Pattern Recognition 48 (2015) 3969–3982

Contents lists available at ScienceDirect

Pattern Recognition

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

# On the usefulness of one-class classifier ensembles for decomposition of multi-class problems

Bartosz Krawczyk [a,*], Michał Woźniak [a], Francisco Herrera [b,c]

[a] Department of Systems and Computer Networks, Wroclaw University of Technology, Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland
[b] Department of Computer Science and Artificial Intelligence, University of Granada, P.O. Box 18071, Granada, Spain
[c] Faculty of Computing and Information Technology - North Jeddah, King Abdulaziz University, 21589 Jeddah, Saudi Arabia

ARTICLE INFO

*Article history:*
Received 12 January 2014
Received in revised form
24 April 2015
Accepted 7 June 2015
Available online 19 June 2015

*Keywords:*
One-class classification
Multi-class classification
Classifier ensemble
Decomposition strategies
Classifier fusion
Binary classification

ABSTRACT

Multi-class classification can be addressed in a plethora of ways. One of the most promising research directions is applying the divide and conquer rule, by decomposing the given problem into a set of simpler sub-problems and then reconstructing the original decision space from local responses.

In this paper, we propose to investigate the usefulness of applying one-class classifiers to this task, by assigning a dedicated one-class descriptor to each class, with three main approaches: one-versus-one, one-versus-all and trained fusers. Despite not using all the knowledge available, one-class classifiers display several desirable properties that may be of benefit to the decomposition task. They can adapt to the unique properties of the target class, trying to fit a best concept description. Thus they are robust to many difficulties embedded in the nature of data, such as noise, imbalanced or complex distribution. We analyze the possibilities of applying an ensemble of one-class methods to tackle multi-class problems, with a special attention paid to the final stage – reconstruction of the original multi-class problem. Although binary decomposition is more suitable for most standard datasets, we identify the specific areas of applicability for one-class classifier decomposition.

To do so, we develop a double study: first, for a given fusion method, we compare one-class and binary classifiers to find the correlations between classifier models and fusion algorithms. Then, we compare the best methods from each group (one-versus-one, one-versus-all and trained fusers) to draw conclusions about the overall performance of one-class solutions. We show, backed-up by thorough statistical analysis, that one-class decomposition is a worthwhile approach, especially in case of problems with complex distribution and a large number of classes.

© 2015 Elsevier Ltd. All rights reserved.

## 1. Introduction

Multi-class problems are abundant in real-life applications. Ranging from several classes in chemometrics [4], medicine [10], by dozens in object recognition [16] and computer vision [17] to hundreds in biometrics [11]. Often with the increase in the number of classes, comes the increased complexity of the estimated decision rules. This leads to the possibility of overfitting and increasing the computational cost of recognition system. Additionally, the difficulties in classification may be present only for some classes, while others can be separated with minimal error.

Building a classifier that handles only a reduced subset of classes may be a solution to these problems. As binary classification itself is well-studied in the last years [47], binary decomposition methods have gained a significant attention of the machine learning community [1]. Binary classifiers return simpler decision boundaries and allow us to reduce the competence areas of each classifier, thus producing locally specialized learners. This leads to a creation of an ensemble of binary learners [63], each dedicated to a sub-problem. From their local decision, the dedicated fusion method must reconstruct the original multi-class problem. While binary decomposition has been proven to perform very well in most multi-class problems [21], it has some limitations as being very dependent on the selected fusion method or low robustness to imbalanced and sparse distribution. That is why novel methods for tackling multi-class data need to be examined, having in mind that binary decomposition is a well-established point of reference.

In this work, we turn the attention towards one-class classification (OCC), which is quite young, yet challenging machine learning domain [35]. OCC seems a natural way of decomposing a multi-class dataset. We consider each class as independent and

* Corresponding author. Tel./fax: +48 071 320 21 06.
*E-mail addresses:* bartosz.krawczyk@pwr.edu.pl (B. Krawczyk), michal.wozniak@pwr.edu.pl (M. Woźniak), herrera@decsai.ugr.es (F. Herrera).

http://dx.doi.org/10.1016/j.patcog.2015.06.001
0031-3203/© 2015 Elsevier Ltd. All rights reserved.

train a one-class model on each of them. Then, we reconstruct the original problem with a dedicated fusion algorithm, just as in binary decomposition.

Application of OCC to problems, where generating negative examples can be costly, dangerous or simply impossible [24] is obvious. However, one may ask: What is the point of applying OCC to multi-class problems, where the representatives of all classes are given beforehand? Some studies show that when objects from all classes are available it is preferable to use binary classifiers than their one-class counterparts [8,26].

However, in this work we propose to evaluate a hypothesis that one-class ensembles can achieve high accuracy for handling multi-class datasets, despite discarding counterexamples during the training phase.

OCC methods have several desirable properties that may aid the process of multi-class decomposition. Their primal difference with binary learners lie in the nature of the training phase. Binary classifiers try to find such a decision boundary that will minimize the error on objects from both classes. OCC aims at capturing the unique properties of the target class, by finding the best possible description that at the same time will describe the analyzed set and not be overfitted to the given data. Due to this different principles, the decomposition with OCC presents itself as an interesting tool for handling complex data structures. Our previous studies point out that for some specific cases OCC decomposition may be of better use than other methods [37].

We aim at presenting an exhaustive study on usefulness and effectiveness of multi-class decomposition with one-class classifiers. To put our findings into context, we compare our proposed approach with state-of-the-art methods for binary decomposition, which is contemporary the most popular approach for this task. We show that for specific multi-class cases OCC can be a better choice than binary approach, as opposed to the reports in the literature [8,26]. Building up on our previous experiences, we propose to analyze the influence of combination methods on the decomposition ensembles and compare the most popular binary and one-class classifiers.

The contribution of this work is as follows:

- We identify the areas of applicability for OCC decomposition. We give an outlook on the potential high usefulness of OCC in this domain. We emphasize that OCC is not a universal solution for handling multi-class datasets. However, we aim at presenting the cases, in which OCC can outperform the binary approach, despite having less information during the learning phase. We discuss the reasons behind this and show that for complex data OCC offer a worthwhile alternative to binary methods.
- We study three types of aggregation methods for re-building original multi-class problem – one-versus-one, one-versus-all and trained fusers. Although such a comparison for binary classifiers exists [21] (and we use its findings in our work), such analysis in case of one-class classifiers is missing. So far only combination strategies for one-class problems (not multi-class decomposition) have been examined [52].
- We present a comparison of multi-class decomposition approaches, carried out with ensembles of one-class and binary classifiers. We apply state-of-the-art combination strategies over a diverse set of real-life datasets. With the use of thorough statistical analysis, we look for best combinations of base classifiers and aggregation methods. To the best of our knowledge, so far such a comparison spanning over binary and one-class decomposition ensembles and their fusion algorithms has not been carried out.

We want to answer some question: Can OCC, despite discarding information about counterexamples, be of use for dealing with multi-class datasets? When doing the multi-class decomposition, when it is preferable to try the OCC approach and when one should use standard binarization? This paper does not concentrate on the problem of when to use the decomposition (this topic has been discussed in [21]) – it deals with the issue on how to use it.

We base our discussion on an extensive empirical study. A diverse set of 19 multi-class benchmarks is used. We test two one-class classifiers, representing main groups of methods in this area (density estimation and boundary estimation), and compare them with well-established binary methods [22]. Based on our previous experience with binary [21] and one-class [61] fusion methods, we selected the best representatives of each of the three combination groups. We include an in-depth analysis of results to draw conclusions about the performance of analyzed methods and areas of suitability of OCC decomposition.

The rest of this paper is organized as follows. The next section presents the background for one-class classification area. In Section 3, we describe the essentials of decomposition techniques for handling multi-class data, while in Section 4 we present in detail the selected state-of-the-art combination methods for reconstructing the multi-class task. Section 5 presents the setup of the used experimental framework and its individual elements. The results of experimental investigations, together with a thorough discussion, are given in Section 6. The final Section 7 concludes the paper.

## 2. One-class classification

In this section, we present an overview of the topic of one-class classification. We show the unique properties of this pattern recognition problem and discuss the most popular families of methods used to tackle this task.

Here it is assumed that during the training stage only objects coming from a single class are available. These are called the target concept and are denoted by $\omega_T$. The purpose of OCC is to calculate a decision boundary that encloses all available data samples, thus describing the concept [56]. During the exploitation phase, new objects, unseen at the training phase, may appear. They may come from one or more distributions and represent data outside the target concept. Such objects are labeled as outliers and denoted by $\omega_O$. Therefore, the OCC is often referred to as learning in the absence of counterexamples.

OCC aims to distinguish the target concept objects from these possible outliers. It is quite similar to binary classification but the primary difference is how the one-class classifier is trained. In the standard dichotomy problems we may expect objects from the other classes to predominantly come from one direction. Here, the available class should be separated from all the possible outliers – this leads to a situation in which a decision boundary should be estimated in all directions in the feature space around the target class. An example of a OCC problem is depicted in Fig. 1.

OCC is a solution to many real-life problems where data from a single class is abundant but is hard or even impossible to obtain for other objects. This is often the case in problems such as intrusion detection [24], machine fault diagnosis [7], or solid-state fermentation [32].

Several methods dedicated to solving OCC problems have been recently introduced. In the relevant literature two main approaches can be distinguished:

- Methods based on density estimation of a target class, which can be simple and effective in some cases. However, this approach has limited applications, as it requires a high number of available samples and the assumption of a flexible density model [51]. Among the most popular density methods for OCC

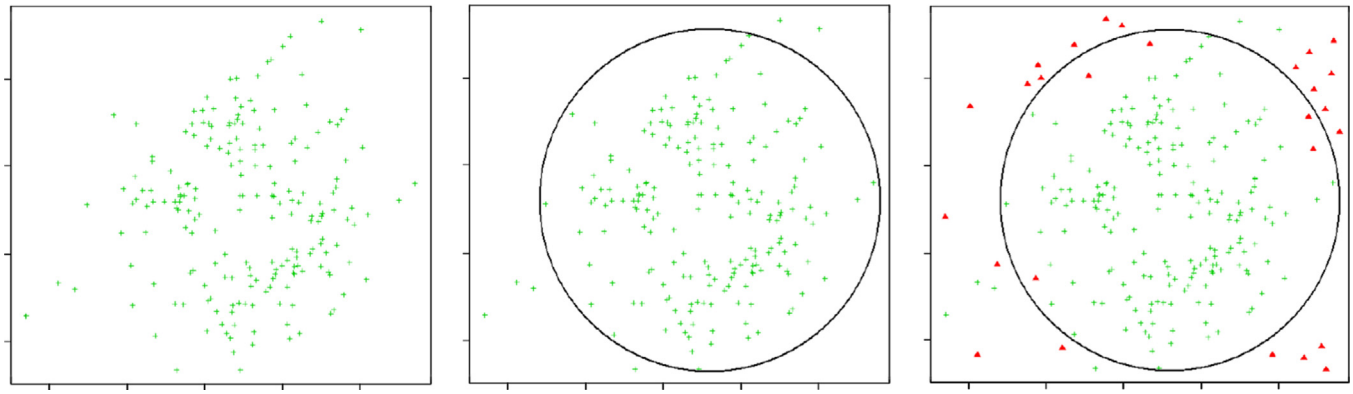

**Fig. 1.** The idea of one-class classification. (*Left*) Data available during the classifier training procedure (*green*) representing the target concept. (*Center*) Boundary one-class classifier with volume enclosing all the relevant samples. (*Right*) outlier objects (*red*) that appear during the exploitation of the model. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

the Gaussian model, the mixture of Gaussians [65], and the Parzen density [15] can be mentioned.

- Estimating the complete density or structure of a target concept in one-class problems may very often be too demanding or even impossible. Therefore, boundary methods have been proposed in recent years. They concentrate on estimating only the close boundary for a given data, assuming that such a boundary will describe sufficiently the target class [34]. The main aim of these methods is to find the optimal size of the volume enclosing given training points [54], because one that is too small can lead to an overtrained model, while one that is too big may lead to an extensive acceptance of outliers into the target class. These methods rely strongly on the distance between objects, therefore proper feature scaling is a very important data preprocessing step. On the other hand, boundary methods require a smaller number of objects to properly estimate the decision criterion in comparison with two previous groups of methods. The most popular methods of this group include the Support Vector Data Description [53] and the One-class Support Vector Machine [13].

In the literature sometimes a third group of methods is mentioned. It is known as reconstruction methods which were originally introduced as a tool for data modeling [14]. This group of algorithms makes assumptions about the object distribution. Use of reconstruction methods for OCC is based on the idea that possibly the unknown outliers do not satisfy those assumptions about the structure of objects under consideration. The most popular techniques are the k-means [12], the self-organizing maps [57] and the auto-encoder networks [45]. However, one can notice that the basis of operation for this group of algorithms is similar to the density-based methods estimating some distribution/structure of the data. That is why it can be considered as a sub-group of this family of one-class classifiers.

## 3. Decomposition techniques for multi-class problems

In the following section, we give a necessary background in the area of multi-class decomposition and binarization. We describe the groups of techniques for aggregating the binary classifiers, that are used in this study – OVO, OVA and trained fusers. Finally, we introduce the concept of applying one-class classifiers for handling multi-class datasets and the differences and similarities between this approach and binarization.

### 3.1. Binary classifiers for decomposing multi-class datasets

According to divide and conquer rule, we should aim at solving each complex problem by dividing it into a series of subproblems, each easier to solve than the original task [9]. This strategy can be easily applied in machine learning, where dealing with complex, multi-class datasets is a common practice [30]. The most popular approach is to decompose the original dataset into a number of binary problems [20,28]. This way we achieve locally specialized classifiers that deal with simplified tasks [46]. The crucial part of such a decomposition is the reconstruction of original multi-class problem. Dividing the data into binary groups is relatively simple – reconstructing the problem from individual outputs is far from trivial. Therefore, a proper combination method plays a very important role in the success of the decomposition scheme.

The most popular decomposition schemes are OVO and OVA. The former creates all possible pairwise combinations of classes, while the latter selects one class as the positive class and uses all the remaining ones as the negative class. For both cases the simplest aggregation strategies are voting methods. For OVO the class with the highest number of votes win. In OVA, one expect that all but one classifiers will point out to the negative class. Hence, this is identical to the Winner-Takes-All (WTA) strategy. However, the limitations of simple voting methods were soon discovered and more sophisticated methods, specialized for the decomposition task, have been proposed.

In recent years the attention of researchers shifts towards the OVO approach. It divides an $M$-class dataset into $M(M-1)/2$ binary pairwise problems. Each classifier is trained on the basis of the reduced training dataset, which consists only of two corresponding classes. In the testing phase the new object is presented to each of the classifiers from the pool. The possible output of the classifier is given by $y_{ij} \in [0,1]$. This stands for a classifier discriminating between classes $i$ and $j$ in favor of the former. The outputs are stored in the confidence matrix for each possible pairwise combinations. The OVO combination methods work directly on this matrix.

In [21] it was showed that this aggregation strategy offers very good results, regardless of the base classifier used. Most of the works here were inspired by Support Vector Machines, yet they are elastic enough to be applied to any kind of binary learner. Some works were inspired by limitations of majority voting – among the most important Decision Directed Acyclic Graph [42] and Nesting OVO [43] should be mentioned. Alternative way was proposed in Binary Trees [20] and similar hierarchical structures [44] that discriminate against a group of classes at each node, finally returning the final decision in the leaf. An interesting hybrid

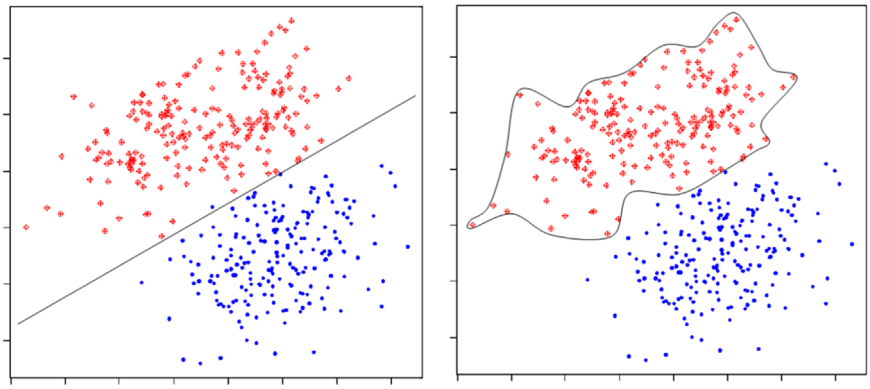

**Fig. 2.** The difference between binary and one-class classifier. (*Left*) A toy data problem handled by the binary classifier. (*Right*) The same dataset analyzed with the usage of the one-class classifier, with a single class serving as the target concept.

approach, combining OVO with adaptive resonance theory networks was described in [59]. In OVO with the increase of the number of classes, increases significantly the quantity of binary ensembles in the committee. New ways for managing and reducing a large number of individual OVO decisions were proposed in [48].

OVA strategy did not receive the same attention as its counterpart, although some report its similar performance [49]. By its nature, OVA return a lower number of base classifiers for the decomposition ensemble, yet they are more complex. It decomposes an $M$-class problem into $M$ binary problems. During the training phase each classifier selects its positive class. All other examples are transformed to serve as negative examples. During the testing phase, a new object is presented to each classifier and the classifier with positive vote is selected as the winner. Using simple binary voting, as in OVO, may lead to ties if more than one classifier gives a positive prediction. That is why often the continuous support function values of classifiers are used and the classifier with the highest confidence value wins.

The main problem of OVA approach is the introduction of imbalance into the training set. As objects from all but one classes are used as negative examples, their quantity is much greater than the positive examples. Negative effects of imbalanced distribution on classifier systems are well-known [31]. This may be one of the reasons behind reports about inferior performance of OVA in comparison to OVO.

The aggregation strategies here are but a few. One should mention the WTA, based on the maximum value of the support function, and the Dynamically Ordered OVA, which is based on a Naive Bayes classifier for establishing the order in which the OVA learners are executed [27]. In [29] authors propose to combine OVA with evolutionary computation to improve the aggregation quality.

The third approach, that grew from the former ones, can be described as trained combiners. They are based on constructing prototypes for each classes, based on the training data, that will guide the fusion process [3]. The most popular approach is the Error-Correcting Output Codes (ECOC) [19], which creates a unique code for each binary classifiers [62]. Many works regarding ECOC concentrate on the design of compact and reliable codewords to allow handling many classes [5]. Another popular approach from this group is Decision Templates [39] that create prototypes of support function values of each classifier for a given class. Then during the prediction a new decision profile is formed from the individual outputs and is compared to existing templates. The template with the lowest distance from the profile is selected as the output [40].

### 3.2. One-class classifiers for decomposing multi-class datasets

So far OCC was used for problems, when the counterexamples were hard or impossible to collect. To the best of our knowledge there exist only few works in applying OCC for handling multi-class datasets. In [6] authors proposed a simple scheme based on minimal distance from the class. An interesting extension of boosting, where one-class classifiers were built iteratively on the order of classes, was introduced in [64]. SVDD, with aim of maximizing the separation margin, were implemented for multi-class problems in [58]. Authors of this paper showed that introducing diverse one-class classifiers may be a useful tool for handling multi-class datasets [37], and that using multi-criteria ensemble pruning may result in a more stable one-class [36] decomposition.

One may doubt the idea of using OCC for decomposing multi-class datasets. OCC uses only information about the target class, therefore, we discard available useful information. Some articles report that in case of sufficient number of counterexamples one should abstain from using OCC and apply binary methods [8,26]. We would like to present a contrary opinion and prove that one-class classifiers may be a useful tool for tackling multi-class problems. However, we emphasize that OCC is by no means a superior method to binary classifiers. For standard datasets, binary classifiers will perform better, due to their access to counterexamples. The applicability of OCC lies in complex data, where standard binary classifiers tend to fail.

To understand the possible advantages of OCC, let us first take a look on the differences between a binary and an one-class classifier. Examples of each of these methods are given in Fig. 2.

Both types of classifiers consider two classes – positive (in OCC called target class) and negative (in OCC called outlier class). The difference lies in the training procedure. Binary classifier has an access to object coming from both classes, while in OCC gathering the counterexamples sufficient for training is impossible. Both types of classifiers output a binary decision $[+1,-1]$, where for OCC, $-1$ stands for an object that did not satisfied the description of the target concept.

Binary classifiers shape their decision boundary in such a way, that will minimize the error made on both classes. OCC aims at capturing the unique properties of the target class, hoping that they will allow for a sufficient dichotomization from unknown outliers. While OCC uses less information about the problem being considered, its properties allow us to deal with difficulties, embedded in the nature of the data: imbalance, class noise or inner outliers, to name a few [38,56].

Using OCC for decomposing a multi-class dataset is very intuitive – each class is considered as independent and delegated

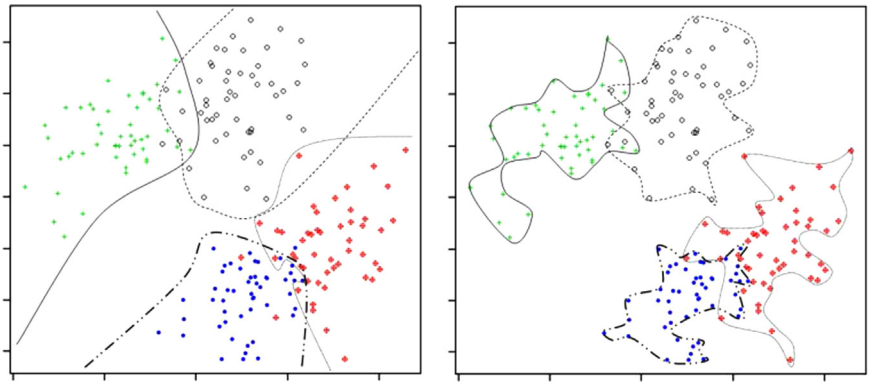

**Fig. 3.** The difference between decomposing a multi-class problem with binary and one-class classifiers. (*Left*) A toy problem with four classes decomposed with binary classifiers applying OVA procedure. (*Right*) The same dataset decomposed with one-class classifiers, each delegated to a different class. Note that in this example we assumed that all the data should be included in the decision boundaries, while some of the one-class methods can discard irrelevant objects.

to a different one-class model. Therefore, for an $M$-class problem, we get $M$ separate one-class tasks. This can be recognized as similar to OVA approach. An illustrative example of OCC decomposition is given in Fig. 3.

In fact, OCC decomposition can be viewed as a special case of OVA decomposition – its one class against the rest, with the difference that it uses only the class for training and omit all remaining ones. Although this may seem as a disadvantage, in practice lifts one of the biggest problems related to OVA – imbalanced object quantity between the target class and the remaining ones.

Each base classifier aims at adjusting itself to the given target class. In OCC the classifier is fit in such a way, that will allow for a best possible separation form potential outliers. On the other hand, OCC algorithms tend to avoid overfitting by not having tight description around the data [41]. This is an especially important feature in multi-class decomposition, as it allows us to discriminate between the classes and at the same time preserves the generalization abilities of the base classifiers.

Some OCC methods, as SVDD have build-in methods for dealing with irrelevant data in the target class (i.e., objects lying too far from the main distribution). This is a desirable property for handling multi-class problems, as such objects will cause the decision boundary to become too big, leading in turn to a high overlap between classifiers [37].

The training phase is independent for each classifier and thus can be easily run in parallel environment to reduce the computational time [60]. The testing phase is based on presenting a new object to each classifier. Then OCC decides if the target object belongs to a target class or is labeled as an outlier. Then from each of $M$-decisions the final multi-class decision is aggregated. Notice that it is straightforward to implement OVA combiners to this case, while OVO combiners must be slightly modified to deal with such classifiers. The training set of each classifier consists only of objects from a specific class. Testing set should have objects from both the outlier and the target class, in order to evaluate both the false acceptance and false rejection rates.

In case of all classifiers in the ensemble point out the new object as an outlier, we assign it the class represented by the closest one-class boundary, as expressed by Euclidean distance.

Some one-class classifiers (like SVDD) base their decision on the distance from the decision boundary. To apply fusion methods that require support function values for each class, one must use the following heuristic mapping (from distance to probability):

$$F(x,\omega_T)=\frac{1}{c_1}\exp(-d(x|\omega_T)/c_2), \tag{1}$$

which models a Gaussian distribution around the classifier, where $d(x|\omega_T)$ is a distance (in case of this paper the Euclidean distance is used) from the evaluated object to the support vectors describing the target concept, $c_1$ is the normalization constant and $c_2$ is the scale parameter. Parameters $c_1$ and $c_2$ should be fitted to the target class distribution.

Finally, one must note that for standard one-class classifiers one cannot compute the accuracy during the training phase and other measures must be used [55]. For multi-class decomposition however, during the testing phase, we have objects representing all the classes available and we can use the accuracy as the performance measure.

## 4. Ensemble fusion methods for aggregating local decisions

In this section, we will describe the five fusion methods for aggregating decomposed decisions used in this paper. They represent three different groups of fusers: OVA, OVO and trained combiners. Our choice of OVO and OVA methods was dictated by the results found in the comprehensive study on binary decomposition [21]. Following suggestions presented in this paper, we have selected three well-performing aggregation schemes. Additionally, we have expanded our study with two trained fusers that were omitted in the former comparison. In our opinion they represent an important direction in ensemble learning and should be taken into consideration.

### 4.1. OVO and OVA fusers

- *Maximum confidence strategy* (MAX) is an OVA aggregation scheme. It is developed in order to handle tie situations, in which more than one classifier give a positive answer. The final output is taken from the classifier with highest value of the support function:

$$Class_x=\arg\max_{m=1,2,\ldots,M} F(x,m). \tag{2}$$

- *Pairwise coupling* (PC) [25] is an OVO aggregation scheme. It is based on the estimation of the joint probability for $M$ classes from the pairwise probabilities of all possible binary combinations. Therefore, for given Prob($Class_i|Class_i$ or $Class_j$), the fuser approximates the posteriori probabilities $\hat{p}(x)=(\hat{p_1}(x),\ldots,\hat{p_M}(x))$, based on the individual classifier outputs. The class with the highest posteriori probability is selected as the final

output of the system:

$$Class_x = \arg \max_{m=1,2,\ldots,M} \hat{p}_m. \quad (3)$$

In order to calculate the posteriori probabilities, the Kullback–Leibler distance between $r_{ij}$ and $\mu_{ij}$ is minimized:

$$l(p) = \sum_{1 \le i \ne j \le M} n_{ij} r_{ij} \log \frac{r_{ij}}{\mu_{ij}} = \sum_{i<j} n_{ij} \left( r_{ij} \log \frac{r_{ij}}{\mu_{ij}} + (1-r_{ij}) \log \frac{1-r_{ij}}{1-\mu_{ij}} \right), \quad (4)$$

where $\mu_{ij} = p_i/(p_i+p_j)$ and $n_{ij}$ stands for the number of objects in the $i$-th and $j$-th classes.

For OCC methods, one needs to estimate the acceptance/rejection probabilities for the target class [54].

- *Decision Directed Acyclic Graph* (DDAG) [42] is an OVO aggregation scheme. It constructs a rooted binary acyclic graph with each node having assigned a list of classes and a corresponding classifier (binary or OCC). At each level the designated classifier dichotomizes between given two classes, and the class that is not predicted is removed. In case of OCC if the classifiers predict the outlier, the target class is removed, and vice versa. The last class remaining on the list is the final output of the system.

### 4.2. Trained fusers

- *Error-Correcting Output Codes* (ECOC) [19] framework is a simple yet effective framework created for dealing with the multi-class categorization problem the reconstruction from the decisions of binary classifiers. The basis of the ECOC framework consists of designing a codeword for each of the classes. These codewords encode the membership information of each class. Arranging the codewords as rows of a matrix, we obtain an encoding matrix. Each of these binary problems (or dichotomizers) splits the set of classes into two partitions (coded by $+1$ or $-1$ according to their class set membership or 0 if the class is not considered by the current binary problem). Then, at the decoding step, applying the $n$ trained binary classifiers, a code is obtained for each data point in the test set. This code is compared to the base codewords of each class denned in the encoding matrix and the data point is assigned to the class with the closest codeword [62]. ECOC can be easily used for OCCs ensemble, as we can map the target class as $+1$ and the unknown, outlier class by $-1$.
- *Decision templates* (DT) [39] consists of typical, often average values of discriminant functions, returned by classifiers from a pool for each of the classes. During the classification step each of the templates is compared to the input objects. A decision template for the $j$-th class, labeled as $DT_j$, is created by averaging values of discriminant functions for objects in the training set from the $j$-th class. The responses of classifiers are organized as a matrix, with the number of columns equal to the number of $L$ classifiers in a pool and with the number of rows equal to the number of $M$ classes in the problem under consideration. A decision profile for a new object $x$, labeled as $DP(x)$, is then compared to all existing decision templates:

$$y_{dt}(x) = \Im(DT_j, DP(x)), \quad j = 1, \ldots, M, \quad (5)$$

where $\Im$ is called a *similarity measure* (or a *distance from the template*). The most popular one is the Euclidean measure, for which the value of discriminant function for the $j$-th class is expressed as follows:

$$y_{dt}(x) = 1 - \frac{1}{M \times L} \sum_{i=1}^{M} \sum_{k-1}^{L} [DT_j(k,i) - g_{k,i}(x)]^2, \quad (6)$$

where $DT_j(k,i)$ indicates an element at the $(k,i)$ position in the $j$-th decision template and $g_{k,i}(x)$ stands for the $k$-th classifier for the $i$-th class.

**Table 1**
Details of datasets used in the experiments.

| No. | Name | Objects | Features | Classes |
|---|---|---|---|---|
| 1. | Autos | 159 | 25 | 6 |
| 2. | Car | 1728 | 6 | 4 |
| 3. | Cleveland | 297 | 13 | 5 |
| 4. | Dermatology | 366 | 33 | 6 |
| 5. | Ecoli | 336 | 7 | 8 |
| 6. | Flare | 1389 | 10 | 6 |
| 7. | Glass | 214 | 9 | 6 |
| 8. | Led7digit | 500 | 7 | 10 |
| 9. | Lymphography | 148 | 18 | 4 |
| 10. | Nursery | 1296 | 8 | 5 |
| 11. | Page-blocks | 548 | 10 | 5 |
| 12. | Penbased | 1099 | 16 | 10 |
| 13. | Satimage | 643 | 36 | 7 |
| 14. | Segment | 2310 | 19 | 7 |
| 15. | Shuttle | 2175 | 9 | 7 |
| 16. | Vehicle | 846 | 18 | 4 |
| 17. | Vowel | 990 | 13 | 11 |
| 18. | Yeast | 1484 | 8 | 10 |
| 19. | Zoo | 101 | 16 | 7 |
| 20. | Auslan | 2565 | 128 | 95 |

## 5. Experimental setup

In this section, we describe the set-up of used experimental framework. We give the details about the used data, classification algorithms and statistical test used.

### 5.1. Datasets

For the purpose of the presented study, we selected 20 diverse datasets from the UCI repository. Our aim was to chose datasets, to which the decomposition would be useful – in other words datasets consisting of a larger number of classes. Selected data are described by 4–95 classes. The details of used datasets are presented in Table 1. Some of the largest datasets (nursery, page-blocks, penbased, satimage, shuttle and led7digit) were sampled with stratification and reduced to 10% of the original size, in order to reduce the overall computational cost required for training. For datasets with missing values, instances without full set of features available were removed.

The selection of datasets for this study was motivated by selecting sets with more than three classes, on which the standard machine learning algorithms (without the decomposition procedure) achieve an accuracy rate over 50%.

Experiments were done with the usage of $5 \times 2$ cross-validation. In case of one-class classifiers, the classifier was trained on the training fold consisting only objects from the target class and tested on objects from all other classes (labeled as outliers) and a testing sample from the target class (to check the false rejection rate).

### 5.2. Classification algorithms

For this study, we aimed at a comparison between the one-class and binary ensembles for decomposing multi-class data. Therefore, we selected a well-known and popular machine learning algorithm, in order to evaluate their performance over a diverse set of multi-class benchmarks. Our choice of binary classifiers was dictated by the lessons learned from a survey on binary decomposition techniques

[21], which pointed out C4.5 and SVM as stable and robust base classifiers for the decomposition. As for the one-class models, we described in Section 2 that they come from two main groups. We decided to chose two representative from each of them, based on their well-documented performance and popularity. We decided that density methods will be represented by Parzen Density Data Description and Mixture of Gaussians Data Description, while boundary methods are represented by One-Class Support Vector Machine and Support Vector Data Description. Below each of the used classifiers is described shortly:

- Parzen Density Data Description [15] is a kernel density estimator. It is a flexible density model with a Gaussian model around each of the training objects. For OCC purposes it estimates the complete density of the target class, and checks if the new, unseen object may be drawn from the estimated distribution.
- Mixture of Gaussians Data Description [65] is another density estimator that uses a combination of $K$ Gaussians to achieve a more flexible description of the target class. *EM* algorithm is used to optimize parameters of this classifier. For high-dimensional data, one needs to remember that the number of free parameters grows significantly.
- One-Class Support Vector Machine [50] is a modification of popular support vector classifier for one-class problems. It maps the training data onto an enclosing hyperplane with the usage of function $f_\chi : \mathbb{R}^d \mapsto \mathbb{R}$ such that most of the data in $\chi$ belong to the set $\mathcal{R}_\chi = \{x \in \mathbb{R}^d; f_\chi(x) \geq 0\}$ and the volume of $\mathcal{R}_\chi$ is minimal. This problem is known as *minimal volume set* (MVS) estimation. Because we are considering an $M$-class recognition problem, we have to learn $M$ membership functions $f_{\chi_i}$ – one for each class. A kernel function can be used to estimate MVS.
- Support Vector Data Description (SVDD) [53] is a model which gives a closed boundary around the data in a form of a hypersphere. It is characterized by a center $a$ and radius $R$. In its basic form it assumes that all objects from the training set must be enclosed by this hypersphere. Yet this approach often leads to a poor performance due to too big enclosing volume. Therefore, identically as in canonical Support Vector Machine one may introduce slack variables to include the possibility inner outliers in the training set. It uses a kernel function for mapping the data into higher dimensions with a better representation.
- C4.5 is a decision tree induction algorithm. It constructs a hierarchical classification rule structure in a form of top-down tree construction, with the normalized information gain for splitting criterion. The feature with the highest value of this criterion is used to split the tree at the considered level.
- Support Vector Machine (SVM) is a maximum separating margin-based classifier. It applies the kernel function to map the original data into a high-dimensional feature space, in which the linear separation will be possible. A maximum margin between the classes is selected in the new, artificial space to minimize an upper bound of the expected risk, and avoid using empirical risk.

Parameters used for the algorithms are given in Table 2. They are standard values used normally for these types of classifiers, examined in our previous works on binary and one-class ensembles [21,37]. We notice that careful tuning of each classifier may result in an improvement of the accuracy. However, our aim was to propose a decomposition method comparison, not a detailed study on optimizing classifier features. Furthermore, in a framework where no method is tuned, winner methods tend to correspond to the most robust, which is also a desirable characteristic.

We used identical parameters for each dataset.

**Table 2**
Details of classifier parameters used in the experiments.

| Algorithm | Parameters |
|---|---|
| Parzen | kernel type = normal |
| | parameter optimization = max. likelihood |
| MoG | no. of components = [2;10] (best setting selected) |
| OSCVM | kernel type = RBF |
| | C = 1.0 |
| | Tolerance = 0.05 |
| | Epsilon = 1.0E − 12 |
| | parameter optimization = SMO |
| SVDD | kernel type = RBF |
| | C = 5.0 |
| | Tolerance = 0.01 |
| | Epsilon = 1.0E − 12 |
| | parameter optimization = quadratic programming |
| C4.5 | pruning = true |
| | confidence level = 0.25 |
| | Minimum number of items per leaf = 2 |
| SVM | kernel type = RBF |
| | polynomial degree = 1 |
| | C = 1.0 |
| | Tolerance = 0.01 |
| | Epsilon = 1.0E − 12 |
| | parameter optimization = SMO |

### 5.3. Statistical tests

In order to present a detailed comparison among a group of machine learning algorithms, one must use statistical tests to prove that the reported differences among classifiers are significant [23]. We use both pairwise and multiple comparison tests. Pairwise tests give as an outlook on the specific performance of methods for a given dataset, while multiple comparison allows us to gain a global perspective on the performance of the algorithms over all benchmarks. With this, we get a full statistical information about the quality of the examined classifiers.

- For a pairwise comparison, we use a $5 \times 2$ combined CV F-test [2]. It repeats five-time two fold cross-validation so that in each of the folds the size of the training and testing sets is equal. This test is conducted by comparison of all versus all. As a test score the probability of rejecting the null hypothesis is adopted, i.e. that classifiers have the same error rates. As an alternative hypothesis, it is conjectured that tested classifiers have different error rates. A small difference in the error rate implies that the different algorithms construct two similar classifiers with similar error rates; thus, the hypothesis should not be rejected. For a large difference, the classifiers have different error rates and the hypothesis should be rejected.
- For assessing the ranks of classifiers over all examined benchmarks, we use a Friedman ranking test [18]. It checks, if the assigned ranks are significantly different from assigning to each classifier an average rank.
- We use the Shaffer post hoc test to find out which of the tested methods are distinctive among an $n \times n$ comparison. The post hoc procedure is based on a specific value of the significance level $\alpha$. Additionally, the obtained $p$-values should be examined in order to check how different given two algorithms are.

We fix the significance level $\alpha = 0.05$ for all comparisons.

## 6. Experimental results

We have carried an extensive experimental comparison, using the framework described in Section 5. By this, we wanted to answer the following questions:

**Table 3**
Results (accuracy [%]) for one-class and binary classifiers combined using Maximum Confidence Strategy (MAX). Small numbers under accuracies stand for indexes of methods, from which the considered one is statistically superior.

| Dataset | Parzen[1] | MoG[2] | OCSVM[3] | SVDD[4] | C4.5[5] | SVM[6] |
|---|---|---|---|---|---|---|
| 1. | 68.94 5 | 68.02 5 | 70.88 1,2,5,6 | 71.95 1,2,5,6 | 65.43 – | 67.42 5 |
| 2. | 85.23 2 | 83.47 – | 87.90 1,2 | 88.64 1,2 | 89.24 1,2 | 89.12 1,2 |
| 3. | 47.32 – | 47.32 – | 55.87 1,2,5 | 55.87 1,2,5 | 54.21 1,2 | 57.53 1,2,5 |
| 4. | 90.36 – | 90.36 – | 95.41 1,2,5 | 95.41 1,2,5 | 90.12 – | 95.75 1,2,5 |
| 5. | 71.18 – | 71.76 – | 72.90 1,2 | 72.74 1,2 | 77.58 1,2,3,4 | 77.43 1,2,3,4 |
| 6. | 75.56 5 | 75.56 5 | 74.91 – | 74.91 – | 73.73 – | 75.49 5 |
| 7. | 63.22 6 | 63.22 6 | 62.89 – | 62.89 – | 65.32 1,2,3,4,6 | 60.84 – |
| 8. | 70.45 2 | 68.18 – | 75.18 1,2,5,6 | 75.65 1,2,5,6 | 69.86 2 | 71.20 2,3 |
| 9. | 67.32 – | 67.82 – | 76.72 1,2 | 76.72 1,2 | 74.94 1,2 | 82.27 1,2,4,5 |
| 10. | 90.21 4,5 | 91.16 1,4,5 | 85.64 5 | 86.62 5 | 79.35 – | 91.05 3,4,5 |
| 11. | 88.48 – | 88.48 – | 92.05 1,2 | 92.05 1,2 | 94.53 1,2,3,4 | 95.34 1,2,3,4 |
| 12. | 95.02 5,6 | 94.26 5,6 | 96.11 5,6 | 95.76 5,6 | 84.80 – | 90.45 5 |
| 13. | 76.26 – | 77.14 1 | 81.72 1,2,5 | 81.96 1,2,5 | 80.01 1,2 | 82.02 1,2,5 |
| 14. | 89.67 2 | 87.82 – | 90.47 1,2 | 90.47 1,2 | 94.72 1,2,3,4,6 | 91.21 – |
| 15. | 96.23 3,4,6 | 96.23 3,4,6 | 94.25 6 | 94.25 6 | 97.51 1,2,3,4,6 | 92.74 – |
| 16. | 65.21 – | 65.79 – | 70.93 1,2,4 | 69.33 1,2 | 70.06 1,2 | 75.15 1,2,3,4,5 |
| 17. | 71.78 4,5,6 | 71.78 4,5,6 | 70.09 6 | 70.09 6 | 75.76 1,2,3,4,6 | 51.23 – |
| 18. | 55.23 2 | 53.82 – | 57.07 1,2,5,6 | 57.98 1,2,5,6 | 54.24 – | 56.20 1,2 |
| 19. | 83.22 – | 83.22 – | 87.14 1,2 | 87.96 1,2 | 88.27 1,2 | 96.05 1,2,3,4,5 |
| 20. | 78.39 5,6 | 77.83 5,6 | 81.58 1,2,5,6 | 82.86 1,2,5,6 | 67.90 – | 74.12 5 |
| Avg. rank | 4.26 | 4.88 | 2.89 | 2.59 | 3.94 | 2.44 |

**Table 4**
Results (accuracy [%]) for one-class and binary classifiers combined using classification by Pairwise Coupling (PC). Small numbers under accuracies stand for indexes of methods, from which the considered one is statistically superior.

| Dataset | Parzen[1] | MoG[2] | OCSVM[3] | SVDD[4] | C4.5[5] | SVM[6] |
|---|---|---|---|---|---|---|
| 1. | 66.89 – | 66.24 – | 69.42 1,2 | 70.45 1,2 | 77.71 1,2,3,4,6 | 74.12 1,2,3,4 |
| 2. | 83.24 2 | 81.97 – | 82.05 – | 82.91 – | 92.38 1,2,3,4 | 92.76 1,2,3,4 |
| 3. | 39.37 – | 39.37 – | 48.29 1,2 | 48.29 1,2 | 50.47 1,2,3,4 | 58.62 1,2,3,4,5 |
| 4. | 87.38 – | 87.38 – | 91.24 1,2 | 91.24 1,2 | 96.03 1,2,3,4,6 | 93.82 1,2,3,4 |
| 5. | 71.45 – | 72.00 – | 72.74 – | 72.30 – | 79.27 1,2,3,4,6 | 77.26 1,2,3,4 |
| 6. | 72.05 3,4 | 72.05 3,4 | 69.32 – | 69.32 – | 74.19 1,2,3,4 | 74.95 1,2,3,4 |
| 7. | 59.23 – | 59.23 – | 65.11 1,2,6 | 65.11 1,2,6 | 71.48 1,2,3,4,6 | 63.22 1,2 |
| 8. | 69.32 2 | 67.04 – | 73.61 1,2,5,6 | 74.13 1,2,5,6 | 72.24 1,2 | 72.76 1,2 |
| 9. | 67.48 – | 68.32 – | 73.74 1,2 | 73.74 1,2 | 74.97 1,2 | 81.54 1,2,3,4,5 |
| 10. | 87.23 3,4 | 88.46 1,3,4 | 84.02 – | 84.59 – | 88.63 3,4 | 91.82 1,2,3,4,5 |
| 11. | 88.32 – | 88.32 – | 90.68 1,2 | 90.68 1,2 | 95.20 1,2,3,4,6 | 94.58 1,2,3,4 |
| 12. | 94.45 5 | 93.74 5 | 94.26 5 | 93.87 5 | 90.36 – | 95.15 1,2,3,4,5 |
| 13. | 72.34 – | 74.01 1 | 76.28 1,2 | 76.64 1,2 | 81.98 1,2,3,4 | 83.47 1,2,3,4,5 |
| 14. | 90.15 1,3,4 | 88.66 – | 88.02 – | 88.02 – | 96.79 1,2,3,4,6 | 92.81 1,2,3,4 |
| 15. | 95.23 3,4 | 95.23 3,4 | 92.71 – | 92.71 – | 99.34 1,2,3,4,6 | 96.17 3,4 |
| 16. | 62.18 – | 62.48 – | 67.02 1,2,4 | 65.44 1,2 | 71.03 1,2,3,4 | 72.93 1,2,3,4,5 |
| 17. | 68.28 3,4 | 68.28 3,4 | 65.32 – | 65.32 – | 80.00 1,2,3,4,6 | 69.78 3,4 |
| 18. | 55.47 2 | 54.03 – | 54.88 – | 55.21 2 | 57.04 1,2,3,4 | 58.39 1,2,3,4,5 |
| 19. | 80.41 – | 80.41 – | 83.97 1,2 | 84.39 1,2 | 92.77 1,2,3,4 | 95.12 1,2,3,4,5 |
| 20. | 76.14 5,6 | 75.22 5,6 | 79.18 1,2,5,6 | 80.23 1,2,5,6 | 61.25 – | 70.38 5 |
| Avg. rank | 4.10 | 4.97 | 3.51 | 3.19 | 2.86 | 2.37 |

- Can applying one-class classifiers for decomposing multi-class datasets bring better results than using binarization, despite rejecting the information about counterexamples?
- Is there a difference from the decomposition point of view, between the density and boundary-based one-class methods?
- What kind of aggregation method is most suitable for reconstructing an original multi-class dataset from one-class responses?

The results for examined methods are given in Tables 3–7, one table dedicated to each fusion method under consideration. Apart from the accuracies, the results of statistical pairwise and ranking tests are presented. Table 8 presents collected results – for each fusion method statistically best one-class and binary classifiers were chosen.

Results of the Shaffer post hoc test are depicted in Tables 9 and 12. Note that we are doing such a comparison only for one-class classifiers. Post hoc analysis for binary classifiers and their fusion methods was presented in [21].

### 6.1. Can one-class ensembles outperform binary approach?

The main aim of this study was to answer if investigating the decomposition methods with one-class classifiers is worthwhile, and if this approach has any advantages over the well-established binarization.

Our initial assumption was that OCC will not be superior to binarization for all cases, and that was confirmed. For standard, well-sampled and balanced datasets, with lack of noise, binary classifiers tend to outperform significantly their one-class counterparts. This is an obvious situation, as binary classifiers have an access to counterexamples during the training phase. Both SVM and C4.5 work reasonably well for standard decomposition scenarios. As there are no difficulties embedded in the nature of the data, that can deteriorate their performance, binary methods can estimate an efficient separation plane that allows for a good dichotomization, and in result for a high quality multi-class recognition. SVM works very well, due to its natural binary nature and efficient maximum margin separation. C4.5 is a weak classifier, but can work surprisingly well for simplified problems, due to its space partitioning nature which is proven by these tests and other done in the literature [21].

However, the results clearly show that one cannot assume the lack of applicability of OCC for the decomposition approach. For more than half of the datasets one-class methods achieved not worse performance than binary classifiers. This proves that they

**Table 5**
Results (accuracy [%]) for one-class and binary classifiers combined using Decision Directed Acyclic Graph (DDAG). Small numbers under accuracies stand for indexes of methods, from which the considered one is statistically superior.

| Dataset | Parzen[1] | MoG[2] | OCSVM[3] | SVDD[4] | C4.5[5] | SVM[6] |
|---|---|---|---|---|---|---|
| 1. | 67.56 – | 67.04 – | 70.45 1,2 | 71.05 1,2 | 78.56 1,2,3,4,6 | 74.73 1,2,3,4 |
| 2. | 83.11 2 | 81.62 – | 83.24 2 | 84.02 2 | 92.03 1,2,3,4 | 92.51 1,2,3,4 |
| 3. | 40.05 – | 40.05 – | 50.78 1,2 | 50.78 1,2 | 53.26 1,2,3,4 | 58.69 1,2,3,4,5 |
| 4. | 87.12 – | 87.12 – | 93.05 1,2 | 93.05 1,2 | 95.31 1,2,3,4,6 | 93.95 1,2 |
| 5. | 72.38 – | 73.04 – | 73.38 1 | 72.89 – | 78.64 1,2,3,4 | 77.36 1,2,3,4 |
| 6. | 72.87 – | 72.87 – | 71.95 – | 71.95 – | 74.11 1,2,3,4 | 75.19 1,2,3,4,5 |
| 7. | 60.41 – | 60.41 – | 67.36 1,2,6 | 67.36 1,2,6 | 70.24 1,2,3,4,6 | 62.88 1,2 |
| 8. | 70.20 2 | 67.84 – | 73.92 1,2,5,6 | 74.51 1,2,5,6 | 71.06 1,2 | 73.05 1,2,5 |
| 9. | 66.89 – | 67.32 – | 75.69 1,2 | 74.96 1,2 | 75.69 1,2 | 81.57 1,2,3,4,5 |
| 10. | 87.23 3,4 | 88.82 1,3,4 | 84.67 – | 85.13 – | 88.82 3,4 | 91.43 1,2,3,4,5 |
| 11. | 87.92 – | 87.92 – | 91.30 1,2 | 91.30 1,2 | 95.78 1,2,3,4 | 94.29 1,2,3,4 |
| 12. | 94.62 5 | 93.38 5 | 94.62 5 | 94.04 5 | 87.38 – | 95.54 1,2,3,4,5 |
| 13. | 73.04 – | 74.19 1 | 79.03 1,2 | 79.46 1,2 | 81.39 1,2,3,4 | 83.98 1,2,3,4,5 |
| 14. | 90.32 2 | 88.40 – | 90.02 2 | 90.02 2 | 96.77 1,2,3,4,6 | 92.35 1,2,3,4 |
| 15. | 95.18 3,4 | 95.18 3,4 | 93.47 – | 93.47 – | 99.64 1,2,3,4,6 | 96.23 3,4 |
| 16. | 62.42 – | 63.08 – | 68.00 1,2 | 67.18 1,2 | 70.15 1,2,3,4 | 71.83 1,2,3,4,5 |
| 17. | 67.23 4 | 67.23 – | 65.89 – | 65.89 – | 77.05 1,4,6 | 68.47 4 |
| 18. | 53.28 2 | 51.67 – | 53.99 1,2 | 55.45 1,2 | 57.46 1,2,3,4 | 60.33 1,2,3,4,5 |
| 19. | 80.41 – | 80.41 – | 84.16 1,2 | 84.89 1,2 | 92.32 1,2,3,4 | 94.79 1,2,3,4,5 |
| 20. | 77.06 5,6 | 76.72 5,6 | 80.48 1,2,5,6 | 80.96 1,2,5,6 | 62.96 – | 72.89 5 |
| Avg. rank | 4.25 | 4.92 | 3.53 | 3.21 | 2.87 | 2.19 |

**Table 6**
Results (accuracy [%]) for one-class and binary classifiers combined using Error-Correcting Output Codes (ECOC). Small numbers under accuracies stand for indexes of methods, from which the considered one is statistically superior.

| Dataset | Parzen[1] | MoG[2] | OCSVM[3] | SVDD[4] | C4.5[5] | SVM[6] |
|---|---|---|---|---|---|---|
| 1. | 67.54 – | 66.82 – | 73.55 1,2 | 74.38 1,2 | 76.34 1,2,3,4,6 | 74.23 1,2 |
| 2. | 85.89 3,4 | 83.72 – | 83.96 – | 84.21 – | 93.23 1,2,3,4 | 93.46 1,2,3,4 |
| 3. | 44.56 – | 44.56 – | 57.28 1,2,5 | 57.28 1,2,5 | 54.23 1,2 | 58.04 1,2,5 |
| 4. | 87.65 – | 87.65 – | 92.30 1,2 | 92.30 1,2 | 93.65 1,2 | 91.28 1,2 |
| 5. | 70.58 – | 71.18 – | 72.28 1,2 | 71.93 1 | 77.43 1,2,3,4,6 | 75.12 1,2,3,4 |
| 6. | 75.56 3,4,5,6 | 75.56 3,4,5,6 | 73.42 – | 73.42 – | 73.85 – | 74.05 – |
| 7. | 58.72 – | 58.72 – | 64.51 1,2,6 | 64.51 1,2,6 | 66.36 1,2,6 | 62.05 1,2 |
| 8. | 72.76 2,5,6 | 70.18 – | 74.53 1,2,5,6 | 75.03 1,2,5,6 | 70.84 – | 71.27 1 |
| 9. | 68.72 2 | 66.31 – | 75.98 1,2,5 | 76.36 1,2,5 | 73.48 1,2 | 78.25 1,2,5 |
| 10. | 89.90 3,4,5,6 | 90.21 3,4,5,6 | 87.68 5 | 88.04 5 | 86.73 – | 87.98 5 |
| 11. | 88.19 – | 88.19 – | 91.36 1,2 | 91.36 1,2 | 92.27 1,2 | 93.99 1,2,3,4,5 |
| 12. | 95.02 5 | 94.26 5 | 96.11 1,2,5,6 | 96.11 1,2,5,6 | 91.89 – | 94.80 5 |
| 13. | 72.68 – | 74.56 1 | 77.78 1,2 | 78.03 1,2 | 79.20 1,2 | 81.25 1,2,3,4,5 |
| 14. | 92.38 2 | 90.47 – | 91.74 2 | 91.74 2 | 95.00 1,2,3,4,6 | 91.82 2 |
| 15. | 96.46 6 | 96.46 6 | 95.18 6 | 95.18 6 | 97.23 3,4,6 | 94.29 – |
| 16. | 65.70 – | 66.28 – | 70.06 1,2 | 68.94 1,2 | 72.22 1,2,3,4 | 74.01 1,2,3,4,5 |
| 17. | 75.68 6 | 75.68 6 | 78.92 1,2,5,6 | 78.92 1,2,5,6 | 77.31 1,2,6 | 70.76 – |
| 18. | 57.10 5 | 55.23 5 | 57.98 1,2,5,6 | 58.69 1,2,5,6 | 55.32 – | 57.04 5 |
| 19. | 80.63 – | 80.63 – | 86.69 1,2 | 87.02 1,2 | 90.74 1,2,3,4 | 93.60 1,2,3,4,5 |
| 20. | 83.86 5,6 | 82.86 5,6 | 88.00 1,2,5,6 | 88.29 1,2,5,6 | 68.82 – | 82.03 5 |
| Avg. rank | 3.99 | 4.95 | 3.25 | 2.58 | 3.63 | 2.60 |

are able to successfully capture the nature of their target classes and sufficiently cover the decision space. Even without the access to counterexamples, one-class ensembles can display at the same time robustness to novelties (high true rejection rate, in our case ability to recognize objects from other classes) and a good generalization ability (which allows for a high true acceptance rate). This means that OCC can be in many cases an alternative to binary approaches, with easy decomposition, parallel implementation and high robustness to complex data structures.

But what is of greatest interest, are the situations in which one-class ensembles outperform in a statistically significant way binary counterparts. This happens in case of seven datasets: glass, led7digit, penbased, shuttle, vowel, yeast and auslan. Let us take a closer look at these particular situations.

First observation is the fact that five of these datasets (led7digit, penbased, vowel, yeast and auslan) are the four datasets with the highest number of classes among all the used benchmarks (from 10 to 95). In all these cases specific one-class ensembles returned statistically better performance than binary methods. This can be explained by the nature of used OVA and OVO decompositions. In case of OVO for a large number of classes the number of ensemble members in a committee rises significantly (for 10 classes, we need to train 45 classifiers). It is known that ensemble with a large number of base classifiers will have a high computational complexity and can behave unstable, especially with respect to their fusion method [63]. It is very possible that in such a large pool of classifiers some model will be non-competent [22], thus causing the aggregation function to drop the performance quality. One-class ensembles return significantly smaller ensembles (for 10 classes, we need to train 10 classifiers). This quality is also displayed by OVA decompositions, as they also train a number of base learners equal to the number of classes. However, for a large number of classes they will create a highly imbalanced dataset (for 10 classes with similar quantities of objects, after applying OVA, we get an imbalance ratio 1:9). This may cause a significant drop of the final accuracy. OCC is robust to such situations, as the negative examples are not used during the training process, thus there is no possibility of bias towards the majority class. Additionally, one-class ensembles have proven themselves as capable tools for dealing with imbalanced datasets [37].

In case of the two remaining datasets (glass and shuttle) can be explained by the robustness of one-class algorithms to some atypical properties of data. In these datasets there are several inner outliers within each class that can influence the shape of the

**Table 7**
Results (accuracy [%]) for one-class and binary classifiers combined using Decision Templates (DT). Small numbers under accuracies stand for indexes of methods, from which the considered one is statistically superior.

| Dataset | Parzen[1] | MoG[2] | OCSVM[3] | SVDD[4] | C4.5[5] | SVM[6] |
|---|---|---|---|---|---|---|
| 1. | 66.76 – | 65.49 – | 73.26 1,2 | 73.89 1,2 | 75.89 1,2 | 74.71 1,2 |
| 2. | 84.72 2 | 82.68 – | 83.04 – | 83.88 2 | 92.21 1,2,3,4 | 92.30 1,2,3,4 |
| 3. | 43.25 – | 43.25 – | 57.18 1,2,5,6 | 57.18 1,2,5,6 | 52.31 1,2 | 55.90 1,2,5 |
| 4. | 87.51 – | 87.51 – | 92.48 1,2,5,6 | 92.48 1,2,5,6 | 91.26 1,2,6 | 90.75 1,2 |
| 5. | 72.63 – | 73.11 – | 75.20 1,2,6 | 74.59 1,2,6 | 74.26 1,2,6 | 73.84 1,2 |
| 6. | 74.97 3,4 | 74.97 3,4 | 73.80 – | 73.80 – | 75.49 3,4 | 76.60 1,2,3,4,5 |
| 7. | 59.21 – | 59.21 – | 64.77 1,2,6 | 64.77 1,2,6 | 65.38 1,2,6 | 63.01 1,2 |
| 8. | 75.39 5,6 | 73.20 5,6 | 76.64 1,2,5,6 | 77.02 1,2,5,6 | 72.76 – | 74.11 5 |
| 9. | 67.73 – | 68.36 – | 75.60 1,2,5 | 75.60 1,2,5 | 70.24 1,2 | 75.60 1,2,5 |
| 10. | 87.69 5,6 | 88.21 5,6 | 86.62 5,6 | 87.06 5,6 | 84.56 – | 85.90 5 |
| 11. | 88.39 – | 88.39 – | 92.28 1,2,5 | 92.28 1,2,5 | 91.16 1,2 | 93.78 1,2,3,4,5 |
| 12. | 96.11 5,6 | 95.76 5,6 | 98.02 1,2,5,6 | 97.67 1,2,5,6 | 93.41 – | 95.23 5 |
| 13. | 74.16 – | 75.52 1 | 82.44 1,2,5,6 | 82.93 1,2,5,6 | 79.95 1,2 | 81.03 1,2,5 |
| 14. | 95.00 3,4,6 | 94.18 3,4,6 | 92.94 6 | 92.94 6 | 96.03 1,2,3,4,6 | 91.75 – |
| 15. | 97.89 6 | 97.89 6 | 96.32 6 | 96.32 6 | 96.99 6 | 95.38 – |
| 16. | 65.26 – | 65.79 – | 70.93 1,2 | 69.04 1,2 | 70.07 1,2 | 71.87 1,2,3,4,5 |
| 17. | 78.43 6 | 78.43 6 | 80.29 1,2,5,6 | 80.29 1,2,5,6 | 78.92 6 | 75.66 – |
| 18. | 60.06 1,5,6 | 57.98 – | 61.17 1,2,5,6 | 61.64 1,2,5,6 | 57.38 – | 58.85 5 |
| 19. | 82.19 – | 82.19 – | 90.38 1,2 | 90.65 1,2 | 91.09 1,2 | 94.26 1,2,3,4,5 |
| 20. | 85.39 5,6 | 84.72 5,6 | 88.68 1,2,5,6 | 89.37 1,2,5,6 | 69.18 – | 83.18 5 |
| Avg. rank | 3.79 | 4.87 | 2.58 | 2.19 | 4.45 | 2.72 |

decision plane, without carrying any useful information about the class distribution. In case of using boundary methods (SVDD and OCSVM), such examples can be easily filtered out and removed from the process of calculating the spherical decision boundary. Additionally, it is possible that OCSVM/SVDD, due to applied kernel mapping, was able to found a compact representation of data that was enclosed in a atomic hypersphere. This is allowed for a good separation of overlapping classes that could not be handled by a linear decision boundary in classical SVM or by too complex rules generated by the decision trees.

This shows us that applying the OCC for decomposition is a promising research track. However, it is not always straightforward to identify why exactly OCC displays such a good performance. By this analysis, we see a need for new measures that could give insight into the level of data complexity and its relation with OCC decomposition.

### 6.2. What type of one-class classifier is preferable for decomposition?

In previous section, we have showed that applying OCC for multi-class problems can lead to an increase in accuracy for specific cases. The next question that must be answered is which type of one-class classifier should be used for such a task? We have examined the representatives of two major families of OCC methods – density and boundary-based.

For density-based methods, we have tested Mixture of Gaussians Data Description or Parzen Density Data Description. By examining the results in the tables together with outputs of statistical tests, one can see that there does not exist a significant difference between them. For many cases both methods return identical performance, being able to similarly capture the density description of the classes. Although for few individual datasets one method outperforms the other, one cannot find a general trend among a number of comparisons that will point out the superiority of one method over the other.

The same observations hold for boundary-based methods. Here, we have tested One-Class Support Vector Machine and Support Vector Data Description. They both originate from binary SVM and work by mapping the target class data onto an enclosing hyperplane. However, the differences lie in the way the training procedure is conducted. However, for the considered multi-class decomposition purposes these two methods return very similar performance. For several datasets SVDD return slightly better accuracy, however statistical tests show that these differences are below the significance level.

Having a look at Table 8, which selected the best types of binary and one-class classifiers for each fusion method, gives a good outlook on the problem. For each case the boundary-based method (SVDD) was selected. It has many desirable properties that can be of great use in decomposing multi-class datasets:

- With the usage of kernel function, support-vector one-class classifiers are able to find a more compact representation of the data. This leads to a smaller volume of the enclosing hypersphere, which reduces the chance of overlap between classifiers assigned to different classes. Additionally, with well-selected kernel they are quite robust to the problem of so-called empty sphere [33] – region covered by the decision boundary, with no training objects within it. A classifier cannot be deemed as competent in such an empty sphere – thus in multi-class decomposition may have a high support value for an object in empty sphere, that in fact belong to a different class. Minimizing empty sphere size is one of the main challenges of OCC and SVDD/OCSVM optimization process handles it in a satisfactory way.
- SVDD/OCSVM can discard objects lying further from the center of target class distribution. Such objects can be treated as in-class outliers and lead to an over-expanded decision boundary. Density-based methods are very sensitive to such objects and may create a high overlap between ensemble members.
- SVDD/OCSVM works very well with a small number of objects, being able to derive a satisfactory decision boundary. Density methods fail in case of smaller datasets and this is one of their major disadvantages.

Although all this speaks in advantage of boundary-based methods, one cannot omit the density-based classifiers. The fact that SVDD completely dominated Table 8 is caused by the fact that algorithms were selected based on their average performance. Density-based methods, such as examined Mixture of Gaussians Data Description or Parzen Density Data Description, require a significant number of objects to work properly. In cases, where the number of objects per class is high and the ratio of in-class outliers is very low or equal to zero, density methods can outperform boundary-based classifiers. This happens for four datasets (flare, nursery, segment and shuttle). So in case of well-sampled datasets, it is worthwhile to check the performance of the density methods for decomposition purposes. For

**Table 8**
Comparison of best models (one-class and binary) for each of the five examined fusion methods. Small numbers under accuracies stand for indexes of methods, from which the considered one is statistically superior.

| Dataset | MAX | | PC | | DDAG | | ECOC | | DT | |
|---|---|---|---|---|---|---|---|---|---|---|
| | SVDD[1] | SVM[2] | SVDD[3] | SVM[4] | SVDD[5] | SVM[6] | SVDD[7] | SVM[8] | SVDD[9] | SVM[10] |
| 1. | 71.95 2,3 | 67.42 – | 70.45 2 | 74.12 1,2,3,5 | 71.05 2 | 74.73 1,2,3,5 | 74.38 1,2,3,5 | 74.23 1,2,3,5 | 73.89 1,2,3,5 | 74.71 1,2,3,5 |
| 2. | 88.64 3,5,7,9 | 89.12 3,5,7,9 | 82.91 – | 92.76 1,2,3,5,7,9 | 84.02 – | 92.51 1,2,3,5,7,9 | 84.21 – | 93.46 1,2,3,5,7,9,10 | 83.88 – | 92.30 1,2,3,5,7,9 |
| 3. | 55.87 3,5 | 57.53 1,3,5,10 | 48.29 – | 58.62 1,3,5,10 | 50.78 3 | 58.69 1,2,3,5,7,9,10 | 57.28 1,3,5,10 | 58.04 1,3,5,10 | 57.18 1,3,5,10 | 55.90 3,5 |
| 4. | 95.41 3,4,5,6,7,8,9,10 | 95.75 3,4,5,6,7,8,9,10 | 91.24 – | 93.82 3,7,8,10 | 93.05 3,7,8,10 | 93.95 3,7,8,10 | 92.30 3,7,8,10 | 91.28 – | 92.48 3,7,8,10 | 90.75 3,7,8,10 |
| 5. | 72.74 – | 77.43 1,3,5,7,8,9,10 | 72.30 – | 77.26 1,3,5,7,8,9,10 | 72.89 – | 77.36 1,3,5,7,8,9,10 | 71.93 – | 75.12 1,3,5,7,9,10 | 74.59 1,2,3,5,7,10 | 73.84 1,2,3,5,7 |
| 6. | 74.91 3,5,7 | 75.49 3,5,7,9 | 69.32 – | 74.95 3,5,7 | 71.65 3 | 75.19 3,5,7,9 | 73.42 3,5 | 74.05 3,5 | 73.80 3,5 | 76.60 *ALL* |
| 7. | 62.89 2 | 60.84 – | 65.11 1,2,3,6,8,9 | 63.22 2 | 67.36 *ALL* | 62.88 2 | 64.51 1,2,3,6,8,9 | 62.05 2 | 64.77 1,2,3,6,8,9 | 63.01 2 |
| 8. | 75.65 2,3,4,6,8,9 | 71.20 – | 74.13 2,4,6,8 | 72.76 2,7 | 74.51 2,4,6,8 | 73.05 2,7 | 75.03 2,4,6,7 | 71.27 – | 77.02 *ALL* | 74.11 2,4,6,8 |
| 9. | 76.72 3,5 | 82.27 1,3,5,7,8,9,10 | 73.74 – | 81.54 1,3,5,7,8,9,10 | 74.96 3 | 81.57 1,3,5,7,8,9,10 | 76.36 3,5 | 78.25 1,3,5,7,9,10 | 75.60 3 | 75.66 3 |
| 10. | 86.62 3,5 | 91.05 1,3,5,7,8,9,10 | 84.59 – | 91.82 1,3,5,7,8,9,10 | 85.13 – | 91.43 1,3,5,7,8,9,10 | 88.04 1,3,5,9,10 | 87.98 1,3,5,10 | 87.06 3,5 | 85.90 3 |
| 11. | 92.05 3 | 95.34 1,3,5,7,8,9,10 | 90.68 – | 94.58 1,3,5,7,9,10 | 91.30 3 | 94.29 1,3,5,7,9,10 | 91.36 3 | 93.99 1,3,5,7,9 | 92.28 3 | 93.78 1,3,5,7,9 |
| 12. | 95.76 2,3,5 | 90.45 – | 93.87 2 | 95.15 2,3,5 | 94.04 2 | 95.54 2,3,5 | 96.11 2,3,4,5 | 94.80 2,3,5 | 97.67 *ALL* | 95.23 2,3,5 |
| 13. | 81.96 3,5,7 | 82.02 3,4,7,10 | 76.64 – | 83.47 1,2,3,5,7,8,10 | 79.46 3,7 | 83.98 1,2,3,5,7,8,10 | 78.03 3 | 81.25 3,5,7 | 82.93 1,2,3,5,7,8,10 | 81.03 3,5,7 |
| 14. | 90.47 3 | 91.21 3,5 | 88.02 – | 92.81 1,2,3,5 | 90.02 3 | 96.77 *ALL* | 91.74 1,2,3,5 | 91.82 1,2,3,5 | 92.94 1,2,3,5,7,8,10 | 91.75 3,5 |
| 15. | 94.25 2,3 | 92.74 – | 92.71 – | 96.17 1,2,3,5,8 | 93.47 – | 96.23 1,2,3,5,8 | 95.18 1,2,3,5,8 | 94.29 2,3 | 96.32 1,2,3,5,7,8,10 | 95.38 1,2,3,5 |
| 16. | 69.33 3,5 | 75.15 *ALL* | 65.44 – | 72.93 1,3,5,7,9 | 67.18 3 | 71.83 1,3,5,7,9 | 68.94 3,5 | 74.01 1,3,4,5,6,7,8,9,10 | 69.04 3,5 | 71.87 1,3,5,7,9 |
| 17. | 70.09 2,3,4,5,6 | 51.23 – | 65.32 2 | 69.78 2,3,5 | 65.89 2 | 68.47 2,3,5 | 78.92 1,2,3,4,5,6,8,10 | 70.76 2,3,4,5,6 | 80.29 *ALL* | 75.66 1,2,3,4,5,6,8 |
| 18. | 57.98 2,3,5 | 56.20 3,5 | 55.21 – | 58.39 2,3,5,8 | 55.45 – | 60.33 1,2,3,4,5,6,8,10 | 58.69 1,2,3,5,8 | 57.04 2,3,5 | 61.64 *ALL* | 58.85 1,2,3,5,8 |
| 19. | 87.96 3,5 | 96.05 *ALL* | 84.39 – | 95.12 1,3,5,7,8,9 | 84.89 – | 94.79 1,3,5,7,9 | 87.02 3,5 | 93.60 1,3,5,7,9 | 90.65 1,3,5,7 | 94.26 1,3,5,7,9 |
| 20. | 82.86 2,3,4,5,6,8,10 | 74.12 4,6 | 80.23 2,4,6 | 70.38 – | 80.96 2,4,6 | 72.89 4 | 88.29 1,2,3,4,5,6,8,10 | 82.03 2,3,4,5,6 | 89.37 *ALL* | 83.18 2,3,4,5,6,8 |
| Avg. rank | 5.64 | 4.42 | 10.00 | 3.33 | 8.89 | 3.68 | 4.25 | 5.72 | 3.86 | 5.21 |

**Table 9**
Shaffer test for examined combination methods with Parzen Density Data Description as base classifier. Symbol '=' stands for classifiers without significant differences, '+' for situation in which the method on the left is superior and '−' vice versa.

| Hypothesis | *p*-Value |
|---|---|
| MAX vs PC | + (0.0402) |
| MAX vs DDAG | + (0.0369) |
| MAX vs ECOC | −(0.0287) |
| MAX vs DT | −(0.0108) |
| PC vs DDAG | =(0.3379) |
| PC vs ECOC | −(0.0196) |
| PC vs DT | −(0.0182) |
| DDAG vs ECOC | −(0.0311) |
| DDAG vs DT | −(0.0486) |
| ECOC vs DT | =(0.3981) |

datasets with smaller number of objects however, this family of classifiers should be omitted.

One should note that OCSVM/SVDD could be optimized for each dataset separately, which could lead to a further improvement in quality. However, even with standard setting the performance of this classifier is very good – therefore it proves its robustness to different kinds of data.

### 6.3. Which type of fusion method should we chose?

Results of the Shaffer post hoc test between the different aggregation methods are depicted in Tables 9–12.

From them, we can see that in general the performance of fusion methods is independent from the type of base classifier used. The only exception is the DDAG fuser, which works better (in a statistically significant way) with boundary-based classifiers.

However, one should take a closer look on the obtained *p*-values. Although general output of the test is independent from the classifier, we can see that for OCSVM/SVDD one gets a more significant tests, as *p*-values indicate a more stable performance for SVDD. For an example let us take a test DDAG vs DT. For Parzen classifier, we get an information that DT are better with *p*-value equal to 0.0494 – very close to the significance level $\alpha = 0.05$. For OCSVM/SVDD, we get the same output but with *p*-value equal to 0.0208, which shows that in this case the differences are much more significant.

**Table 10**
Shaffer test for examined combination methods with Mixture of Gaussians Data Description as base classifier. Symbol '=' stands for classifiers without significant differences, '+' for situation in which the method on the left is superior and '−' vice versa.

| Hypothesis | *p*-Value |
|---|---|
| MAX vs PC | + (0.0417) |
| MAX vs DDAG | + (0.0326) |
| MAX vs ECOC | −(0.0274) |
| MAX vs DT | −(0.0132) |
| PC vs DDAG | =(0.3589) |
| PC vs ECOC | −(0.0174) |
| PC vs DT | −(0.0191) |
| DDAG vs ECOC | −(0.0332) |
| DDAG vs DT | −(0.0501) |
| ECOC vs DT | =(0.4104) |

**Table 11**
Shaffer test for examined combination methods with OCSVM as base classifier. Symbol '=' stands for classifiers without significant differences, '+' for situation in which the method on the left is superior and '−' vice versa.

| Hypothesis | *p*-Value |
|---|---|
| MAX vs PC | + (0.0306) |
| MAX vs DDAG | + (0.0414) |
| MAX vs ECOC | −(0.0407) |
| MAX vs DT | −(0.0146) |
| PC vs DDAG | −(0.0489) |
| PC vs ECOC | −(0.0220) |
| PC vs DT | −(0.0141) |
| DDAG vs ECOC | −(0.403) |
| DDAG vs DT | −(0.0192) |
| ECOC vs DT | =(0.4074) |

OVO schemes delivered the worst performance amongst all of the fusion methods. This is in opposite to findings presented in [21], which stated that OVO is a preferable choice. This can be explained by the nature of one-class learners – they are de facto a specific case of OVA (target class vs all possible outliers). Hence, the OVA methods had been slightly modified in order to work with OCC. As statistical test prove, they are on average significantly inferior to OVA and trained fusers.

Max scheme turns out a very suitable fusion method for one-class ensembles, despite its simplicity. Although in some cases it is outperformed by ECOC and DT, we should remember that it is much less computationally expensive than trained fusers. Hence can be of use in areas, where we need a fast training of recognition system, e.g., in data stream classification.

Trained fusers, represented by ECOC and DT, returned the best results among the examined methods. Compared between each other, we can see that they behaved mutually exclusive on some datasets. From the analysis of results in Table 8, one may see that their performance is strongly dependent on the number of classes. ECOC outperforms DT in cases when the number of classes is $\leq 5$. This can be explained by high correction power in case of shorter codewords. However, with the increase of the number of classes, increases the advantage of DT over ECOC. For all cases with 10, 11 or 95 classes, DT were statistically superior. Generating codewords for many classes leads to over-lengthy codewords with reduced correction power. Templates do not suffer from this problem, as they are based on averaged outputs of classifiers for a given class.

**Table 12**
Shaffer test for examined combination methods with SVDD as base classifier. Symbol '=' stands for classifiers without significant differences, '+' for situation in which the method on the left is superior and '−' vice versa.

| Hypothesis | *p*-Value |
|---|---|
| MAX vs PC | + (0.0298) |
| MAX vs DDAG | + (0.0407) |
| MAX vs ECOC | −(0.0382) |
| MAX vs DT | −(0.0128) |
| PC vs DDAG | −(0.0462) |
| PC vs ECOC | −(0.0181) |
| PC vs DT | −(0.0104) |
| DDAG vs ECOC | −(0.377) |
| DDAG vs DT | −(0.0200) |
| ECOC vs DT | =(0.3829) |

## 7. Conclusions and future works

In this paper, we have evaluated a hypothesis that one-class classification can be effectively used for handling multi-class datasets. Although OCC discards information about the counter-examples, its major advantage lies in its training principle. It captures the unique properties of the target class. Therefore, for multi-class decomposition it does not try to find best separation boundaries – it aims at creating individual descriptions of each of considered classes. This leads to a completely different competence space of ensemble classifier, that seems an attractive solution for complex data in which the standard classifiers are biased towards one of the classes.

We have presented a thorough study on the applicability of one-class classifiers for the process of decomposing multi-class datasets. We have compared one-class ensembles with popular binarization algorithms. We have checked the performance of four most popular classifiers from these groups and their correlation with five examined fusion methods, dedicated to reconstructing the original dataset from individual decisions.

Experiments, backed-up with a series of statistical tests, confirmed our hypothesis. For many cases one-class ensembles achieve similar performance to binary committees, and for some more complex cases significantly outperform traditional methods. This is especially interesting, due to the fact that one-class classifiers do not utilize the information about classes other than the target concept.

After careful experiments, we can present the following set of rules for the end-user for addressing decomposition problems with one-class methods:

1. One-class classification is not a universal solution. For standard data, with no difficulties embedded in them, binary decomposition performed much better having access to all the data. However, if you have a more complex problem, OCC is a worthwhile direction.
2. One-class decomposition works exceptionally well with datasets having a high number of classes. This is due to the lower number of classifiers in a committee in comparison to OVO, and avoiding class imbalance in comparison to OVA.
3. One-class classifiers can deal with problems embedded in the nature of the data - imbalanced distribution, feature and label noise or small number of objects. These problems are common in multi-class classification.
4. The differences between one-class classifiers from the same family of models are very small. It is more worthwhile to consider different groups of OCC models than examine many models from a single group.

5. When having a dataset with moderate or low number of examples and/or risk of being affected by noise/in-class outliers, it is preferable to apply boundary methods, like OCSVM or SVDD. However, when the class objects are plentiful, density-based methods can give a better description of the data.
6. Fusion is as important in one-class decomposition as in binarization. In contrary to binary classifiers, OVO aggregation methods do not deliver satisfactory results. It is recommended to use OVA fuser or trained combiners (if additional training time is not a problem in the specific application).

In future, we would like to gain more insight into the areas of applicability of one-class classifier decomposition. We plan to use novel measures for grading the levels of complexity for multi-class datasets and search for correlations between the complexity and performance of one-class learners.

## Conflict of interest

None declared.

## Acknowledgments

Bartosz Krawczyk and Michał Woźniak were partially supported by the Polish National Science Centre under the Gant PRELUDIUM no. DEC-2013/09/N/ST6/03504 realized in years 2014-2016.

Francisco Herrera was partially supported by the Spanish Ministry of Education and Science under Project TIN2011-28488 and the Andalusian Research Plan P10-TIC-6858, P11-TIC-7765.

## References

[1] E.L. Allwein, R.E. Schapire, Y. Singer, Reducing multiclass to binary: a unifying approach for margin classifiers, J. Mach. Learn. Res. 1 (2) (2001) 113–141.
[2] E. Alpaydin, Combined 5 × 2 cv f test for comparing supervised classification learning algorithms, Neural Comput. 11 (8) (1999) 1885–1892.
[3] E. Alpaydin, E. Mayoraz, Learning error-correcting output codes from data, in: IEEE Conference Publication, vol. 2, 1999, pp. 743–748.
[4] M. Badura, A. Szczurek, P.M. Szecówka, Statistical assessment of quantification methods used in gas sensor system, Sens. Actuators B: Chem. 188 (2013) 815–823.
[5] M.A. Bagheri, G.A. Montazer, E. Kabir, A subspace approach to error correcting output codes, Pattern Recognit. Lett. 34 (2) (2013) 176–184.
[6] T. Ban, S. Abe, Implementing multi-class classifiers by one-class classification methods, in: IEEE International Joint Conference on Neural Networks—Conference Proceedings, 2006, pp. 327–332.
[7] A. Bartkowiak, R. Zimroz, Outliers analysis and one class classification approach for planetary gearbox diagnosis, J. Phys. Conf. Ser. 305 (1) (2011).
[8] C. Bellinger, S. Sharma, N. Japkowicz, One-class versus binary classification: which and when? in: 2012 11th International Conference on Machine Learning and Applications (ICMLA), vol. 2, 2012, pp. 102–106.
[9] R. Burduk, P. Trajdos, Construction of sequential classifier using confusion matrix, in: Computer Information Systems and Industrial Management—Proceedings of 12th IFIP TC8 International Conference, CISIM 2013, Krakow, Poland, September 25–27, 2013, pp. 401–407.
[10] R. Burduk, M. Zmyslony, Decomposition of classification task with selection of classifiers on the medical diagnosis example, in: 7th International Conference on Hybrid Artificial Intelligent Systems, HAIS 2012, vol. 7209, Lecture Notes in Artificial Intelligence of Lecture Notes in Computer Science, 2012, pp. 569–577.
[11] K. Cao, L. Pang, J. Liang, J. Tian, Fingerprint classification by a hierarchical classifier, Pattern Recognit. 46 (12) (2013) 3186–3197.
[12] B. Chen, A. Feng, S. Chen, B. Li, One-cluster clustering based data description, Jisuanji Xuebao/Chin. J. Comput. 30 (8) (2007) 1325–1332.
[13] Y. Chen, X.S. Zhou, T.S. Huang, One-class svm for learning in image retrieval, in: IEEE International Conference on Image Processing, vol. 1, 2001, pp. 34–37.
[14] K.F. Cheung, Fuzzy one-mean algorithm: formulation, convergence analysis, and applications, J. Intell. Fuzzy Syst. 5 (4) (1997) 323–332.
[15] G. Cohen, H. Sax, A. Geissbuhler, Novelty detection using one-class parzen density estimator. An application to surveillance of nosocomial infections, in: Studies in Health Technology Informatics, vol. 136, 2008, pp. 21–26.
[16] B. Cyganek, Color image segmentation with support vector machines: applications to road signs detection, Int. J. Neural Syst. 18 (4) (2008) 339–345.
[17] B. Cyganek, S. Gruszczynski, Hybrid computer vision system for drivers eye recognition and fatigue monitoring, Neurocomputing 126 (2014) 78–94.
[18] J. Demsar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[19] T.G. Dietterich, G. Bakiri, Solving multiclass learning problems via error-correcting output codes, J. Artif. Int. Res. 2 (January) (1995) 263–286.
[20] B. Fei, J. Liu, Binary tree of svm: a new fast multiclass training and classification algorithm, IEEE Trans. Neural Netw. 17 (3) (2006) 696–704.
[21] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, F. Herrera, An overview of ensemble methods for binary classifiers in multi-class problems: experimental study on one-vs-one and one-vs-all schemes, Pattern Recognit. 44 (8) (2011) 1761–1776.
[22] M. Galar, A. Fernández, E. Barrenechea Tartas, H. Bustince Sola, F. Herrera, Dynamic classifier selection for one-vs-one strategy: avoiding non-competent classifiers, Pattern Recognit. 46 (12) (2013) 3412–3424.
[23] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Inf. Sci. 180 (10) (2010) 2044–2064.
[24] G. Giacinto, R. Perdisci, M. del Rio, F. Roli, Intrusion detection in computer networks by a modular ensemble of one-class classifiers, Inf. Fusion 9 (January) (2008) 69–82.
[25] T. Hastie, R. Tibshirani, Classification by pairwise coupling, Ann. Stat. 26 (2) (1998) 451–471.
[26] K. Hempstalk, E. Frank, Discriminating against new classes: one-class versus multi-class classification, Lecture Notes in Artificial Intelligence of Lecture Notes in Computer Science, vol. 5360, 2008, pp. 325–336.
[27] J. Hong, J. Min, U. Cho, S. Cho, Fingerprint classification using one-vs-all support vector machines dynamically ordered with naidotlessve bayes classifiers, Pattern Recognit. 41 (2) (2008) 662–671.
[28] C. Hsu, C. Lin, A comparison of methods for multiclass support vector machines, IEEE Trans. Neural Netw. 13 (2) (2002) 415–425.
[29] H. Jabeen, A.R. Baig, Two-stage learning for multi-class classification using genetic programming, Neurocomputing 116 (2013) 311–316.
[30] A.K. Jain, R.P.W. Duin, Jianchang Mao, Statistical pattern recognition: a review, IEEE Trans. Pattern Anal. Mach. Intell. 22 (January (1)) (2000) 4–37.
[31] N. Japkowicz, S. Stephen., The class imbalance problem: a systematic study, Intell. Data Anal. 6 (5) (2002) 429–449.
[32] H. Jiang, G. Liu, X. Xiao, C. Mei, Y. Ding, S. Yu, Monitoring of solid-state fermentation of wheat straw in a pilot scale using ft-nir spectroscopy and support vector data description, Microchem. J. 102 (2012).
[33] P. Juszczak, Learning to recognise. A study on one-class classification and active learning (Ph.D. thesis), Delft University of Technology, 2006.
[34] P. Juszczak, D.M.J. Tax, E. Pekalska, R.P.W. Duin, Minimum spanning tree based one-class classifier, Neurocomputing 72 (7–9) (2009) 1859–1869.
[35] M.W. Koch, M.M. Moya, L.D. Hostetler, R.J. Fogler, Cueing, feature discovery, and one-class learning for synthetic aperture radar automatic target recognition, Neural Netw. 8 (7–8) (1995) 1081–1102.
[36] B. Krawczyk, One-class classifier ensemble pruning and weighting with firefly algorithm, Neurocomputing 150 (2015) 490–500.
[37] B. Krawczyk, M. Woźniak, Diversity measures for one-class classifier ensembles, Neurocomputing 126 (2014) 36–44.
[38] B. Krawczyk, M. Woźniak, B. Cyganek, Clustering-based ensembles for one-class classification, Inf. Sci. 264 (2014) 182–195.
[39] L. Kuncheva, J.C. Bezdek, R.P.W. Duin, Decision templates for multiple classifier fusion: an experimental comparison, Pattern Recognit. 34 (2) (2001) 299–314.
[40] L.I. Kuncheva, Using measures of similarity and inclusion for multiple classifier fusion by decision templates, Fuzzy Sets Syst. 122 (3) (2001) 401–407.
[41] T.C.W. Landgrebe, D.M.J. Tax, P. Paclík, R.P.W. Duin, The interaction between classification and reject performance for distance-based reject-option classifiers, Pattern Recognit. Lett. 27 (8) (2006) 908–917.
[42] K.L. Li, H.K. Huang, S.F. Tian, A novel multi-class svm classifier based on ddag, in: Proceedings of 2002 International Conference on Machine Learning and Cybernetics, vol. 3, 2002, pp. 1203–1207.
[43] B. Liu, Z. Hao, E.C.C. Tsang, Nesting one-against-one algorithm based on svms for pattern classification, IEEE Trans. Neural Netw. 19 (12) (2008) 2044–2052.
[44] Y. Liu, Z. You, L. Cao., A novel and quick svm-based multi-class classifier, Pattern Recognit. 39 (11) (2006) 2258–2264.
[45] L. Manevitz, M. Yousef, One-class document classification via neural networks, Neurocomputing 70 (7–9) (2007) 1466–1481.
[46] F. Masulli, G. Valentini, Comparing decomposition methods for classification, in: Proceedings of International Conference on Knowledge-Based Intelligent Electronic Systems, KES, vol. 2, 2000, pp. 788–791.
[47] H. Nickisch, C.E. Rasmussen, Approximations for binary gaussian process classification, J. Mach. Learn. Res. 9 (2008) 2035–2078.
[48] S. Park, J. Fürnkranz, Efficient prediction algorithms for binary decomposition techniques, Data Min. Knowl. Discov. 24 (1) (2012) 40–77.
[49] R. Rifkin, A. Klautau, In defense of one-vs-all classification, J. Mach. Learn. Res. 5 (2004) 101–141.
[50] B. Schölkopf, A.J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, Adaptive Computation and Machine Learning, MIT Press, Cambridge, MA, USA, 2002.
[51] S. Sonnenburg, G. Rtsch, C. Schfer, B. Schlkopf, Large scale multiple kernel learning, J. Mach. Learn. Res. 7 (2006) 1531–1565.

[52] D.M.J. Tax, R.P.W. Duin, Combining one-class classifiers, in: Proceedings of the Second International Workshop on Multiple Classifier Systems, MCS '01, Springer-Verlag, London, UK, 2001, pp. 299–308.
[53] D.M.J. Tax, R.P.W. Duin, Support vector data description, Mach. Learn. 54 (1) (2004) 45–66.
[54] D.M.J. Tax, P. Juszczak, E. Pekalska, R.P.W. Duin, Outlier detection using ball descriptions with adjustable metric, in: Proceedings of the 2006 Joint IAPR International Conference on Structural, Syntactic, and Statistical Pattern Recognition, SSPR'06/SPR'06, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 587–595.
[55] D.M.J. Tax, K. Müller, A consistency-based model selection for one-class classification, in: Proceedings of the International Conference on Pattern Recognition, vol. 3, 2004, pp. 363–366. Cited By (since 1996):12.
[56] D.M.J. Tax, Robert P. W. Duin, Characterizing one-class datasets, in: Proceedings of the Sixteenth Annual Symposium of the Pattern Recognition Association of South Africa, 2005, pp. 21–26.
[57] O. Taylor, J. MacIntyre, Adaptive local fusion systems for novelty detection and diagnostics in condition monitoring, in: Proceedings of SPIE—The International Society for Optical Engineering, vol. 3376, 1998, pp. 210–218.
[58] M. Tohme, R. Lengelle, Maximum margin one class support vector machines for multiclass problems, Pattern Recognit. Lett. 32 (13) (2011) 1652–1658.
[59] A. Wang, W. Yuan, J. Liu, Z. Yu, H. Li., A novel pattern recognition algorithm: combining art network with svm to reconstruct a multi-class classifier, Comput. Math. Appl. 57 (11–12) (2009) 1908–1914.
[60] T. Wilk, M. Woźniak, Complexity and Multithreaded Implementation Analysis of One Class-classifiers Fuzzy Combiner, in: E. Corchado, M. Kurzyński, M. Woźniak (Eds.), Hybrid Artificial Intelligent Systems of Lecture Notes in Computer Science, vol. 6679, Springer, Berlin/Heidelberg, 2011, pp. 237–244.
[61] T. Wilk, M. Woźniak, Soft computing methods applied to combination of one-class classifiers, Neurocomputing 75 (January) (2012) 185–193.
[62] T. Windeatt, R. Ghaderi, Coding and decoding strategies for multi-class learning problems, Inf. Fusion 4 (1) (2003) 11–21.
[63] M. Woźniak, M. Grana, E. Corchado, A survey of multiple classifier systems as hybrid systems, Inf. Fusion 16 (1) (2014) 3–17.
[64] C. Yeh, Z. Lee, S. Lee, Boosting one-class support vector machines for multi-class classication, Appl. Artif. Intell. 23 (4) (2009) 297–315.
[65] H. Zuo, O. Wu, W. Hu, B. Xu, Recognition of blue movies by fusion of audio and video, in: Proceedings of 2008 IEEE International Conference on Multimedia and Expo, ICME 2008, 2008, pp. 37–40.

**Bartosz Krawczyk** received a B.Sc. Engineering degree in Computer Science in 2011 and M.Sc. degree with distinctions in 2012 from Wroclaw University of Technology, Poland. He was awarded as the best M.Sc. graduate by the Rector of Wroclaw University of Technology. He is currently a Research Assistant and a Ph.D. Candidate in the Department of Systems and Computer Networks at the same university. His research is focused on machine learning, multiple classifier systems, one-class classifiers, class imbalance, and interdisciplinary applications of these methods. So far, he has published more than 90 papers in international journals and conferences. He was awarded with numerous prestigious awards for his scientific achievements like IEEE Richard E. Merwin Scholarship, PRELUDIUM and ETIUDA grants from Polish National Science Center, Scholarship of Polish Minister of Science and Higher Education or START award from Foundation for Polish Science among others. He served as a Guest Editor in four special issues of journals devoted to ensemble learning and data stream classification. He is a member of Program Committee for over 40 international conferences and a reviewer for dozen of journals.

**Michał Woźniak** is a Professor of Computer Science in the Department of Systems and Computer Networks, Wroclaw University of Technology, Poland. He received a M.S. degree in biomedical engineering in 1992 from the Wroclaw University of Technology, and Ph.D. and D.Sc. (habilitation) degrees in computer science in 1996 and 2007 respectively from the same university. His research focuses on machine learning, distributed algorithms and teleinformatics. He has published over 200 papers, three books, and has edited eight ones. He has been involved in several research projects related to the above-mentioned topics, moreover, he has been a consultant on several commercial projects for well-known Polish companies and for public administration. He is a senior member of the IEEE and a member of International Biometric Society.

**Francisco Herrera** received his M.Sc. in mathematics in 1988 and Ph.D. in mathematics in 1991, both from the University of Granada, Spain. He is currently a Professor in the Department of Computer Science and Artificial Intelligence at the University of Granada. He has been the supervisor of 28 Ph.D. students. He has published more than 240 papers in international journals. He is coauthor of the book "Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases" (World Scientific, 2001).

He currently acts as Editor in Chief of the international journal "Progress in Artificial Intelligence" (Springer). He acts as an area editor of the International Journal of Computational Intelligence Systems and associated editor of the journals: IEEE Transactions on Fuzzy Systems, Information Sciences, Knowledge and Information Systems, Advances in Fuzzy Systems, and International Journal of Applied Metaheuristics Computing; and he serves as a member of several journal editorial boards, among others: Fuzzy Sets and Systems, Applied Intelligence, Information Fusion, Evolutionary Intelligence, International Journal of Hybrid Intelligent Systems, Memetic Computation, and Swarm and Evolutionary Computation.

He received the following honors and awards: ECCAI Fellow 2009, IFSA 2013 Fellow, 2010 Spanish National Award on Computer Science ARITMEL to the "Spanish Engineer on Computer Science", International Cajastur "Mamdani" Prize for Soft Computing (Fourth Edition, 2010), IEEE Transactions on Fuzzy System Outstanding 2008 Paper Award (bestowed in 2011), and 2011 Lotfi A. Zadeh prize best paper Award of the International Fuzzy Systems Association.

His current research interests include computing with words and decision making, bibliometrics, data mining, big data, data preparation, instance selection, fuzzy rule based systems, genetic fuzzy systems, knowledge extraction based on evolutionary algorithms, memetic algorithms and genetic algorithms.