers, and they can be further categorized as: (1) acyclic graphs and (2) directed binary trees. A key differentiating factor among the two categories is that each node (binary classifier) in the acyclic graphs is trained to classify data of 2 classes only, whereas a node for the directed binary tree is trained to classify two groups of data wherein each group may have samples from more than one classes.

Examples of acyclic graph decomposition methods are decision directed acyclic graph (DDAG) [23] and adaptive directed acyclic graph [15]. The disadvantages of DDAG as pointed by Kijsirikul et. al. [15] are as follows: (a) Results are dependent on the sequence of binary classifiers, and (b) there is a possibility of shadow region being present. Another deficiency of DDAG is that the number of classifiers evaluated is independent of the true class's position in the graph. For example, if correct class is already present in root node, it would still be evaluated with $k-1$ binary classifiers before obtaining the output. This increases the probability of error due to the buildup of cumulative error. To understand cumulative error here, consider a situation

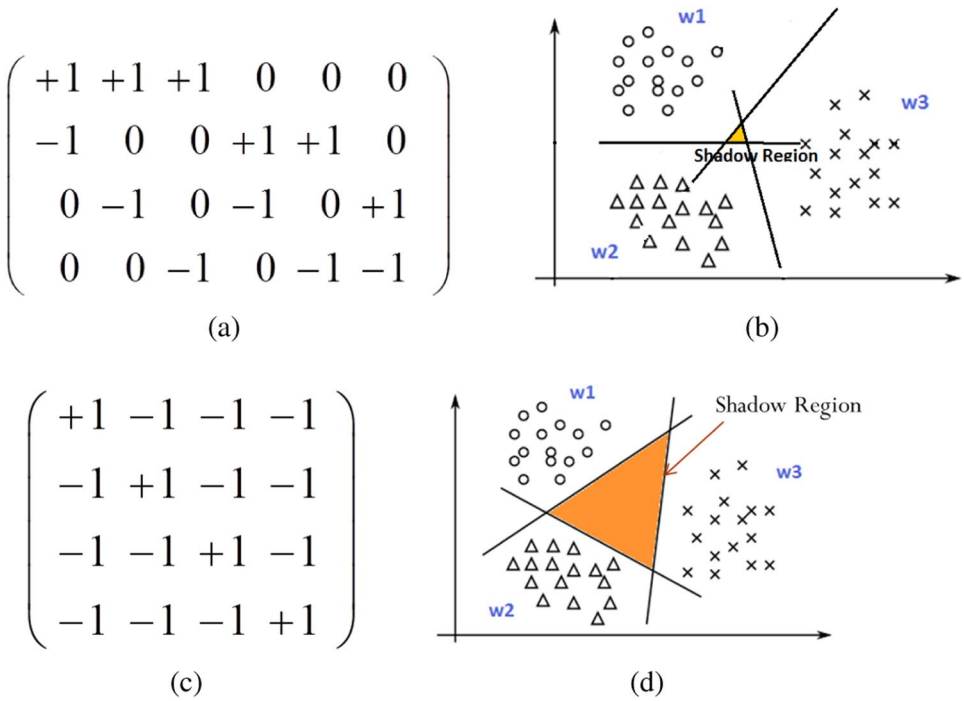

Fig. 2 Illustration: **a** code matrix for OAO, **b** illustration for OAO, **c** code matrix for OAA, **d** illustration for OAA

where each binary node/classifier has an error rate of α; the cumulative error rate for DDAG would then equal $(1-\alpha)^{k-1}$. Cumulative error therefore can become critical for data having large number of classes.

In the case of directed binary tree decomposition methods, the number of classifiers evaluated for test samples can range from $\log_2 k$ to a maximum of $k-1$binary classifiers [5, 16, 19, 31, 33, 35]. Clustering methods have been popularly used for creating the directed binary tree [16, 33, 35]. Lorena et al. [19] used minimum spanning tree for making the tree/graph. Some drawbacks of these earlier methods [16, 19, 33, 35] are: (1) Earlier methods could often lead us to having an imbalanced binary tree decomposition, and the main advantage of achieving low cumulative error and quick execution of test cases could be lost, (2) measures used for partitioning of classes could fail in simple cases, e.g., classes made up of concentric circles.

It is understood that a directed binary tree decomposition which leads to a balanced binary tree-like structure, would require only $\log_2 k$ classifiers for test sample evaluation, and such a decomposition would have the least amount of redundancy in the code matrix. This paper extends our recently introduced decomposition algorithm (introduced in conference paper [31]), which brings out a balanced binary tree decomposition of classes. For partitioning of the classes, Hausdorff distance (HD) is used to identify the classes which are most separable from each other. As these two most separable classes form two partitions, remaining classes are assigned to each partition based on their affinity to the partition w.r.t. their Hausdorff distances. Special considerations ensure that a balanced binary tree-like structure is formed, which theoretically gives least redundant code matrix.

The novelty of this paper includes the consideration for class noise. Since class noise (outliers) can pose a major challenge during decomposition and binary classification, two checks have been performed: (1) when effect of possible outliers is reduced during the decomposition process and (2) when possible outliers are appropriately modeled while training of binary classifiers. For the first check, a clustering-based approach has been used, and for the second check, fuzzy support vector machine (FSVM) [17] has been used. Elaborate experimentation on 6 datasets shows that proposed method achieves comparable classification performance with OAO and OAA. Further, the proposed method was shown to achieve the mentioned classification performance with least number of nonzero support vectors (SVs) during testing, which translates to faster test performance. It was further seen that reducing the effect of possible outliers led to improvements in classification performance on both accounts, namely in the decomposition process, as well as in the individual binary classification.

The paper further proceeds as follows: Section 2 explains the Hausdorff metric, and Sect. 3 describes the proposed decomposition algorithm. Section 4 mentions the performed experimentation, results achieved and conclusions derived from the same. Section 5 ends the paper with Conclusions.

2 Hausdorff distance

Hausdorff distance is a popular measure to find out how close two non-empty sets of a metric space are to each other [13, 25, 30]. Distance between two sets can be pictured as distance between a representative point from each set. It is important here that the two sets between whom Hausdorff distance is to be computed, be non-empty sets. Closed and bounded properties of the space are very essential as it allows one to take limits. In the absence of such restrictions, it cannot be insured that limit exists and if it exists, it will be finite. The restriction also allow us to have $d(A, B) = 0$ for non-empty sets A and B, when $A = B$, which is a necessary criteria for a metric space to be defined.

There exist two mathematical definitions for Hausdorff distance [13]. The first definition mentions that two sets are said to be close when every point of either set is close to some point in the other set. Given a compact metric space S, consider X to be the space of non-empty closed subsets of S. Hausdorff metric is defined on pairs of elements in X whose expression is given in Eq. 1. In the definitions below, $d_H(.)$ denotes the function computing the Hausdorff distance.

$$d_H(A, B) = \max\left\{\sup_{x\in A}\inf_{y\in B} d(x, y), \sup_{y\in B}\inf_{x\in A} d(x, y)\right\} \tag{1}$$

The second definition which is graphically more appealing is defined as follows: Given $A \in X$, let its ϵ-expansion be defined as the union of all ϵ-open spheroids around points in A and denoted as $E_\epsilon(A)$. Hausdorff distance between the two sets is defined as the smallest ϵ that allows the expansion of one set to cover the other and vice versa [25]. The expression is given in (2), and its illustration is shown in Fig. 3.

$$\begin{aligned} &E_\epsilon(A) = \cup_{x\in A} B(x, \epsilon) \\ &d_H(A, B) = \inf\left\{\epsilon > 0 | E_\epsilon(A) \supset B \text{ and } E_\epsilon(B) \supset A\right\} \end{aligned} \tag{2}$$

2.1 Distance between a non-empty set and a point

We present here two popular metrics to measure distance between a non-empty set and a point. Consider a metric space S with metric d. Let e and A be, respectively, a point and a non-empty closed subset of S. The definition for the first metric would be by computing the Hausdorff distance between A and the non-empty set having a single point e. The definition is mentioned below and shall be referred here as HD_{pt} distance.

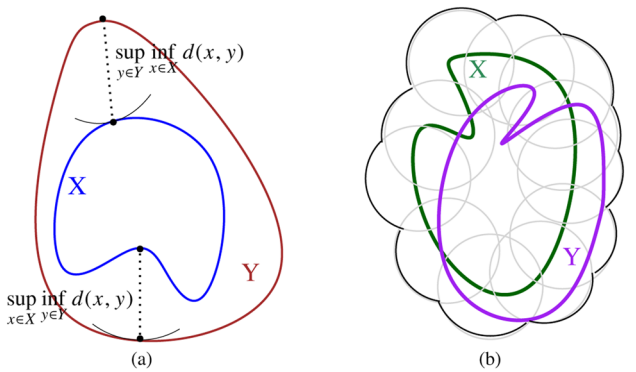

Fig. 3 Illustration to depict: **a** first definition of Hausdorff distance, **b** second definition of Hausdorff distance

$$\begin{aligned} d_H(\{e\}, A) &= max\left\{\inf_{a\in A} d(e, a), \sup_{a\in A} d(e, a)\right\} \\ &= \sup_{a\in A} d(e, a) \end{aligned} \tag{3}$$

The second metric is defined as the greatest lower bound of distances from points a in A to point e. This measure which we shall refer here as *pt-set* distance is more popular than HD_{pt} due to its greater relevance to physical reality. In the definition below, $d_P(.)$ denotes the function computing the *pt-set* distance.

$$d_P(e, A) = \inf\{d(e, a) : a \in A\} \tag{4}$$

3 Proposed method

The proposed method for decomposition is presented below in Algorithm 1. The primary concept behind the method is to find the classes that are most distant from each other and then segregate other classes into two groups based on their proximity to each of these two classes. This process is recursively repeated to bring out a binary tree-like structure during the decomposition process. Considering that it is very likely that each of the classes comprises of more than one training sample, metrics which can measure distance between two non-empty sets, could be useful for our decomposition algorithm.

Some popular options for this objective are: (1) Jaccard distance, (2) Earth mover's distance after estimating probability distributions for each of the two non-empty sets and (3) the Hausdorff distance. Jaccard distance has a limitation that when the two sets are mutually exclusive, no matter how far or close the two sets be, the Jaccard distance would always be 1. This limits the ability to compare distance between more than 2 mutually exclusive non-empty sets. The second option of finding Earth mover's

distance after estimating probability distributions can be a very computationally expensive procedure, as estimating probability distributions is not an easy task. On considering these options, the easy to compute Hausdorff distance appeared to be the most suitable metric here.

Algorithm 1: Decomposition algorithm using Hausdorff Metric

Step 1: Compute the Hausdorff distance between each pair of classes. Given k classes, we shall have $\frac{k(k-1)}{2}$ pairs of classes between which the Hausdorff distance must be found.

Step 2: Make two empty sets of classes, namely *left_node* and *right_node* (the way we have in Binary Search Trees). Find the pair of classes which are most distant from each other w.r.t. Hausdorff distance. From this pair, assign one class to *left_node* and the other to *right_node*.

Step 3: Now based on the Hausdorff distances already found between different class pairs, check the class, i.e., closest to either left node or right node and then assign it to the closer node.

Step 4: Step 3 is repeated until the left node or the right node has half of the total number of classes present in previous node. In case, the previous node has an odd number of classes, half will be rounded off to the higher integer value. The remaining classes are automatically assigned to the node which has fewer number of classes.

Step 5: Once all the classes have been divided (equally) into left node and right node, we recursively perform the Steps 2-4, to further divide each of these nodes. This recursive operation continues until the leaf nodes have only one class in them.

Following the proposed method gives a balanced binary tree-like structure. To make a multiclass decision making system, a binary classifier needs to be generated for all the nodes except for the leaf nodes, such that classes in the left node and classes in the right node are maximally separated. A pictorial presentation of how classes are divided into binary tree structure is shown in Fig. 4a, and the effective code matrix for number of classes $k = 4$ is shown in Fig. 4b.

During testing, starting from the root node which has all classes, decision of binary classifier at each node will direct us to the next node. This operation continues until a leaf node is reached, which provides the determined/predicted class. An important point to be noted is that a case of shadow region, i.e., ambiguity during final class assignment, does not exist with proposed approach.

3.1 Identifying the possible outliers

Outliers are defined as samples which appear to have been generated from a distribution other than what they are denoted to. Hausdorff distance suffers from a limitation that the distance between two non-empty sets can be seriously affected by outliers present in both classes.

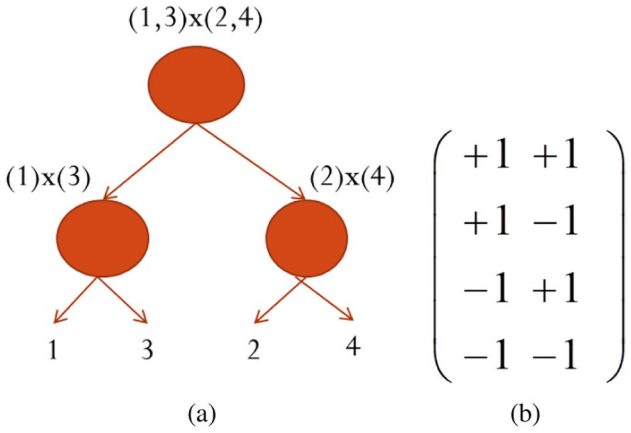

Fig. 4 **a** Partitioning into two groups of classes, **b** code matrix of proposed approach

For example, consider a situation where most samples of the two classes are close to each other, except for a single outlier which is very far from all samples of the other class; Hausdorff distance in such a case would be much larger than when it is computed in the absence of that outlier. Keeping such situations in mind, we now explore the proposed decomposition algorithm after identifying the possible outliers.

In the absence of prior knowledge, the procedure of identifying possible outliers would be entirely unsupervised. Most existing outlier identification methods [4, 21] need some form of training data containing non-outliers (for unsupervised methods) or both non-outliers and outliers (for semi-supervised methods). For this reason, when prior knowledge is absent, possible outliers are identified based on a common notion here. The notion used in this paper is that after performing clustering procedures, the samples which do not show strong belongingness to any of the clusters can be seen as possible outliers. For segregating the training samples that are possibly non-outliers and outliers, density-based clustering methods, namely DBSCAN [11] and OPTICS-OF [3], have been tested for use here. The reasons for using density-based clustering methods are as follows:

- noise objects can be identified even within broken structures and irregular shaped structures.
- prior knowledge of the number of clusters is not required.
- simple and fast to implement.

Tests on various toy datasets showed that DBSCAN is much more effective than OPTICS-OF, in identifying the possible outliers. Therefore, only DBSCAN approach was used for our possible outlier identification strategy. References

[26, 30] provides justification on how performing DBSCAN-based outlier identification on input feature space, and later performing classification in kernel space (which is used by classifiers like support vector machine), is effective in improving the classification performance. Accordingly, the procedure to identify possible outliers gets its primary ideas from [30].

Introduced by Ester et al. [11], DBSCAN is a density-based clustering method, which is designed with a notion that clusters should be formed at locations where samples are more densely connected than other locations. The density of a sample is defined as the number of samples k in its ϵ neighborhood, i.e., within its radius of ϵ. DBSCAN introduces the following concepts for explaining the algorithm. A point or an object is denoted as a "core point" when the density in its ϵ neighborhood is above a defined threshold, τ. "Border points" are those objects whose density within the ϵ neighborhood is less than τ, and which have at least one core point in each of their ϵ neighborhood. Objects/points which neither fall into the category of core points nor that of border points, are considered to be "noise objects/points."

The algorithm starts with a random sample, which is allowed to expand as a cluster, only when it is a core point. A point or cluster expands its cluster by absorbing the untouched points in its ϵ neighborhood. This expansion ends when there are no more core points or border points to absorb in their ϵ neighborhood. The same process is repeated with a random untouched core point, to form a new cluster group. This process continues until all the core points are either exhausted or are absorbed by the existing clusters, after which the clustering process is stopped. The noise points which remain unabsorbed can be treated as possible outliers. Figure 5 shows a plot of cluster formation.

Before using DBSCAN to identify the possible outliers, it is required to decide the values for parameters k and ϵ. Let *out_ratio* be defined as the ratio of number of noise objects (as found after running DBSCAN) to the total number of samples in the class, as shown in Eq. 5. Let *max_or* be a user-defined parameter that limits the maximum allowed *out_ratio*. As would be seen later, defining *max_or* would limit the number of possible outliers allowed. The algorithm begins by initializing $k = n + 1$, a common heuristic where n is the number of features or dimensions in the space. ϵ is initialized to 0.03 times the largest diameter of the ellipsoid that can encompass all the samples of the class, *diam_lar* as given in Eq. 6. In Eq. 6, *max_vec* refers to the vector having maximum value of each feature and *min_vec* refers to the vector having the minimum of each feature value among all samples of the class.

$$out\_ratio = \frac{\text{no. of samples that are probablyoutliers}}{\text{total number of data samples}} \tag{5}$$

$$diam\_lar = \|max\_vec - min\_vec\| \tag{6}$$

The algorithm proposed to identify possible outliers proceeds as follows: DBSCAN is first run with these initial parameter values and *out_ratio* is found. If the *out_ratio* found is above the estimate, then DBSCAN is rerun on the same class with modified parameters. k is remained fixed, and ϵ is revised to a new value equal to *incr_fac* times the previous ϵ, i.e., $\epsilon_{new} = incr\_fac \times \epsilon_{old}$. This process is repeated until *out_ratio* falls just below *max_or*. By the end of this procedure, two groups, namely *in_class* which refers to the set of samples that are considered to be part of the class and *out_class* which refers to the set of possible outlier samples, are formed.

The objective of defining new variables, namely *out_ratio* and *max_or*, was to substitute the decision of

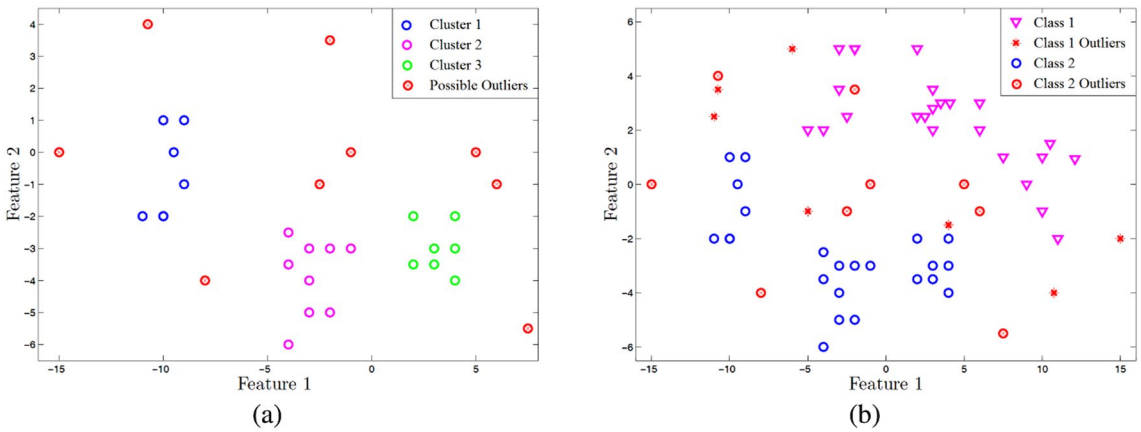

Fig. 5 **a** DBSCAN run on class 1 of a toy dataset, **b** DBSCAN run on class 1 and class 2 of a toy dataset

defining value for k, ϵ, with *max_or*, which is intuitively more easier. The user is asked to give an estimate for *max_or*, based on prior knowledge, i.e., by having an idea of how noisy the dataset is. In case there is no prior knowledge about the dataset, *max_or* can be varied between 0 and 1 in small steps and value giving best performance could be selected. The second option would obviously be a time-consuming process.

3.2 When possible outliers do not take part in the decomposition algorithm

The effectiveness and sufficiency of the above outlier detection strategy are checked on a toy problem, as shown in Fig. 6. The data of the toy problem were randomly generated between a band of concentric circles of radius 7 to 10 for class 1, and radius 0 to 2 for class 2, with center at (0, 0). It was also made sure that 10% of the samples from both classes were outliers. The problem is non-separable and would need nonlinear classification with kernel classifiers. As shown in Fig. 6b, the above-mentioned outlier detection technique is able to correctly identify the *out_class* and *in_class* groups of both classes, with user given *max_or* of 0.15.

In our experiments, in addition to the proposed decomposition method, we tried a variant where only possible non-outliers are considered while computing the Hausdorff distance between the individual classes. The idea behind the variant is that when outliers are absent during the computation of Hausdorff distance, their effect on the proposed binary tree decomposition would be significantly reduced, which is desirable. Once the decomposition of the multi-class problem to multiple binary class problems is complete, all samples (including the possible outliers) would take part during training of the binary classifiers. In other words, outliers' effect is reduced only during the decomposition procedure, and not during training of the binary classifiers.

3.3 Accounting possible outliers within binary classification

In the previous two subsections, we worked on ways on how to reduce the effect of possible outliers during the multiclass decomposition procedure. In this subsection, methods to appropriately reduce the effect of possible outliers during the binary classification, would be worked upon.

FSVM is a significant improvement over *C*-SVM which allows one to model possible outliers. FSVM allows one to assign fuzzy membership values (MVs) to each data sample. Such facility provides greater flexibility in accounting the data samples differently, w.r.t. their individual perceived importance. FSVM with appropriate fuzzy membership functions (MFs) has repeatedly shown that it can perform statistically better than regular *C*-SVM [2, 30]. For our experiments, we shall use four of the MFs introduced in our earlier paper [30], namely μ_{hd_lin}, μ_{hd_exp}, μ_{ptset_lin} and μ_{ptset_exp}.

For assigning MVs, the MFs perform the procedure of outlier detection as mentioned in Sect. 3.1, to divide the data into two groups, namely the *in_class* group and the *out_class* group. Further, Hausdorff distance between the *in_class* groups of the two classes, and *pt-set* and HD_{pt} between possible outliers and their *in_class* group are computed as shown below:

$$d_H class_{12} = d_H(in\_class_1, in\_class_2) \tag{7}$$

$$d_H classA_i = d_H(i\text{th}sample, in\_classA) \tag{8}$$

$$d_P classA_i = d_P(i\text{th}sample, in\_classA) \tag{9}$$

In Eqs. 7–9, $d_H class_{12}$ refers to the Hausdorff distance between *in_class* subset of *class*1 and *in_class* subset of *class*2. $d_H classA_i$ refers to the HD_{pt} (as derived in Eq. 3) between the *in_class* subset of class *A* and *i*th sample, where *A* is the class label of the sample. $d_P classA_i$ refers

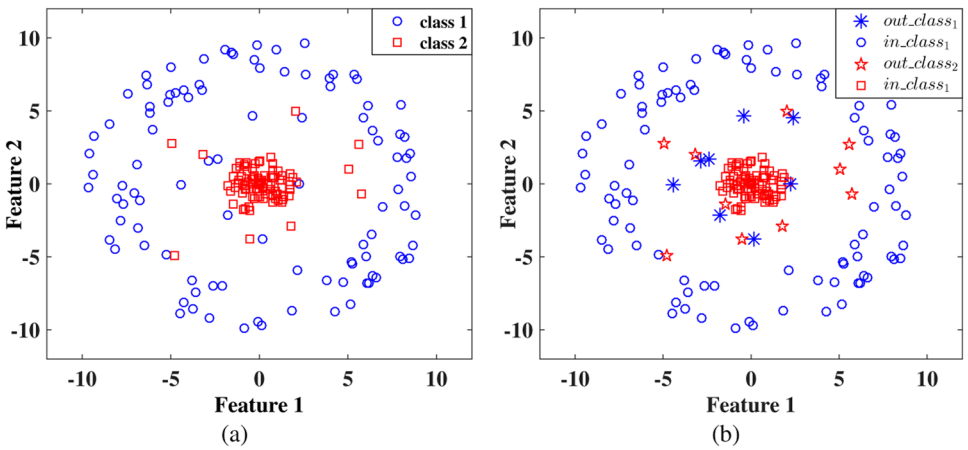

Fig. 6 Visually depicting the effectiveness of proposed methodology : **a** nonlinear and non-separable classification problem, **b** possible outliers and non-outliers identified using proposed strategy with DBSCAN

to the *pt-set* distance (as mentioned in Eq. 4) between *in_class* subset of class A and *i*th sample.

MFs are defined separately for each class of the binary classification problem. Accordingly, the procedure of identifying possible outliers and assigning them MVs is performed independently for each class. The four MFs we use here are defined below:

For $\mathbf{x_i} \in out\_class$,

$$\mu_{hd\_lin}(\mathbf{x_i}) = max\left(1 - \frac{d_H classA_i}{\lambda \times d_H class_{12}}, 0.01\right) \tag{10}$$

$$\mu_{hd\_exp}(\mathbf{x_i}) = exp\left(-\frac{\beta_{exp\_hd} \times d_H classA_i}{d_H class_{12}}\right), \quad \beta \in (0, 1] \tag{11}$$

$$\mu_{ptset\_lin}(\mathbf{x_i}) = max\left(1 - \frac{d_P classA_i}{\lambda \times d_H class_{12}}, 0.01\right) \tag{12}$$

$$\mu_{ptset\_exp}(\mathbf{x_i}) = exp\left(-\frac{\beta_{exp\_ptset} \times d_P classA_i}{d_H class_{12}}\right), \quad \beta \in (0, 1] \tag{13}$$

and for $\mathbf{x_i} \in in\_class$,

$$\begin{aligned} \mu_{hd\_lin}(\mathbf{x_i}) &= \mu^{prop}_{hd\_exp}(\mathbf{x_i}) = 1 \\ \mu_{ptset\_lin}(\mathbf{x_i}) &= \mu^{prop}_{ptset\_exp}(\mathbf{x_i}) = 1 \end{aligned} \tag{14}$$

As shown in Eq. 14, for all four MFs, the samples from *in_class* group are assigned a MV of '1', and the samples belonging to *out_class* group are assigned MVs based on heuristics mentioned in Eqs. 10–13. Equations (10)–(11) use HD_{pt}, and Eqs. (12)–(13) use *pt-set* distance for finding distance of possible outliers from their *in_class* group of samples. These distances are then normalized w.r.t. the Hausdorff distance between the *in_class* groups of the two classes. The idea behind these MFs is that if a sample is as much far from its own class as the second class is, then it is quite possible for the sample to belong to the second class. Hence, the sample's belongingness to the class can be doubted, and it may be given low MV. Heuristics $\mu_{hd\_lin}(\mathbf{x_i})$ and $\mu_{ptset\_lin}(\mathbf{x_i})$ decay linearly with HD_{pt} and *pt-set* distance, respectively, and parameter λ decides the extent of decay. Heuristics $\mu_{hd\_exp}(\mathbf{x_i})$ and $\mu_{ptset\_exp}(\mathbf{x_i})$ decay exponentially with HD_{pt} and *pt-set* distance, respectively, and parameter β decides the extent of decay.

4 Case study and results

Experiments were performed on six datasets from the UCI repository [10] to infer the conclusions. The purpose of the experiments was threefold: (i) to see how proposed decomposition algorithm compares to other traditional decomposition methods like OAA and OAO, (ii) to observe the effect of using only possible non-outliers to perform the decomposition and (iii) effect of using FSVM with outlier detection embedded in the membership functions for reducing the effect of outliers in the training of individual binary classifiers.

Details of the six experimented datasets are reported in Table 1. The first five datasets shown in Table 1 do not have a separate test set. Accordingly, the classifiers' performances for these five datasets were evaluated based on tenfold cross-validation (we call this as external cross-validation performance), and the mean classification accuracy across the tenfold was reported. The Segment dataset had a distinct training and test dataset; accordingly, classifiers were trained on the training data, and their performance was evaluated and reported on the test data.

For all experiments, SVM and FSVM have been used as the binary classifiers. The experiments were performed on MATLAB, and LibSVM [30] was used for training each of binary SVM classifiers. Radial basis function (RBF) kernel was used in all the experiments. For tuning the hyperparameters in SVM, a grid search-based procedure was used in each of the experiments. The value for γ was logarithmically searched across powers of 2, from 2^{-10} to 2^4, and C was searched across powers of 2, from 2^{-2} to 2^{12}, i.e., across a total of 225 pairs of (C, γ) values. For identifying the optimal parameter values, the fivefold cross-validation (internal cross-validation) performance on the training data was ascertained with each parameter pair; and the pair giving best accuracy value was considered as optimal. While using FSVM, the parameters *max_or*, λ and β wherever applicable, were also tuned by searching values across $\{0.1, 0.2, \ldots, 0.9\}$, $\{1, 2, 3\}$, and $\{0.1, 0.2, \ldots, 1.0\}$, respectively. The parameter values

Table 1 Details of benchmark datasets

Datasets	# Train samples	# Test samples	# Classes	# Features
Iris	150	0	3	4
Glass	214	0	6	13
Vowel	528	0	11	10
Vehicle	846	0	4	18
Segment	2310	0	7	19
Optical	3823	1797	10	64

Table 2 Results—classification error (in %)

Datasets	C-SVM				FSVM-μ_{hd_lin}				FSVM-μ_{hd_exp}				FSVM-μ_{ptset_lin}				FSVM-μ_{ptset_exp}			
	OAA	OAO	Prop.	PropO	OAA	OAO	Prop.	PropO	OAA	OAO	Prop.	PropO	OAA	OAO	Prop.	PropO	OAA	OAO	Prop.	PropO
Iris	4.00	4.00	4.00	4.00	2.00	2.67	2.67	2.67	2.67	2.67	2.67	2.67	2.67	3.33	3.33	3.33	2.67	3.33	3.33	3.33
Glass	32.20	33.17	33.63	26.63	31.27	28.47	29.40	27.59	30.81	27.08	28.95	27.53	31.28	29.42	29.87	27.10	31.27	28.48	30.33	26.64
Vowel	43.51	33.33	35.93	35.50	43.51	33.12	35.93	35.93	43.29	33.33	35.93	35.93	43.51	33.33	35.71	35.93	43.51	33.33	35.93	35.93
Vehicle	15.72	14.42	17.73	14.31	14.90	15.25	16.67	15.01	14.78	15.13	17.85	14.42	14.66	14.31	17.02	14.18	14.42	14.54	17.38	14.66
Segment	8.95	8.81	10.38	9.29	8.71	8.24	9.81	9.10	8.52	8.19	9.19	9.14	8.62	8.00	9.57	9.14	8.57	8.24	10.14	9.14
Optical	1.22	2.06	2.06	1.61	1.22	1.95	2.06	1.56	1.28	1.95	2.06	1.56	1.17	2.00	2.06	1.56	1.22	2.00	2.06	1.56

Smaller error indicates better performance

which gave the best overall external cross-validation accuracy values were selected, and these same accuracy values are reported in Table 2.

There are three measures across which the performance of the decomposition methods has been evaluated: (i) classification performance as measured by classification error, (ii) average number of nonzero support vectors required to pass through, before determining the class, and (iii) sum of the execution times during training and testing. The decision function for SVM is given below where x refers to a test data sample, n refers to the number of training samples in training SVM, α^* refers to optimal value of SVs found after training, b^* refers to optimal value of b found with training, and $K()$ refers to the kernel function.

$$f(x, \alpha^*, b^*) = \sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + b^* \tag{15}$$

The computational complexity in classifying test samples is directly and linearly proportional to the number of nonzero SVs α^* [6, 28]. Therefore, the second measure of average number of nonzero SVs indicates the speed at which classification would take place during testing.

The three measures as reported for each of the experiments are shown in Tables 2, 3 and 4, respectively. In all the tables, 'Prop' is used to refer to the proposed decomposition, and 'PropO' is used to refer to the case when only the possible non-outliers take part in deriving the decomposition structure with proposed decomposition algorithm.

We now perform statistical tests to infer conclusions from the available results of six datatsets. Some commonly used statistical tests, which we use here to compare classifier performance across multiple datasets/cases, are: (i) the Friedman test to check for Null Hypothesis and (ii) the Bonferroni–Dunn test [8, 32]. While statistically comparing two or more classifiers, the Null Hypothesis tests checks if the differences in classifier performances are by chance or if they have any statistical significance. Null Hypothesis states that the differences are only by chance. Using the exact performance values for statistical analysis can create limitations and doubts on the commensurability of the performance measure [32]. For this reason, a more viable method is to rank the classifiers for the given dataset/case (wherein better performance is awarded a lower rank) and then use the average of ranks taken across all datasets/cases, to perform statistical analyses.

The Friedman test is performed by computing the Friedman statistic χ_F^2 as shown below. As seen from the below equation, χ_F^2 depends on the number of datasets/cases N, the degrees of freedom l which equals the number of classifier being compared, and the average ranks of classifiers r_j. When the computed χ_F^2 is greater than a defined critical value, the Null Hypothesis is rejected; otherwise, it

SN Applied Sciences (2020) 2:1130 | https://doi.org/10.1007/s42452-020-2853-6

Table 3 Average number of support vectors used per class during testing

Datasets	*C*-SVM				FSVM-μ_{hd_lin}				FSVM-μ_{hd_exp}				FSVM-μ_{ptset_lin}				FSVM-μ_{ptset_exp}			
	OAA	OAO	Prop.	PropO	OAA	OAO	Prop.	PropO	OAA	OAO	Prop.	PropO	OAA	OAO	Prop.	PropO	OAA	OAO	Prop.	PropO
Iris	42	41	26	26	44	43	28	28	45	38	24	24	42	40	26	26	39	38	24	24
Glass	234	249	98	77	225	232	89	76	234	227	102	70	237	233	85	66	237	225	85	72
Vowel	602	1614	184	160	602	1614	184	184	603	1614	184	184	602	1614	184	184	602	1614	184	184
Vehicle	573	392	283	191	592	400	272	181	580	391	261	175	585	400	288	177	582	396	283	191
Segment	243	389	64	79	284	424	88	87	288	424	84	84	244	421	74	84	250	453	64	84
Optical	3217	10029	2964	1973	2906	10697	2964	1996	3062	10234	2964	1996	3282	10421	2964	1996	2905	11163	2964	1996

Smaller number is preferable

Table 4 Overall execution time during training and testing (in s)

Datasets	*C*-SVM				FSVM-μ_{hd_lin}				FSVM-μ_{hd_exp}				FSVM-μ_{ptset_lin}				FSVM-μ_{ptset_exp}			
	OAA	OAO	Prop.	PropO	OAA	OAO	Prop.	PropO	OAA	OAO	Prop.	PropO	OAA	OAO	Prop.	PropO	OAA	OAO	Prop.	PropO
Iris	9	4	4	4	8	4	4	4	8	5	4	4	9	5	4	4	8	5	4	4
Glass	77	36	41	30	66	35	32	28	67	45	33	33	76	35	52	37	55	33	36	6087
Vowel	95	32	24	28	113	37	25	28	106	36	26	31	105	36	26	28	99	33	22	28
Vehicle	943	399	5348	365	908	336	478	367	823	359	500	349	972	398	680	407	5632	402	683	421
Segment	122	72	51	41	158	95	47	53	132	78	43	49	143	87	49	47	128	89	70	53
Optical	18276	1657	7505	5997	19136	2324	8426	3240	27868	1714	6999	3117	13212	1890	2717	6673	20500	2309	3174	7923

Smaller number is preferable

is understood that Null Hypothesis is true and the differences in classifier performance are by chance. The critical values for Friedman test are unique to the degrees of freedom l and the desired significance level α; these can be found from Table A4 of [32].

$$\chi_F^2 = \frac{12N}{l(l+1)}\left[\sum_{j=1}^{l} r_j^2 - \frac{l(l+1)^2}{4}\right] \tag{16}$$

If Null Hypothesis is rejected, we perform Bonferroni–Dunn (BD) post hoc test to make individual comparisons across algorithms and check if one statistically performs better than the other [8]. Given the degrees of freedom l and the desired significance level α, the BD post hoc test provides a statistic called critical difference CD_α. The statistic signifies that any classifier algorithm can be considered as statistically superior to another at α, if the difference in their performance average ranks is greater than CD_α. CD_α can be computed using the below equation; the value of q_α for the equation can be taken from Table 5(b) of [8].

$$CD_\alpha = q_\alpha\sqrt{\frac{l(l+1)}{6N}} \tag{17}$$

We now perform statistical tests to see: (i) how the proposed decomposition approach performs in comparison with OAO and OAA decomposition approaches, (ii) how "PropO" (proposed decomposition performed while accounting only the possible non-outliers) compares to Prop (proposed decomposition performed while accounting for all samples), (iii) how FSVM compares with C-SVM, and (iv) how the OAO, OAA and the proposed

Table 5 Statistical analysis to compare OAO, OAA and proposed decomposition method

Classifier	Dataset	Test error (in %)			Ranks (lower error is better)		
		OAA	OAO	PropO	OAA	OAO	PropO
C-SVM	Iris	4.00	4.00	4.00	2	2	2
	Glass	32.20	33.17	26.63	2	3	1
	Vowel	43.51	33.33	35.50	3	1	2
	Vehicle	15.72	14.42	14.31	3	2	1
	Segment	8.95	8.81	9.29	2	1	3
	Optical	1.22	2.06	1.61	1	3	2
FSVM-μ_{hd_lin}	Iris	2.00	2.67	2.67	1	2.5	2.5
	Glass	31.27	28.47	27.59	3	2	1
	Vowel	43.51	33.12	35.93	3	1	2
	Vehicle	14.90	15.25	15.01	1	3	2
	Segment	8.71	8.24	9.10	2	1	3
	Optical	1.22	1.95	1.56	1	3	2
FSVM-μ_{hd_exp}	Iris	2.67	2.67	2.67	2	2	2
	Glass	30.81	27.08	27.53	3	1	2
	Vowel	43.29	33.33	35.93	3	1	2
	Vehicle	14.78	15.13	14.42	2	3	1
	Segment	8.52	8.19	9.14	2	1	3
	Optical	1.28	1.95	1.56	1	3	2
FSVM-μ_{ptset_lin}	Iris	2.67	3.33	3.33	1	2.5	2.5
	Glass	31.28	29.42	27.10	3	2	1
	Vowel	43.51	33.33	35.93	3	1	2
	Vehicle	14.66	14.31	14.18	3	2	1
	Segment	8.62	8.00	9.14	2	1	3
	Optical	1.17	2.00	1.56	1	3	2
FSVM-μ_{ptset_exp}	Iris	2.67	3.33	3.33	1	2.5	2.5
	Glass	31.27	28.48	26.64	3	2	1
	Vowel	43.51	33.33	35.93	3	1	2
	Vehicle	14.42	14.54	14.66	1	2	3
	Segment	8.57	8.24	9.14	2	1	3
	Optical	1.22	2.00	1.56	1	3	2
	Sum of Ranks				61	58.5	60.5
	Average Rank				2.03	1.95	2.02

decomposition approach compare against each other, w.r.t. the number of nonzero support vectors in the trained classifier.

For deciding how proposed decomposition approach works in comparison with OAA and OAO, let us compare and rank the classifier performances of OAA, OAO and PropO, with each classifier, namely C-SVM, F-SVM with MF μ_{hd_lin}, F-SVM with MF μ_{hd_exp}, F-SVM with MF μ_{ptset_lin}, and F-SVM with MF μ_{ptset_exp}. The ranking for these decomposition methods, as observed with each dataset and classifier, is shown in Table 5. The sum of all ranks for OAA, OAO and PropO is 61, 58.5 and 60.5, respectively; this gives an average rank of 2.03, 1.95 and 2.02, respectively. Performing the Friedman test, we get

$$\chi_F^2 = \frac{12 \times 30}{3(3+1)}\left[2.03^2 + 1.95^2 + 2.02^2 - \frac{3 \times (3+1)^2}{4}\right] = 0.114$$

As per [32], for $l = 3$ and $N = 30$, the threshold required to reject Null Hypothesis at $\alpha = 0.05$, is 7.81. Since χ_F^2 is below the threshold value, Null Hypothesis is true. With this, we can infer that the proposed decomposition method provides comparable performance to those of OAA and OAO decomposition methods, and none of the approaches can be considered superior w.r.t. test classification scores.

For the next analysis, we shall compare the proposed decomposition methods when: (1) applied on all samples, Prop, and (2) applied only on the possible non-outliers, PropO. Similar to the earlier analysis, we shall compare the performances of Prop and PropO, with each classifier, namely C-SVM, F-SVM with MF μ_{hd_lin}, F-SVM with MF μ_{hd_exp}, F-SVM with MF μ_{ptset_lin}, and F-SVM with MF μ_{ptset_exp}. In this paper, including additional tables to compare the performances and rank the methods like in Table 5, would make the paper too long. For this reason, without mentioning all details, we directly mention the sum of all ranks, and the average ranks for each of the individual methods.

The sum of ranks across $N = 30$ for Prop and PropO is 55 and 35, respectively; this gives us average ranks of 1.83 and 1.17, respectively. Thus, $\chi_F^2 = \frac{12 \times 30}{2 \times 3}[1.83^2 + 1.67^2 - 4.5] = 13.2$. For $l = 2$ and $N = 30$, the threshold required at $\alpha = 0.05$ to reject Null Hypothesis is 5.99. Since the χ_F^2 value is greater than the desired threshold, Null Hypothesis is rejected. Now we shall perform the BD post hoc test. q_α for $l = 2$ and $\alpha = 0.05$ is 1.96. Accordingly, the critical difference value would be $CD_{0.05} = 1.96 \times \sqrt{\frac{2(2+1)}{6 \times 30}} = 0.358$. The difference between the average ranks of PropO and Prop is $(1.83 - 1.17) = 0.66$, which is greater than $CD_{0.05}$. It can be therefore statistically ascertained that PropO is better than Prop, i.e., proposed decomposition method truly performs better when the decomposition procedure is performed only with the possible non-outliers. This is a very important inference for us and indicates that the unsupervised method for identifying possible outliers and possible non-outliers, works pretty well. Furthermore, it indicates that our hypothesis of using only possible non-outliers for the decomposition process has been effective. In fact, other heuristics-based decomposition techniques could also benefit from this finding, and one could try using only the possible non-outliers for the specific decomposition.

To compare the performance of FSVM with C-SVM, we took the best performance of FSVM across all MFs, as found for each dataset and each decomposition strategy and compared them with the respective C-SVM performance. Six datasets and four decomposition strategies give us a total of $N = 24$ cases. Ranking the performance of C-SVM and FSVM, we found the sum of ranks to be 45.5 and 26.5, respectively; this gives us average ranks of 1.896 and 1.104, respectively. Performing Friedman test, the χ_F^2 value is $\chi_F^2 = \frac{12 \times 24}{2 \times 3}[1.896^2 + 1.104^2 - 4.5] = 15.07$. We know the threshold required for $l = 2$ at $\alpha = 0.05$ to reject Null Hypothesis is 5.99; hence, Null Hypothesis can be rejected at $\alpha = 0.05$. To see if FSVM is statistically better than C-SVM, we shall perform the BD post hoc test. q_α for $l = 2$ and $\alpha = 0.05$ is 1.96. Accordingly, the critical difference value would be $CD_{0.05} = 1.96 \times \sqrt{\frac{2(2+1)}{6 \times 24}} \approx 0.4$. The difference between the average ranks of C-SVM and F-SVM is $(1.896 - 1.104) = 0.892$, which is greater than $CD_{0.05}$. We can therefore statistically ascertain with $\alpha = 0.05$ that FSVM, with the tested MFs, gives better performance than C-SVM. It may be noted that this inference is in line with the conclusion mentioned in [30].

Finally, we compare the average number of nonzero support vectors required by the SVM classifier when trained with each decomposition methods, before determining the class of a sample. This statistical analysis would be very similar to the analysis performed in Table 5 of this paper; the only difference is that we would compare the number of nonzero support vectors mentioned in Table 3 for OAA, OAO and PropO methods. Ranking the methods for $N = 30$ cases, we find the sum of ranks for OAA, OAO and PropO to be 73, 77 and 30, respectively, which give the average ranks of 2.43, 2.57 and 1.0, respectively. Performing the Friedman test, we get $\chi_F^2 = \frac{12 \times 30}{3 \times 4}[2.43^2 + 2.57^2 + 1^2 - 12] = 45$. The threshold required for $l = 3$ and $\alpha = 0.05$ is 7.81. Since the χ_F^2 value is greater than the threshold, Null Hypothesis can be rejected. q_α for $l = 3$ and $\alpha = 0.05$ is 2.343. On performing the BD post hoc, we arrive at a critical difference value of

$CD_{0.05} = 2.343 \times \sqrt{\frac{3(3+1)}{6 \times 30}} \approx 0.61$. The difference between the average rank of PropO and OAA is 1.43, and difference between PropO and OAO is 1.57. From the BD test, we can statistically ascertain with $\alpha = 0.05$ that PropO provides fewer nonzero support vectors than both OAO and OAA.

This is another very important inference. As mentioned in Eq. (15), it has direct implication on the computational complexity during test phase of the classifier. We can therefore expect that the proposed decomposition technique is quicker in execution during test phase. It should be noted that the quicker test execution is achievable with the proposed decomposition technique, while ensuring a classification performance which is comparable to that of OAA and OAO (as learned from the first statistical analysis). Interestingly, we also see that PropO, in 27 of $N = 30$ cases, has an equal/fewer number of nonzero support vectors than Prop decomposition.

A non-conclusive observation which is relevant for the tested datasets, is that except for the 'Optical' dataset, the total execution time for training and testing together is always lowest in the case of proposed decomposition approach. In the case of 'Optical' dataset, the execution time for train and test is best with 'OAO' approach, and second best with 'PropO' approach. To understand this point, let us compare the training time complexity for different decomposition algorithms. The worst case time complexity for training a binary kernel SVM classifier with n data samples is $O(n^3)$, and the average time complexity is $O(n^2)$ [28]. Consider a case where we have n samples which are equally divided amongst k classes. In the case of OAA decomposition method, all n samples take part in forming k classifiers (one for each class). Therefore, the training time complexity for OAA would be $O(kn^2)$. In the case of OAO, with all classes having an equal number of samples (we stated this assumption while defining the case) of $\frac{n}{k}$, a total of $\frac{2n}{k}$ samples will take part to generate a single SVM classifier. Further, OAO requires $\frac{k(k-1)}{2}$ such classifiers. Therefore, the time complexity for training SVM with OAO decomposition method would be

$$
\begin{aligned}
&O\left(\left(\frac{2n}{k}\right)^2 * \frac{k(k-1)}{2}\right) \\
&\Rightarrow O\left(\frac{4n^2}{k^2} * \frac{k^2-k)}{2}\right) \\
&\Rightarrow O\left(2n^2\left(1-\frac{1}{k}\right)\right)
\end{aligned}
$$

Now let us consider the case of proposed decomposition method which leads to a balanced binary tree (say a complete balanced binary tree). For the first/root node, all n samples would take part in training a single SVM classifier, which requires $O(n^2)$ operations on average. In the second level, two binary classifiers would be trained with $\frac{n}{2}$ samples each. Similarly, for each lower tth level, 2^{t-1} classifiers would be trained with $\frac{n}{2^{t-1}}$ samples each. This leads us to the following

$$
\begin{aligned}
&O\left(n^2 + 2 * \left(\frac{n}{2}\right)^2 + \vdots + 2^{\log_2 k-1} * \left(\frac{n}{2^{\log_2 k-1}}\right)^2\right) \\
&\Rightarrow O\left(n^2\left(1 + \frac{1}{2} + \vdots + \frac{1}{2^{\log_2 k-1}}\right)\right) \\
&\Rightarrow O\left(n^2\left(\frac{1-(\frac{1}{2})^{\log_2 k}}{1-\frac{1}{2}}\right)\right) \\
&\Rightarrow O\left(2n^2\left(1-\frac{1}{k}\right)\right)
\end{aligned}
$$

In the above time complexity for proposed decomposition method, if we even include the time complexity for computing the Hausdorff distance between individual classes, then total time complexity would turn out to be

$$
\begin{aligned}
&O\left(2n^2\left(1-\frac{1}{k}\right)\right) + O\left(\left(\frac{2n}{k}\log\frac{2n}{k}\right) * \frac{k(k-1)}{2}\right) \\
&\Rightarrow O\left(2n^2\left(1-\frac{1}{k}\right)\right) + O\left(n(k-1)\log\frac{2n}{k}\right) \\
&\approx O\left(2n^2\left(1-\frac{1}{k}\right)\right), \text{when } k << n
\end{aligned}
$$

Further, if we consider the time complexity of identifying possible outliers and non-outliers from each class, the additional time complexity based on [30] would be $O(2n \log \frac{2n}{k})$, which is negligible compared to $O\left(2n^2\left(1-\frac{1}{k}\right)\right)$. It may be observed from Table 4 that while the time complexity of proposed decomposition method is similar to that of OAO, in reality, it is observed that proposed decomposition method takes lesser time to train and test for smaller sized datasets. This is expected because the constant time operations required during training of each binary SVM classifier, would play a big role for smaller sized datasets. OAO has many more binary SVM classifiers to be trained than the proposed decomposition method; thus, OAO ends up taking larger execution time in smaller sized datasets.

To compare the proposed approach with other directed binary tree decomposition methods, our results with

Table 6 Comparing directed binary tree decomposition methods (in %)

Datasets	Proposed method	Method 1 centroid distance	Method 2 balanced subsets	Method 3 scatter measure
Sat. image	91.80	90.90	91.90	91.80
Optical	98.39	96.50	96.90	96.20

C-SVM have been compared with those presented in Lorena et al.'s paper [19] (which followed a similar experimentation procedure). The comparison is presented in Table 6. Looking at the table, it seems that the proposed method provides either comparable or superior performance to that of the methods presented in [19]. Further, an added advantage of the proposed approach is that it assures a balanced binary tree-like decomposition structure, which leads to faster execution speed during testing.

4.1 Limitations and possible future work

This paper proposed a method which decomposes a multi-class classification problem to binary classification problems with balanced binary search tree structure, and we found that it execute quickly during testing. Arguably, there are multiple ways to decompose the multi-class classification problem to a balanced binary search tree-like structure, and this paper is limited to describing and testing only one heuristic-based method. An important future work would be to explore other heuristic-based decomposition methods, which ensure a balanced binary search tree structure, and statistically find which method does best. Another limitation of the proposed method is that optimizing the hyper-parameters of FSVM during classifier training, can be computationally very expensive. Hence, an important avenue for future research would be to develop heuristic-based methods that can quickly (i.e., in fewer iterations) bring out a reasonable set of hyper-parameters for classifier training. It is known from the literature [12] that the 0-1 loss function, the sigmoid loss function and the ramp loss function can provide good robustness to class noise; accordingly, in the future, it would be a good idea to compare the multi-class classification performance over noisy data, with SVM trained over the different loss functions and that with FSVM.

5 Conclusions

This paper extended our recently proposed Hausdorff distance-based decomposition algorithm, by studying the effect of excluding possible non-outliers during decomposition process and/or using fuzzy SVM as the binary classifier. Two studies on use of outlier detection techniques in this scenario were made while keeping class noise into consideration. The first study compared the performance of proposed decomposition approach, when performing the decomposition procedure on: (1) all data and (2) only the possible non-outliers. A second study compared the multiclass classification performances of the decomposition approaches, while using *C*-SVM or FSVM as the binary classifier. The objective of both studies was to see if there is an improvement in overall classification performance when effect of class noise is reduced with outlier detection techniques. The study concluded that proposed approach statistically performs better when only possible non-outliers were considered while deriving the decomposition. The study also indicated that the unsupervised method used for identifying possible outliers and possible non-outliers, works well. The study then suggested that other decomposition techniques, which use heuristics to determine the decomposition structure, can benefit from this finding, and that they could try using only the possible non-outliers for the learning of decomposition structure. The conclusion from the second study conformed with our earlier paper, which mentioned that FSVM with appropriate MFs can provide statistically better classification results than *C*-SVM.

The proposed decomposition approach was elaborately evaluated and compared with the OAO and OAA decomposition methods. Inferring from the results, the proposed approach statistically required fewer nonzero support vectors to ascertain the class during testing, and comparable performance, to those of OAA and OAO. As expected theoretically, testing speed of the proposed method was empirically observed to be faster than OAO and OAA methods for all tested datasets. Theoretically speaking, execution time for training SVMs with OAO is expected to be lower than that of the proposed approach. However, during our experiments, it was found that for most datasets, combined training and testing time with SVMs using proposed approach is lesser than that with the OAO method. This is because OAO has more number of classifiers, and more number of constant time operations to perform, which especially matter for smaller sized training data. The proposed approaches have an added advantage that there is zero ambiguity during final class assignment, i.e., without shadow region. Possible future work could include the use of hierarchical clustering techniques on individual classes w.r.t. the Hausdorff distances between them, for deriving the binary tree decomposition.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Research involving human participants and/or animals This article does not contain any studies with human participants or animals performed by any of the authors.

Informed consent This article does not contain any studies with human participants.

References

1. Allwein EL, Schapire RE, Singer Y (2000) Reducing multiclass to binary: a unifying approach for margin classifiers. J Mach Learn Res 1(DEc):113–141
2. Batuwita R, Palade V (2010) FSVM-CIL: fuzzy support vector machines for class imbalance learning. IEEE Trans Fuzzy Syst 18(3):558–571
3. Breunig MM, Kriegel HP, Ng RT, Sander J (1999) Optics-of: identifying local outliers. In: Principles of data mining and knowledge discovery, Springer, pp 262–270
4. Breunig MM, Kriegel HP, Ng RT, Sander J (2000) LOF: identifying density-based local outliers. ACM SIGMOD Record, ACM 29:93–104
5. Chen J, Wang C, Wang R (2008) Combining support vector machines with a pairwise decision tree. IEEE Geosci Remote Sens Lett 5(3):409–413
6. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
7. Crammer K, Singer Y (2002) On the learnability and design of output codes for multiclass problems. Mach Learn 47(2–3):201–233
8. Demsar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
9. Dietterich TG, Bakiri G (1995) Solving multiclass learning problems via error-correcting output codes. J Artif Intell Res 2:263–286
10. Dua D, Graff C (2019) UCI machine learning repository http://archive.ics.uci.edu/ml. University of California, School of Information and Computer Science, Irvine, CA
11. Ester M, Kriegel HP, Sander J, Xu X (1996) A density based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of international conference on knowledge discovery and data mining (KDD'96), pp 226–231
12. Ghosh A, Manwani N, Sastry PS (2015) Making risk minimization tolerant to label noise. Neurocomputing 160:93–107
13. Henrikson J (1999) Completeness and total boundedness of the hausdorff metric. MIT Undergrad J Math 1:69–80
14. Hsu CW, Lin CJ (2002) A comparison of methods for multiclass support vector machines. IEEE Trans Neural Netw 13(2):415–425
15. Kijsirikul B, Ussivakul N (2002) Multiclass support vector machines using adaptive directed acyclic graph. In: Proceedings of the 2002 international joint conference on neural networks, 2002. IJCNN'02. IEEE, vol 1, pp 980–985
16. Lei H, Govindaraju V (2005) Half-against-half multi-class support vector machines. In: International workshop on multiple classifier systems. Springer, pp 156–164
17. Lin CF, Wang SD (2002) Fuzzy support vector machines. IEEE Trans Neural Netw 13(2):464–471
18. Liu B, Hao Z, Tsang ECC (2008) Nesting one-against-one algorithm based on SVMs for pattern classification. IEEE Trans Neural Netw 19(12):2044–2052
19. Lorena AC, de Carvalho AC (2005) Minimum spanning trees in hierarchical multiclass support vector machines generation. In: International conference on industrial engineering and other applications of applied intelligent systems. Springer, pp 422–431
20. Lorena AC, De Carvalho AC, Gama JMP (2008) A review on the combination of binary classifiers in multiclass problems. Artif Intell Rev 30(1–4):19
21. Perdisci R, Gu G, Lee W (2006) Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems. In: IEEE sixth international conference on data mining (ICDM'06), pp 488–498
22. Pimenta E, Gama J (2005) A study on error correcting output codes. In: Portuguese conference on artificial intelligence, 2005. epia 2005. IEEE, pp 218–223
23. Platt JC, Cristianini N, Shawe-Taylor J (2000) Large margin dags for multiclass classification. In: Advances in neural information processing systems, pp 547–553
24. Pujol O, Radeva P, Vitria J (2006) Discriminant ECOC: a heuristic method for application dependent design of error correcting output codes. IEEE Trans Pattern Anal Mach Intell 28(6):1007–1012
25. Scharf L (2003) Computing the Hausdorff distance between sets of curves
26. Scholkopf B, Mika S, Burges CJ, Knirsch P, Muller KR, Ratsch G, Smola AJ (1999) Input space versus feature space in kernel-based methods. IEEE Trans Neur Netw 10(5):1000–1017
27. Schwenker F (2000) Hierarchical support vector machines for multi-class pattern recognition. In: Proceedings of Fourth international conference on knowledge-based intelligent engineering systems and allied technologies. vol 2, IEEE, pp 561–565
28. Sevakula RK, Verma NK (2012) Support vector machine for large databases as classifier. In: International conference on swarm, evolutionary, and memetic computing. Springer, pp 303–313
29. Sevakula RK, Verma NK (2017) Assessing generalization ability of majority vote point classifiers. IEEE Trans Neural Netw Learn Syst 28(12):2985–2997
30. Sevakula RK, Verma NK (2017) Compounding general purpose membership functions for fuzzy support vector machine under noisy environment. IEEE Trans Fuzzy Syst 25(6):1446–1459
31. Sevakula RK, Verma NK (2017) Hausdorff distance based binary search tree multiclass decomposition algorithm. In: 2017 Intelligence, Springer, pp 239–249
32. Sheskin DJ (2003) Handbook parametric nonparametric statistical procedures. CRC Press, Boca Raton
33. Takahashi F, Abe S (2002) Decision-tree-based multiclass support vector machines. In: Proceedings of the 9th international conference on neural information processing, 2002. ICONIP'02. vol 3, IEEE, pp 1418–1422
34. Vapnik VN (1999) An overview of statistical learning theory. IEEE Trans Neural Netw 10(5):988–999
35. Vural V, Dy JG (2004) A hierarchical method for multi-class support vector machines. In: Proceedings of the twenty-first international conference on machine learning. ACM, p 105