Forming and pruning one-class classifier ensembles

Wrocław University of Technology

Bartosz Krawczyk

Department of Systems and Computer Networks

Wrocław University of Technology, Wyb. Wyspianskiego 27, 50-370 Wrocław, Poland.

A thesis submitted for the degree of

PhilosophiæDoctor (PhD)

year month

This thesis is dedicated to people who are most dear to me: my mother Grażyna, my grandmother Alina and to the loving memory of my grandfather Bogusław for all their love, encouragement and understanding. Words cannot express my gratitude.

Rozprawę doktorską dedykuję najbliższym mi osobom: mojej Mamie Grażynie, Babci Alinie oraz pamięci mojego Dziadka Bogusława w podziękowaniu za całą miłość, wsparcie oraz zrozumienie. Słowa nie są w stanie wyrazić mej wdzięczności.

03.07.2015, Wrocław.

Acknowledgements

This thesis marks an ending of an important period in my life. These three years of doctoral studies were a stimulating experience and a definite step forward, yet at the same being a great and fun adventure. I had an invaluable luck of my work being at the same time an exciting hobby.

I would like to extend my deepest gratitude to a person who started this all - my supervisor Prof. Michał Woźniak. He is the sole person who got me interested in machine learning and convinced me to start an academic career, opening a new path in my life. One could not have a better supervisor and mentor. For all the time and discussions not only about scientific matters, for all the support, inspiration, motivation, endless patience, and cordiality I extend my deepest gratitude. One can hardly ask for more.

I am most thankful to my supervisor Prof. Francisco Herrera for a fantastic collaboration, being a great mentor on both professional and personal level, numerous stimulating discussions, and making Granada such a special place in my heart.

I would like to show appreciation to all fantastic people I have pleasure working with: Prof. Bogusław Cyganek, Prof. Salvador Garcia, Dr. Mikel Galar, Dr. Isaac Triguero, Dr. Jose Antonio Saez, Prof. Andrzej Szczurek, Prof. Monika Maciejewska, Dr. Paweł Filipczuk, and Dr. Łukasz Jeleń. Thank you for all our joint research, ideas and inspirations.

I give thanks to my colleagues from Department of Systems and Computer Networks: Róża Goścień, Tomasz Kucofaj, Dr. Wojciech Kmiecik, Dr. Konrad Jackowski, Paweł Ksieniewicz and others for friendly atmosphere and all the help during my studies.

To all my friends for their support, love, commitment and patience with my various antics. To my personal animal farm of two dogs and five cats for always finding a way to pull me away from writing this thesis.

This is the end of something, yet at the same time a beginning of a new journey.

I eagerly look forward to it.

Contents

Abbreviations Used

AUC	area under curve
CONS	consistency measure
DIV	diversity measure
EM	expectation-minimization
G-mean	geometric mean
MCS	multiple classifier system
OCC	one-class classification
OCSVM	One-Class Support Vector Machine
PCA	principal component analysis
RBF	Gaussian radial basis function
ROC	receiver-operating characteristic
SOM	self-organizing map
SVD	singular value decomposition
SVDD	support vector data description
SVM	support vector machine
WOCSVM	weighted one-class support vector machine

Symbols Used

$\mathbf{A}$	vector with feature subsets
a	center of the sphere
$a^{(l)}(x^{(q)})$	q-th feature used by l-th individual classifier
α_i	i-th Lagrange multiplier
B	bootstrap sample of given size
$\mathbf{B}$	binary mask for classifier selection
β	importance weighting factor in multi-criteria optimization
C	number of clusters
$\mathbf{C}$	vector with cluster centroids
c_i	i-th centroid
$\mathbf{CSM}$	classifiers support matrix
d	number of features (attributes)
$\mathrm{dst}(\cdot,\cdot)$	distance between two objects (Euclidean measure used)
Δ_m	mutation range factor
E	number of pairwise overlaps between a set of spherical classifiers
η	mutation probability
ϵ	minimal firefly movement
$f_j(x)$	probability density function for j-th class
F_k	support for k-th class
G	number of classifiers that can correctly classify i-th training object
γ	mutation probability
H	number of competence areas
$\mathcal{HC}$	hypercube
$\mathcal{HS}$	hypersphere
$I(\cdot)$	indicator function
K	number of classifiers in the competence area ensemble
$K(\cdot,\cdot)$	kernel function
L	number of classifiers in the ensemble
$L(i,j)$	loss function for incorrectly assigning object from class j to class i
λ	exponential distribution
ι	number of slack variables for considered classifier

M	number of classes
$\mathcal{M}$	set of classes
μ_{ij}	membership value of j-th object into i-th cluster
N	number of objects in the learning set
N_c	the upper limit of algorithm cycles
N_f	number of free parameters
N_p	the population quantity
N_S	quantity of features in a subset
ν	penalization factor of slack variables
O	number of iterations without results improvement that indicate overfitting
p_j	prior probability for j-th class
Ψ	classifier
$\bar{\Psi}$	ensemble classifier
$\hat{\Psi}_h$	local ensemble in h-th competence area
Π	pool of classifiers
$Q(\cdot)$	optimization goal function
R	radius of a sphere
$\mathbf{R}$	rotation matrix
S	number of feature subsets
s	scale parameter for mapping distance into support
T	temperature for simulated annealing
τ	light absorption coefficient
θ	one-class classification threshold
Θ	distance threshold between fireflies groups
$\mathbf{U}$	membership matrix
Υ	firefly light intensity
$\mathbf{W}$	vector with classifier weights
$w_i^{(l)}$	weight assigned to l-th classifier for i-th class
$\mathfrak{w}_i$	weight assigned to i-th object
$V.$	volume of a given object
$\varkappa$	firefly attractiveness
ϱ	cooling factor for simulated annealing
ς	size of the tabu list
x	object (example)
ξ	slack variable
$\mathcal{X}$	feature space
y_i	i-th coefficient returned by a feature extraction / reduction method
ζ	threshold for One-Class Energy diversity measure
$\|\cdot\|$	Euclidean norm
$\langle\cdot,\cdot\rangle$	inner product

1

Introduction

In this chapter, we will present an introduction to the topics discussed in this thesis. We will provide a formulation of the pattern classification task on which the entire content of this thesis is based.

Then background information about the one-class classification task will be presented. We show the formulation of the classification without an access to counterexamples and discuss its potential outputs. We present how to evaluate such classifiers and next a detailed presentation of a number of one-class algorithms is given. A taxonomy of existing single-class learning algorithms is proposed. Most important algorithms are presented in details, with special focus on two methods frequently used in this thesis: One-Class Support Vector Machine and Support Vector Data Description.

We will discuss the idea of classifier ensembles, reviewing the methods for combining classifiers. We focus on how to create a pool of useful classifiers and present a thorough literature review from this domain. Next, a ensemble pruning task is characterized and state-of-the-art approaches from different groups of methods are given in detail. We show how to combine the outputs of individual classifiers and present a review of different fusion approaches used in ensemble learning.

On the basis of these sections, we introduce the concept of ensemble learning in one-class classification. We present the current state-of-the-art in this domain, discuss their potential drawbacks and formulate directions to be explored in this area. Finally a research hypothesis is being formulated, together with aims and goals of this thesis.

1.1 Machine Learning Task

Learning is the crucial part of the development of human beings. With this process were are able to gather new information and improve our base of knowledge. New concepts, ideas, tools and solutions can extend our possessed information and allow us to make more efficient decisions. The increased learning abilities were one of the main factors that allowed for the rapid expansion and development of the human race.

However, such boost in intellectual capabilities comes not only from the amount of stored data, but also from how one is able to efficiently use this information. It should to analyze and draw conclusions from incoming situations that may differ from the ones we were previously in. Additionally, a feedback from the changing environment must also be taken into consideration and one must quickly adapt to the new conditions, select useful information and forget the outdated one. Only then we are able to generalize our knowledge and use it in new unseen situations.

Let's quote a definition of learning given by Marvin Minsky "*changes in behavior caused by observation so that solve similar problem better in the future (i.e. according to a criterion)*".

People learn from data. This may be provided in a plethora of forms: books, pictures, movies, conversations or lectures, to name a few. We are able to extract what we want to learn from these sources and use this specific information to serve our purposes.

The quality of the learning material has a significant impact on the process. With little or no prior knowledge in a considered domain we are dependent on the source of incoming information. Thus to learn efficiently we should knowledge coming from expert sources that has been verified and tested. Thus often one requires a supervisor, who will provide a guide in the starting phase. When our knowledge increases, we are able to critically evaluate the learning material and discard uncertain sources.

Not only the quality of the knowledge itself is a major factor in the effectiveness of the learning process, but also the knowledge representation. One may present the same information in a clear, structured and easy-to-follow form, or in a chaotic, inconsistent and details-lacking manner. That is why humans have always looked for efficient representation of information that will concentrate on the core of the idea in a understandable and reproducible way. This would also allow to automatize the decision making process on the basis of available knowledge. In 1862 in Luxor (Egypt) American egyptologist Edwin Smith found the papyrus (circa 1650-1550 BC) which is able to deal with cranial and spine injuries. here the knowledge was presented in a form of "if-then" rules that allowed for an easy reproduction of the diagnosis process. In the 17th century mathematician Gottfried F. Leibnitz expressed a hope that at some point it will be possible to solve each problem with the usage of logic by using logic. His popular quote says "*Let us calculate [calculemus], without further ado, to see who is right*". Thence, according to Leibnitz's postulate *Calculemus!*, we are looking for computational methods of solving the real problems which could be "algorithmized" efficiently.

Modern computing systems generate massive amounts of data that needs to be transferred, processed and stored. It is estimated that the volume of generated data doubles each year[1]. This forces a rapid development in computer networks, distributed computing systems, data warehouses, and decision support systems.

Such data often carries useful and highly valuable information that can be efficiently put to use once extracted. However, the complexity, amount and velocity of arriving data makes it impossible for a human expert to analyze efficiently in a reasonable amount of time. This has lead to the development of algorithms for mining collections of data and searching for underlying patterns. Such methods are able not only to simply analyze the data, but to learn from it in order to improve their competences and be able to generalize the extracted knowledge over new samples that may appear in the future.

This domain is known as machine learning, pattern recognition or data mining and originate from statistics and computer science [5]. It is a focus of intense research in the last decades, as there is an ever-increasing need for new and efficient automatic algorithms for this task. When designing learning systems one may distinguish three main types of the learning step, depending on the available information [132]:

- **Supervised learning** assumes that we have *a prior* knowledge about a supplied dataset in a form of labels. These labels are usually provided by an expert. Such a information allows us to guide the learning process and used the supplied labels to learn dependencies between the characteristics of data and its true state. Then a trained model is used on new, previously unseen objects to output a decision on

[1]http://www.ni.com/newsletter/51649/en/

the basis of the learned function or rules. This shows that the supplied dataset must be allow us to generalize the extracted knowledge to new data.

- **Unsupervised learning** assumes that we have no access to labels. Thus we need to explore the given dataset in order to find some relations, dependencies or structures within the set that may allow us to gain an insight into its properties. Usually, output of an unsupervised learning must be further analyzed by a domain expert in order to verify and evaluate it.
- **Semi-supervised learning** assumes that the cost of labeling the entire dataset is high, as it requires an external expert. Therefore, we only need to have a part of our data labeled and use it to guide the learning step with the help of a larger collection of unlabeled data. For example one may use the labeled examples to automatically label the remaining objects on the basis of their similarity or consistency, creating a self-training systems.

This types of learning can be used in several main data analytics tasks:

- **Classification** aims at identifying to which of a set of categories (classes) a new observation belongs, on the basis of a training set of data containing labeled observations whose true class is known (usually given by an expert) [76].
- **Clustering** aims at discovering some underlying structures and connections in the analyzed dataset. It groups a set of objects in such a way that objects in the same group (cluster) are more similar (according to a given metric) to each other than to those in other clusters [131].
- **Regression** aims at estimating the relationships among variables, with the focus on the relationship between a dependent variable and one or more independent variables to form a regression function [123]. While the classification process has a discrete output, regression outputs a continuous value.
- **Association rule learning** aims at discovering interesting relations between variables in large databases [312]. It identifies the rules observed within databases, allowing for an interpretable mining of dependencies describing some processes or events.

In this thesis we will concentrate on the classification task.

Despite being present in the research community for several decades, the classification algorithms are still being developed as intensely as ever. Contemporary machine learning must adopt to new problems in the age of big data. Canonical classifiers have been developed with a view to manage thousands of objects. However, nowadays such datasets are considered as small ones. Advent of big data brought the idea of 4Vs (Variety, Velocity, Veracity and Volume) into view [317]. Such datasets are massive collections of objects that can arrive in real-time and change their nature periodically. This has lead to development of two important tracks in machine learning: big data processing and data stream mining [157]. Other novel problems and research directions emerge constantly, such as imbalanced classification [258], deep learning [6], or multi-label classification [231]. This makes machine learning a vital and exciting field.

1.1.1 Pattern Classification Model

We will now present the formal model of classification. $\mathcal{X}$ denotes feature space and $x \in \mathcal{X}$ is the example, i.e., x is the so-called feature vector which informs about attribute values. We will assume that we have d attributes at our disposal:

$$x = \left\{x^{(1)}, x^{(2)}, ..., x^{(d)}\right\}, \text{ and } x \in \mathcal{X} = \mathcal{X}^{(1)} \times \mathcal{X}^{(2)} \times ... \times \mathcal{X}^{(d)} \tag{1.1}$$

In many notations the feature set is replaced by the feature vector:

$$x = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ ... \\ x^{(d)} \end{bmatrix}, \text{ and } x \in \mathcal{X} = \mathcal{X}^{(1)} \times \mathcal{X}^{(2)} \times ... \times \mathcal{X}^{(d)} \tag{1.2}$$

where $x^{(l)} \in \mathcal{X}^{(l)}$.

From the formal point of view we assume that:

$$x \in \mathcal{X} \subseteq \mathbf{R}^d \tag{1.3}$$

Each of the attributes is used in the classification process to partition the feature space into decision regions associated to each of the classes according to feature values. However, one needs to be careful when working with examples described by a high number of features. Correct class generalization becomes exponentially harder as the dimensionality of the examples grows. This phenomenon is known as the so-called the *curse of dimensionality* [19, 73]. Also, the Hughes effect may take place. It stands for a situation where we have at disposal a finite number of observations in a high-dimensional feature space with each feature having a number of possible values. In such a case an enormous amount of training data are required to ensure that there are several samples with each combination of values.

To counter these drawbacks, one may reduce the dimensionality of the feature space. There are two main approaches for dimensionality reduction in classification [102]:

- **Feature selection** finds a reduced subset of s features from the original feature space:

$$x = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ \dots \\ x^{(d)} \end{bmatrix} \rightarrow \overline{x} = \begin{bmatrix} \overline{x}^{(1)} \\ \overline{x}^{(2)} \\ \dots \\ \overline{x}^{(s)} \end{bmatrix}, \quad s < d \tag{1.4}$$

- **Feature reduction** establish a mapping from the original d dimensional feature space into a new s dimensional one.

$$x = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ \dots \\ x^{(d)} \end{bmatrix} \rightarrow \overline{x} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(s)} \end{bmatrix} = f\left(\begin{bmatrix} x^{(1)} \\ x^{(2)} \\ \dots \\ x^{(d)} \end{bmatrix}\right), \quad s < d \tag{1.5}$$

  The disadvantage of this approach lies in forming a completely new feature space. Thus we lose any interpretability, as we usually do not know what is the meaning behind new features.

As mentioned before, the classification goal is to assign a given object described by its features x into one of the predefined categories, also called labels. Let $\mathcal{M} = \{1, ..., M\}$ denotes set of labels.

The classification algorithm is a given function Ψ with domain $\mathcal{X}$ and codomain $\mathcal{M}$:

$$\Psi \, : \, \mathcal{X} \rightarrow \mathcal{M}. \tag{1.6}$$

In most cases the decision of the classifier is made on the basis of support functions that return support values for each of considered classes. To make a decision, we usually use the maximum rule

$$\Psi(x) = \max_{k \in \mathcal{M}} \left(F_k(x)\right), \tag{1.7}$$

where $F_k(x)$ is the support for k-th class.

In order to construct a classifier one needs to have a proper dataset to be used in the learning process. The learning set $\mathcal{LS}$ groups observations from a given domain in a form of pairs:

$$\mathcal{LS} = \{(x_1, j_1), (x_2, j_2), ..., (x_N, j_N)\}, \tag{1.8}$$

where x_k denotes the feature vector of the k-th learning pattern and j_k is its correct classification.

Each element in this set comprises the observation of a given object and its true state (class label). However, one must be aware of the fact that the supplied learning set is usually obtained through manual analysis and labelling, thus being prone to error. If the source of labels is a human expert, then he may be tired, inattentive, or simply mistaken. If labels are provided by some automatic manner, noises and mistakes can appear. Therefore an incorrect labeling may occur [239], leading to a decrease in the quality of the learning process. The second major problem is the correctness of the features values [63], which could be related to measurement error, operator (who introduced the data) mistakes, or malicious activity. These factors and their impact on the quality of the learning set must be taken into consideration when designing a system based on the machine learning algorithms.

1.1.2 Probabilistic Approach

Due to the imperfect nature of the training information as well as classifier models, we need to deal with the problem of uncertainty in the pattern classification and machine learning domains. When classifier outputs a class label, we do not have any information about how certain it is in making this prediction. However, most of the learning models may return a continuous decision support. This can be canonically understood as the *posterior* probability typically associated with probabilistic models of the pattern recognition task [67], although any continuous output of classifier may be used [35].

Statistical decision theory is a highly popular and effective approach for managing uncertainty in the pattern classification problems [76]. Here it is assumed that both the feature vector $x \in \mathcal{X}$ and its class label $j \in \mathcal{M}$ should be considered as observed values of a pair of random variables $(\mathbf{X}, \mathbf{J})$. The probability distribution of this pair of variables is given by the *prior* class probabilities:

$$p_j = P(\mathbf{J} = j),\ j \in \mathcal{M} \tag{1.9}$$

and class-conditional probability density function of $\mathbf{X}$

$$f_j(x) = f(x|j),\ x \in \mathcal{X},\ j \in \mathcal{M}. \tag{1.10}$$

From the statistical point of view, our designed classifier should minimize the average cost of misclassification. If the cost related to the error on all of classes is equal, then this becomes equivalent to making the smallest number of mistakes. However, in many real-life application cost connected with some mistake is higher than for others. Let us consider a medical example of breast cancer diagnosis [156]. Here we deal with a two class problem: benign and malignant. If we diagnose a benign cases as a malignant one, then the cost is connected with additional medical examinations and discomfort imposed upon the patient. When diagnosing a malignant case as a benign one, we delay the treatment procedure and put the patients' health and life in jeopardy. Therefore, one may associate a higher cost with the second scenario in order to penalize such mistakes.

To embed this in the pattern classification framework, we have to formulate the so-called loss function which allows us to measure the cost of decisions between two classes:

$$L : \mathcal{M} \times \mathcal{M} \to \mathcal{X}, \tag{1.11}$$

where $L(i, j)$ returns the lost from incorrectly assigning object from class j to class i.

This allows us to formulate the criterion of classification task for the optimal Bayes classifier:

$$\min_{\Psi} Risk(\Psi) = Risk(\Psi^*), \tag{1.12}$$

where

$$Risk(\Psi) = E[L(i, j)] = \int_{\mathcal{X}} \sum_{j=1}^{M} L(\Psi(x), j) p_j f_j(x) dx. \tag{1.13}$$

The $Risk(\Psi)$ is the average risk of the classifier Ψ, that should be minimized. Let us notice that it is enough to minimize the so-called conditional risk:

$$r_i(x) = \underset{\mathbf{J}|x}{E}[L(i,j)] = \sum_{j=1}^{M} L(i,j)p_i(x). \tag{1.14}$$

This allows us to formulate the following decision rule for the optimal Bayes classifier:

$$\Psi^*(x) = i \text{ if } \sum_{j=1}^{M} L(i,j)p_j(x) = \min_{k \in \mathcal{M}} \sum_{j=1}^{M} L(k,j)p_j(x) \tag{1.15}$$

where the *posterior* probability $p_j(x)$ can be calculated from the Bayes formula:

$$p_j(x) = \frac{p_j f_j(x)}{\sum\limits_{k=1}^{M} p_k f_k(x)} \tag{1.16}$$

For this one needs to input the proposed loss function. As for most of the classification problems we do not have the costs given beforehand, a very popular 0-1 loss function is commonly used. It returns 1 in the case of error and 0 otherwise:

$$L(i,j) = \begin{cases} 0 & if \;\; i = j \\ 1 & if \;\; i \neq j \end{cases} \tag{1.17}$$

This leads to the following formulation of the optimal (Bayes) recognition algorithm Ψ^* decision rule that aims at minimizing the probability of misclassification:

$$\Psi^*(x) = i \text{ if } p_i(x) = \max_{k \in \mathcal{M}} p_k(x), \tag{1.18}$$

Please note that this defined loss function is connected to class label with the highest *posterior* probability. In this case, the conditional risk is equal to the probability of the failure to recognize an object x and it makes sense to the average risk of a misclassification probability:

$$Risk\,(\Psi^*) = P_e(\Psi^*) = \sum_{j=1}^{M} p_j \int\limits_{D_j} f_j(x)dx = 1 - \int\limits_{X} \max_{j \in \mathcal{M}} p_j f_j(x)dx = 1 - P_c(\Psi^*) \tag{1.19}$$

One should remember that this is an theoretical model of a classifier - in practice we do not have the full probabilistic information about the problem and thus must use classifiers that will be able to approximate the properties of classes, at the cost of the increased error possibility. There is a plethora of methods for conducting the learning and classification process. From simple minimal-distance classifiers, ones based on density and probability, through decision rules and trees, up to *Neural Networks* and *Support Vector Machines* [132]. Nevertheless, there is not a single pattern recognition algorithm that is appropriate for all the tasks we are faced with, since each classifier has its own domain of competence and is bound to make different errors [291].

1.1.3 Overfitting

During the classifier training and exploitation phases we may observe two types of errors to occur:

- The training error is the ratio of the misclassified object from the learning set to its cardinality:

$$P_e^{\mathcal{LS}}(\Psi) = \frac{\sum_{k=1}^{N} [\Psi(x_k) = j_k]}{|\mathcal{LS}|} \tag{1.20}$$

- The real error, called also generalization error, is the number of misclassified objects drawn from the general population that occurs during the testing phase of a trained classifier:

$$P_e(\Psi) = \int_X P(\Psi(x) \neq i|x) f(x) dx \tag{1.21}$$

As we have at our disposal the learning set $\mathcal{LS}$ during the classifier training step we can precisely establish the training error, but only estimate the generalization error. Of course, the real aim of the pattern classification is to obtain the lowest possible real error. This can be achieved by assuming that the training set is a representative subsample of the problem domain and using it to train a model that can efficiently generalize the discovered properties over the entire decision space. This is not directly translatable into achieving a low classification error on the training set, as the classifier my not learn the general properties but the properties of individual examples. This problem is known as overfitting.

This happens when the classifier loses its generalization skills, i.e., in the given moment it starts to memorize learning set instead of generalizing knowledge contained in it. When the learning process was conducted excessively, was too complicated or the training set is not sufficient the learner may adjust to extremely specific random features of the training data that have no causal relation to the target function. Once overfitting occurs the accuracy performed on the training examples still increases while the accuracy on new data degrades (and thus the real error increases). Therefore, it is important to prevent overfitting through proper learning procedures and validation.

This can be presented formally as [213]:

Definition 1 *Classifier Ψ overfits learning data $\mathcal{LS}$ if there is an alternative classifier Ψ' such that*

$$P_e(\Psi) > P_e(\Psi') \ \ \textit{and} \ \ P_e^{\mathcal{LS}}(\Psi) < P_e^{\mathcal{LS}}(\Psi') \tag{1.22}$$

in practice instead of P_e, we should use its estimator P_e^{VS} based on validation set (VS) i.e., set of demarcated objects from the learning set which are not presented during learning procedure.

Often overfitting is being connected with too complex learning model. Thus, finding a bias between the complexity and accuracy may alleviate this problem. In machine learning one often cites the Occam's razor rule that means *simpler hypothesis less susceptible to overfitting* [1], which can be interpreted that the simplest classifiers are the most preferable ones because the simplest explanation is most likely to be the correct one. In practice there are many contrary examples to it [73] and according to Wolpert's *No Free Lunch theorem* [291] implies that simplicity does not imply accuracy. Simpler hypotheses should be preferred because simplicity is a highly desirable property, not because of any hypothetical direct link with accuracy.

1.1.4 Bias and Variance

We have defined two types of errors. However, one should keep in mind that they may be connected with different properties of our learning task. One may decompose the error into three components [145]:

- Error related to a given problem that cannot be eliminated and it is lower bounded by the error of optimal Bayes classifier (1.18).

[1]The principle was formulated by logician and Franciscan friar William of Occam (1287-1347) *Entia non sunt multiplicanda praeter necessitatem* (entities must not be multiplied beyond necessity)

- Bias related to the made assumptions about classifier model.
- Variance related to a given learning set.

Let us take a closer look on these components.

The (...) bias of a learning algorithm is the set of assumptions that the learner uses to predict outputs given inputs that it has not encountered [1]. As mentioned in previous sections, there is no universal scheme on how to design a classification algorithm [291], due to different areas of competence of each of them. On the other hand the *Ugly Ducking theorem* [76] states that each classifier is biased towards some type of specific problems. Assumptions are necessary for classifier training, but they increase the bias at the same time. Reducing the bias causes the variance to be very high, as we need a large number of training examples to do this. This is called the *bias-variance dilemma* [94] - how to find an optimal trade-off between bias and variance of a given classification model. This is also known as the model selection procedure. Several approaches are considered. Firstly, we have to mention the practical approach that tests several models using cross validation. Vapnik [280] proposed to order the available models according to their complexity, e.g., *Vapnik-Chervonenkis* dimension and choose the simplest one which achieve the best quality.

1.1.5 Evaluating Classifiers

One may compare classifiers according to the number of possible criteria:

- Accuracy, misclassification error or risk (for loss functions different from the 0-1 version).
- Computational or memory complexities of training or decision phases.
- Cost of classifier construction and decision making.
- Interpretability of outputted decision and ability to gain an insightful knowledge from it.
- Robustness to changes and drifts in data.

[1]Tom M. Mitchell, Machine learning, McGraw-Hill, New York, 1997.

Let us concentrate on the first case. As mentioned before, the training error must not be used to compare two algorithms (their respective errors on training set $\mathcal{TS}$), so we need an external validation set ($\mathcal{VS}$) that is different from the training one. The most common approach to obtain this is to divide the set of available examples $\mathcal{LS}$ into $\mathcal{TS}$ and $\mathcal{VS}$.

A single run of training and validation scheme is not enough due to the following factors:

- For many practical tasks our access to data is limited, so $\mathcal{TS}$ and $\mathcal{VS}$ may be of small size and spoiled by outliers and noise.
- Many classifiers are stochastic in nature and depend on some random factors that affect strongly their generalization abilities (such as the choice of starting point or stopping criteria).
- To alleviate the problem of randomness, one may use an identical algorithm to generate several classifiers that are going to be validated over different $\mathcal{VS}$s to obtain a sample of validation errors.
- We can assess the expected error of learning method for a given problem or compare it with other learning algorithms on the basis of distribution of these validation errors.

Therefore, when designing a machine learning experiment a method that will allow for a robust estimation of error via multiple independent validations is required. One must chose the most proper one according to the size and type of analyzed data. Let us focus on the most popular one, namely **cross validation** (CV).

CV requires a fixed number of $\mathcal{TS}$s and $\mathcal{VS}$s obtained from our learning set $\mathcal{LS}$. If our original dataset is of huge size, then we could divide it into k parts and each part into $\mathcal{TS}$ and $\mathcal{VS}$. Unfortunately, in real tasks we usually use the same data differently splitted k times due to the limited viability of data. One should keep $\mathcal{VS}$ and $\mathcal{TS}$ as large as possible, because it guarantees that error estimates are robust. At the same time we also have to keep the overlap between different sets as small as possible. Further, classes can be represented in the right proportions as the subsets of data are drawn (*prior* probabilities are not disturbed) and there are two most popular frameworks as follows:

1. ***k*-Fold Cross-Validation**, where k is typically set to 10. To get more reliable error estimators, we could repeat this procedure several times (e.g. 10x10-fold cross validation). *Leave-one-out cross validation* (LOOCV) is a method dedicated for very small datasets that uses only a single sample for validation in each replication.

2. **5×2 fold cross-validation** [72], where LS is randomly divided into two equal-size subsets. First part is used as $\mathcal{TS}$, second as $\mathcal{VS}$. Then we swap their roles. To get the next fold, we shuffle LS randomly and repeat the previous step. Further, we can do it more than five folds, but Dietterich [72] points out that after five folds sets have shared many instances and overlap so much. Thus statistics calculated from the sets are too dependent.

One needs to assure the following conditions in order to make the results significant and suitable for statistical comparison:

- We cannot affect the sampling in any way and the order of appearance of data should be randomly determined.

- When setting the values of parameters the experiment should be run a number of times to average the influence of both human-determined and underlying factors. Parameters selection should be carried out with respect to their influence on each other, e.g., via the gird-search.

- When comparing different classifiers, each of them must be trained and tested on exactly the same subsets of data. This way we ensure that obtained differences in accuracies are dependent only on the properties of algorithms and not on the different distributions of datasets.

1.2 One-Class Classification

Methods mentioned in the previous section are designed for scenarios where we have two or more classes. In order to properly train a classifier one needs to have a representative sample of each class in the training set. In many real-life problems we have an access only to objects from a single class.

Let us consider an example in form of a process of monitoring the nuclear power plant. Data arrive as a continuous stream of sensor outputs. We would like to conduct a non-stop surveillance of the parameters of the plant for an early detection of any irregularities. It is easy to gather labelled examples of proper behaviour of such a plant, but counterexamples are obviously dangerous to collect. Even if we simulate some potential failures we cannot be sure that this set is exhaustive. In such a critical application one cannot allow for a poorly-designed classification system. On the other hand, a fully automatic and intelligent monitoring system would certainly benefit the safety and effectiveness of work in such a plant.

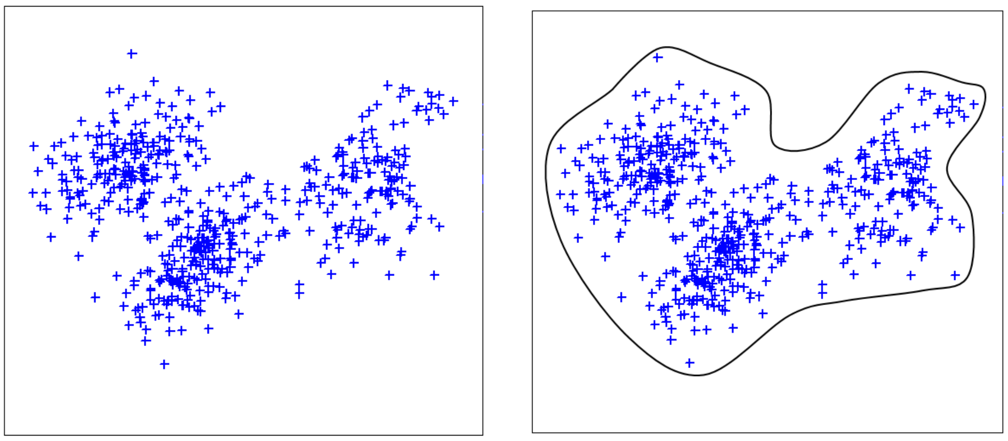
For such scenarios the one-class classification (OCC) approach has been developed [133, 267]. Let us now briefly describe this area of machine learning.

1.2.1 Learning in the Absence of Counterexamples

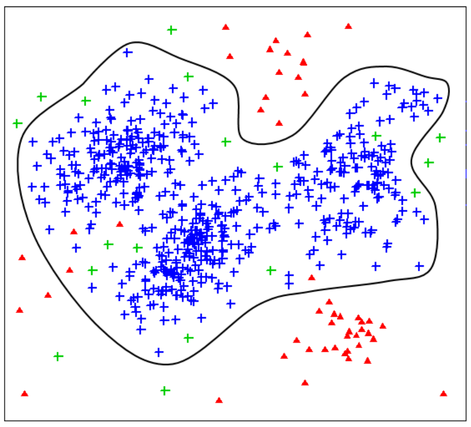
One-class classification is a specific area of machine learning that aims at distinguishing a given class (denoted as target concept ω_T) from a more broad set of classes [214]. All other objects that do not satisfy the conditions of ω_T are labeled as outliers ω_O. OCC assumes that the counterexamples are unavailable during the training, therefore an OCC learning algorithm needs to estimate the classification rules only on the basis of positive samples [24]. At the same time it must display good generalization properties as during the exploitation phase both objects from the target concept and unseen outliers may appear. OCC aims at finding a trade-off between capturing the properties of the target class (too fit or too lose boundary may lead to high false rejection / false acceptance rates) and maintaining a good generalization (as overfitting is likely to occur when having only objects from one class for training). The idea of learning in the absence of counterexamples is depicted in Figure 1.1.

One-class classification is an attractive solution for many real-life problems where data from a single class is easily obtained but access to counterexamples is limited or restricted, such as in intrusion detection systems [140, 144]. It has also gained interest as a powerful tool for analyzing data streams for novel concepts [200] and for handling imbalanced datasets [209].

Let us formulate the OCC task. We assume that our model is working in a d-dimensional feature space $\mathcal{X}$ and deals with a one-class problem described by a set of class labels $\mathcal{M} = \{\omega_T, \omega_O\}$.



(a) Single-class data. **(b)** Trained one-class classifier.



(c) Exploitation phase.

Figure 1.1: One-class toy problem with three main stages: (**a**) during the training phase only positive objects are available; (**b**) a one-class classifier is trained on the data, enclosing all the relevant samples, while not being overfitted to data; (**c**) during the exploitation phase new objects appear that can be labeled as the target concept (positive samples that should be accepted) or outliers (negative objects that should be rejected).

In case of OCC, we may have four different classification outcomes:

- Object belonging to the target class is labeled as such (correct classification). We will denote the fraction of such objects by TP.
- Object belonging to the target class is labeled as outlier (incorrect classification). We will denote the fraction of such objects by FN.

- Object belonging to the outlier class is labeled as outlier. (correct classification). We will denote the fraction of such objects by TN.
- Object belonging to the outlier class is labeled as one belonging to the target class (incorrect classification). We will denote the fraction of such objects by FP.

Table 1.1 summarizes the possible decisions made in OCC scenario.

Table 1.1: Four possible classification outcomes in one-class scenarios.

	object from ω_T	object from ω_o
classified as ω_T	true positive TP	false positive FP
classified as ω_O	false negative FN	true negative TN

This may seem as a binary classification task, but the biggest difference lies in the learning procedure [271]. In the standard dichotomy problems we may expect objects from the other classes to predominantly come from one direction. The available class should be separated from all the possible outliers - this leads to a situation where a decision boundary should be estimated in all directions in the feature space around the target class. A description of differences between binary and one-class classifiers is depicted in Figure 1.2.

To compute the error rate when training a classifier, one needs to have an access to both the probability density function for the target class $f_{\omega_T}(x)$ and the probability density function for the outlier class $f_{\omega_O}(x)$. As in OCC during the training step only objects from a given class may be used, only the $f_{\omega_T}(x)$ may be estimated [133]. This allows us to minimize the false rejection rate, but nothing else. W can trivially satisfy this criterion by accepting all objects as target class representatives. Without example outlier objects or an estimate of the outlier distribution $f_{\omega_O}(x)$, it is not possible to estimate the number of outlier objects which will be accepted by the OCC model.

To avoid a naive solution of accepting every possible object, one must make some assumptions about the nature of possible outliers. The commonly used solution is to assume that the outliers are uniformly distributed around the representatives of the target class [268].

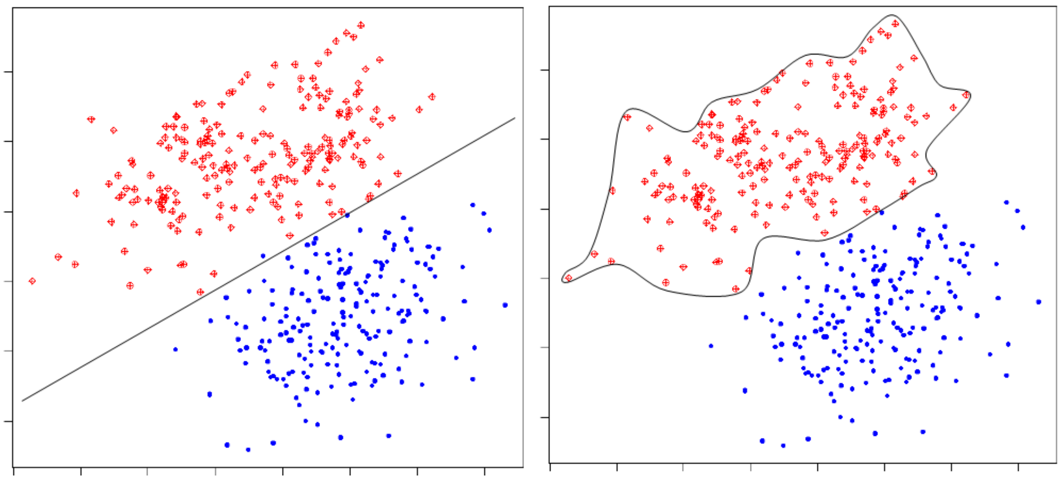


Figure 1.2: The difference between binary and one-class classifier. (*Left*) A toy data problem handled by the binary classifier. (*Right*) The same dataset analyzed with the usage of the one-class classifier, with a single class serving as the target concept.

The posterior probability for the target class can be computed by usage of Bayes rule:

$$p(\omega_T|x) = \frac{p(\omega_T)p(x|\omega_T)}{p(x)} = \frac{p(\omega_T)p(x|\omega_T)}{p(\omega_T)p(x|\omega_T) + p(\omega_O)p(x|\omega_O)}, \tag{1.23}$$

where with the assumption of the uniform distribution of outliers, we may compute this with only information about ω_T. Additionally, we can use $p(x|\omega_T)$ instead of $p(\omega_T|x)$.

When an uniform distribution is assumed, minimization of FP factor will lead to obtaining a description of target class that finds a trade-off between the minimal volume encompassing all of the objects from ω_T and generalization capabilities over ω_T. This can be used to train a one-class classifier without any knowledge about FP. By minimizing FN and the volume of the descriptor, one can obtain a good data description [269]. When the true distribution of outliers differs from the uniform one such a data description may not be the optimal one. Without an access to the counterexamples during the classification stage one needs to accept such a solution.

Due to the lack of counterexamples the training error can only be assumed on the target class. To define the training error on the outlier data, artificial counterexamples must be created [270] or a measure of the volume of the data description must be used.

Another problem lies in the complexity of a one-class model. For canonical classification, smoothness constraints on the output of a given classifier are often enforced. For one-class classifiers not only the smoothness is required, but also additional constrains on outputting a closed boundary around the target data. It increases the complexity of the problem and enhances the effects of other difficulties, such as the curse of dimensionality [267].

One may also use a definition of one-class decision making process on the basis of discriminants (or support functions). In such a case a classifier Ψ should base its decision on a set of support functions $\Big(F_{\omega_T}(x), F_{\omega_O}(x)\Big)$. This discriminant value represents a support of the considered classifier for object x belonging either to target or outlier classes. As we deal with OCC, we only need $F_{\omega_T}(x)$ to make a decision considering object x.

Classification algorithm Ψ makes a decision using the following rule:

$$\Psi(x) = \begin{cases} \omega_T \text{ if } F_{\omega_T}(x) \geq \theta \\ \omega_O \text{ otherwise} \end{cases} \tag{1.24}$$

where θ stands for a classification threshold, responsible for the degree of acceptance assigned to an one-class classifier. Such a measure should be fitted according on how certain should be classifier in order to accept new sample as one belonging to ω_T. In case of $\theta = 0.5$ and usage of a 0-1 loss function, we will get a threshold equal to the one used in binary classification.

To apply the mentioned above decision rule, we require the knowledge about the values of support functions of each individual classifier from the pool. Not all of one-class classifiers can output it directly - some of them work on the basis of distance between the new sample and its decision boundary (known also as reconstruction error). Therefore, a heuristic mapping is used:

$$F_{\omega_T}(x) = exp(-dst(x, \omega_T)/s), \tag{1.25}$$

where $dst(x, \omega_T)$ stands for a distance (usually Euclidean metric is used) between the considered object x and decision boundary for ω_T (this depends on the nature of used classifier, e.g., support vectors or nearest neighbor may be used) and s is the scale parameter that should be fitted to the target class distribution. This scale factor is related to how spread out your data points are. When the distance between objects tend to get very high (e.g., in high dimensional spaces) small value of s is used to control the stability of the mapping. Therefore, in most cases $s = \frac{1}{d}$. This mapping has the advantage that the outputted support value is always bounded between 0 and 1.

1.2.2 Evaluating One-Class Classifiers

The role of threshold θ (from Eq. (1.24)) is of a high importance to OCC domain. Selecting its different values will directly result in a different trade-off between TP and TN. Different values of threshold will give different accepted distances from the decision boundary (in case of distance-based classifiers) or resemblance measures (in case of model-based classifiers). Majority of one-class classifiers compute their own characteristics on the basis of learning set $\mathcal{TS}$ independently of the threshold. The threshold can also be derived directly from the $\mathcal{TS}$ being adjusted to accept a predefined fraction of the target class. For a given target acceptance rate $TPR = \frac{TP}{TP+FN}$ the threshold θ can be defined as follows:

$$\theta = \min_{\theta \in [0,1]} \left(\left(\frac{1}{N} \sum_{i=1}^{N} I\Big(F_{\omega_T}(x_i) \geq \theta \Big) \right) = TPR \right), \tag{1.26}$$

where I stands for an indicator function.

An ideal situation would occur when we could use a separate validation set $\mathcal{VS}$ for estimating this threshold. Such a separate set would increase the robustness to overfitting. This comes with the requirement of additional data - which can be a problem in many real-life OCC applications. Most of the one-class methods can use the same set for estimating the distance or support values and threshold θ.

With the use of Eq.(1.26) we can compute a threshold θ on the training set for different values of TP. Then, we can measure the FP on a set of example outliers (e.g., generated artificially). When for all values of TP we measure FP, we get a *Receiver-Operating Characteristic* curve (ROC) [80]. An ideal one-class classifier would obtain a vertical ROC curve with $FP = 0$, while still accepting the predefined TP. It is a purely theoretical classifier - in practice all of methods will always accept some outliers. This depend on the fit of the learner to the data and the volume of the target class (assuming uniform distributions of outliers over the feature space, there will always be a fraction of outliers that fall within the target concept). An example of a ROC curve for a one-class classifier is given in Figure 1.3.

Each one-class classifier will perform differently for different thresholds. A given learner may output a tight description (aiming at obtaining a high TP), but fail when large parts of the target class should be rejected (e.g., due to noise or high overlap with potential outliers). To find a single error measure for fitting one-class classifiers into

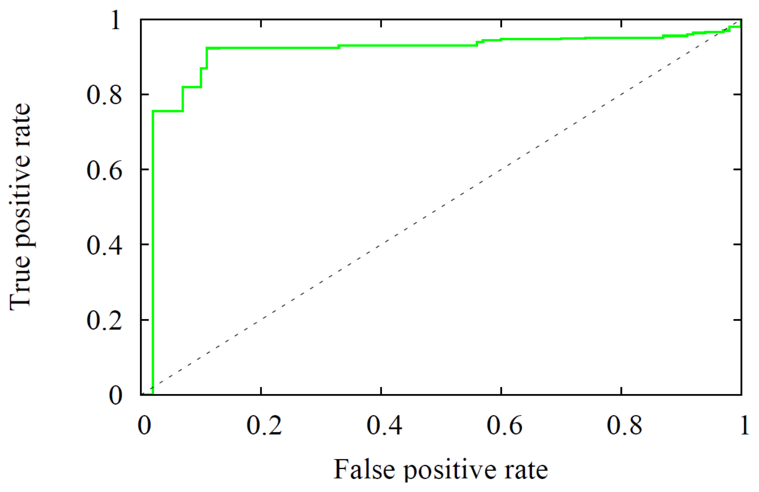


Figure 1.3: Example of a ROC curve for a one-class classifier over a toy dataset. Different values of θ are used as cut-off points. Dotted line indicates the performance of a random classifier.

given target concept, a 1-dimensional error measure from ROC curve may be derived, known as *Area Under Curve* (AUC) [32]. One needs to set $FP = 1 - TP$ and $FN = 1 - TN$. One can integrate FP over varying thresholds (i.e., all possible errors of the first kind). This outputs the error:

$$AUC = \int_0^1 FP \times FN dFN = \int_0^1 \int_0^{2R} I(F_{\omega_T}(x) \geq \theta) dx d\theta, \tag{1.27}$$

where R is the radius of a d-dimensional hypersphere enclosing all objects from ω_T and θ is measured over the target concept. This measure integrates the performance of a given one-class classifier over all thresholds values (without the need to select one specific for error estimation).

The error FP from Eq. (1.27) is of equal importance (has identical weight) over all of the thresholds. In many practical applications we need to deal with some restrictions posed by the nature of the problem. FP may be constrained to a smaller range or may be affected by some weighting function $w(\theta)$ [1] to create a more robust method :

$$AUC = \int_0^1 \int_{-R}^{R} I(F_{\omega_T}(x) \geq \theta) w(\theta) dx d\theta. \tag{1.28}$$

Weighting allows us to exclude extreme cases. For $\theta_f < 0.05$ less than 5% of objects from the target class can be rejected. In case of samples with quantity smaller than 20, no target object can be rejected and the threshold is determined by the most dissimilar object in $\mathcal{TS}$, as experimentally proven in [267]. It may lead to an overfitting over small datasets. We can avoid this, by discarding any thresholds with values lower than 0.05. On the other hand when $\theta > 0.5$ is used, more than a half of the target class representatives may be rejected. As the primary goal of OCC is to create an efficient description of the target concept rejection of more than 50% of training samples is unacceptable. Thus with the usage of a weighting function that penalizes such high thresholds one may prevent this from occurring.

However, for AUC we still need some approximation of the FP objects, which would require to generate artificial outliers. Therefore, more interesting would be a fully unsupervised measure that can work only on the basis of objects from ω_T. Consistency measure [266] fulfils this requirement. It indicates how consistent a pool of classifiers is in rejecting a pre-set fraction t of the target data.

Let us assume that we have a one-class classifier Ψ trained to reject the fraction t of objects and a validation set $\mathcal{VS}$.

We can estimate the error of this classifier as:

$$FN = \sum_{i=1}^{|\mathcal{VS}|} \big(1 - I(F_{\omega_T}(x_i) \geq \theta)\big). \tag{1.29}$$

We can model this as $|\mathcal{VS}|$ binominal experiments. This will allow us to compute the expected number of rejected objects and the variance:

$$E[FN] = \lfloor |\mathcal{VS}| t \rfloor, \tag{1.30}$$

$$V[FN] = |\mathcal{VS}| t(1-t). \tag{1.31}$$

When the number of rejected objects from the target class exceeds some bounds around this average (usually 2σ is used), the examined classifier Ψ can be deemed as inconsistent. We may say that Ψ is inconsistent at level t when:

$$FN \geq t + 2\sqrt{|\mathcal{VS}| t(1-t)}. \tag{1.32}$$

One may compute the consistency for an examined one-class classifier by comparing the rejected fraction t with an estimate of the error on the target class FN:

$$\mathrm{CONS}(\Psi) = |FN - t|. \tag{1.33}$$

To use this approach, we need to have a number of models to be selected. We train and test them, ordering them by their complexity. The model for which the boundary could be estimated with highest reliability, will be selected. Consistency measure prefers the classifier with highest complexity (in order to offer the best possible data description) that can still be considered as consistent. Therefore, we look for a maximum value of consistency that still satisfies Eq. (1.32).

1.2.3 Estimating the Volume of Data Description

By randomly drawing objects from an uniform distribution over the target data, one may estimate the volume captured by the trained data description. The FP informs about the fraction of the outlier space volume covered (erroneously) by the data description under consideration. Unfortunately, the number of testing objects for such an estimation can increase into a prohibitive size, especially for problems described in high-dimensional spaces.

We assume a target class distribution in a form of a d-dimensional hypersphere $\mathcal{HS}$ with radius R: $\mathcal{HS} = \{x : \|x\|^2 \leq R^2\}$. Let us further assume that outliers are present in a form of a d-dimensional hypercube: $\mathcal{HC} = [0, 2R]^d$. The volumes of such datasets are equal to:

$$V_{\mathcal{HS}} = \frac{2R^d \pi^{d/2}}{d\Gamma(d/2)}, \tag{1.34}$$

where:

$$\Gamma(n) = 2\int_0^\infty e^{-r^2} r^{2n-1} dr, \tag{1.35}$$

and

$$V_{\mathcal{HC}} = (2R)^d. \tag{1.36}$$

One can see that for $d = 1$ these volumes are equal ($V_{\mathcal{HS}} = V_{\mathcal{HC}} = 2$). For $d = 2$, these volumes are similar to each other. For small values of d ($d < 6$) they are of the same order. However, for higher dimensionalities $V_{\mathcal{HS}}$ decreases to zero while $V_{\mathcal{HC}}$ exponentially increases.

In practical applications, both $\mathcal{HS}$ and $\mathcal{HC}$ will have more complex shapes, but when the volumes of these two differ their will diverge with increasing dimensionality d. Therefore, for problems with a high number of dimensions standard one-class classifiers will tend to accept an increased ratio of outliers. To counter this problems we should find a low-dimensional representation of the considered problem that will be used for classifier training. This observation is the root for interest of researchers working with one-class classifiers in solutions that are able to divide the feature space into smaller partitions.

1.2.4 One-Class for Multi-Class Classification

In previous sections, we have presented the outlook on the usage of OCC for learning in the absence of counterexamples. We must point out that there is a second view on OCC, concentrating on the usage of these methods for handling multi-class problems [215]. 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. This strategy can be easily applied in machine learning, where dealing with complex, multi-class datasets is a common practice [132]. The most popular approach is to decompose the original dataset into a number of binary problems [84]. We achieve locally specialized classifiers that deal with simplified tasks. The crucial part of such a decomposition is the reconstruction of original multi-class problem.

However, instead of binary classifiers, one may use one-class classifiers [176]. Using one-class classification algorithms for decomposing a multi-class dataset is very intuitive - each class is considered as independent and delegated 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 *one-versus-all* (OVA) approach [84] . An illustrative example of OCC decomposition is given in Figure 1.4.

One may doubt the idea of using one-class classification for decomposing multi-class datasets. Methods from this group use only information about the target class, therefore we discard available useful information [121]. One-class classification is by no means

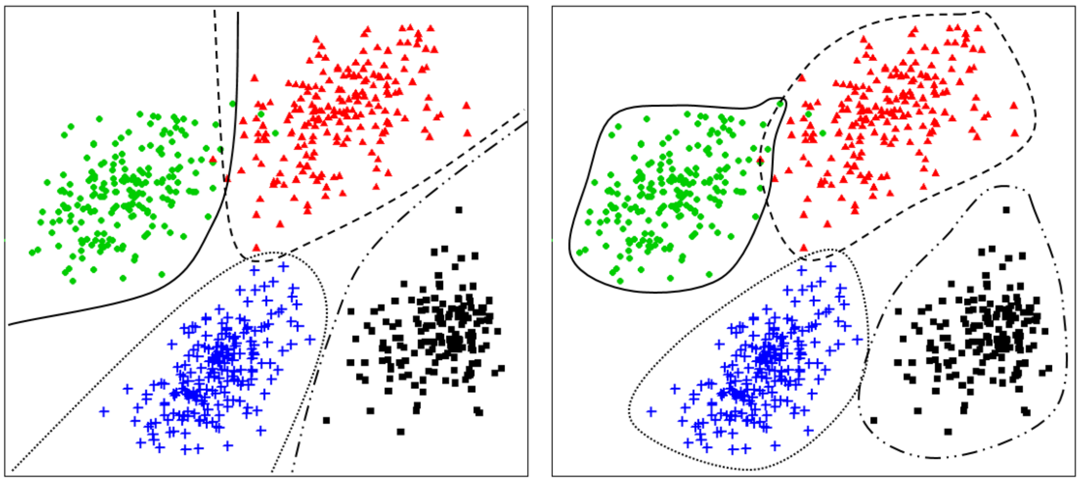


Figure 1.4: The difference between decomposing a multi-class problem with binary and one-class classifiers. (*Left*) A four-class toy problem decomposed with binary classifiers applying OVA procedure. (*Right*) The same dataset decomposed with one-class classifiers, each delegated to a different class.

a superior method to binary classifiers. For standard datasets, binary classifiers will perform better due to their access to counterexamples. The applicability of one-class decomposition lies in complex data, where standard binary classifiers tend to fail. One-class learning aims at capturing the unique properties of the target class, hoping that they will allow for a sufficient dichotomization from unknown outliers. While one-class classifier uses less information about the problem being considered, its properties allows to deal with difficulties, embedded in the nature of the data: imbalance, class noise or inner outliers, to name a few [135].

1.2.5 Overview of One-Class Classifiers

In the last decade a plethora of one-class classifiers have been proposed. Each of them has tried to approach differently the problem of fitting a data description without an access to counterexamples. According to the principle of their description volume estimation method, one can propose a following taxonomy for one-class learners:

- The first group comprises methods based on density estimation of a target class [39]. This is a simple, yet surprisingly effective method for handling concept learning. This approach has limited application as it requires a high number of available examples and the assumption of a flexible density model [251]. The most widely used methods from this group are the *Gaussian model*, the *mixture of Gaussians* [324], and the *Parzen Density Data Description* [53].

- The second group is known as reconstruction methods [178]. They were originally introduced as a tool for data modeling. These algorithms estimate the structure of the target class and their usage in OCC tasks is based on the idea that unknown outliers differ significantly from this established positive class structure. The most popular techniques are the *k-means* algorithm [44], *self-organizing maps* [273], and *auto-encoder neural networks* [210].

- The third group consists of boundary methods [225]. Estimating the complete density or structure of a target concept in a one-class problem can very often be too demanding or even impossible. Boundary methods instead concentrate on estimating only the closed boundary for the given data, assuming that such a boundary will sufficiently describe the target class [138]. The main aim of these methods is to find the optimal size of the volume enclosing the given training points [272]. It allows to find trade-off between robustness to outliers and generalization over positive examples. These methods are based on calculating the distance between objects. This makes selecting a proper distance metric and use of feature scaling very important steps in the classifier design [168]. Additionally, internal noisy samples should be excluded in order to improve the compactness of the classifiers [323]. On the other hand boundary methods require a smaller number of objects to estimate the decision criterion correctly compared with the two previous groups of methods. The most popular methods in this group include the *Support Vector Data Description* (SVDD) [265] and the *One-Class Support Vector Machine* (OCSVM) [48].

A more detailed schema of one-class classifiers taxonomy, with the respect to most popular models is presented in Figure 1.5.

Each of these methods use a different learning paradigm for data description task. Thus, the outputted shapes of decision boundary can differ significantly among one-class learners as seen in Figure 1.6.

Let us describe main representatives from each of the groups, with special attention paid to boundary-based methods (as they will be used most commonly in this thesis).

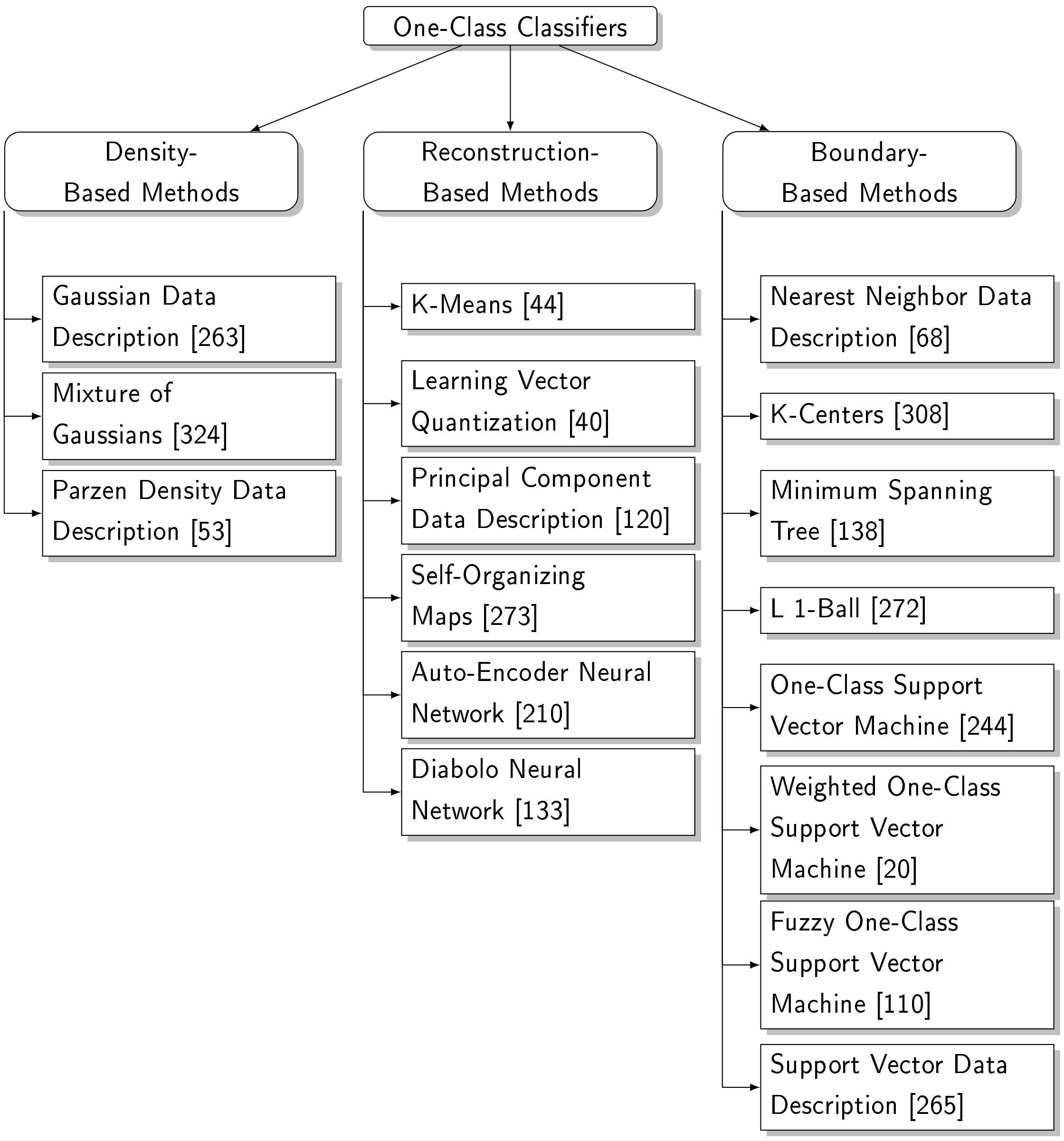

One-Class Classifiers
Density-Based Methods
Reconstruction-Based Methods
Boundary-Based Methods
Gaussian Data Description [263]
Mixture of Gaussians [324]
Parzen Density Data Description [53]
K-Means [44]
Learning Vector Quantization [40]
Principal Component Data Description [120]
Self-Organizing Maps [273]
Auto-Encoder Neural Network [210]
Diabolo Neural Network [133]
Nearest Neighbor Data Description [68]
K-Centers [308]
Minimum Spanning Tree [138]
L 1-Ball [272]
One-Class Support Vector Machine [244]
Weighted One-Class Support Vector Machine [20]
Fuzzy One-Class Support Vector Machine [110]
Support Vector Data Description [265]

Figure 1.5: Taxonomy of popular one-class classifiers, divided into three main groups based on the learning paradigm.

1.2.5.1 Density-Based Methods

The most straightforward take on one-class problem is to estimate the density of the target class data available for training [263]. User needs to set a threshold on this density, in order to be able to discriminate between wanted objects and outliers. One

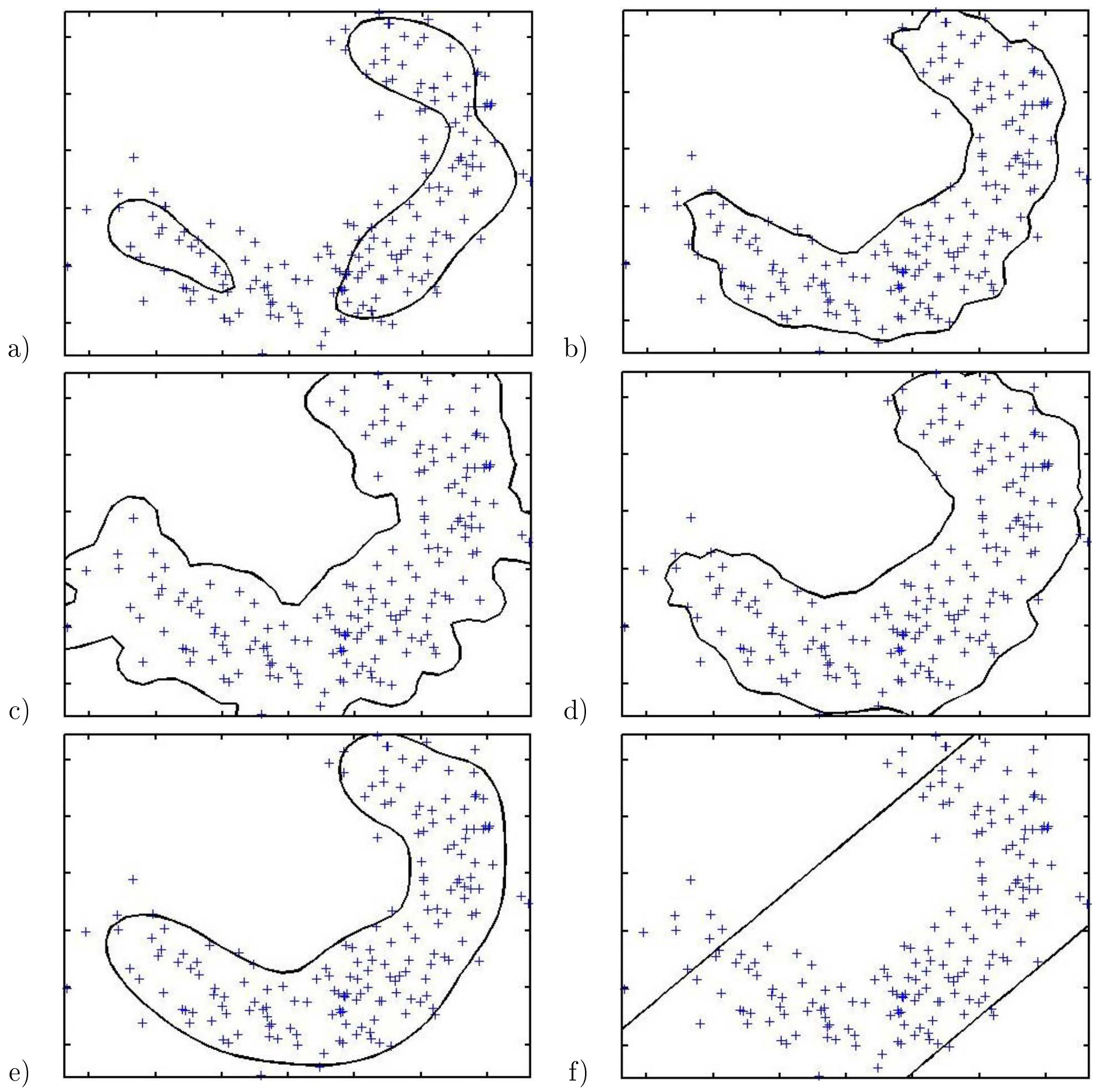


Figure 1.6: Exemplary differences between decision boundaries created by different one-class classifiers: (*a*) Support Vector Data Description, (*b*) Parzen Density Data Description, (*c*) Minimum Spanning Tree Data Description, (*d*) Nearest Neighbor Data Description, (*e*) Mixture of Gaussians Data Description and (*f*) Principal Component Data Description.

may assume several different distributions of data (such as Gaussian or Poisson) and then use one of many discordancy tests [16] to examine new object according to a given density. In one-class classification three models are most popular: *Gaussian model*, *Mixture of Gaussians model* and *Parzen density*. When the sample size is sufficiently high and a flexible density model is used (for example a *Parzen density* estimation), methods from this group return a highly satisfactory performance [116]. Unfortunately, in order to work properly they require large quantities of training examples to overcome the curse of dimensionality [19]. One may avoid this by restricting the dimensionality of the data and the complexity of the density model, but at the cost of introduction of a large bias when the model (under given restrictions) does not fit the data very well. Finding the right model to describe the target distribution and the given sample size is a typical incarnation of the bias-variance dilemma [82]. This highly limits the practical use of density-based data descriptors. Let us discuss each of three mentioned models.

Gaussian Model Data Description or the normal density model is the simplest density-based one-class classifier. According to the Central Limit Theorem, this model is correct when one assume that objects originate from a single prototype and is additively distributed by a large number of small independent disturbances. This method forces a strict unimodal and convex model of density on the given data. It requires the biggest computational effort for inverting the covariance matrix Σ. When we deal with badly scaled data or one with singular directions, the inverse Σ cannot be calculated and should be replaced by the pseudo-inverse $\Sigma^{+} = \Sigma^{T}(\Sigma\Sigma^{T})^{-1}$ or by applying a regularization factor [246].

Mixture of Gaussians Data Description is a method develop to alleviate the restrictions imposed by using a Gaussian model. Assumptions made by *Gaussian Data Description* approach are very strong and will be violated for most of real-life datasets. A *Mixture of Gaussians* model offers a much more flexible density method, by linearly combining a given number of normal distributions [25]. It offers a much more smaller bias than the single Gaussian approach, bu requires a significantly larger training dataset. Thus, for smaller data the variance of this method increases. When the number of components is predefined by the user, both μ and Σ of its individual components can be estimated with the use of *Expectation-Minimization* (EM) algorithm [23].

Parzen Density Data Description estimates the density on the basis of Gaussian kernels centered on the individual training examples with (usually) diagonal covariance matrices. This method assumes an equal width of kernel in each feature direction. It means that the *Parzen Density* estimator works on equally weighted features, thus being sensitive to scaling of the feature values (this is most prominent for smaller datasets). Training *Parzen Density Data Description* requires determination of a single parameter

- the optimal width of the kernel h. This parameter is optimized using the maximum likelihood solution. As this is a non-parametric model, therefore its quality strongly depends on the representativity of the training set. The computational cost connected with the training process of *Parzen Density Data Description* is minimal. However, the testing procedure bears a very high cost. All objects from the training set must be stored and distances to them must be calculated and sorted. This can be seen as severely limiting applications of this method for large and high dimensional datasets.

1.2.5.2 Reconstruction-Based Methods

The reconstruction methods were primarily designed as a data modeling tool. However, they can be easily transformed into data descriptors. Methods from this group generate and fit a model to the training data on the basis of data characteristics and assumptions about the source generating such data. have not been primarily constructed for OCC, but rather to model the data. We assume that a more compact representation of the target data can be obtained and that it simplifies further processing without harming the information content.

Most of algorithms from this group are based on clustering schemes. They provide a set of subspaces or prototypes and then minimize the reconstruction error. When applying such a model for OCC we assume that outliers do not satisfy the assumptions about the target distribution. The outliers should be represented worse than true target objects and their reconstruction error should be high. The reconstruction error therefore can be used as a distance to the target class or support value. Reconstruction-based methods require to obtain an empirical threshold θ with the usage of training set. Let us discuss each of three mentioned models.

K**-means** is a very popular clustering algorithm that still attracts attention of machine learning community [31]. This method assumes that data is distributed in a form of chunks and can be sufficiently characterized by a few prototypes μ_k. Objects from the target class are represented by an associate prototype (centroid) that has the lowest Euclidean distance from them. To obtain a one-class data description, one needs to define the distance between an object and target concept as an Euclidean distance of that object to its nearest prototype. Different routines for minimizing the *k-means* error were proposed in last decades [131]. For OCC purposes, the batch algorithm for optimizing the prototypes is most frequently used [254]. This procedure starts with a random placement of the prototypes. All objects from the training set are then assigned to the nearest prototype and the prototype is updated to the mean of this set of examples. This continues until all of prototypes are stable [30]. This method strongly relies on setting the proper number of k prototypes.

Self-Organizing Map (SOM) is a method lying on a border between neural networks and clustering schemes [146]. The location of prototypes is optimized with the respect to training data and constrained to form a low-dimensional manifold. After the convergence of this algorithm, prototypes corresponding to nearby vectors on the manifold are located closely to each other. Often a 2- or 3-dimensional regular grid is selected, in order to allow a visualization of the data [117].

Auto-Encoder and Diabolo Neural Networks are approaches to learn the representation of data [133, 210]. Both methods are trained to reproduce the input patterns as in their output (they have roots in signal compression or dimensionality reduction). Thus, for one-class classification they can decide if a new object resembles the target concept or differs from it significantly. The main difference between *Auto-Encoders* and *Diabolo Networks* lie in their neural architecture. Both networks have an equal number of input and output units and differ in their definition of the hidden layers. *Diabolo networks* have larger number of hidden layers (three hidden layers are popular), while *Auto-Encoders* have one hidden layer (called the bottleneck layer). Determining the number of neurons in layers is crucial for the performance of both networks. However, as there are no clear indicators on how to do this it can be considered as a magic parameter.

Both networks are trained by minimizing the mean square error. The assumption for one-class classification is that an object from the target concept will be reconstructed with a smaller error than outliers. This works on the basis of calculating the distance between reconstructed and training objects.

1.2.5.3 Boundary-Based Methods

One can notice that in OCC estimating the complete density or structure of given data may pose a highly demanding problem, especially in case of smaller and high-dimensional datasets. This may also impose an excessive computational cost when in fact only an enclosed boundary around the target class is required [269]. Boundary-based methods are based on this observation as they only optimize the shape of an enclosing decision boundary enclosing training samples.

These methods have a bias towards a minimal volume solution, in order to avoid so-called empty spaces (regions covered by the boundary but without any training examples) that may increase the false acceptance rate. Fit of the model to the data controls the volume of the boundary. Most of these classifiers works on the basis of computing a distance between a new object and training set representatives. The threshold θ is obtained directly from the training set.

Boundary-based methods tend to be sensitive to different metrics and feature scaling [168]. On the other hand, they can work efficiently with a much smaller datasets than methods from previous two groups. Thus, boundary-based methods alleviate many limitations of density-based and reconstruction-based methods, but at the same time pose a difficulty of well-defining the distances.

Let us describe in details two popular boundary-based classifiers that will be widely used thorough this thesis: OCSVM and SVDD.

1.2.5.4 One-Class Support Vector Machine

One-class SVM classifier (OCSVM) [244] can deal with datasets containing only patterns from one target class. OCSVM classification aims at discriminating one class of target samples from all other ones. It consists of learning the minimum volume contour that encloses most of the data in a given dataset. Its original application is the outlier detection finding data that differ from most of the data within a dataset. The schema of OCSVM method is presented in Fig. 1.7.

The idea behind OCSVM is to find a hyperplane $\langle \mathbf{w}, x \rangle + \rho$ that separates the training data from the origin with the maximal margin. We can formulate this problem as a convex optimization task:

$$Q(\mathbf{w}, \xi_1, \cdots, \xi_\iota, \rho) = \min \left(\frac{1}{2} \|\mathbf{w}\|^2 + \frac{1}{\nu\iota} \sum_{i=1}^{\iota} \xi_i - \rho \right) \tag{1.37}$$

subject to:

$$\forall i \in [1, \iota] \quad \langle \mathbf{w}, x \rangle \geq \rho - \xi_i, \tag{1.38}$$

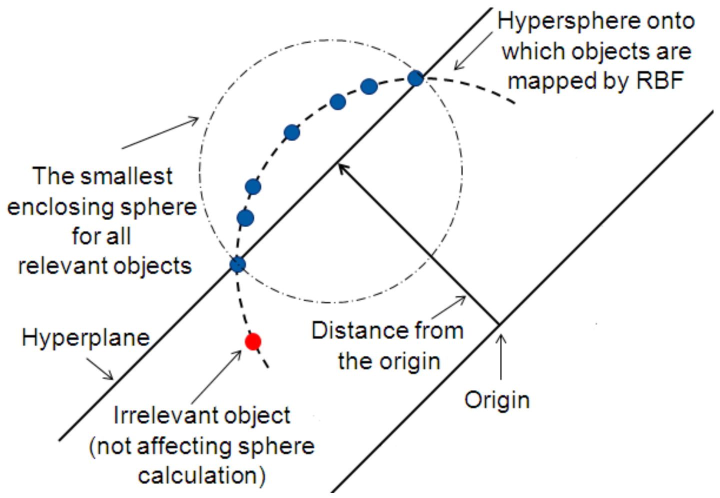

Hypersphere onto which objects are mapped by RBF
The smallest enclosing sphere for all relevant objects
Hyperplane
Distance from the origin
Origin
Irrelevant object (not affecting sphere calculation)

Figure 1.7: Idea of a OCSVM classifier, which maps relevant training points onto a smallest enclosing hypersphere.

where $\|\cdot\|$ denotes the Euclidean norm, $[\xi_1, \cdots, \xi_\iota]$ are the slack variables that must satisfy the condition $\xi_i \geq 0$ and ν is the penalization factor incurred by these slack variables.

The corresponding Wolfe dual [244] is subject to optimization:

$$Q(\alpha_1, \cdots, \alpha_\iota) = min\left(\sum_{i=1}^{\iota}\sum_{j=1}^{\iota}\alpha_i\alpha_j\langle x_i, x_j\rangle\right), \tag{1.39}$$

subject to:

$$\forall i \in [1, \iota] \quad 0 \leq \alpha_i \leq 1/(\nu\iota), \tag{1.40}$$

and

$$\sum_{i=1}^{\iota}\alpha_i = 1, \tag{1.41}$$

where $\langle\cdot,\cdot\rangle$ is the inner product and $[\alpha_1, \cdots, \alpha_\iota]$ are Lagrange multipliers.

One may denote the solution of the problem from Eq. 1.39 as $[\alpha_1^*, \cdots, \alpha_\iota^*]$. Then according to $\mathbf{w} = \sum_i \alpha_i^* x_i$, we may compute the distance between a new point x and the separating hyperplane using:

$$dst(x, \omega_T) = \sum_{i=1}^{\iota} \alpha_i^* \langle x_i, x \rangle - \rho. \tag{1.42}$$

Parameter ρ can be calculated with the usage of property that for any α_i^* satisfying $0 < \alpha_i^* < 1/(\nu\iota)$, the corresponding example x_i satisfies:

$$\rho = \sum_{j=1}^{\iota} \alpha_i^* \langle x_i, x_j \rangle. \tag{1.43}$$

OCSVM makes its decision about a new object x on the basis of distance $dst(x, \omega_T)$. This can be seen as a measure of similarity between the object and the target class. This formulation uses only inner product. It allows for an easy kernelization, by replacing each inner product $\langle x_i, x_j \rangle$ by the selected kernel function $K(x_i, x_j)$.

This concept can be further extended to a *Weighted One-Class Support Vector Machine* (WOCSVM) [20] by the introduction of weights $\mathfrak{w}_i$ that allows for an association of an importance measure to each of the training objects. This forces slack variables ξ_i, to be additionally controlled by $\mathfrak{w}_i$. If with object x_i there is associated a small weight $\mathfrak{w}_i$ then the corresponding slack variable ξ_i indicates a small penalty. In effect, the corresponding slack variable will be larger, allowing x_i to lie further from the center a of the hypersphere. This reduces an impact of x_i on the shape of a decision boundary of WOCSVM.

By using the above mentioned ideas we can reformulate the WOCSVM optimization task:

$$Q(\mathbf{w}, \xi_1, \cdots, \xi_\iota, \rho) = \min \left(\frac{1}{2} \|\mathbf{w}\|^2 + \frac{1}{\nu\iota} \sum_{i=1}^{\iota} \mathfrak{w}_i \xi_i - \rho \right) \tag{1.44}$$

subject to:

$$\forall i \in [1, \iota] \quad \langle \mathbf{w}, x \rangle \geq \rho - \mathfrak{w}_i \xi_i, \tag{1.45}$$

where slack variables must satisfy the condition $\mathfrak{w}_i \xi_i \geq 0$.

For establishing weights we may use techniques dedicated to a weighted multi-class support vector machines. Most often one uses the following formula:

$$\mathfrak{w}_i = \frac{|x_i - x_{mean}|}{R + \delta}, \tag{1.46}$$

where $\delta > 0$ is used to prevent the case of $\mathfrak{w}_i = 0$. The value of x_{mean} is computed with the usage of all available objects from $\mathcal{TS}$. The difference between OCSVM and WOCSVM is depicted in Figure 1.8.

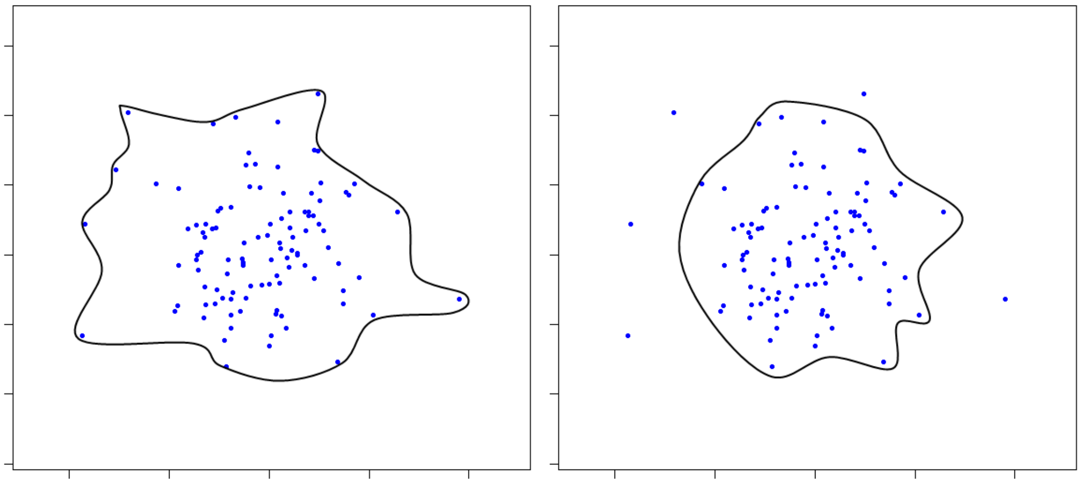


Figure 1.8: Difference between established decision boundaries for a toy problem by (*left*) OCSVM and (*right*) WOCSVM. The latter model strongly reduces the influence of noisy samples on the shape of the decision boundary.

OCSVM can be considered as the most popular one-class classifier due to its many parallels with popular SVMs for binary and multi-class problems. However, one of the main problems connected to a proper setting of OCSVM is the selection of kernel parameters. As kernel is responsible for mapping object onto the decision hypersphere, therefore its proper tuning can affect the robustness and compactness of the volume. In the recent years methods based on greedy search [302] or heuristic optimization [301] have been successfully introduced.

Another problem lies in the lack of robustness of OCSVM to outliers and atypical data distribution. The problem of internal noise in the training concept was discussed in [20] (by introducing the mentioned WOCSVM classifier). Other works concentrated on different classification margin calculation to get a better robustness [305], extracting different data properties to guide the boundary estimation process [323] or including the covariance of training data to get more sparse representation of the classification volume [142].

Additionally, a new version of OCSVM based on least squares optimization (LS-OCSVM) was recently introduced [50]. LS-OCSVM achieves a faster training time, but at the cost of a much slower response - as in LS-SVM all of training objects become support vectors. This drawback almost completely prohibits the usage of LS-OCSVM for larger datasets.

An interesting proposal was presented in [115] where a multi-task learning scheme was applied to OCSVM training in order to combine more than a single fitness criterion. OCSVM can also be trained incrementally in batch mode [141] or in on-line mode [321] in order to process massive datasets.

OCSVM has proven its quality in many real-life problems, ranging from medical domain (biomedical signal monitoring [92] or medical imaging [190]), image analysis [99], to remote sensing [218].

1.2.5.5 Support Vector Data Description

A *Support Vector Data Description* (SVDD) [265] 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 $\mathcal{TS}$ must be enclosed by this hypersphere. Yet this approach often leads to an poor performance due to too big enclosing volume. Therefore, identically as in canonical Support Vector Machine one may introduce slack variables ξ [38] to include the possibility of outliers in $\mathcal{TS}$. The schema of SVDD method is presented in Fig. 1.9.

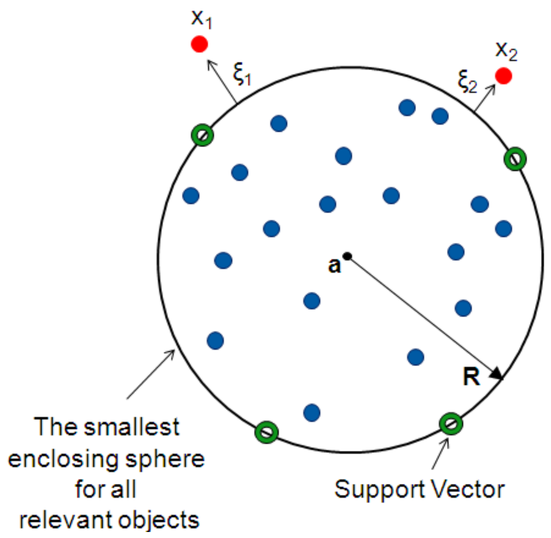


Figure 1.9: Idea of a SVDD classifier with two outliers in the training set, denoted by slack variables ξ_1 and ξ_2.

We can formulate SVDD training problem as a convex optimization task:

$$Q(\mathbf{a}, R, \xi_1, \cdots, \xi_\iota) = R^2 + \nu \sum_{i=1}^{\iota} \xi_i, \tag{1.47}$$

subject to:

$$\forall i \in [1, \iota] \quad \|x_i - \mathbf{a}\|^2 \leq R^2 + \xi_i, \tag{1.48}$$

where each slack variable must satisfy the condition $\xi_i \geq 0$.

The corresponding Wolfe dual is subject to optimization:

$$Q(\alpha_1, \cdots, \alpha_\iota) = min\left(\sum_{i=1}^{\iota} \alpha_i \langle x_i, x_i \rangle - \sum_{i=1}^{\iota}\sum_{j=1}^{\iota} \alpha_i \alpha_j \langle x_i, x_j \rangle\right), \tag{1.49}$$

subject to:

$$\forall i \in [1, \iota] \quad 0 \leq \alpha_i \leq \nu, \tag{1.50}$$

and

$$\sum_{i=1}^{\iota} \alpha_i = 1. \tag{1.51}$$

One may denote the solution of the problem from Eq. 1.49 as $[\alpha_1^*, \cdots, \alpha_\iota^*]$. Then the sphere center is given as $\mathbf{a} = \sum_i \alpha_i^* x_i$. With this, we can calculate the squared distance between the new point x and center $\mathbf{a}$:

$$dst(x, \omega_T) = \|x - \mathbf{a}\|^2 = \langle x, x \rangle - 2\sum_{i=1}^{\iota} \alpha_i^* \langle x_i, x \rangle + \sum_{i=1}^{\iota}\sum_{j=1}^{\iota} \alpha_i^* \alpha_j^* \langle x_i, x_j \rangle. \tag{1.52}$$

The decision about the new object x is made by comparing the distance $dst(x, \omega_T)$ with threshold θ.

Similar as OCSVM, this formulation uses only inner product. This allows for an easy kernelization, by replacing each inner product $\langle x_i, x_j \rangle$ in Eq. (1.49) and Eq. (1.52) by the selected kernel function $K(x_i, x_j)$.

SVDD can be viewed as a different approach to obtaining an enclosing hypersphere than OCSVM. SVDD grows the hypersphere from the estimated center, while OCSVM maps points onto pre-calculated hypersphere. Due to useful properties of SVDD (ease of tuning and simple operation mode) and open-source implementation, this classifier has become a popular one-class solution. A thorough analytical analysis of the properties of SVDD was presented in [285].

A big advantage of SVDD is it shorter training time than OCSVM. Recently a modified version of this classifier was introduced, changing the way the hypersphere estimation is conducted. This allowed for a faster estimation of the volume, further reducing the training time [204]. An entropy-based kernel can be applied in SVDD, creating a more compact decision boundary in lower time than when using RBF kernel [224].

SVDD can deal with some outliers in the training set much better than OCSVM, however not nearly as efficiently as WOCSVM. To deal with this limitation, a hybrid implementation of SVDD was proposed, which used local density estimators [191]. The idea is to check the density distribution in subspaces and reject ones that can be a potential noise [192]. This method automatically discarded objects, without considering their importance. A soft approach with density-based weighting was recently proposed [42]. It uses density estimators to calculate weights assigned to objects, allowing to control their degree of importance (a very similar idea to WOCSVM). Another solution to this approach is to estimate a degree of certainty of the training and incoming objects [199] (a solution rooted in fuzzy logic). This can be efficiently applied to non-stationary environments, especially when combined with incremental learning mode for SVDD [303].

SVDD has found application in fault detection [320], biometrics [193], and hyperspectral data analysis [11, 240].

1.3 Classifier Ensembles

There are a number of proposals on how to automate the classification process [132]. Nevertheless, according to Wolpert's theorem [291], there is not a single pattern recognition algorithm that is appropriate for all the tasks we are faing, since each classifier has its own domain of competence [291]. Usually we can pool different classifiers to solve a given problem. Therefore, methods that can exploit the strengths of individual classifiers are currently the focus of intense research [295]. It is worth noting that the incompetence area, i.e., the subset of the feature space where all individual classifiers make the wrong decision, is typically small [227], as presented in Figure 1.10.

The presented approach is called a *multiple classifier system* (MCS), *combined classifier*, or *classifier ensemble* [182], and its main components are depicted in Figure 1.10. MCS assumes that a combination of weak classifiers may return an efficient compound

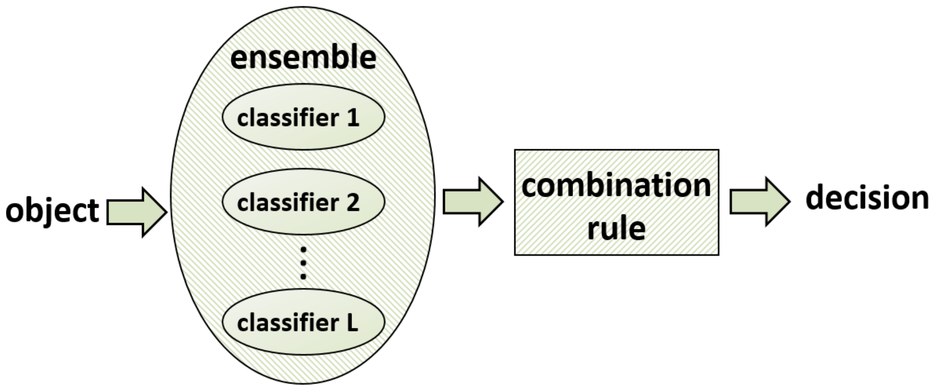


Figure 1.10: Overview of multiple classifier system.

classifier due to the possibility of exploiting local competencies of base learners. An example of such a situation is presented in Figure 1.11.

In this concept the greatest effort is concentrated on combining the outputs of elementary classifiers at our disposal for a given classification problem. This concept was first presented by Chow [51], who proved that the decision of independent classifiers with appropriately defined weights is optimal. Listed below are some of the advantages of an MCS:

- The design of an MCS does not differ from that of a classical pattern recognition [97] application. In the standard approach we select the most valuable features and choose the best classification method from the set of available ones. The design of a classifier ensemble aims to create a set of complementary/diverse classifiers and assign an appropriate fusion method, which can combine the individual classifiers' outputs optimally.

- Some works report that MCSs can improve the overall performance compared with the best individual classifier, because they are able to exploit unique strengths of each of the individual classifiers [188]. In some cases (e.g., majority voting by a group classifiers that make independent errors) their characteristics have been

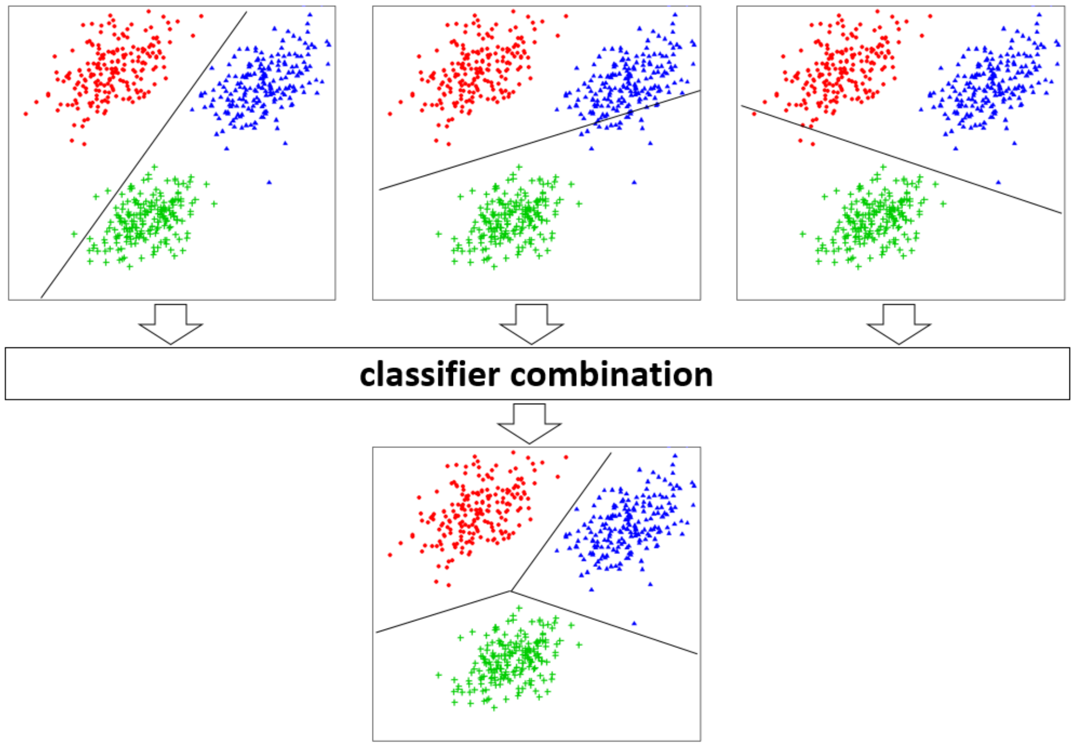


Figure 1.11: An example of MCS paradigm. Three weak linear classifiers are given, each characterized by a high individual error. By combining them we obtain a compound classifier with significantly improved accuracy.

proven in an analytical way [278]. Additionally, an MCS protects against selection of the worst classifier for a small sample [96].

- Many machine learning algorithms (e.g., C4.5 based on a top down induction decision tree concept) are $de facto$ heuristic search algorithms, which cannot guarantee that an optimal model is found. Therefore, the combined approach, which could start searching from different points of the search space, seems to be an attractive proposition [128].

- Combined classifiers could be used in efficient computing environments such as parallel and multithreaded computer architectures [286]. Another attractive area of application is distributed computing systems (P2P, GRID) [139], especially in the case of a database that is partitioned for privacy reasons [159, 163] and only the final decision is available at each node of the computer network [282].

There are number of important issues that must be taken into consideration when building multiple classifier systems.

These can be grouped into the following problems:

- Creating a pool of mutually complementary and competency classifiers for the ensemble and selecting the most valuable and diverse members in order to form a sparse and efficient committee.
- Designing a combination rule, aimed at creating a mechanism that can exploit the strengths of the selected classifiers and combine them optimally.
- Proposing the topology i.e., interconnections between classifiers in the ensemble.

We do not address the last issue because most of the combined classifiers are based on a parallel topology, which has a good methodological background [182] and is used in this thesis.

1.3.1 Creating a Pool of Classifiers

In order to form a classifier ensemble, we require a pool of elemental classifiers to combine. As the aim of the ensemble is to benefit from different competence areas of each base classifiers, simply multiplying the same model would not contribute anything. Therefore, there is a need for a careful preparation of the pool of classifiers [13]. One needs to select a proper method to create a number of models that display high individual accuracy while being mutually complementary to each other [236].

There are several proposals on how to enforce the diversity of an individual classifier pool:

- We could use different partitions of a dataset or generate a number of datasets through data splitting [322], a cross-validated committee [179], *Bagging* [33], or *Boosting* [81], in the hope that classifiers trained on different inputs would be complementary. Selected features are then used to train a pool of classifiers to assure diversity of the pool. There are several propositions based on this principle such as the *Random Subspace* [119]. It is worth pointing out the interesting proposition presented in [274], where the authors proposed a hierarchical method of ensemble creation, based on feature space splitting and then assigning binary classifiers (such as Support Vector Machines) locally.

- We could train each individual classifier to recognize a subset of only predefined classes (e.g., binary classifier: one class against the rest strategy) and then choose a fusion method that can recover the whole set of classes [84].
- We could train individual classifiers based on different parameters (like kernels [10]) or different versions of models [93, 276].

Among methods based on the different partitions of the dataset *Bagging, Boosting* and *Random Forest* [34] are the most popular. They all work on the basis of creating complementary ensembles on the basis of subsets (either randomly drawn or weighted according to some rule). These methods usually employ one classification model as a base learner and manipulate its input to create a pool of base classifiers. Such methods that work on the basis of one type of classifier are known as homogeneous ensembles [70].

Bagging, despite its simplicity, has become one of the most popular ensemble methods [250] due to its proven efficiency [37]. What is highly interesting, *Bagging* is a method still being intensively developed. It can be effectively adapted to many contemporary difficult problems in machine learning. *Bagging* has been efficiently applied in imbalanced classification, outperforming many more complex methods [85], especially when one modify the probability of drawing an observation from the minority and majority classes [118] or analyze the neighborhood of each sample [27]. *Bagging* can deal with noisy and uncertain data by sub-sampling the original dataset and selecting the best subspaces [143]. Recently *Bagging* has been successfully used for on-line data streams [123] and big data analytics [21].

Boosting [242] is often a popular counterpart of *Bagging*. Here, classifiers are created in an iterative manner. Each classifier aims at correctly capturing the misclassified objects by its predecessor. Therefore, after a number of iterations Boosting should return a pool of classifiers that have complementary competence areas [243]. The most popular implementation of *Boosting* family is *AdaBoost* [81] and *AdaBoost.M2*, which was designed for multi-class problems [217, 238]. A very big advantage of Boosting is abundance of theoretical and analytical studies of its performance [205, 216]. Boosting has been successfully applied to the imbalanced classification domain by adding a cost-sensitive paradigm [257] or applying sub-sampling [86, 245] or over-sampling [43, 46]. Despite its non-parallel nature, there are some efforts to adapt *Boosting* schemes for mining data streams [62].

Random Forests are considered as one of the best off-the-shelf methods for general-purpose machine learning [281]. They are a a combination of *Bagging* and *Random Subspaces*. They can generate a large pool of simple classifiers in a very limited training time. At the same time, a combination of such a decision forest leads to an excellent classification results, even for high dimensional [69] or massive data [64]. Due to this properties, it has become a very popular classifier in bioinformatics and life science domains [275]. Recently, several modifications of this method were proposed, such as *Fuzzy Random Forest* [29] or *Unsupervised Random Forests* [309]. This technique has also been applied to on-line data analysis [283].

Here, we should mention two very important ensemble algorithms that have roots in *Random Forest* approach: *Rotation Forest* and *Random Ferns*. *Rotation Forest* [235] creates an ensemble of decision trees on the basis of mapping original data into lower-dimensional and orthogonal spaces - usually with the usage of *Principal Component Analysis* (PCA). This method attracts an increased attention of machine learning community due to increasing number of reports that it can outperform *Random Forest* for many benchmark and real-life problems [300]. *Random Ferns* were originally designed for image classification with binary features [220], but recently have been modified for general-purpose machine learning [187]. They can bee seen as a crossover between *Random Forest* and *Naive Bayes* methodologies, thus combining an efficient ensemble forming mechanism with probabilistic nature. It has been shown that *Random Ferns* achieve competitive accuracy to *Random Forest*, while having a much faster training procedure [187].

1.3.2 Classifier Selection

Having a given pool of classifier, one usually assumes that they are sufficient to form an ensemble. In many cases such assumption is far from being true as the quality of classifiers at our disposal may significantly vary. Let us consider several scenarios, where we may have an uncertain pool of learners:

- We are given a pool of classifiers beforehand, without any control over their training process. This often occurs in real-life scenarios, where we gather multi-source data, e.g., in sensor networks. Here, each model is trained over a designated node or sensor without considering the quality or importance of the collected data.

- Many methods for forming a pool of classifiers rely on some free parameters without any indication on how to set them properly. Thus for boosting it is often difficult to select the proper number of iterations [252]. *Bagging* and *Random Forest* may use out-of-bag error to stop the creation of new classifiers, but this was shown to only be an estimate, not a true factor of the ensemble performance. *Random Subspaces* and related methods have no clear indicator on the optimal number of subspaces for a given dataset.

- When creating heterogeneous ensembles, one varies the used model or the model's parameters. However, this cannot guarantee us that we will get competent or complementary classifiers.

Combining all models at our disposal is the most straightforward approach, however often not the one returning the best performance [185]. An ideal ensemble should consist of models that display a high individual accuracy, while being complementary to other members of the committee [196]. Combining incompetent classifiers will degrade the performance of the ensemble. On the other hand adding classifiers that do not differ from other members will contribute nothing to the formed MCS, but will increase its computational and memory requirements.

To deal with this problem, we need to chose the valuable committee members from a wider pool of learners. This process is known as classifier selection [98] or ensemble pruning [277].

One can trace the origin of these methods to feature selection [103] and pruning decision trees [228]. Classifier selection works on identical assumptions as feature selection - out of abundant number of entities at our disposal (features or classifiers) one needs to choose a much smaller subset that will preserve or improve the final model. Pruning assumes that the model that we currently have at our disposal is too complex and overfitted to data and can be simplified without negative effects on its quality [232]. Therefore some of existing entities (tree nodes or classifiers) must be removed to reduce the complexity and robustness of the final model [45].

One of popularly used techniques is known as overproduce and select [75]. Here, we build a large pool of classifiers and then conduct a pruning step in hope that we were able to sufficiently cover the decision area with some of the trained classifiers.

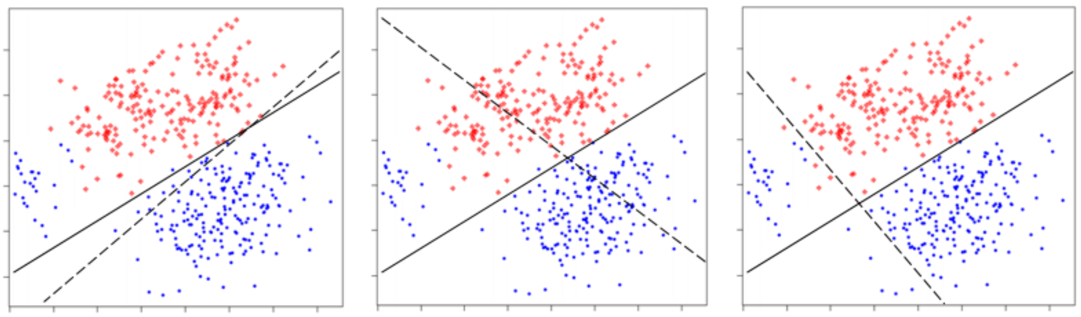


Figure 1.12: Three examples of combining two classifiers: (*left* classifiers with high individual accuracy but low diversity); (*centre*) classifiers returning a high value diversity measure but of weak individual quality; (*right*) mutually complementary classifiers. Please note that the best combination is returned neither by learners with the highest accuracy nor with the highest diversity.

This is connected with the idea of so-called *diversity* in ensemble learning. Diverse classifiers can be seen as ones significantly varying from each other, having different areas of competence or approaching the problem from various viewpoints. Many algorithms covering this subject were inspired by the guides on how to design reliable software, among which [221]. The diversity is an attractive concept, but the main problem is how to properly measure it. A concept called diversity measures [12] was introduced in last decade in order to provide a way to evaluate the degree of diversity among given classifiers. We can divide the existing diversity measures into two groups: pairwise [186] (measuring the diversity between two models) and global or non-pairwise [183] (measuring the diversity of a pool of classifiers). This idea became an attractive direction in ensemble learning, but soon the limits of it were discovered. Diversity itself is not the ideal determinant of the quality of the committee - often maximizing only the diversity leads to a weaker ensemble [36]. Many researchers now agree that we still do not perfectly grasp the idea of diversity and currently used measures are not well-designed for universal creation of competent ensembles [294]. An example of combining two classifiers presenting the role of diversity and diversity measure is presented in Figure 1.12. However, when properly used, diversity measures can be an efficient tool for forming compact and accurate MCSs [206]. An interesting solution is to apply a combination of accuracy and diversity to evaluate subsets of classifiers. Such ensembles usually display good recognition rate, while being more compact than others [248].

The methods for classifier selection and ensemble pruning can be divided into three main categories: *optimization-based, clustering-based* and *ordering-based.*

Optimization-based ensemble pruning treats the process as a search problem [237]. We aim at finding a subset of classifiers that will maximize a given criterion function. Genetic algorithms are especially popular for this task [83]. They offer a natural binary representation (which allows for an easy coding of which classifier was selected) and can easily work with both simple and compound criteria [74]. These methods are especially useful in case of large pools of classifiers, where an exhaustive search becomes prohibitive due to time complexity. Additionally, one may combine this with cost-sensitive classifier selection [158], in order to minimize the cost related to the classification procedure [124]. Other nature-inspired methods were successfully applied in this task, such as swarm algorithms that allow to directly treat an ensemble pool as an self-interacting population [315]. Another track in this group relies on the use of convex optimization techniques, such as semi-definite programming [319]. These methods are usually faster than nature-inspired ones, but more suitable for smaller pools of classifiers [61].

Clustering-based pruning assumes that it is possible to aggregate similar classifiers in form of clusters [314]. Such cluster should consist of classifiers that are as much as possible similar to each other and as much as possible different from classifiers in other clusters [189]. Then, for each cluster we select a representative classifier. This reduces the number of classifiers in the pool to the number of established clusters. Three main methods of choosing a representative classifier are used: to select the classifier that is closest to the clusters' centroid, to select the classifier that lies furthest from other centroids, or to train a new classifier for each cluster. The problem with the last method is how to merge several classifiers from one group into a single, new model. Therefore, first two solutions are much more popular [52]. Another problem lies in establishing the proper number of clusters for given dataset. This has a crucial influence on the performance of the pruning, as the number of clusters reflects the final size of the ensemble. Recent proposals concentrate on the usage of some statistical information embedded in the training subsets of each classifier [290]. Interesting sub-group of this method is Clustering and Selection scheme [184], where for each cluster of data a most competent classifier is selected from the pool [197, 202].

The third group consists of *ordering-based methods.* Here, we create an ordered list of classifiers at our disposal, according to some pre-defined criterion [211]. Then, the pool is extended in each step with a new classifier, until a given stop criterion is meet (usually, the bound is put onto the ensemble size or classification error) [101]. These methods usually apply a greedy search to chose the classifiers, thus can be related to greedy forward-search in feature selection.

Another important concept of classifier selection assumes a local specialization of individual classifiers [17]. According to this proposal, a single classifier that achieves the best results is chosen from a pool for each demarcated partition of the feature space. Its answer is treated as the system answer, for all objects included in the partition. This methodology was described by Rastrigin and Erenstein [229]. Certain proposals based on this idea assume a local specialization of particular classifiers and only search for locally optimal solutions [14, 54, 97], while other methods propose dividing the feature space and training a classifier for each partition. This approach is known as *Clustering-and-Selection* [184]. Classifiers can also be specialized on a reduced subset of classes by *divide and conquer* technique. This is usually done in a form of binary decomposition [239] or hierarchical decision cascade [226].

1.3.3 Combining Classifiers

Another important issue is the choice of collective decision making method. We can divide the combination algorithms mentioned above into two groups:

- Methods that make decisions on the basis of outputs (labels) of individual classifiers.
- Methods that propose constructing new discriminant functions based on continuous outputs (supports) of individual classifiers.

The former group includes voting algorithms [22, 304]. Initially only majority voting schemes were implemented, but in later works more advanced methods were proposed. These take the importance of decisions coming from particular committee members into consideration [279].

Many known conclusions regarding the classification quality of MCSs have been derived analytically, but these are typically valid only under strong restrictions, such as particular cases of the majority vote [109] or make convenient assumptions, such as a classifier committee consisting only of independent classifiers. Unfortunately, such assumptions and restrictions are in most cases not very useful for solving practical problems. Here, we should mention those works that propose training the weights, which seems to be an attractive alternative method [292, 296].

The second group of fusers is based on discriminant analysis. The main form of the discriminants is a posterior probability typically associated with probabilistic pattern recognition models, although outputs of neural networks or other functions whose values are used to establish the decision of the classifier (so called support functions) could be considered as well. Aggregation methods that do not require learning use simple operators, like minimum, maximum, product, or mean. However, they are typically subject to very restrictive conditions [77], which limit their practical use. Therefore, the design of new classifier combination models [169], especially trained fusers, are currently the focus of intense research [298].

Assume that we have a pool of L classifiers $\Pi = \{\Psi^{(1)}, \ \Psi^{(2)}, \ ..., \ \Psi^{(L)}\}$. For a given object $x \in \mathcal{X}$, each individual classifier decides whether it belongs to class $i \in \mathcal{M} = \{1, \ ..., M\}$ based on the values of discriminants. Let $F_i^{(l)}(x)$ denote a function that is assigned to class i for a given value of x and that is used by the l-th classifier $\Psi^{(l)}$. The combined classifier Ψ uses the following decision rule [129]

$$\Psi(x) = i \iff \hat{F}_i(x) = \max_{k \in \mathcal{M}} \hat{F}_k(x). \tag{1.53}$$

For establishing $\hat{F}_i(x)$ one may use a weighted combination of support functions. Assigning a weight to each classifier allows to control the level of its influence over the final decision. We describe four methods of establishing weights. All of these methods must fulfil the following constraint on the values of weights:

$$\sum_{i=1}^{L} w^{(l)} = 1. \tag{1.54}$$

1. Weights dependent on the classifier: This is the traditional approach where weights are connected with a classifier and each discriminant of the l-th classifier is weighted by the same value $w^{(l)}$:

$$\hat{F}_i(x) = \sum_{l=1}^{L} w^{(l)} F_i^{(l)}(x). \tag{1.55}$$

The probability error of such a classifier can be estimated e.g., as in [293].

2. Weights dependent on the classifier and feature vector: Weight $w^{(l)}(x)$ is assigned to the l-th classifier and for a given x has the same value for each discriminant function used by it:

$$\hat{F}_i(x) = \sum_{l=1}^{L} w^{(l)}(x) F_i^{(l)}(x). \tag{1.56}$$

   In this type of model, known as a "mixture of experts", parametric estimation is normally used to establish the weights [130].

3. Weights dependent on the classifier and class number: Weight $w^{(l)}(i)$ is assigned to the l-th classifier and the i-th class:

$$\hat{F}_i(x) = \sum_{l=1}^{L} w_i^{(l)} F_i^{(l)}(x). \tag{1.57}$$

   Here, the given classifier weights assigned to different classes may differ.

4. Weights dependent on the classifier, class number, and feature vector: Weight $w^{(l)}(i, x)$ is assigned to the l-th classifier, but for a given x its value could differ for discriminants assigned to each class:

$$\hat{F}_i(x) = \sum_{l=1}^{L} w_i^{(l)}(x) F_i^{(l)}(x). \tag{1.58}$$

Stacking is one of the most popular trained classifier combination methods based on discriminants [311].

It is worth to notice that most of works connected to classifier combination assume a fixed pool of classifiers. At the same time, works dealing with ensemble pruning usually concentrate on simple combination methods, such as voting-based schemes. There is little research done on methods combining these two paradigms. Integrating pruning and fusion may lead to a more specified ensemble and a combination block tailored for a valuable pool of classifiers. One may use weighted fusion as an input for pruning scheme, by discarding classifiers with assigned weights below some specified threshold [171]. Even more interesting approach would be to apply ensemble optimization scheme, where both classifier selection and fuser training are integrated as a single compound optimization problem [167].

Separate paragraph should be devoted to specific fusion methods for classifiers working on decomposed datasets [212]. Here, we require a method for reconstructing an original multi-class problem from a set of simpler local decisions. Most popular are *one-versus-one* (OVO) [114] and *one-versus-all* (OVA) [113] combiners. Most of these works originated from multi-class methods for binary SVMs [78], but were later expanded for any type of binary decomposition. OVO has gained a more attention from the community [84] due to its better accuracy for datasets with higher number of classes. Recently dynamic OVO schemes were introduced [87, 88] in order to discard non-competent classifiers and reduce the size of the ensemble pool. The third approach that grew from the former ones, can be described as trained reconstruction combiners. They are based on constructing prototypes for each classes, based on the training data that will guide the fusion process [4]. The most popular approach is the *Error-Correcting Output Codes* (ECOC) [71], which creates a unique code for each binary classifiers [289]. Many works regarding ECOC concentrate on the design of compact and reliable codewords to allow handling many classes [9]. Another popular approach from this group are *Decision Templates* [180] 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 [181].

1.3.4 Ensembles for One-Class Classification

Ensembles are a promising research direction for OCC problems, as using a number of classifiers instead of a single one may lead to an improvement in robustness to outliers (as each base classifier will contribute a different competence). Additionally, MCS approach allows us to train less complex individual classifiers, thereby reducing the risk of model overfitting which is one of the major concerns in OCC. Finally, they are an ideal solution for implementation in a distributed environment. Most of the OCC classifiers (especially the boundary-based ones) are computationally expensive and therefore relying on several weak models that run independently may significantly reduce the training cost of the recognition system.

One should notice that due to the specific nature of one-class problems, canonical ensemble solutions from binary and multi-class task cannot be straightforwardly applied. Diversity measures, pruning schemes or ensemble creation methods require an access to counterexamples, which is impossible in OCC.

There are some works done on creating combined classifiers for one-class problems [264]. However, most of them concentrate on practical applications [89, 95, 198, 201, 316], not on creating solid methodological background on how to form efficient one-class ensembles [247].

Let us review the works done so far in the filed of one-class committees.

The idea of combining one-class classifiers was introduced in 2001 in the work of Tax and Duin [264], where authors conducted some simple fusion experiments with conclusion that one-class classifiers trained on different feature sets can be highly complementary. They also introduced five ways to combine the individual outputs of one-class classifiers that are nowadays widely used.

One can use one of the following combination rules, which can be applied to both heterogeneous and homogeneous one-class ensembles:

Mean vote, which combines votes in form of support functions of one-class classifiers. It is expressed by:

$$F_{\omega_T}(x) = \frac{1}{L}\sum_{l=1}^{L} I(F_{\omega_T}^{(l)}(x) \geq \theta^{(l)}), \tag{1.59}$$

where $F_{\omega_T}^{(l)}(x)$ stands for the discriminant function value returned by the l-th individual classifier for a given observation x and class ω_T. When θ is equal to 0.5, this rule transforms into a majority vote for binary problems.

Mean weighted vote which introduces the weighting of base classifiers by TP fraction of each classifier:

$$F_{\omega_T}(x) = \frac{1}{L}\sum_{l=1}^{L}(TP^{(l)} \times I(F_{\omega_T}^{(l)}(x) \geq \theta^{(l)}) + (1 - TP^{(l)})I(F_{\omega_T}^{(l)}(x) \leq \theta^{(l)}), \tag{1.60}$$

which is a smoothed version of the mean vote method.

Product of the weighted votes, which is defined as:

$$F_{\omega_T}(x) = \frac{\prod_{l=1}^{L} TP^{(l)} \times I(F_{\omega_T}^{(l)}(x) \geq \theta^{(l)})}{\prod_{l=1}^{L} TP^{(l)} \times I(F_{\omega_T}^{(l)}(x) \geq \theta^{(l)}) + \prod_{l=1}^{L}(1 - TP^{(l)})} \tag{1.61}$$

Mean of the estimated supports which is expressed by:

$$F_{\omega_T}(x) = \frac{1}{L}\sum_{l=1}^{L} F_{\omega_T}^{(l)}(x). \tag{1.62}$$

Product combination of the estimated supports, which is expressed by:

$$F_{\omega_T}(x) = \frac{\prod_{l=1}^{L} F_{\omega_T}^{(l)}(x)}{\prod_{l=1}^{L} F_{\omega_T}^{(l)}(x) + \prod_{l=1}^{L} \theta^{(l)}}. \tag{1.63}$$

A relation between the training process of OCSVM and boosting schemes was discussed in [230], where authors proposed naive one-class classifiers that could be efficiently boosted (OCSVM can be considered as a strong classifier and thus does not correspond in a satisfactory manner to being boosted).

Next work on combining one-class classifier appeared in 2004 [137], where it has been shown how to use ensemble of one-class classifiers to deal with missing data in multi-class problems. Then a linear combination of one-class classifiers was applied to novelty detection in gene expression data extracted from microarrays [253]. Authors showed that a combination of one-class models display a better robustness to noise and outliers in biological data.

A first note of diversity in one-class ensembles can be found in 2008 in work of Reyes and Gilbert [233]. Here, authors enforced diversity by training each one-class model on the basis of data originating from different biological sources. This concept was interesting, but restricted only to this specific application. No rules on how to create and measure diversity in general-purpose one-class ensembles were given.

Bagging scheme for combining OCSVMs using kernel density estimation to decrease the weight given to noise was presented in [247]. During the same year a novel boosting method for one-class classifiers was introduced [307], but with the purpose of using it for multi-class classification.

Recent years saw some new developments in one-class ensembles. An efficient combined classifiers was proposed to handle uncertain data streams [313]. One-class classifiers were applied in multi-class scenarios, in order to allow an open-set classification scenario [104] (possibility of objects outside of the pre-defined set of classes) or to make a possibility of rejecting unknown samples [108].

Two interesting schemes dedicated to generating one-class ensembles were recently introduced. First one aims at fitting an ensemble of approximate polytopes [41] in order to efficiently describe the distribution of target class objects. Second method is the one-class modification of Random Forest [66]. However, this method can be seen as a little controversial as it uses binary trees as base classifiers. They are adapted to one-class problem by artificial generation of outliers - however making this not a pure one-class approach. A big limitation and weakness of this method is its reliance on the generated outliers that may often be far from the true outlier distribution.

One should notice that ensembles of one-class classifies have been also proposed for data stream mining [58], where to each class a one-class decision tree is delegated in order to capture the evolving properties of the incoming objects [59].

Pruning one-class ensembles was so far addressed only once in the literature [49], where authors proposed two exhaustive search schemes based on consistency measure or an approximation of AUC for one-class problem. The main limitation of this work was the high computational cost of carrying out a full search over a large pool of classifiers.

A soft computing-based combination module for one-class classifiers was introduced by Wilk and Woźniak [288]. They proposed to train fuzzy rules in order to efficiently harvest the support values outputted by each of the base classifiers. Their approach returned highly satisfactory results for both one-class and binary problems [287]. A similar, but much more simple approach based on fuzzy integral was later proposed in [105].

It is worthwhile to mention two ensemble approaches that were not designed for one-class classification but take inspiration from it. A SVDD model was applied as a measure of diversity in the ensemble, assuming that outlier classifiers will differ significantly from the ones already in the pool and thus add diversity [106]. SVDD was also used in the process of forming an ECOC combiner. Authors proposed to construct the data-driven coding matrix with the help of SVDD and binary tree [194]. SVDD was used to measure the class separability quantitatively to obtain the inter-class separability matrix.

1.3.5 Challenges in One-Class Ensemble Learning

With these works analyzed, one sees that the ensemble learning paradigm in one-class classification is a promising direction, but there is a need for introducing general methods for constructing compound classifiers without an access to counterexamples. General solutions for combining one-class learners should be developed and analyzed from both theoretical and experimental points of view.

Let us now identify the main areas of one-class ensembles that require a scientific development.

- New methods for forming a pool of one-class classifiers are required. Bagging and Boosting were used for one-class learning, yet a closer examination of these algorithms show that they are not well-adapted to learning in the absence of counterexamples. Bagging creates random sub-samples of data. However, as one-class methods rely on distance between object using sub-spaces consisting of distant training samples is unadvised. This may lead to training classifiers with excessive size of the classification volume (boundary) and high rate of outliers acceptance. Boosting is an approach burdened with a high risk in one-class classification. As we work only on objects from a single class iterative classifiers may become overtrained and too fitted to the training data. This may lower the generalization abilities of the ensemble. Additionally, methods that require creation of artificial samples (like one-class Random Forest) should be avoided, as artificial outliers may introduce an unwanted bias. That is why new methods for forming robust, complementary and compact one-class classifiers must be introduced.

- There are no measures on how to evaluate the diversity of one-class classifiers. Intuitively, the ensemble diversity is of as much importance as in multi-class problems. Standard measures cannot be applied, because they usually rely on the differences in errors or decisions of base learners. As we do not have an access to counterexamples, we cannot use these measures. That is why new methods must be developed to exploit the specific properties of one-class models.

- There are no dedicated pruning methods for efficient search over sets of one-class classifiers. Random subspace method is an attractive tool for one-class ensembles, but it works in an overproduce-and-select manner. Out of a plethora of generated models, one needs to chose a compact subset of learners. That is why new pruning schemes must be proposed. They should take into consideration different performance criteria of one-class learners, have acceptable computational complexity and be able to efficiently process large collections of classifiers.

- Alternative methods for combining one-class classifiers should be investigated. Weighted and trained combiners seems especially attractive, as they allow to modify the level of influence of each base learner on the final output. Currently used methods are static and do not adjust themselves to the analyzed dataset. Trained combiners can offer a more flexible characteristic of ensembles being constructed.

This thesis deals with all four of mentioned task, by proposing novel solutions and algorithms that will be described in the following sections.

1.4 Research Hypothesis

One may design such methods for forming and pruning one-class classifier ensembles that are able to outperform state-of-the-art one-class classification algorithms.

1.5 Research Aims and Goals

In order to prove the proposed research hypothesis the following aims and goals are formulated:

- To design efficient methods for constructing one-class ensembles on the basis of feature space partitioning that take into consideration the spatial relations between data and can use locally specialized classifiers.
- To propose methods for criterion-driven creation of pool of mutually complementary learners.
- To introduce pruning schemes suited for the specific nature of one-class classifiers and appropriate measures to evaluate the selection procedure.
- To prove the claims of research hypothesis by a thorough evaluation of the proposed methods on a set of diverse benchmarks with a rigorous statistical analysis of obtained results.

- To further evaluate the proposed algorithms on a set of real-life problems from the domains of environmental engineering, computer vision and medicine.

This dissertation is organized as follows:

Chapter 2 introduces four novel algorithms dedicated to forming diverse and efficient one-class classifier ensembles.

Chapter 3 formulates three diversity measures designed specifically for one-class classification task and introduces three approaches dedicated to one-class ensemble pruning.

Chapter 4 presents three real-life applications of methods introduced in this thesis to problems originating from the domains of environmental engineering, computer vision and medicine.

Chapter 5 concludes the thesis and presents potential research directions for future works.

2

Forming One-Class Ensembles

This chapter focuses on introducing new algorithms for forming ensembles of one-class classifiers that are proposed in this thesis. These methods were developed in order to counter the drawbacks of multiple classifier systems used so far for learning in the absence of counterexamples and to offer more robust and accurate classification rate.

The following four ensemble systems are proposed:

- *One-Class Clustering-Based Ensemble (OCClustE), which aims at training a pool of compact one-class classifiers on the basis of local areas of competence.*
- *One-Class Evolutionary Clustering Ensemble (OC-EvoClust), which uses an evolutionary optimization technique in order to adjust itself to the properties of analyzed dataset.*
- *One-Class Rotation Forest (OC-RotF), which is an adaptation of this ensemble technique to the OCC domain.*
- *Ensemble of Soft One-Class Classifiers based on Weighted Bagging (OC-Wagg) that uses weighted bagging (wagging) in order to introduce diversity into an ensemble of soft one-class classifiers.*

These methods will be described in details in the following sections.

2.1 One-Class Clustering-Based Ensemble

The first method proposed in this thesis is One-Class Clustering-Based Ensemble (OC-ClustE) [173]. We propose an approach based on the idea of data clustering in the feature space.

The main contributions related to this method are as follows:

- We propose building an ensemble of one-class classifiers based on clustering of the target class. This ensures initial diversity among the classifiers in the pool (as they are based on different inputs) and the correct handling of possible issues embedded in the nature of data, such as a rare distribution or chunks of objects [57]. This approach allows us to maintain the spatial relations between data (contrary to Bagging) and generate compact and robust base classifiers [174].
- We propose an elastic and efficient framework for this task, which requires only the selection of several components, namely, the clustering algorithm, individual classifier model, and classifier combination method. These can easily be chosen by the user, as there are practically no limitations on their nature. All other parameters for the method are selected automatically.
- We introduce a novel method for calculating weights for WOCSVMs (when they serve as base models for OCClustE) by utilizing the results of the space partitioning procedure.
- We discuss the possibility of extending our one-class ensemble to multi-class problems.
- We propose a number of methods for automatic detection of optimal number of clusters (competence areas).

2.1.1 OCClustE Architecture

W propose a new architecture for creating ensembles of one-class classifiers based on the clustering of a feature space into smaller partitions. The idea behind OCClustE comes from the problem on how to create a pool of accurate and diverse one-class classifiers. Due to the specific nature of OCC problems, one would like to preserve the spatial relations between data and do not allow a sutiation in which base classifier has too complex boundary or too extensive volume of the data description (which may lead to overfitting or low robustness to new, unseen observations).

In this thesis we propose using a clustering algorithm to partition the feature space. In the next step each of these clusters is used to train a one-class classifier. This leads to the formation of a pool of L classifiers assigned to the target class, as follows.

Through this we achieve the goal of creating a pool of several one-class predictors for the target class and at the same time we ensure their diversity (as a result of using different inputs in their training), which leads to better performance of the ensemble. Subsequently, a classifier fusion method combines the outputs of the classifiers to deliver a final decision. An overview of the proposed OCClustE method is illustrated in Fig. 2.1.

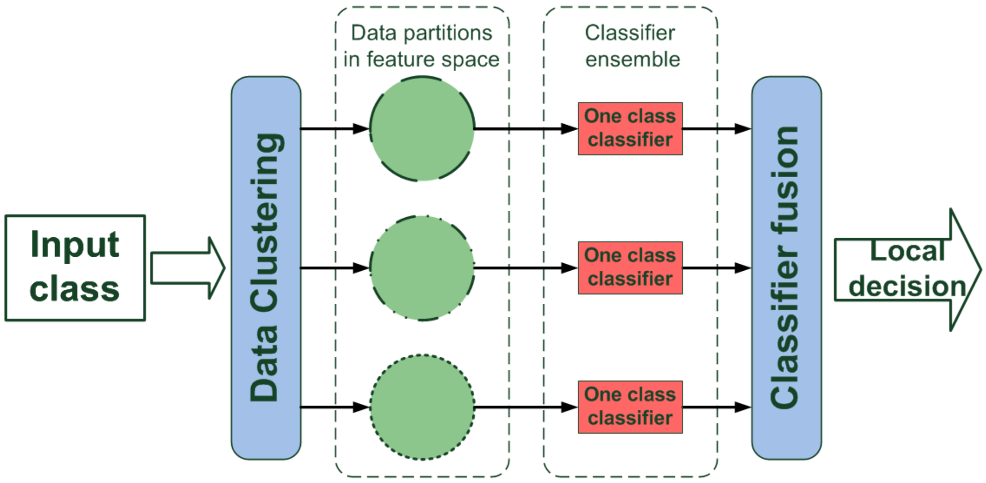


Figure 2.1: Overview of the OCClustE architecture.

Differences between the single OCC and OCClustE models for an example dataset are presented in Fig. 2.2.

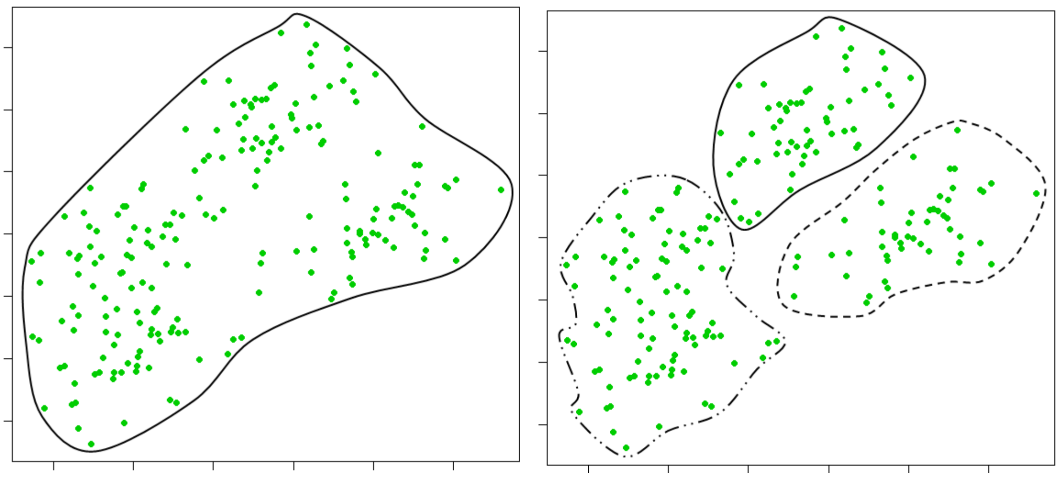


Figure 2.2: Differences between the outputs of a standard approach and the proposed one for a one-class toy problem. (*Left*) Target concept enclosed by a single model approach. (*Right*) Target concept after OCClustE classification with three clusters.

In summary, the proposed approach leads to several improvements compared with the standard OCC models:

- Boundary approaches (such as the one-class support vector machine (OCSVM)) were proven to have better generalization abilities than clustering-based (reconstruction) OCC [265], but are prone to atypical data distributions. Therefore, a hybrid method utilizing both approaches combines the advantages of each while reducing their drawbacks.
- As each classifier is trained only on a partition of the data, its complexity is lower than in the case of a single model approach. This leads to reduced probability of overtraining. Additionally, a number of individual classifiers can easily be applied in a distributed environment [241], leading to a significant decrease in execution time.
- Partitioning ensures the initial diversity and mutual complementariness of the classifier ensemble. Additionally, they become locally specialized models. Thus one may relate the detected clusters to the areas of competence of each base learner.
- Using chunks of data as the classifier input leads to a reduction in the problem known as the empty sphere; that is, the area covered by the boundary in which no objects from the training set are located [136].
- A boundary classifier trained on a more compact data partition usually has a lower number of support vectors.

2.1.2 OCClustE for Multi-Class Problems

In the case of a single-class problem, only one OCClustE model is created. Yet, this approach can easily be applied to solving multi-class problems. We now present the architecture for a multi-class classification system based on class decomposition with the local OCClustE.

The proposed architecture for multi-class problems comprises three main steps:

1. Class decomposition: in this step an M-class problem is decomposed into M one-class problems. This approach is valid for multi-class classification. In the case of

single-class classification, the decomposition can be omitted as we already have a one-class problem as the input.

2. Classification: in this step each of the classes is considered to be an independent recognition task. To solve each of these, the OCClustE algorithm is employed. Therefore, we have M local OCClustE models, each assigned to a different class.

3. Classifier combination: after the classification step we have M separate local decisions, one for each of the classes in the problem under consideration. Therefore, each of the local ensembles outputs whether the considered object x belongs to its target class ω_T or is an outlier. In this step the original multi-class problem is reconstructed by the fusion method. In the case of single-class problems the output of the local ensemble trained on the target class is also the global output of the whole system.

An outline of the global approach is presented in Fig. 2.3.

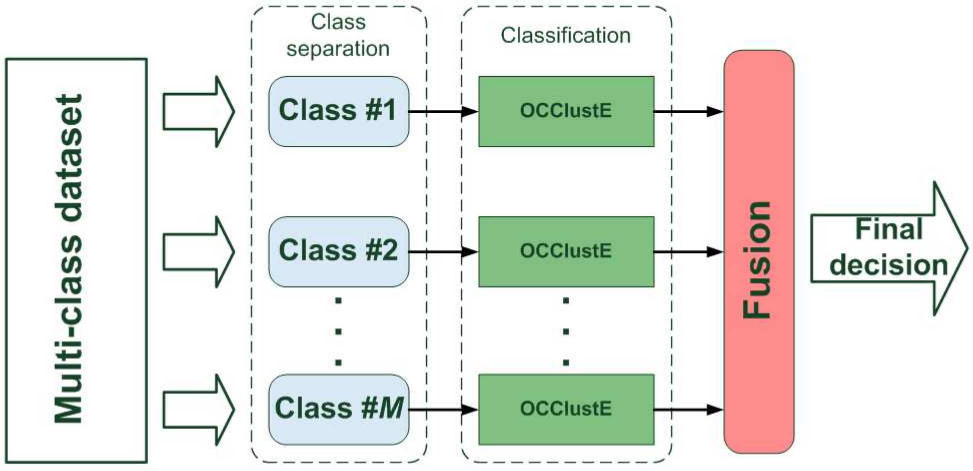


Figure 2.3: Overview of the architecture for handling multi-class problems with combined OCClustE algorithms.

Differences between single-model fusion and the OCClustE global multi-class approach are illustrated in Fig. 2.4.

Using the proposed architecture for multi-class data decomposition leads to a significantly smaller overlap between the OCC predictors assigned to each of the classes.

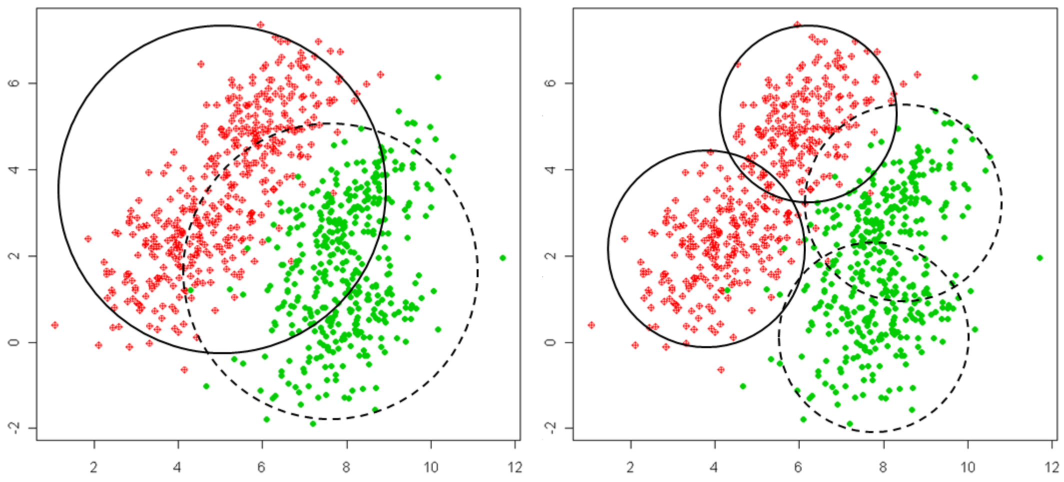


Figure 2.4: Differences between the fusion of single models assigned to each of the classes and the OCClustE for a multi-class toy problem. (*Left*) Binary problem solved by fusion of two one-class classifiers. (*Right*) OCClustE output with two clusters per class. Please note that this toy problem is solved by simple spherical one-class classifiers. One may use more complex methods (such as OCSVM) in order to get a different shape of the boundary and reduce the overlapping between decision regions.

We will now discuss the usefulness of one-class classifier ensemble for multi-class problems [176]. The main advantages of using one-class classifiers are seen in problems for which not all classes are known, or those with highly imbalanced data distributions (e.g., there are large differences in the available training patterns for different classes). Additionally, using one-class classifiers can lead to a different decision boundary; multi-class classifiers search for a optimal separation boundary, while one-class methods focus on capturing properties of a given class. Therefore, the former could be used in cases with high class overlap or where the class distribution is spread over several disjoint data chunks. There are many examples of such classification problems, e.g., in computer vision the appearance model of the object being tracked is often known, whereas those of other objects that can be encountered in the images are unknown. In some other cases, gathering training data is possible only for selected conditions of the system, e.g., when collecting data for normal engine operation it is generally not possible to collect data for failure conditions, which are rare and expensive to simulate. In these cases interpretability is also much better since other classes may not even be known. In several situations, we may have several classes at our disposal during the training phase. Yet, during the execution phase, frequent outliers, concept drift, or new classes may appear. Canonical multi-classification algorithms tend to fail under such conditions, whereas one-class ensembles are robust to such situations. Therefore, our proposed method combines the high recognition rate for multi-class problems with improved robustness to unexpected changes in data.

2.1.3 OCClustE Components

As discussed in the previous section, the proposed framework is very flexible, as it places no restrictions on the nature of its three main components:

- Clustering: any clustering algorithm can be applied to the OCClustE. However, the algorithm should be chosen carefully as data partitioning has a critical impact on the classification step, since badly defined clustering objectives may lead to the formation of a pool of weak classifiers with low diversity. Additionally, the correct number of clusters should be selected for the considered problem. A proposal on how to achieve this is presented in a later section.

- Classification: the OCClustE is designed to work with one-class classifiers, especially those based on boundary estimation. However, the choice of the type of classifier is left to the end-user of the proposed method.

- Classifier combination: classifier fusion methods must be chosen for two purposes: first, to combine the outputs of individual classifiers in the OCClustE ensemble and second (in the case of a multi-class decomposition) to combine the local outputs of each of the OCClustE ensembles to reconstruct the original multi-class task from several one-class ones.

2.1.4 Suggested Settings for OCClustE

As it can be seen from the previous section, OCClustE is a flexible framework suitable for working with almost any type of one-class classifiers, partitioning methods and combination rules. Nevertheless, on the basis of extensive computational experiments [173], we were able to establish a highly efficient setting of components for this ensemble method [174].

For space partitioning (detecting the areas of competence), we suggest to use kernel fuzzy c-means [318], which is a modification of the fuzzy c-means algorithm that operates in an artificial feature space created by a kernel function. It is more robust to atypical data distribution than standard fuzzy c-means and has a very fast convergence [318].

As the kernel function we use the *Radial Basis Function* (RBF). Here very essential is choice of the parameter σ which controls a spread of that kernel. It highly depends on the distribution of data in the training set. In our method σ was set to half a value of data variance. Such an approach was verified to operate well in practice [299].

To further boost the quality of OCClustE, we propose to use WOCSVM (presented in Section 1.2.5.4) as the base learner. It has been shown that weighted one-class classifiers can outperform the canonical ones, due to manipulating the influence degree that each object has on the shape of the decision boundary. Additionally, weighted methods are insensitive to internal outliers that may be present in the target class (as it may contain irrelevant, noisy objects). By assigning them a low weight we minimize the impact on the process of shaping the decision boundary.

The crucial element in using WOCSVM is the process of establishing weights, which is heuristic and time-consuming (see Eq. (1.46)). We introduce a novel approach for establishing the degree of importance of objects, based on the output of clustering algorithm. We use fuzzy clustering algorithm that returns the membership functions for each object in the given cluster. We use these membership values as weights for WOCSVM. This way, the new weights reflect the degree of importance of a given object in a cluster. As they are directly outputted by kernel fuzzy c-means method no additional calculations are needed and we reduce the computational time needed for training WOCSVMs.

Finally, we need to combine the individual outputs of base classifiers at our disposal. WOCSVM is based on computing the distance between the object x and the decision boundary that encloses the target class ω_T. To apply fusion methods we require the support function of object x for a given class. To obtain this, we need to use a mapping described in Eq. (1.25). In order to fuse the outputs of one-class classifiers, we propose to use the mean of the estimated support functions (described in Eq. (1.62)).

The summary of the suggested settings for OCClustE is given in Figure 2.5

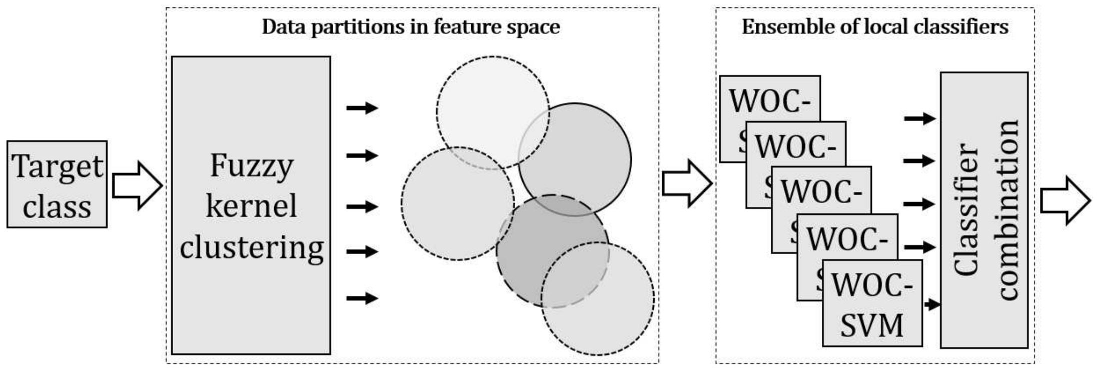


Figure 2.5: The schema of OCClustE algorithm with implemented suggested components, such as kernel fuzzy c-means and WOCSVMs.

2.1.5 Methods for Automatic Detection of the Number of Competence Areas

We would like to point out a limitation of the proposed method. OCClustE relies strongly on the number of competence areas (clusters), used during the training step. This number directly translates into a quantity of base classifiers in the ensemble. We noticed that the variance in accuracy is high even for small changes of this parameter value. An exemplary influence of the number of clusters on the OCClustE structure is depicted in Figure 2.6.

Selecting a proper number of clusters is time consuming and requires some specialist knowledge that cannot be always assumed (e.g., in case of real-life applications in decision support systems and their end-users).

Therefore, an efficient and fully automatic method for estimating a number of competence areas for OCClustE must be proposed.

As OCClustE works on the basis of clustering the object space, one may treat the problem of determining the number of competence areas as model selection for clustering algorithms [256]. This allows for an automatic selection of the optimal number of groups in data.

We investigate 10 methods for automatic selection of the clustering model that in our case represents the number of competence areas. The selection of such methods was dictated by a recent survey on their performance [310]. They form three groups: ones using only the membership matrix, ones using membership matrix and dataset and ones

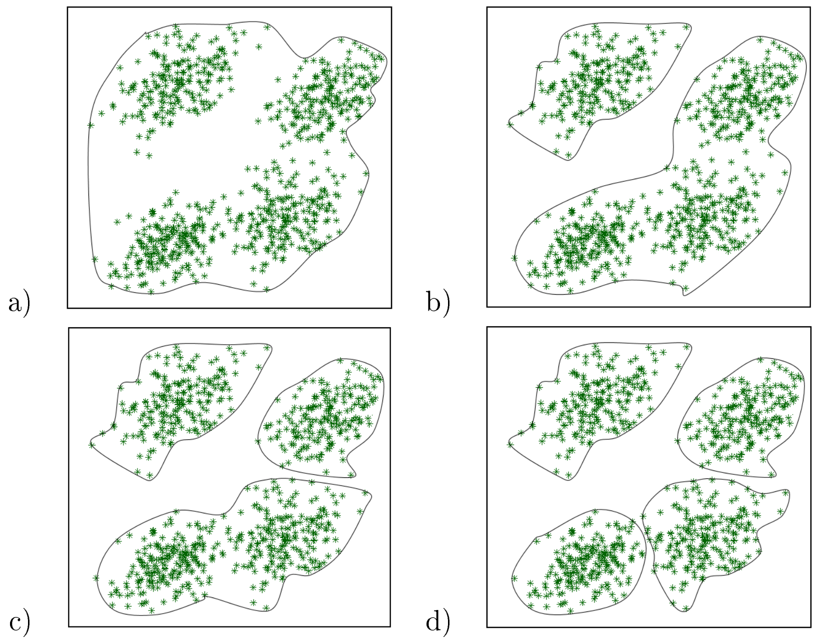


Figure 2.6: Exemplary differences between the structures of OCClustE for a different number of competence areas and base classifiers: (*a*) single one-class classifier, (*b*) two clusters detected, (*c*) three clusters detected, (*d*) four clusters detected.

based on statistical indexes.

For the description of the following methods, we assume that membership values coming from kernel fuzzy c-means are collected in the membership matrix $\mathbf{U} = [\mu_{ij}]$, where μ_{ij} stands for a membership value of j-th object into i-th cluster. We assume that we investigate a number of clusters in the range of $[1, C]$. We store the centroids of our clusters in a vector $\mathbf{C} = [c_1, c_2, \cdots, c_C]$.

The presented 10 indexes are used to evaluate the output of the clustering algorithm. Their sole purpose is to automatically identify the most suitable number of clusters among the input models.

2.1.5.1 Indexes Based on Membership Values

Partition Coefficient.

The partition coefficient (PC) can be defined as follows:

$$PC(\mathbf{C}, \mathbf{U}) = \frac{1}{N} \sum_{i=1}^{C} \sum_{j=1}^{N} \mu_{ij}^2. \tag{2.1}$$

The PC ranges between $[1/C, 1]/$ The closer the index to unity, the crisper the clustering is. Therefore, a PC value close to $1/C$ indicates that there is no clustering tendency in the analyzed data or the clustering method failed to detect one.

Partition Entropy

The partition entropy coefficient (PE) can be defined as:

$$PE(\mathbf{C}, \mathbf{U}) = -\frac{1}{N} \sum_{i=1}^{C} \sum_{j=1}^{N} \mu_{ij} \cdot \log(\mu_{ij}). \tag{2.2}$$

This index is calculated only for a number of clusters greater than 1 and ranges between $[0, \log C]$. Values of PE close to 0 indicate the difficulties in clustering of the analyzed data. Once again, the values close to the upper bond indicate $1/C$ indicates that there is no clustering tendency in the analyzed data or the clustering method failed to detect one.

Modified Partition Coefficient

The modified partition coefficient (MPC) method originates from an observation about a weakness of the PC method. It is monotonously dependent on the number of clusters C. To alleviate this, we should look for a significant knees of increase of the criterion based on the number of clusters versus PC values. One can reduce the monotony tendency by using the following formula:

$$MPC(\mathbf{C}, \mathbf{U}) = 1 - \frac{C}{C-1}(1 - C * PC(C, \mathbf{U})), \tag{2.3}$$

where $0 \leq MPC \leq 1$.

2.1.5.2 Indexes Based on Membership Values and Dataset

I Index

The I index can be defined as follows:

$$I(\mathbf{C}, \mathbf{U}, \mathcal{TS}, D_{max}) = \left(\frac{D_{max}}{C \times E_C(C, \mathbf{C}, \mathbf{U}, \mathcal{TS})} \right)^p, \tag{2.4}$$

where

$$E_C(\mathbf{C}, \mathbf{U}, \mathcal{TS}) = \sum_{i=1}^{C} \sum_{j=1}^{N} \mu_{ij} \|x_i - c_i\|, \tag{2.5}$$

where c_i is the centroid of the v_i cluster. The factor D_{max} stands for a maximum distance between the cluster prototypes. It will increase with the number of clusters.

The second factor $\frac{1}{C}$ will be responsible for reducing the value of index with the increase in the number of clusters. The third factor $\frac{1}{EC}$ measures the total fuzzy dispersion and will penalize the index with its increase. The power of p controls the contrast between the different cluster configurations. We set $p = 2$, as suggested in the literature [310].

Cluster Validity Measure

The cluster validity measure (CVM) is defined as:

$$CVM(\mathbf{C}, \mathbf{U}, \mathcal{TS}) = C + (f \times G(2,1) + 1)\frac{D_a(C, \mathbf{C}, \mathbf{U}, \mathcal{TS})}{D_e(\mathbf{C})}, \tag{2.6}$$

where$f$ is some natural constant, $G(2,1)$ is a Radial Basis Function with mean value equal to 2 and standard deviation equal to 1 and D_a is the measure of compactness of clusters:

$$D_a(\mathbf{C}, \mathbf{U}, \mathcal{TS}) = \frac{1}{N}\sum_{i=1}^{C}\sum_{x \in v_i} \|x - c_i\|^2, \tag{2.7}$$

and D_e is the measure of the average separation between two clusters over all possible pairs of clusters:

$$D_e(\mathbf{C}) = average(\|c_i - c_j\|)^2, \tag{2.8}$$

where $i \in [1, C]$, $j \in [2, C]$. CVM measure should be minimized in order to obtain accurate and compact clusters.

Fukuyama-Sugeno Index

The Fukuyama-Sugeno Index (FS) can be described as:

$$FS(\mathbf{C}, \mathbf{U}, \mathcal{TS}) = \sum_{i=1}^{C}\sum_{j=1}^{N} \mu_{ij}^{m}\|x_i - c_j\|^2 - \|c_j - \bar{c}\|^2, \tag{2.9}$$

where $\bar{c} = \sum_{i=1}^{c} c_i/C$. Small values of FS will indicate compact and separable clusters. The first term in Eq 2.9 measures the compactness of the clusters and the second measures the distances between centroid of two clusters.

Fuzzy Hyper Volume

The fuzzy hyper volume (FHV) is based on the concept of hyper volume and density and can be given as:

$$FHV(\mathbf{C}, \mathbf{U}, \mathcal{TS}) = \sum_{i=1}^{C} V_i(\mathbf{C}, \mathbf{U}, \mathcal{TS}), \tag{2.10}$$

where

$$V_i(\mathbf{C}, \mathbf{U}, \mathcal{TS}) = |\sum_i|^{1/2} = \left(\frac{\sum_{j=1}^{N} (x_j - c_i)(x_j - c_i)^T}{\sum_{j=1}^{N} \mu_{ij}^m} \right). \tag{2.11}$$

Small FH values informs about the presence of compact clusters.

Average Partition Density

The average partition density (APD) can be formulated as follows:

$$APD(\mathbf{C}, \mathbf{U}, \mathcal{TS}) = \frac{1}{C} \sum_{i=1}^{C} \frac{S_j(\mathbf{U}, \mathcal{TS})}{V_j}, \tag{2.12}$$

where $S_i = \sum_{x \in x_i} \mu_{ij}$, x_i being the set of data points within a center of cluster c_i. S_i is called the sum of the central members of the c_i cluster.

Xie-Beni Index

The Xie–Beni index (XBI), known as the compactness and separation validity function is defined as follows:

$$XBI(\mathbf{C}, \mathbf{U}, \mathcal{TS}, D_{min}) = \left\{ \frac{1}{N} \sum_{i=1}^{C} \sigma_i^2(\mathbf{C}, \mathbf{U}, \mathcal{TS}) \right\} / \{D_{min}\}^2, \tag{2.13}$$

where

$$\sigma_i^2(\mathbf{U}, \mathcal{TS}) = \sum_{j=1}^{N} \mu_{ij} \|x_j - c_i\|^2 \tag{2.14}$$

Each σ_i^2 is a fuzzy weighted mean-square error for the i-th cluster and decreases with the increase of cluster's compactness.

2.1.5.3 Methods Based on Statistical Indexes

Akaike Information Criterion

Akaike Information Criterion (AIC) is defined as:

$$AIC(\mathbf{C}, \mathbf{U}, \mathcal{TS}) = D_a + 2\mu(C)\sigma^2(C), \tag{2.15}$$

where $\mu(C) = (C-1)N + C$ is the number of degree of freedom of the model, D_a can be computed from Eq. 2.7 and the noise level σ^2 can be estimated from:

$$\sigma^2(\mathbf{C}) = \frac{D_a(C*)}{pN - \mu(C*)}, \tag{2.16}$$

where $C*$ is the maximum number of clusters, p is the co-dimension of the model (p = 1). The smaller the AIC value, the better the clustering performance for the data set.

2.1.6 Experimental Study and Discussion

In this section, we carry an experimental analysis of the proposed method. We analyze its performance over a set of 20 benchmarks, presented in Appendix A in Table A.1.

We follow an experimental framework described in Appendix B.

The experiments are divided into three parts:

- Evaluating the quality of different methods for automatic detection of the number of competence areas (discussed in Section 2.1.5).
- Checking the performance of OCClustE for one-class problems and comparing it to state-of-the-art one-class ensembles.
- Using OCClustE to decompose multi-class problems, evaluating the quality of different components for this task and comparing it to multi-class classifiers.

2.1.6.1 Automatic Estimation of the Number of Competence Areas

In this section, we compare 10 selected indexes for an automatic detection of the number of competence areas in OCClustE. Please note that these methods can be seen as model selection approaches - they work on the results returned by a single clustering algorithm (kernel fuzzy *c*-means). Their aim is to select the best number of prototypes from a wider set returned by the clustering method.

The detailed results of this experiment are given in Appendix C. In the main body of the thesis, we present only results for an exemplary datasets (to give insight in details of the method) and outcomes of statistical analysis over multiple datasets (Friedman ranking and Shaffer post-hoc), to give an outlook on the global performance of this method (for details see Appendix B).

The average quantity of detected competence areas by each method for an exemplary dataset (CYP2C19 isoform) over 5x2 CV is presented in Figure 2.7.

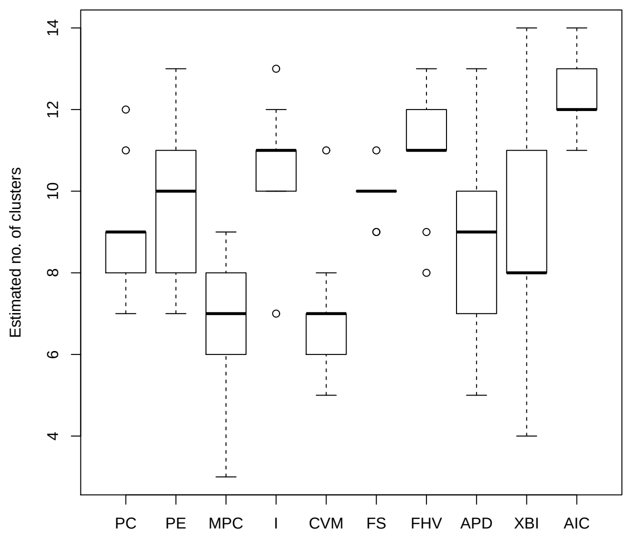


Figure 2.7: The number of competence areas (clusters) automatically detected by different methods over a 5x2 CV for CYP2C19 isoform dataset.

From the results presented in Figure 2.7 one may easily see that there is a little agreement between the methods in relation to the true number of competence areas in the dataset. What is interesting to notice is the stability of different indexes. Here, one can see that PC, I, CVM, FS, FHV and AIC return the most stable results (the number of areas does not have a large variance over the folds of 5x2 CV), while remaining methods are highly prone to the variance in data (with XBI being the most unstable one). This tendency holds for all of the examined datasets (see Appendix C).

The estimated number of competence areas is then used to form an OCClustE. We build a OCClustE classifier for each of 10 examined automatic competence areas detection methods. As for used datasets we do not have a true number of clusters, we need to rely on comparing the performance of OCClustE methods trained on different detected number of competence areas. Thus, the OCClustE model with the best quality will point out for a method that can most efficiently detect the proper number of competence areas.

The comparison of G-mean results (for details see Appendix B) for an exemplary dataset (CYP2C19 isoform) over 5x2 CV is presented in Figure 2.8.

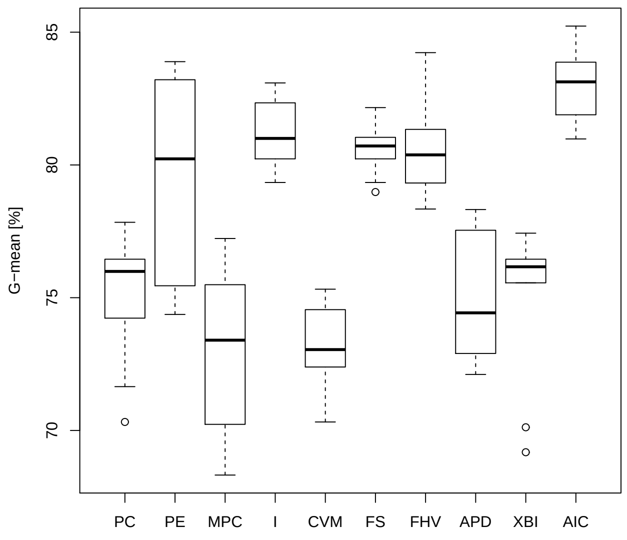


Figure 2.8: The comparison of G-mean values returned by OcclustE modes trained over the number of competence areas returned by examined methods over a 5x2 CV for CYP2C19 isoform dataset.

From the results, we can clearly see that (for the considered dataset) the AIC criterion returns the highest recognition quality from all of the 10 examined indexes. Interestingly, the high stability of the competence area detection method directly translated to a lower variance in the G-mean results over each of folds from 5x2 CV.

However, to be able to give a more global conclusions, we need to check the performance of these methods over a wider set of benchmarks. Detailed results are given in

Appendix C. Here, we present the results of Friedman ranking test, in order to check the quality of the methods over multiple datasets. The results of this ranking test are given in Figure 2.9.

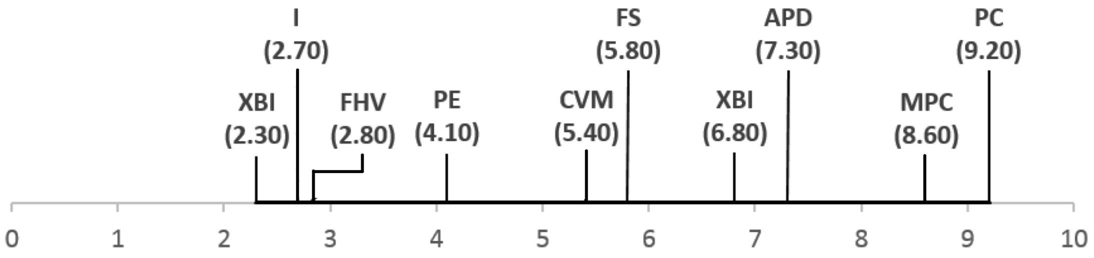


Figure 2.9: The results of Friedman ranking test for 10 different methods for competence area detection over 20 benchmark datasets.

The results of the ranking test point out that when considering a wide spectrum of datasets the AIC criterion return the best performance. As mentioned before, one can relate the obtained recognition rate with the ability of this method to properly select the number of competence areas (one of models returned from the clustering step). To provide an additional verification of this claim, we present the results of post-hoc Shaffer test over multiple datasets with the respect to AIC performance in Table 2.1.

Table 2.1: Shaffer test for comparison between the Akaike Information Criterion (AIC) and remaining nine methods for automatic detection of the number of competence areas. The test is carried over 20 benchmark datasets. Symbol '+' stands for a situation in which the proposed method is superior, '-' for vice versa and '=' represents a lack of statistically significant differences.

hypothesis	p-value
AIC vs PC	+ (0.0028)
AIC vs PE	+ (0.0307)
AIC vs MPC	+ (0.0096)
AIC vs I	+ (0.0432)
AIC vs CVM	+ (0.0242)
AIC vs FS	+ (0.0206)
AIC vs FHV	+ (0.0411)
AIC vs ADP	+ (0.0128)
AIC vs XBI	+ (0.0195)

Experimental study conducted on 10 different methods for determining the number of competence areas for purposes of training OCClustE classifier, allows us to draw several interesting conclusions.

Firstly, one can observe a great variety in the outputs of all of the examined competence area selection methods. This proves that they can return diverse results for the same set of points and thus it is worthwhile to test their behavior over a set of benchmarks. However, for few datasets (see Appendix C) an identical number of areas is returned by all of methods. This can lead to conclusions that for these cases the clustering task is relatively easy and has a single best-performing solution.

When analyzing the accuracy of different OCClustE classifiers built on the basis of different object space partitions, one may observe how crucial is the impact of a proper selection of the number of base classifiers on the final quality of the model. Therefore, looking for methods that will be able to most precisely detect the number of partitions is of great importance.

We can observe that for small datasets some methods tend to output underestimated number of competence areas, failing to discover hidden structures. On the other hand, for higher number of objects, these methods tend to over-cluster the data, locating false groups of objects. Both of these situations highly decrease the accuracy of OCClustE and should be strongly avoided.

One should also take a look at datasets, in which the best performance was delivered when no clustering was done (so a single one-class classifier for each class). Some of the metrics were able to detect that there is no clustering tendency in this data and prevented the construction of the ensemble. Training multiple classifier system in such a case would only consume computational time, without any benefit for the recognition accuracy. Such an example show that a proper evaluation metric for tuning OCClustE can also detect situations, in which committee of classifiers would be unnecessary.

We have examined ten different evaluation methods, originating in three groups. From the tests on both one-class and multi-class problems, we can observe that the best performing ones are PE, I, FVM and AIC. These four methods significantly outperform remaining six, so we will concentrate our discussion on them.

Partition Entropy is the only method that requires just the membership values for being computed. This educes the required information, but at the same time reduces the quality of the method - as PE achieves lower overall rank than the three remaining evaluation approaches. It is easy and straightforward to compute and may be efficiently used in cases with limited computational memory / time.

Both I index and Fuzzy Hyper Volume require at the same time membership values and dataset for being computed. The outperform PE method, due to the additional information embedded in the dataset. Differences between them are pretty small, as both these methods operate on similar assumptions.

Akaike Information Criterion is based on statistical information. This is the only method that succeeded for all of the used datasets. For many cases it returned identical quantity of space partitions as PE, I or FHV, but for some other benchmarks (e.g., Sonar) it was able to outperform all of the other methods.

We may conclude that PE, I, FHV and AIC are the best performing measures for automatic selection of the number of competence areas for OCCustE, with AIC being the most robust one and slightly (but in a statistically significant way) outperforming remaining approaches.

Thus, in next two experiments we will use only AIC approach for an automatic detection of the number of competence areas.

2.1.6.2 One-Class Scenario

In this section, we examine the usefulness of OCClustE classifier in one-class learning scenarios. For the experiments we use OCClustE with components discussed in Section 2.1.5 with Akaike Information Criterion for an automatic detection of the number of competence areas.

We compare our method with a single-classifier approach and other ensemble-based approaches for one-class classification based on space partitioning, namely Bagging, Random Subspace and simple clustering of the target class without any weighting module. Additionally, we add Boosting to this experimental framework, as it is considered as one of the best solutions in this field.

Each of reference ensembles is tuned with the respect to the number of base classifier from the range [5;50] for each dataset independently.

We use a WOCSVM classifier as a base method for each of examined algorithms, in order to provide a fair comparison between them.

We want to check, which of these methods will produce the most competent pool of one-class classifiers.

In Appendix C the parameter tuning, model selection and results for each of 20 datasets are presented. In the main body of the thesis, we present only results for an exemplary datasets (to give insight in details of the method) and outcomes of statistical analysis over multiple datasets (Friedman ranking and Shaffer post-hoc), to give an outlook on the global performance of this method (for details see Appendix B).

The comparison of G-mean results (for details see Appendix B) for an exemplary dataset (Spambase) over 5x2 CV is presented in Figure 2.13.

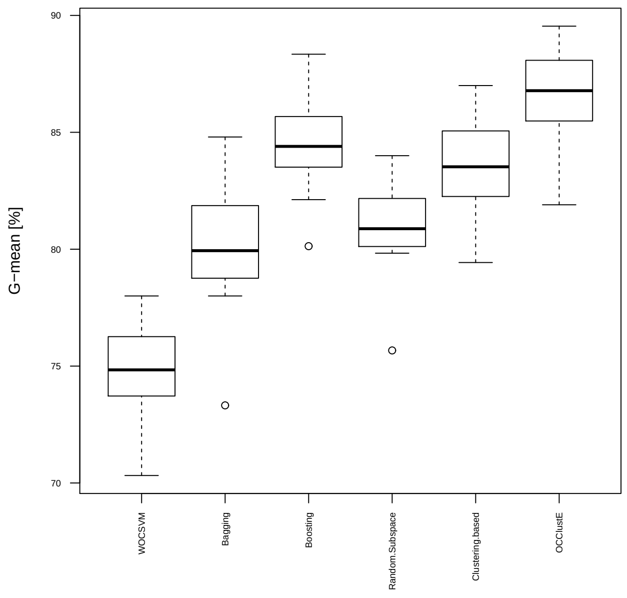


Figure 2.10: The comparison of G-mean values returned by OcclustE and reference ensemble classifiers over a 5x2 CV for Spambase dataset.

From the results, we can clearly see that (for the considered dataset) the OCClustE returns the superior performance in terms of G-mean. This can be related with the ability of OCClustE to exploit local sub-concepts in the target class. Spambase, due to its large number of examples in the target class, may be distributed in a form of chunks. Other ensemble methods cannot exploit this property embedded in the nature of data and thus deliver models that create too large volume of data description, decreasing their robustness to outliers.

To be able to give a more global conclusions, we need to check the performance of OCClustE over a wider set of one-class benchmarks. We present the results of Friedman ranking test, in order to check the quality of the methods over multiple datasets. The results of this ranking test are given in Figure 2.11.

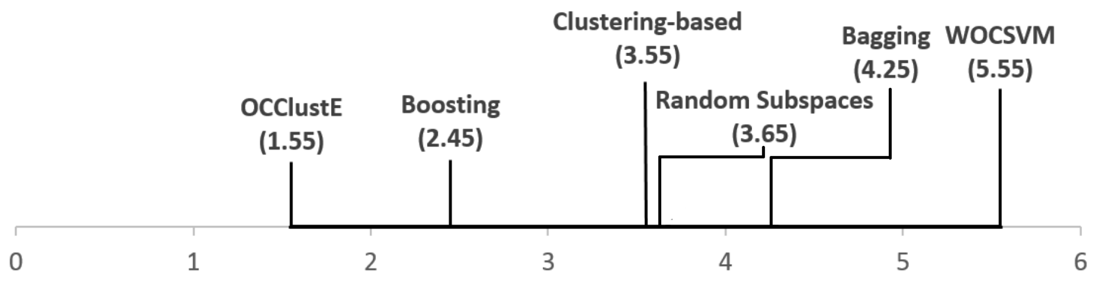


Figure 2.11: The results of Friedman ranking test for OCClustE and reference methods over 20 benchmark datasets.

The results of the ranking test point out that when considering a wide spectrum of datasets OCClustE can outperform Bagging, Random Subspaces and Boosting. This can be achieved by training more compact base classifiers that are able to adjust themselves to local properties of datasets, without producing too complex or overfitted classification boundaries. To provide an additional verification of this claim, we present the results of post-hoc Shaffer test over multiple datasets for comparison between OCClustE and reference one-class classification methods in Table 2.2.

Table 2.2: Shaffer test for comparison between OCClustE and reference one-class classification methods. The test is carried over 20 benchmark datasets. Symbol '+' stands for a situation in which the proposed method is superior, '-' for vice versa and '=' represents a lack of statistically significant differences.

hypothesis	p-value
OCClustE vs WOCSVM	+ (0.0035)
OCClustE vs Bagging	+ (0.0143)
OCClustE vs Random Subspace	+ (0.0184)
OCClustE vs Boosting	+ (0.0306)
OCClustE vs Clustering-based	+ (0.0275)

OCClustE outperforms in a statistically significant way both single WOCSVM and remaining one-class ensemble methods. What is of high importance is the fact that the proposed method, for tested cases, is never inferior to any of the remaining one-class classifiers. Shaffer test shows that OCClustE is statistically superior to all other algorithms, when taking into account its performance over multiple data sets.

OCClustE always outperforms a single WOCSVM. This is due to the fact that single one-class classifier often cannot find a good description boundary. Either the boundary volume is too big (due to the complex distribution), which leads to a high outliers acceptance rate; or it is too fitted to the data, which results in poor generalization. Using a larger number of less complex classifiers seems to reduce this problem and generates a robust classifier with good generalization abilities.

OCClustE often shows better performance than Bagging, Random Subspaces and Boosting. This is due to the fact that all of them were originally introduced for multi-class problems and are not fitted to the specific nature of OCC. Bagging uses sub-groups of objects, but do not assure that they are atomic. This leads to forming classifiers with too big volume of the decision boundary. Boosting reduces the error rate on a single class, which often may lead to the training data overfitting, especially if the number of training objects is small. Random Subspace creates classifiers without considering their individual competencies. OCClustE can alleviate all of these drawbacks by using effective kernel clustering and weighted classification.

In comparison with its simpler version (without using weighted classifiers) OCClustE always scores superior results. This shows the importance of weighting the influence of training objects in OCC. It also proves that our proposed method for calculating weights based on the cluster membership functions, is an effective way for producing accurate and diverse classifiers.

Use of the AIC method allowed for the automatic determination of the most promising number of clusters for an ensemble. The experimental results confirm that this criterion coped well with the proposed classification architecture, eliminating the time-consuming manual tuning phase.

2.1.6.3 Multi-Class Scenario

As described in Section 2.1.2 OCClustE can be directly used for decomposing a multi-class problem. In this section, we present the results of thorough experimental investigation examining the behavior of the proposed one-class ensemble approach. The aim of the experiments was to assess the behavior of different OCClustE components (clustering methods, classification algorithms and fusers) in a multi-class scenario and to compare the proposed method with known approaches for multi-class decomposition using binary and one-class classifiers (i.e., where a single one-class classifier is assigned to each of the classes).

Our aim is to evaluate whether further partitioning of the target class will lead to an improvement in recognition accuracy and to ascertain how well our method works with multi-class datasets.

We test the following components of OCClustE in multi-class scenarios:

- base classifier: OCSVM and WOCSVM;
- clustering methods: k-means, fuzzy c-means and kernel fuzzy c-means;
- combination rules: majority voting, maximum of supports and Error-Correcting Output Codes trained with exhaustive codes [4].

We compare our method with binary SVM, decomposing multi-class classifier with single WOCSVM assigned to each class and with boosting one-class support vector machines for multi-class classification [307]. Boosting is tuned for the best number of base classifiers in range [5;30] for each dataset independently.

In Appendix C the parameter tuning, model selection and results for each of 20 multi-class datasets (for details see Appendix A) are presented. In the main body of the thesis, we present only results for an exemplary datasets (to give insight in details of the method) and outcomes of statistical analysis over multiple datasets (Friedman ranking and Shaffer post-hoc), to give an outlook on the global performance of this method (for details see Appendix B).

From these results one may see that the OCClustE with WOCSVM as a base classifier, kernel fuzzy c-means space partitioning and ECOC fusion gives the best results out of all possible setting of OCClustE. Therefore, in the main body of this thesis we will only use it for comparison with other methods.

Firstly, one needs to establish a proper number of competence areas (clusters) for OCClustE in a multi-class scenario. We propose to calculate AIC criterion for each class and then use mean value to obtain the average number of clusters per class. Detailed results of the usage of AIC criterion for multi-class problems are given in Appendix C. Here, we present only the AIC values for kernel fuzzy c-means to show hwo many competence areas were identified for each of 20 considered datasets. Mean AIC values for several possible settings of the number of competence areas are given in Figure 2.12.

The comparison of accuracy results (for details see Appendix B) for an exemplary dataset (Audiology) over 5x2 CV is presented in Figure 2.13.

From the results, we can clearly see that (for the considered dataset) the OCClustE returns the superior performance in terms of accuracy. One should note that Audiology is a 24 class problem. The excellent performance of OCClustE-based decomposition can be related with the good properties of one-class classifiers applied to the problem of multi-class decomposition. They can be especially useful in case of a large number of classes, as they produce smaller ensembles than binary decomposition methods, while maintaining a good adaptation to the individual properties of each class.

To be able to give a more global conclusions, we need to check the performance of OCClustE over a wider set of one-class benchmarks. We present the results of Friedman ranking test, in order to check the quality of the methods over multiple datasets. The results of this ranking test are given in Figure 2.14.

The results of the ranking test point out that when considering a wide spectrum of datasets OCClustE can outperform other one-class methods applied for handling multi-class datasets. It achieves a similar rank as SVM-based binary decomposition, meaning that for a number of datasets it is inferior to it. This can be explained by the fact that OCC-based decomposition should be aimed at handling datasets with difficult data distributions. To provide an additional verification of our experiments, we present the results of post-hoc Shaffer test over multiple datasets with the respect to AIC performance in Table 2.3.

Table 2.3: Shaffer test for comparison between OCClustE and reference decomposition-based methods for multi-class datasets. The test is carried over 20 benchmark datasets. Symbol '+' stands for a situation in which the proposed method is superior, '-' for vice versa and '=' represents a lack of statistically significant differences.

hypothesis	p-value
OCClustE vs SVM	= (0.2185)
OCClustE vs OCC Boosting	+ (0.0238)
OCClustE vs OCC per class	+ (0.0152)

Analysis of the results (see Appendix C) clearly shows that the proposed method

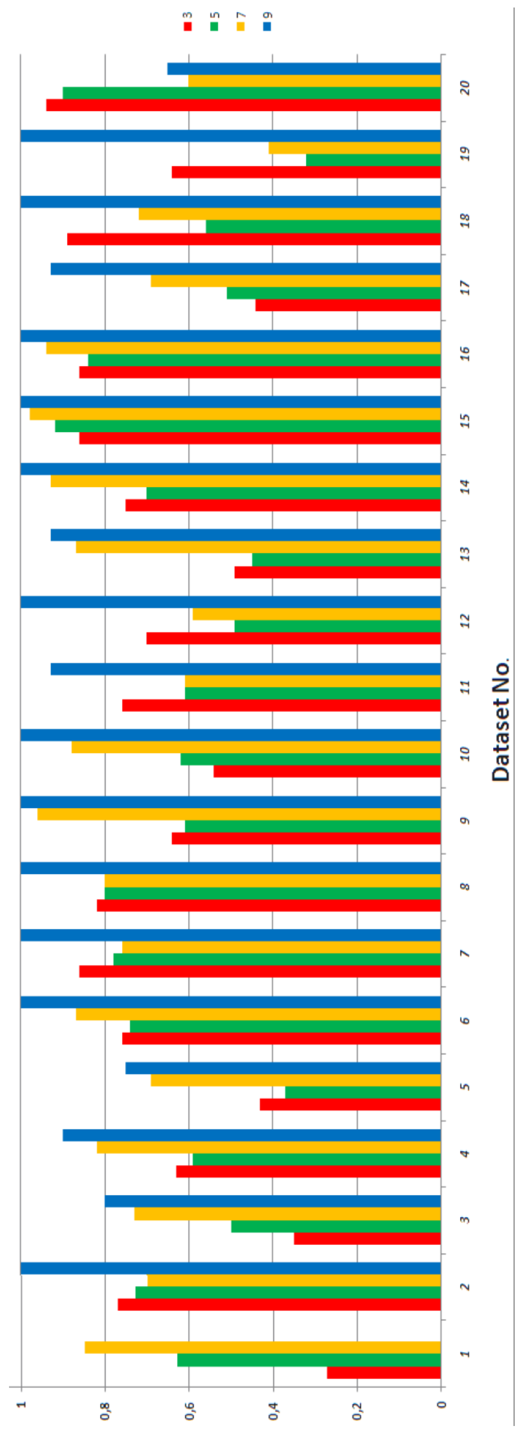


Figure 2.12: Mean AIC values over 20 datasets for kernel fuzzy *c*-means with the number of clusters {3;5;7;9}. Numbers of datasets correspond to ones described in Appendix A.

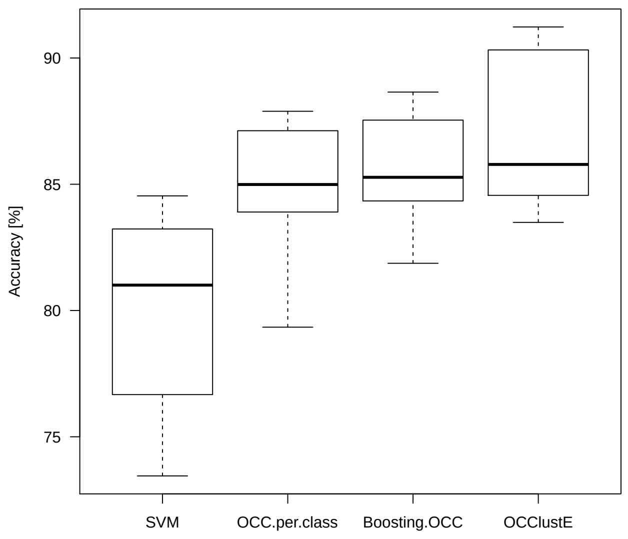


Figure 2.13: The comparison of accuracy values returned by OcclustE and reference mulit-class ensemble classifiers over a 5x2 CV for Audiology dataset.

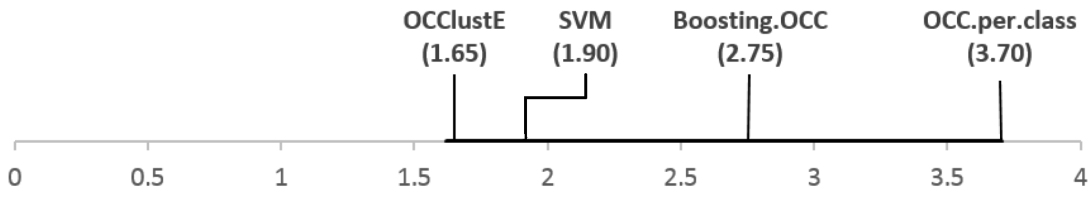


Figure 2.14: The results of Friedman ranking test for OCClustE and reference methods over 20 multi-class benchmark datasets.

displays high quality. In 16 of the 20 benchmark tests the OCClustE significantly outperformed the single OCC predictors. Additionally, in 13 cases it performed better than the multi-class SVM. This is a very interesting result as the SVM had all the data available during the training process, whereas the OCClustE relied on independent recognition of a simplified problem. This confirms our previous statement that OCC ensembles may be a useful tool for multi-class decomposition, especially in the case of difficult and imbalanced datasets.

WOCSVM outperformed the standard crisp model in 17 of the 20 cases. This shows that inclusion of weights based on the clustering membership function can lead to more diverse and at the same time highly accurate, predictors. As expected, kernel fuzzy clustering outperformed the other simpler methods.

Use of the Akaike Information Criterion allowed for the automatic determination of the most promising number of clusters for an ensemble. The experimental results confirm that this criterion coped well with the proposed classification architecture, eliminating the time-consuming manual tuning phase. The AIC is merely a suggestion for the number of clusters and better results may be achieved after manual experimentation with the settings; this was, however, not our goal. We aimed to create an ensemble classifier that would be easy for the end-user to use. To this end, automatic cluster selection by means of the AIC worked satisfactorily.

We tested three classifier combination methods, each of which represented different groups of combined approaches. We examined the behavior of fusers based on discrete (majority voting) and continuous (max and ECOC) outputs of the base classifiers. The experimental results confirmed our suspicions that voting-based approaches return inferior results; owing to the distance-based nature of boundary one-class classifiers they tend to work better with continuous outputs (mapped from the distance to the support function). As for the remaining two methods, there is a steady trend over the majority of databases in favor of the ECOC combiner. For the proposed model, the max method will work as a winner-takes-all approach, thus choosing a single best classifier from the pool. ECOC took advantage of all the classifiers in the pool, returning the best results. This shows that after the proposed clustering space split we obtain a pool of mutually supplementary classifiers that benefit from group decision making.

Compared with the one-class boosting algorithm, dedicated to one-class classification, our method is statistically superior in most cases. The authors in [307] showed that their one-class boosting algorithm was highly suited to multi-class datasets, often outperforming canonical classifiers. We achieved better results, especially when using an ECOC fuser. This is most likely due to the fact that our classifiers work on smaller data partitions and therefore the ensemble is able to fully explore their local competencies, easily dealing with underlying issues in class distributions.

Finally, let us look at the comparison of our proposed method and the multi-class SVM. SVMs are natural tools for class decomposition, as they operate on binary problems. Hence, a multi-class problem is split into several simpler problems. In our approach we split several classes into one-class problems. Each method thus offers a different view of the decomposition task. Intuitively, the SVM should yield better performance, as during the training step it has at its disposal objects originating from both classes, whereas our approach does not use counter-examples. Surprisingly, in 13 of the 20 cases our method outperformed the SVM. This leads to the conclusion that the distribution of objects in classes is too complex for a binary kernel classifier to handle. While trying to find a good trade-off among the classes it fails to capture the individual features of each of the distributions. One-class classifiers, on the other hand, work only on a single class and therefore can more easily adjust to their properties. We can conclude that the clustering step further improves the process of capturing the unique characteristics of the target class by reducing the empty regions within the decision boundary and allowing us to deal more effectively with problems such as sparse distribution, data chunks, or rare objects. A worthwhile research direction would be to compare our approach with other decomposition techniques and different algorithms for multi-class SVMs.

2.2 One-Class Evolutionary Clustering Ensemble

The second algorithm dedicated to forming one-class ensembles presented in this thesis is One-Class Evolutionary Clustering Ensemble (OC-EvoClust).

The idea of OC-EvoClust in rooted in the *Clustering and Selection* (CS) approach [184]. It divides the feature space according to a given clustering algorithm. CS selects the best individual classifier for each cluster according to its local accuracy. This algorithm assumes that the clustering step is done beforehand and then classifiers are selected independently. The main drawback of this method noticed by Jackowski and Woźniak, is the lack of interaction between the clustering step and classifier selection step. The selected clusters may be completely incompatible with the classifiers available in the pool. That is why they have proposed the Adaptive Splitting and Selection algorithm (AdaSS) [127], which fuses partitioning the feature space and assigning classifiers to each partition into one integrated process. The main advantage of using AdaSS over other algorithms from this group is that AdaSS has the ability to incorporate two tasks (feature-space partitioning and assigning individual classifiers to each partition) into a single integrated process. The main advantage of this approach is that the training algorithm considers the localization and the shape of an area to determine the content of a classifier. Consequently, the areas adapt to the competencies of the classifiers. The AdaSS idea was intensively developed by the team of Jackowski, Krawczyk and Woźniak. They proposed several AdaSS modifications, such as using different classfier selection approaches [128] or weighted and trained combination of support functions [297]. A cost-sensitive version of AdaSS that examined how the cost connected with the features impact the process of shaping the decision regions was proposed in [124]. Additionally, it was shown that AdaSS can adapt itself to non-stationary environments with the presence of concept drift, as illustrated with the example of SPAM detection [125]. The most recent version, named AdaSS+ [126] was proven to be an highly efficient classifier comparable to outperform Bagging-based and Boosting-based ensembles over a variety of datasets.

One should point out that both CS and AdaSS algorithms assume that we have a pool of classifiers given beforehand. They concentrate on finding such a space partitioning that will exploit in the best possible way the strong sides of each classifier in the pool. In case of learning in the absence of counterexamples, we are more interested in outputting efficient one-class classifiers, instead of assuming that they will be provided. That is why both CS and AdaSS do not meet the requirements of methods dedicated to learning in the absence of counterexamples.

The idea of training classifiers in some local areas of competence seems similar to OCClustE proposal. However, OC-EvoClust assumes that the partitioning of the feature space into competence areas made by any clustering algorithm may not reflect the true nature of sub-groups in data. Additionally, there is no link between the clustering and classification steps, so we cannot modify the partitioning phase according to the obtained quality of our classifiers. OC-EvoClust aims at overcoming this limitations, by offering an efficient interconnection between the clustering and classification steps. We embed the space partitioning and classifier training step into a hybrid evolutionary optimization task. This allows to change the shape of clusters to better suit the nature of one-class classifiers. An example of such an approach is presented in Figure 2.15. We further augment our method with evolutionary Random Subspace approach, where in each cluster we train a subset of classifiers based on reduced feature space and with a trained weighted fusion.

The main contributions related to OC-EvoClust are as follows:

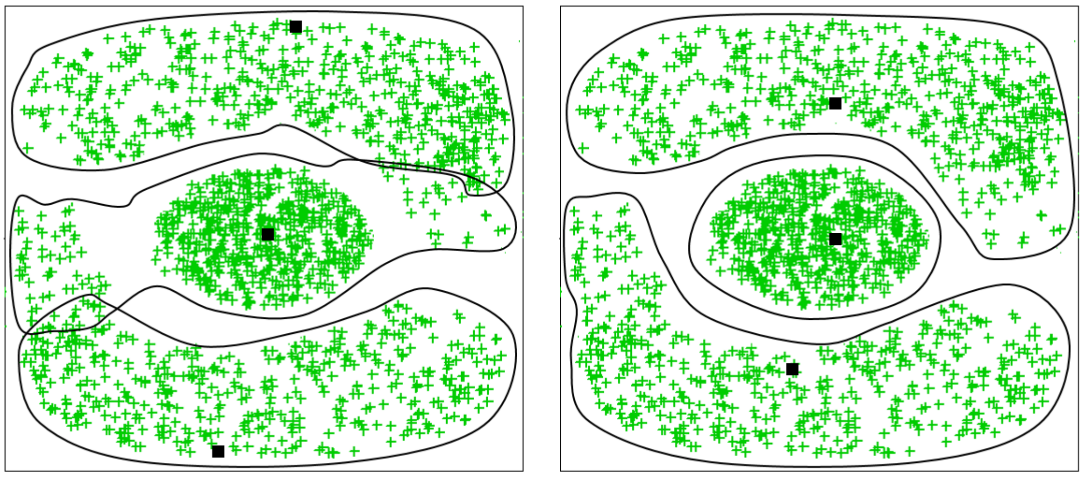


Figure 2.15: An example of different approaches to creating one-class classifiers based on the feature space partitioning. (*Left*) Three one-class classifiers trained on the basis of k-means algorithm. One can see that the partitioning does not reflect the true structure of the target class. This leads to training of incompetent classifiers with overlapping decision areas. (*Right*) Centroids with optimized locations allow for training locally specialized classifiers that are of high individual quality and complementary to each other.

- We propose building an ensemble of one-class classifiers based on clustering of the target class. This ensures initial diversity among the classifiers in the pool (as they are based on different inputs) and the correct handling of possible issues embedded in the nature of data, such as a rare distribution or chunks of objects.
- In order to create a more efficient pool of one-class learners, we propose to optimize the position of clusters' prototypes to achieve such a partitioning that will return the most diverse and robust ensemble.
- We propose to further augment our ensemble system by training a group of classifiers for each of the clusters.
- These classifiers are trained on the basis of different feature subsets and the features for each subset are selected during the training procedure.
- We propose to combine classifiers with the usage of a trained combiner based on support functions. We establish weights assigned to each classifier that will allow for the best exploitation of their local competencies.

- We introduce a hybrid, compound training procedure that embeds space partitioning, feature selection and weighted fusion into one optimization problem realized with the usage of an evolutionary algorithm.
- We propose to embed the process of selecting proper number of clusters into the evolutionary training procedure.

2.2.1 OC-EvoClust Preliminaries

We assume that all of the classifiers use the same classification model (e.g., based on support functions). One of the key issues while forming the pool of classifiers is maintaining a diversity between members. That can be accomplished by the random character of the classifier training process or by exploiting other methods, depending on the selected classifier model.

2.2.1.1 Competence Areas

As stated in the introductory section, we believe that exploiting the local competencies of the classifiers, which constitute an ensemble system, ensures decent diversity which could lead to elevating classification accuracy. For that purpose, the feature space is divided into a set of H competence areas:

$$\mathcal{X} = \bigcup_{h=1}^{H} \hat{X}_h, \tag{2.17}$$

where constituent $\hat{X}_h$ is the h-th area and

$$\forall k, l \in \{1, \ldots, H\} \quad \text{and} \quad k \neq l \quad \hat{X}_k \cap \hat{X}_l = \emptyset. \tag{2.18}$$

In our approach, $\hat{X}_h$ is represented by the centroid c_h

$$c_h = [c_h^{(1)}, c_h^{(2)}, \ldots, c_h^{(d)}]^T \in \mathcal{X}. \tag{2.19}$$

The centroids are gathered in one single set $\mathbf{C}$:

$$\mathbf{C} = \{c_1, c_2, \ldots, c_H\}. \tag{2.20}$$

The distances between the given object x and the centroids are the basis for determining the competence area, i.e., the object belongs to the area indicated by the closest centroid:

$$A(x, \mathbf{C}) = \arg\min_{h=1}^{H} dst(x, C_h) \tag{2.21}$$

Such a metric can be used for the continuous attributes that we are considering in this work; however, if one was to deal with discrete features the only difference in the approach would be to propose an adequate metric for any given type of feature.

The number of the areas plays an essential role in the performance of the system. It is difficult to define strict rules for how to choose the number of areas, because the decision strongly depends on the distribution of objects and characteristics of the classifiers (i.e., their ability to establish a decision border over the feature space). The number of areas can be selected for each problem separately, as follows:

- by an expert, based on a prior knowledge;
- in a series of experiments evaluating the number's influence on system accuracy;
- by incorporating selection of the number into the training procedure of the system to automate the selection.

2.2.1.2 Area Ensembles

The competency levels of the elementary classifiers in the areas vary themselves. Therefore, for each $\hat{X}_h$ respective area ensemble classifier $\hat{\Psi}_h$ should be created. We consider that $\hat{\Psi}_h$ is an ensemble of K classifiers trained over the h-th competence area (cluster).

As each classifier in $\hat{\Psi}_h$ is trained over the same partition of target class data, we need to ensure that these models will be diverse and complementary. We achieve this by using different subsets of available features as an input for base classifiers. Let us propose the following representation of the area ensemble classifier $\hat{\Psi_h(x)}$:

$$A_h = \begin{pmatrix} a_h^{(1)}(x^{(1)}) & \cdots & a_h^{(1)}(x^{(d)}) \\ \vdots & \ddots & \vdots \\ a_h^{(K)}(x^{(1)}) & \cdots & a_h^{(K)}(x^{(d)}) \end{pmatrix}, \tag{2.22}$$

where $a_h^{(k)}(x^{(q)}) = 1$ is the q-th feature used by the k-th individual classifier from $\hat{\Psi}_h(x)$, otherwise $a_h^{(k)}(x^{(q)}) = 0$ indicates that mentioned above attribute is not used by it. This solution allows us to use a pool of homogeneous classifiers as each of them will be trained on a different subset of features, thus ensuring diversity.

An initial solution is taken from Random Subspaces method, but the training procedure of OC-EvoClust will allow to optimize the distribution of features used to train each classifier in h-th competence area. This allows for the classifier-driven feature selection step to be embedded in the OC-EvoClust.

We assume that competencies of the elementary classifiers vary over feature space. The contribution of particular classifier to the decision making should be weighted according to its competency. For that purpose set of weights is defined for each area ensemble $\hat{X}_h$. Values of weights are set in the course of training procedure:

$$W_h = \{w_h^1, w_h^2, \dots, w_h^K, \} \tag{2.23}$$

The decision made by the area ensemble $\hat{\Psi}_h$ relies on weighted fusion of support function:

$$\hat{\Psi}_h(x) = \underset{i \in \{\omega_T, \omega_O\}}{\arg\max} \sum_{k=1}^{K} w_h^k F_i^{(k)}(x). \tag{2.24}$$

Here weights are dependent on the classifier and considered area h.

The decision made by the ensemble $\bar{\Psi}$ based on the response of the area ensemble $\hat{\Psi}_h$ for given object x is given as follows:

$$\bar{\Psi}(x) = \sum_{h=1}^{H} \left(\delta(A(x,C), h) \underset{i \in \{\omega_T, \omega_O\}}{\arg\max} \sum_{k=1}^{K} w_h^k F_i^{(k)}(x) \right), \tag{2.25}$$

where $\delta(a, b)$ denotes the Kronecker delta.

2.2.2 Training Algorithm

First, we should define the basic assumptions that have been made while designing OC-EvoClust training algorithm:

1. **Training as Compound Optimization Process**

   We propose to establish a multilateral relation between shapes and positions of competence areas, feature subsets used for each base classifier in each area ensemble and the classifier weights. The training procedure is implemented as one compound optimization task. It aims at minimization of a combination of a dedicated one-class diversity measure with consistency measure. Such a hybrid, compound training algorithm allows for adjusting areas' positions and shapes of competence areas, selecting diverse feature subsets to build local ensembles on each of established competence areas and computing weights that reflect classifier importance in the ensemble. Additionally, we incorporate the adjustment of the number of competence areas H within the learning process. This approach alleviates the difficulty of selecting H, which often had to be carried out experimentally [127].

2. **Creation of Fine-Tuned Pool of Locally Specialized Classifiers**

   Our approach allows to construct an efficient pool of base classifiers on two levels of local specialization. Firstly, a number of classifiers is specialized within a local competence area with a flexible shape in order to adjust it to the nature of analyzed data. Secondly, we further specialize them by trainingthem on a diverse subset of features. This can be viewed as an hierarchical ensemble, or an ensemble of ensembles.

2.2.2.1 Objective function

The OC-EvoClust training procedure aims at creating locally specialized and diverse classifiers. One need an evaluation criterion to control such a compound training procedure. Usually, some measure based on accuracy or diversity is used for ensemble learning task [126, 295]. In OCC neither of these can be applied. As we have only objects coming from a single class, it is impossible to estimate the accuracy on the training set. Diversity measures for binary and multi-class problems also require representatives from each class, as they are usually based on a disagreement between classifiers.

Instead of accuracy, we propose to use an unsupervised consistency measure (please see Section 1.2.2).

As for the diversity - proper measures of it had not been proposed for one-class classification so far. However, one of the contributions of this thesis is a proposal of three novel diversity measures dedicated specifically for one-class ensembles. The details of these measures can be found in Section 3.1. Any of these measures can be used as a criterion for OC-EvoClust training procedure.

To create both stable and diverse ensemble, a combination of dedicated one-class diversity measure and ensemble consistency evaluated on $\mathcal{TS}$ is used to control the OC-EvoClust training procedure:

$$Q(\bar{\Psi}) = \sqrt{\mathrm{CONS}(\bar{\Psi})\mathrm{DIV}(\bar{\Psi})}. \tag{2.26}$$

where $\bar{\Psi}$ represent the compound OC-EvoClust classifier constructed on the basis of a pool of classifiers Π, $\mathrm{CONS}(\bar{\Psi})$ is the consistency of a given pool of classifiers and $\mathrm{DIV}(\bar{\Psi})$ is the diversity of a given pool of classifiers.

2.2.2.2 OC-EvoClust Training as Compound Optimization Task

Searching for the optimal solution of the objective function from Eq. (2.26) is a compound optimization problem which cannot be simply solved by analytical methods. Therefore, the OC-EvoClust training procedure has to incorporate some heuristic procedures. We applied evolutionary based methods [8] that process population of possible solutions in an iterative manner.

Each of the solutions is encoded in a form of chromosome consisting of three parts that represent the set of set of centroids (as seen in Eq. 2.19), features on the basis of which each classifier is being trained (as seen in Eq. 2.22) and weights assigned to each classifier (2.23):

$$Chr(\mathbf{C}, \mathbf{A}, \mathbf{W}) = \begin{cases} \mathbf{C} \\ \mathbf{A} \\ \mathbf{W} \end{cases} = \begin{cases} \{C_1, C_2, \ldots, C_H\} \\ \{A_1, A_2, \ldots, A_H\} \\ \{W_1, W_2, \ldots, W_H\} \end{cases} \tag{2.27}$$

As it was previously mentioned, the number of competence areas H plays an essential role in the exploration of local competencies of the classifiers. We propose to add an extension of the training procedure which adjusts H automatically. At the beginning of the training procedure the initial value of H is set and it might be increased in the course of the optimization process.

The general form of the training procedure is presented in the Algorithm 1. The OC-EvoClust training algorithm consists of several phases. Let us now describe each of them in detail.

Initialize Population Procedure

OC-EvoClust starts with generating the population of individuals. A size of the population is an input parameter and is chosen arbitrarily. In general the larger the

Algorithm 1 Overview of OC-EvoClust algorithm

```
Require: $\mathcal{TS}$ - training set
    $\mathcal{VS}$ - validation set
    $S$ - population size
    $H$ - number of clusters
    $T$ - number of iterations
    $V$ - upper limit of algorithm iteration with falling quality
 1: Initialize Population;
 2: $V_c := 0$
 3: $Q_t := 0$
 4: for $t = 1$ to $T$ do
 5:     $Q_{t-1} := Q_t$
 6:     Evaluate Population over $\mathcal{TS}$
 7:     $s^* := \arg\max_{s \in \{1,\ldots,S\}} Q(Ch(s), \mathcal{TS})$
 8:     $Q_t := Q(c(s^*), \mathcal{VS})$
 9:     Detect Overfitting
10:     Select Elite
11:     Select Parents
12:     Mutation
13:     Crossover
14:     Create Child Population
15: end for
16: Evaluate Population over $\mathcal{TS}$
17: Postprocessing
```

population the more comprehensive optimization. On the other hand, the larger the population, the higher computational effort is required for processing. The size has to be chosen as the reasonable trade off.

Individuals in the population are initialized with randomly selected numbers with respect to the following constraints:

1. classifier weights must satisfy the following:

$$\sum_{k=1}^{K} w_h^k = 1 \quad \forall h \in \{1, \ldots, H\}; \tag{2.28}$$

2. all centroids C_h have to fall into the space limited by the boundaries of feature space specific for given recognition problem in hands.

$$\begin{gathered} C_h \in < \min_{n=1}^{N} x_n^{(l)}, \max_{n=1}^{N} x_n^{(l)} > \\ \forall h \in \{1, \ldots, H\} \quad \text{and} \quad \forall l \in \{1, \ldots, d\}; \end{gathered} \tag{2.29}$$

3. the features in **A** are initialized with the Random Subspaces procedure in such a way that each classifier has selected 60% of original features. Please note that during the course of OC-EvoClust training the number of features used by classifiers may change and it may vary among individual classifiers.

4. the location of centroids of competence areas is initialized with k-means clustering method. Then, the training algorithm optimizes centroids locations and quantity. In the experiments thorough this thesis, we initialize OC-EvoClust training procedure with initial number of clusters set to 3.

Evaluate Population Procedure

In every step of the algorithm, we train one-class classifiers in a hierarchical architecture. Firstly, we create as many area ensembles as there are currently processed competence areas. In each area classifier, we train K classifiers on the basis of feature space partitions encoded in this specific iteration. Then, the individual outputs of each classifier from each area ensemble is being combined with the weights encoded in this specific iteration.

Individuals in the population are evaluated by functions calculating their fitness value (according to Eq. (2.26)) using samples stored in the learning set. The results form the basis for selecting the elite and parents, which undergo mutation or crossover.

Select Elite Procedure

To maintain the stability of the learning procedure, the elite is selected from the population and promoted to the offspring population not affected by crossover and mutation operators. Two individuals that feature the smallest misclassification rate are chosen from the population to form the elite.

Select Parents Procedure

Only selected individuals from the population are chosen for further processing by mutation and crossover operators. To maintain the diversity of the population, which is essential for ensuring the ability to explore the space of possible solution of the problem effectively, the selection process has to be realized in probabilistic manner i.e., not only are the best chromosomes chosen, but also the probability of selection is straight proportional to their misclassification rate. To meet this assumption, a standard roulette selection procedure was implemented.

Mutation

The first procedure which is essential for population processing is mutation [8]. By assumption, it shall inject some randomness into chromosomes. The mutation procedure is split into two sequentially launched parts:

1. increasing number of competence areas,
2. mutation of existing chromosome parts.

The first one compares the misclassification rate obtained by selected individuals with the results obtained in previous generations. If no improvement has been noticed in the last generation, the algorithms might extend the number of competence areas to elevate flexibility of their boundaries. The length of $\mathbf{C}$, $\mathbf{A}$ and $\mathbf{W}$ in the chromosome are increased by one and the new constituents are filled with a copy of C_h, A_h and W_h of randomly selected existing constituents. Nonetheless, the procedure is not launched automatically, but its possibility increases with time and is directly proportional to number of generations.

The second mutation procedure differs in respect to the components of chromosome. For the binary-encoded part of the chromosome (feature set $\mathbf{A}$), we use a random change of bit value, according to the given probability. For the real-encoded part (location of competence area centroids $\mathbf{C}$ and classifiers weights $\mathbf{W}$) mutation involves adding a vector of numbers randomly generated according to a normal density distribution (with mean of 0 and standard deviation of Δ_m).

Crossover

The crossover operator is the second core procedure of evolutionary algorithms. It exchanges data between two parent individuals to form child chromosomes. Because, the chromosome in OC-EvoClust consists of three parts, we need to process them differently with regard to their type. For the binary-encoded part of our chromosome (feature set $\mathbf{A}$), offsprings are obtained according to the two-point rule. For the real-encoded part (location of competence area centroids $\mathbf{C}$ and classifiers weights $\mathbf{W}$), we apply uniform arithmetic crossover.

Nonetheless, we have to keep in mind that the population can consists of individuals with different lengths of chromosome parts, because the mutation operator happens to adjust the competence area number. Only chromosomes with the same lengths can be processed by the crossover. The crossover procedure is realized in a series of repeated operations until two parents of the same length are combined if it is possible:

1. selecting the first parent from the population
2. selecting the second parent from the population
3. if the parents' lengths are the same perform crossover
4. if the parents' lengths are not the same and there are other parents to select go to step 2.

The weights part of the chromosome are normalized.

Create Child Population Procedure

Elite individuals along with those affected by mutation and crossover operators are joined together and form an offspring population. It is processed in the next generation of the algorithm.

Detect Overfitting Procedure

This procedure protects learning procedure against classifier overfitting [5]. A validation set is required for that purpose which does not overlap with the learning set. The procedure evaluates the population result with Eq. (2.26) over the validation set and compares the results with the ones obtained in the previous generation. If the current results are better than the previous ones, overfitted has not been detected and the current population is saved, because the population is not affected by overfitting. In other cases, training procedures are canceled after O iterations without improvement and the last saved population, not affected by overfitting, replaces current population.

The pseudocode of the procedure is listed in Algorithm 2.

Algorithm 2 Pseudo-code of the Detect Overfitting procedure.

```
Require: O, $O_c$, $t$, $Q_t$, $Q_{t-1}$
1: if $Q_t < Q_{t-1}$ then
2:     $O_c := O_c + 1$
3:     if $O_c = O$ then
4:         break
5:     end if
6: else
7:     $O_c := 0$
8: end if
```

2.2.3 Experimental Study and Discussion

In this section, we carry an experimental analysis of the proposed method. We analyze its performance over a set of 20 benchmarks, presented in Appendix A in Table A.1.

We follow an experimental framework described in Appendix B.

The experiments had two primary goals:

- Evaluating the usefulness of OC-EvoClust for one-class problems and comparing it with state-of-the-art one-class ensmebles.
- Analyzing the convergence of the training algorithm over a given number of iterations.

We compare our method with a single-classifier approach and other ensemble-based approaches for one-class classification based on space partitioning, namely Bagging, Random Subspace and simple clustering-based ensemble. Additionally, we add Boosting to this experimental framework, as it is considered as one of the best solutions in this field.

Each of reference ensembles is tuned with the respect to the number of base classifier from the range [5;50] for each dataset independently.

We use a SVDD classifier as a base method for each of these algorithms, in order to provide a fair comparison between them. We want to check which of these methods will produce the most competent pool of one-class classifiers.

OC-EvoClust uses a Sphere Intersection diversity measure (see Section 3.1.3) for its objective function (see Eq. (2.26)).

In Appendix C the parameter tuning, model selection and results for each of 20 datasets are presented. In the main body of the thesis, we present only results for an exemplary datasets (to give insight in details of the method) and outcomes of statistical analysis over multiple datasets (Friedman ranking and Shaffer post-hoc), to give an outlook on the global performance of this method (for details see Appendix B).

The comparison of G-mean results for an exemplary dataset (p53 Mutants) over 5x2 CV is presented in Figure 2.16.

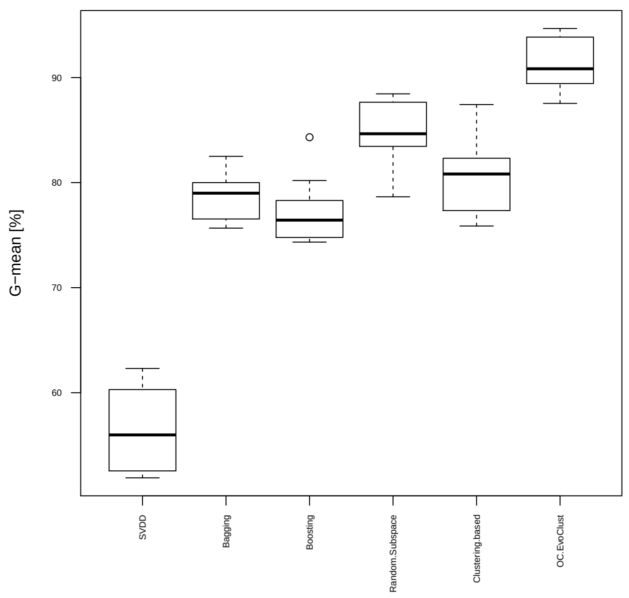


Figure 2.16: The comparison of G-mean values returned by OC-EvoClust and reference ensemble classifiers over a 5x2 CV for p53 Mutants dataset.

From the results, we can clearly see that (for the considered dataset) the OC-EvoClust returns the superior performance in terms of G-mean. One should note that p53 Mutants is a dataset with a relatively high number of objects that are described by several thousands of features each. We can alleviate the problems related to a high number of training objects (too complex decision boundaries, overestimated classifier volume) by creating classifiers on the basis of local areas of competence and we can deal with problems related to a high number of features (high training complexity, increased processing time) by training each classifier on the basis of a reduced subset of features. As both of these objects are controlled by our criterion function (combination of consistency and diversity), we get locally competent and diverse base classifiers. As one can see, other methods return inferior performance in comparison with OC-EvoClust, as they are not able to deal with the quantity of objects or features.

Let us now analyze the converge rate of the OC-EvoClust training procedure, with special attention paid to the influence of the process of automatic creation of new competence areas (clusters). The detailed run over 200 iterations for p53 Mutants dataset is given in Figure 2.17.

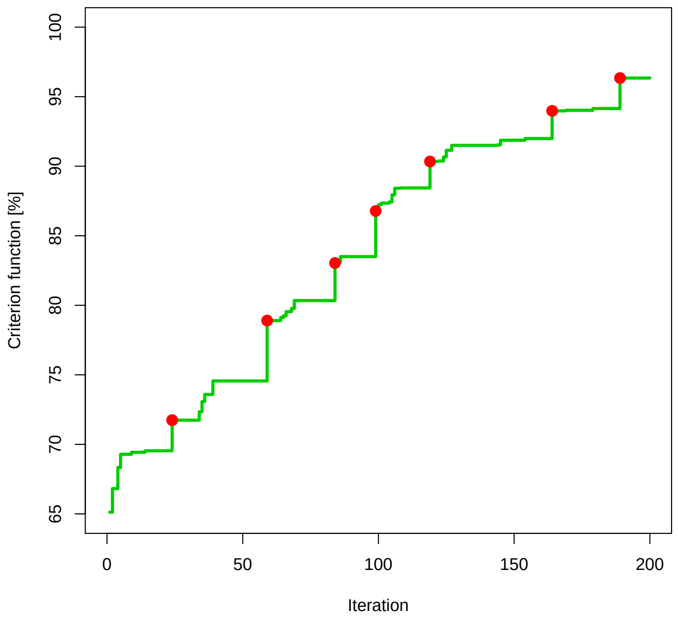


Figure 2.17: The detailed performance of the OC-EvoClust training procedure in 200 iterations for p53 Mutants dataset. Red points mark the moments in which a new competence area was added to the ensemble.

To be able to give a more global conclusions, we need to check the performance of OC-EvoClust over a wider set of one-class benchmarks. We present the results of

Friedman ranking test, in order to check the quality of the methods over multiple datasets. The results of this ranking test are given in Figure 2.18.

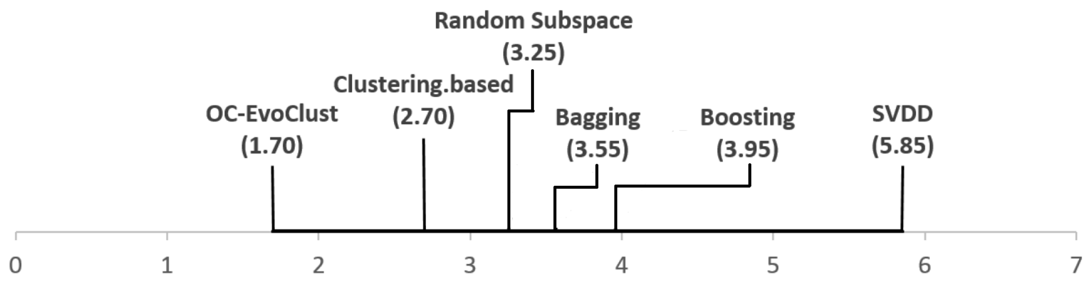


Figure 2.18: The results of Friedman ranking test for OC-EvoClust and reference methods over 20 benchmark datasets.

The results of the ranking test point out that when considering a wide spectrum of datasets OC-EvoClust can outperform Bagging, Random Subspaces, Boosting and simple Clustering-Based ensemble. This can be explained by the ability of OC-EvoClust to adjust the shape of competence areas in order to train compact and diverse base classifiers, to its hybrid architecture, as well as to the usage of trained weighted fusion. To provide an additional verification of this claim, we present the results of post-hoc Shaffer test over multiple datasets for comparison between OCClustE and reference one-class classification methods in Table 2.4.

Table 2.4: Shaffer test for comparison between OC-EvoClust and reference one-class classification methods. The test is carried over 20 benchmark datasets. Symbol '+' stands for a situation in which the proposed method is superior, '-' for vice versa and '=' represents a lack of statistically significant differences.

hypothesis	*p*-value
OC-EvoClust vs SVDD	+ (0.0018)
OC-EvoClust vs Bagging	+ (0.0179)
OC-EvoClust vs Random Subspace	+ (0.0207)
OC-EvoClust vs Boosting	+ (0.0129)
OC-EvoClust vs Clustering-based	+ (0.0264)

Results of Shaffer post-hoc test confirm the high quality of the proposed method.

The main advantages of OC-EvoClust method lies in its adaptability to complex datasets. Therefore, we can easily exploit the local decision areas independently according to the divide-and-conquer principle. The combination of creating classifiers on the basis of smaller subsets of objects together with reduced feature space allows OC-EvoClust to efficiently handle massive and multi-dimensional problems.

We can observe that the number of competence areas on which the feature space split plays a key role in the performance of the OC-EvoClust ensemble. Intuitively, the higher number of competence areas leads to smaller, thus easier to cover, decision regions. Therefore, one may think that by achieving a very dense partitioning we can improve the quality of recognition. Yet the goal of the partitioning is different from a standard clustering procedure. We aim at finding regions that do not consist of similar objects, but that allow for easiest discrimination between the target and outlier classes. In such a case, we can formulate a hypothesis that a high number of competence areas may easily lead to overfitting the OC-EvoClust ensemble, because we get too complex compound decision boundary.

The proposed OC-EvoClust eliminates the problem of manual setting the number of competence areas by introducing an automatic procedure embedded in the evolutionary training algorithm. One should remember that it is recommended to start from a low number of clusters (usually 2 or 3). Most importantly, the proposed automatic selection of the number of the competence areas always performs no worse than the manual setting.

This procedure can be seen as a big advantage of OC-EvoClust, although it has a strong bias. New cluster is constructed on the basis of splitting the biggest one in the previous iteration into two equally sized clusters. This may cause a situation, where we create a new partition in a wrong place of the feature space, as a cluster placed in a different place could improve the ensemble performance. Nevertheless despite this bias the proposed method still return highly satisfactory results.

The increase of the training time of the OC-EvoClust when using this automatic procedure is still lower than the potential time spent on manual tuning. In many cases the end-users of the classification tools are not experts in the field of machine learning. Such people see the classifier as a black-box and have no knowledge of how to set the optimal number of clusters. This leaves them with a time-consuming trial-and-error approach that cannot guarantee to return good results. By introducing an automatic cluster quantity selection for OC-EvoClust, we reduce the number of free parameters that require setting by the users and make our proposal more elastic and easy to use.

The experimental results tend to confirm our assumptions about the overfitting occurring in highly partitioned feature spaces. There is no steady correlation between the increase of the number of clusters and the increase or decrease of the ensemble error. The automatic procedure selected a high number of competence areas at seldom occasions. This is quite an revealing finding, because there was no upper bound on the number of clusters for the training procedure. Additionally, with the increase of the number of competence areas the procedure which protected against overfitting activated itself more and more often.

Automatic selection of the number of the competence areas, adjusting their shapes and positions, training an area ensemble on the basis of reduced subset of features along with weighted classifier fusion models allowed to create a very flexible and efficient hybrid classifier ensemble for one-class classification.

2.3 One-Class Rotation Forest

The third method proposed in this thesis is One-Class Rotation Forest (OC-RotF), a one-class adaptation of the Rotation Forest ensemble [235].

Rotation Forest was introduced in 2006 as a counterpart to the Random Forest classifier [34]. While Random Forest concentrated on creating a large ensemble of trees constructed on the basis of random feature subsets, Rotation Forest proposed to concentrate on smaller-sized committees consisting of more accurate and diverse base learners. It produces base classifiers by splitting the feature space into a given number of subsets and applying Principal Component Analysis on each of them. Rotation Forest retains all principal components in order to preserve the variability information in the data. Thus it uses a given number of axis rotations to form an ensemble. The rotation approach encourages simultaneously individual accuracy and diversity within the ensemble. Rotation Forest uses decision trees as base classifiers, but there are no imposed restrictions that only this model can be applied for this framework.

The idea of OC-RotF originates from the recent proposal of One-Class Random Forest (OC-RandF) [66]. Authors proposed to adapt Random Forest to one-class learning scheme, as using a reduced feature space is an attractive property (due to the growing complexity of one-class learners in higher dimensions). However, the proposed OC-RandF has a major drawback - it uses binary trees as base learners. Authors proposed a novel scheme for generating artificial outliers in reduced feature subsets, in order to train binary trees. They argued that the proposed classifier can outperform many other one-class learners. However, this is not a canonical one-class method. It rather transforms the one-class problem into binary one and solves it with two-class approach. One can see how strong dependency lies between the classification model and the quality of generated outliers. Additionally, binary trees create a dichotomization hyperplane, not a data description. So they operate on different principles than one-class learners. Finally, authors proposed a rather dubious experimental analysis, where no proper metric of the robustness to outliers were used. This observations lead to a proposal of novel one-class ensemble learning method that would use one-class classifiers as base learners and would be in agreement with the principles of learning in the absence of counterexamples.

We selected Rotation Forest as basis of our approach, as it preserves advantages of Random Forest, while using unsupervised feature extraction methods that can work well with one-class classifiers.

The main contributions related to this method are as follows:

- A formulation of Rotation Forest ensemble for one-class classification task in such a way that it does not require artificial outliers to be trained.
- We show that unsupervised feature extraction methods can be efficiently used to prepare input data for one-class classifiers.
- We examine the quality of different feature extraction methods applied in OC-RotF.
- We conduct a thorough comparison with OC-RandF, showing that we can achieve more robust one-class classification with smaller ensembles.

2.3.1 Details of the Algorithm

Let us present the steps for preparing an input datasets for l-th classifier $\Psi^{(l)}$ from the pool:

1. Split $\mathcal{X}$ randomly into S subsets. The subsets may be disjoint or intersecting. Standard Rotation Forest assumed disjoint subspaces. However, this prevented

us from exploring hidden dependencies between features and limited the usage of this ensemble for small-dimensional datasets. In OC-RotF we propose to allow for intersecting subsets, just as in Random Subspaces approach. We assume that all subspaces use identical number of features N_S.

2. For each of S such feature subset draw a bootstrap sample B_s of training objects equal to 75% of the original dataset.

3. Let us run a chosen feature extraction algorithm (see Eq. 1.5) on the s-th subset of features and corresponding bootstrap sample of training objects. Store the extracted coefficients for s-th subset for l-th classifier as a vector $[y_l^{(1)}, y_l^{(2)}, \cdots, y_l^{(N_S)}]$. Please note that it is possible that some of the coefficients will be equal to zero.

4. Let us organize the obtained vectors into a sparse rotation matrix $\mathbf{R}$:

$$\mathbf{R} = \begin{bmatrix} [y_1^{(1)}, y_1^{(2)}, \cdots, y_1^{(N_S)}] & [0] & \cdots & [0] \\ [0] & [y_2^{(1)}, y_2^{(2)}, \cdots, y_2^{(N_S)}] & \cdots & [0] \\ \vdots & \vdots & \ddots & \vdots \\ [0] & [0] & \cdots & [y_S^{(1)}, y_S^{(2)}, \cdots, y_S^{(N_S)}] \end{bmatrix} \tag{2.30}$$

   The rotation matrix will be of dimensionality $d \times N_S$.

5. Train l-th classifier using objects from B_s and $\mathbf{R}$ as the new feature space input.

The pseudocode of the proposed OC-RotF method is given in Algorithm 3.

One should note that the proposed OC-RotF algorithm can work flexibly with any feature extraction method. In this thesis we propose to investigate the performance of three feature extraction methods:

- Principal Components Analysis (PCA) [134];
- Independent Component Analysis (ICA) [122];
- Local Fisher Discriminant Analysis (LFDA) [255];

Algorithm 3 Overview of OC-RotF algorithm

```
Require: $\mathcal{TS}$ - training set
    $\mathcal{X}$ - feature space
    $L$ - size of the ensemble
    $S$ - number of feature subsets
    $N_S$ - size of feature subsets
    $\Psi$ - one-class classifier
1: for $l = 1$ to $L$ do
2:   split $\mathcal{X}$ into $S$ subsets of $N_S$ size
3:   for $s = 1$ to $S$ do
4:     $B_s \leftarrow$ bootstrap sample from $\mathcal{TS}$ with features from $s$-th subset
5:     apply feature extraction method on $B_s$ to obtain coefficients
6:   end for
7:   $R_l \leftarrow$ arrange coefficients as in Eq. (2.30)
8:   $\Psi \leftarrow$ train one-class classifier ($B_s$, $R_l$)
9: end for
```

They transform the input space into a new rotated feature space, thus allowing for introduction of diversity among base learners.

The combination of base classifiers can be done with the usage of any approach discusses in Section 1.3.4 in Eq. (1.59) – (1.63).

OC-RotF has several advantages over standard one-class ensembles and OC-RandF. Unsupervised feature extraction methods are suitable for one-class problems, as they do not require an access to class labels. Additionally, they can transform the feature space into a more compact one, thus reducing the data description volume outputted by each base classifier. OC-RotF is highly flexible and can work with any type of one-class classifier (unlike OC-RandF, which can work only with decision trees). OC-RotF maintains the advantages of multi-class Rotation Forest, i.e., efficient introduction of diversity among the ensemble members and creating competent individual classifiers. Finally, OC-RotF does not require an access to counterexamples, thus being a native one-class method.

2.3.2 Experimental Study and Discussion

In this section, we carry an experimental analysis of the proposed method. We analyze its performance over a set of 20 benchmarks, presented in Appendix A in Table A.1. We follow an experimental framework described in Appendix B.

The experiments had two primary goals:

- Comparing the efficiency of different feature extraction methods.
- Evaluating the quality of OC-RotF in comparison with other one-class ensembles, especially OC-RandF.

2.3.2.1 Examination of Different Feature Extraction Methods

In this section, we would like to compare the quality outputted by three different feature extraction schemes: PCA, ICA and LFDA.

In Appendix C the results for each of 20 datasets are presented. In the main body of the thesis, we present only results for an exemplary datasets (to give insight in details of the method) and outcomes of statistical analysis over multiple datasets (Shaffer post-hoc), to give an outlook on the global performance of this method (for details see Appendix B).

The comparison of G-mean results for an exemplary dataset (Sonar) over 5x2 CV for examined feature extraction methods is presented in Figure 2.19.

From the results, we can clearly see that (for the considered dataset) the PCA feature extraction methods leads to forming of the most accurate ensemble. LFDA returns slightly less accurate ensembles, while ICA is inferior to both of the mentioned methods. This can be explained by efficient PCA mapping into orthogonal spaces. This can positively affect the data description volume outputted by each base one-class classifier and improve the diversity of descriptors (as target class is projected over different directions in space).

To be able to give a more global conclusions, we need to check the performance of examined feature extraction methods over a wider set of one-class benchmarks. We present the results of Shaffer post-hoc test (as conducting ranking test for only three methods is nor recommended [65]), in order to check the quality of the methods over multiple datasets. The results of this ranking test are given in Table 2.5.

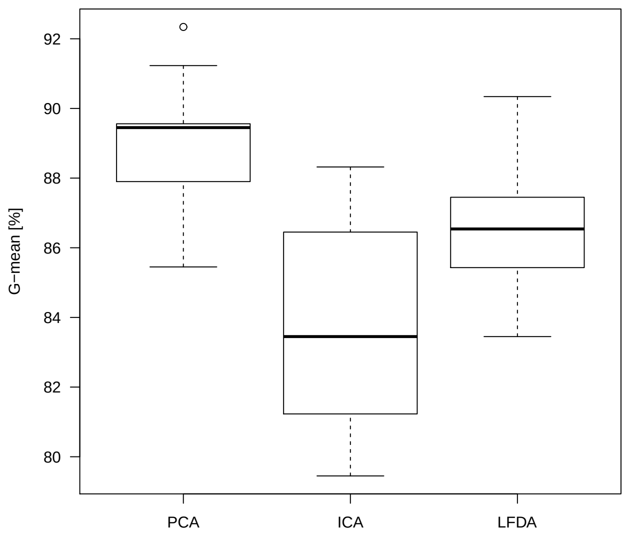


Figure 2.19: The comparison of G-mean values returned by OC-RotF with PCA, ICA and LFDA extraction methods over a 5x2 CV for Sonar dataset.

Table 2.5: Shaffer test for comparison between different feature extraction methods applied in OC-RotF. The test is carried over 20 benchmark datasets. Symbol '+' stands for a situation in which the proposed method is superior, '-' for vice versa and '=' represents a lack of statistically significant differences.

hypothesis	p-value
PCA vs ICA	+ (0.0142)
PCA vs LFDA	+ (0.0469)
ICA vs LFDA	- (0.0218)

Results of Shaffer post-hoc test confirm that PCA delivers the best performance in general classification framework. ICA is inferior over all of datasets to its two counterparts. One should take a look onto the performance of LFDA. Although Shaffer test points out that it is statistically inferior to PCA, the obtained p-value is very close to the significance level of the test. That indicates that there are some datasets in which LFDA is able to outperform PCA. This can be seen after a careful study of results presented in Appendix C. No systematic trend was found that would allow to form recommendations on when to apply LFDA. That is why we recommend to check it manually for analyzed datasets, as it may improve the performance of OC-RotF in some cases. In the next experimental study we will use OC-RotF with PCA to allow for a general comparison with other one-class ensembles.

2.3.2.2 Comparison with Other One-Class Ensembles

In this section, we would like to compare the proposed OC-RotF with state-of-the-art one-class ensembles.

In Appendix C the results for each of 20 datasets are presented. In the main body of the thesis, we present only results for an exemplary datasets (to give insight in details of the method) and outcomes of statistical analysis over multiple datasets (Friedman and Shaffer post-hoc), to give an outlook on the global performance of this method (for details see Appendix B).

We compare our method with a single-classifier approach and other ensemble-based approaches for one-class classification, namely Bagging, Random Subspace, Boosting and OC-RandF. One-Class Random Forest uses standard binary trees, while all remaining methods use a SVDD classifier as a base learner, in order to provide a fair comparison between them. The number of base classifiers used by each ensemble was optimized for each dataset independently from the range [5;50].

We want to check, which of these methods will obtain the highest classification quality (balance between generalization over the target class and robustness to outliers).

The comparison of G-mean results for an exemplary dataset (Sonar) over 5x2 CV for examined feature extraction methods is presented in Figure 2.20.

One may see that OC-RotF can outperform all of the reference one-class methods for Sonar dataset.

To be able to give a more global conclusions, we need to check the performance of OC-RotF over a wider set of one-class benchmarks. We present the results of Friedman ranking test, in order to check the quality of the methods over multiple datasets. The results of this ranking test are given in Figure 2.21.

The results of the ranking test point out that when considering a wide spectrum of datasets OC-RotF can outperform OC-RandF without the use of any counterexamples. Additionally, it is generally ranked higher than other reference methods. To provide an additional verification of this claim, we present the results of post-hoc Shaffer test over

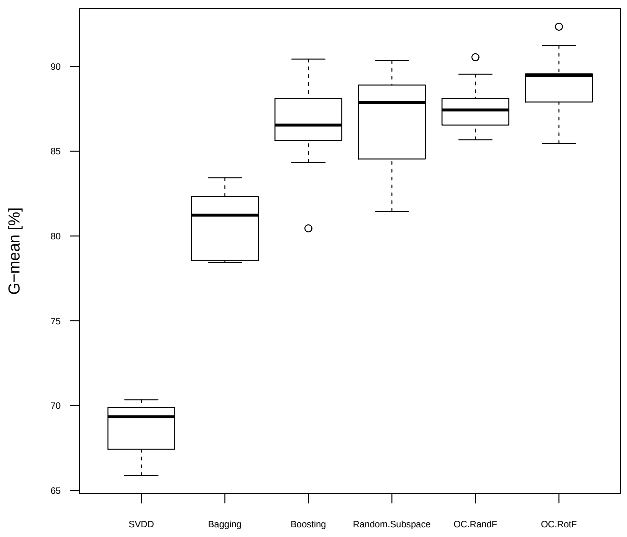


Figure 2.20: The comparison of G-mean values returned by OC-RotF (with PCA) and reference one-class methods for an exemplary dataset (Sonar) over 5x2 CV.

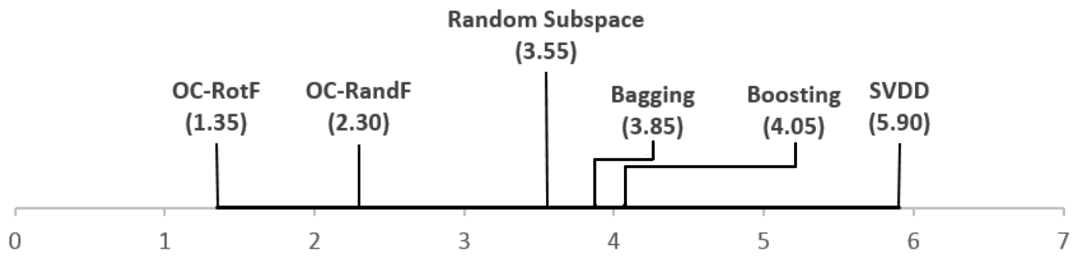


Figure 2.21: The results of Friedman ranking test for OC-RotF and reference methods over 20 benchmark datasets.

multiple datasets for comparison between OC-RandF and reference one-class classification methods in Table 2.6.

One may see that OC-RotF consistently delivers excellent performance when analyzed over a set of benchmarks. OC-RotF approaches the feature space processing in a different approach than Random Subspaces. Instead of selecting at random some features, it transforms the original space into a new one, consisting of linear combinations of original features. When using proper extraction technique (e.g., PCA) we obtain orthogonal spaces, thus assuring a sufficient diversity to the pool of classifiers. As they

Table 2.6: Shaffer test for comparison between OC-RotF and reference one-class classification methods. The test is carried over 20 benchmark datasets. Symbol '+' stands for a situation in which the proposed method is superior, '-' for vice versa and '=' represents a lack of statistically significant differences.

hypothesis	p-value
OC-RotF vs SVDD	+ (0.0041)
OC-RotF vs Bagging	+ (0.0274)
OC-RotF vs Random Subspace	+ (0.0297)
OC-RotF vs Boosting	+ (0.0173)
OC-RotF vs OC-RandF	+ (0.0411)

are trained on such artificial spaces, we may obtain more compact base learners, as PCA may output a sparser representation. Additionally, we should note that OC-RotF may also be trained on smaller dimensions - one just needs to select a desired number of coefficients in order to train less-dimensional classifiers. In this thesis we use all of obtained coefficients.

Finally, it is interesting to notice that OC-RotF outperforms OC-RandF. It can be explained by two main factors: OC-RotF preserves the useful properties of multi-class Rotation Forest (that was shown to be superior to Random Forest for many datasets) and OC-RotF does not require counterexamples in the training procedure. OC-RandF creates artificial outliers, thus changing the one-class problem into a binary one. Additionally it is restricted only to decision trees. OC-RotF may work with any kind of one-class classifiers (density, reconstruction or boundary-based), thus preserving the data description properties that should be associated with OCC.

2.4 Ensemble of Soft One-Class Classifiers based on Weighted Bagging

The fourth proposed algorithm is Wagging Ensemble of Soft One-Class Classifiers (OC-Wagg). Wagging is also known as weighted Bagging. It does not randomly select examples for each bag, but instead draw weights assigned to each observation according to a given probability distribution. We propose to use these weights to directly train a pool of WOCSVMs. Each wagging iteration constructs new weighted one-class classifier, while weights assigned to each object in a given iteration are utilized in the classifier's training process in order to assign a degree of importance to each training sample. With this, we can easily create a hybridization between wagging and weighted one-class learning. Such an approach improves the diversity of the ensemble, as different weights assigned to objects lead to different decision boundaries computed by classifiers.

The main contributions related to this method are as follows:

- A hybridization of Wagging and soft one-class classifiers.
- Efficient scheme for fast computation of weights assigned to each object from the training set that significantly reduces the time complexity of weight estimation for soft one-class classifiers.
- Alleviating drawbacks of Bagging, as Wagging does not discard the spatial relations between objects.

2.4.1 Details of the Algorithm

None of standard ensembles used in OCC can directly benefit from the usage of WOCSVM [20]. WOCSVMs are much more efficient and robust to internal noise than standard one-class methods, but require dedicated ensemble forming algorithms to being combined in an efficient way [173]. This lead us to proposing a novel ensemble of weighted one-class classifiers, based on Wagging.

Wagging [18] is a variant of Bagging method. It is also known as Weighted Bagging. Here each base classifier is trained on the entire training set, but each objects is randomly assigned a weight.

Bagging can be considered as Wagging with weights drawn from the Poisson distribution, as each instance is represented in the bag a discrete number of times. On the other hand, Wagging often uses an exponential distribution to drawn weights. This is because the exponential distribution is the real-value counterpart of the Poisson distribution.

Wagging offers an interesting way to modify the level of influence of each sample on the classifier's training process by differentiating weights. However, Wagging cannot be directly applied in most of the OCC methods, as they consider each object from the target class to be equally important during the training step. We propose to combine Wagging with WOCSVM classifier and input the drawn weights directly into the WOCSVM training phase (see Eq. 1.44).

The pseudo-code for proposed Wagging ensemble for one-class classification is presented in Algorithm 4.

Algorithm 4 Wagging for forming ensembles of Weighted One-Class Support Vector Machines.

```
Require: WOCSVM training procedure,
    number of classifiers $L$ ,
    training set $\mathcal{TS}$,
    exponential distribution $\lambda$
1: $l \leftarrow 1$
2: repeat
3:     $\mathcal{TS}_i \leftarrow \mathcal{TS}$ with random weights drawn from $\lambda$
4:     Train $l$-th WOCSVM on $\mathcal{TS}_i$ according to weights assigned to each object
5:     $l \leftarrow l + 1$
6: until $l > L$
```

After forming such an ensemble, one should map distance to probability (as WOCSVMs are distance-based, see Eq. (1.25)) and apply one of the fusion techniques from Section 1.3.4 (see Eq. (1.59) – (1.63)).

The main advantages of the proposed approach are a significant reduction of the training complexity (weights are given directly by Wagging, instead of calculating them individually, e.g, according to Eq. (1.46) and increase of the diversity of ensemble members. Additionally, WOCSVM training scheme outputs a locally competent classifier, resulting in a committee of diverse and accurate base learners.

2.4.2 Experimental Study and Discussion

In this section, we carry an experimental analysis of the proposed method. We analyze its performance over a set of 20 benchmarks, presented in Appendix A in Table A.1.

We follow an experimental framework described in Appendix B.

We aim at establishing the quality of OC-Wagg ensemble in case of learning in the absence of counterexamples.

In Appendix C the results for each of 20 datasets are presented. In the main body of the thesis, we present only results for an exemplary datasets (to give insight in details of the method) and outcomes of statistical analysis over multiple datasets (Friedman and Shaffer post-hoc), to give an outlook on the global performance of this method (for details see Appendix B).

We compare our method with a single-classifier approach and other ensemble-based approaches for one-class classification based on space partitioning, namely Bagging, Random Subspace and simple clustering of the target class without any weighting module. Additionally, we add Boosting to this experimental framework, as it is considered as one of the best solutions in this field. We use a WOCSVM classifier as a base method for each of these algorithms, in order to provide a fair comparison between them. The number of base classifiers used by each ensemble was optimized for each dataset independently from the range [5;50].

The comparison of G-mean results for an exemplary dataset (Delft Pump 2x2 noisy) over 5x2 CV for examined one-class ensemble methods is presented in Figure 2.22.

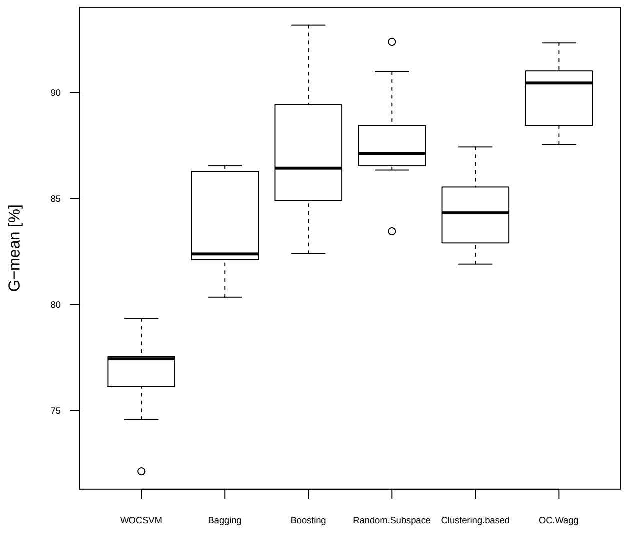


Figure 2.22: The comparison of G-mean values returned by OC-Wagg and reference one-class methods for an exemplary dataset (Delft Pump 2x2 noisy) over 5x2 CV.

One may see that OC-Wagg can outperform all of the reference one-class methods for Delft Pump 2x2 noisy dataset. This can be related to a relatively small size of the training dataset. For such cases Bagging and Clustering-Based approaches tend to create too small partitions of data and thus create overfitted classifiers. Boosting-based

Table 2.7: Shaffer test for comparison between OC-Wagg and reference one-class classification methods. The test is carried over 20 benchmark datasets. Symbol '+' stands for a situation in which the proposed method is superior, '-' for vice versa and '=' represents a lack of statistically significant differences.

hypothesis	p-value
OC-Wagg vs WOCSVM	+ (0.0053)
OC-Wagg vs Bagging	+ (0.0258)
OC-Wagg vs Random Subspace	+ (0.0227)
OC-Wagg vs Boosting	+ (0.0336)
OC-Wagg vs Clustering-based	+ (0.0194)

method may also suffer from the lack if sufficient training set, as classifiers trained in the following iterations will quickly start to process the same examples over again.

To be able to give a more global conclusions, we need to check the performance of OC-Wagg over a wider set of one-class benchmarks. We present the results of Friedman ranking test, in order to check the quality of the methods over multiple datasets. The results of this ranking test are given in Figure 2.23.

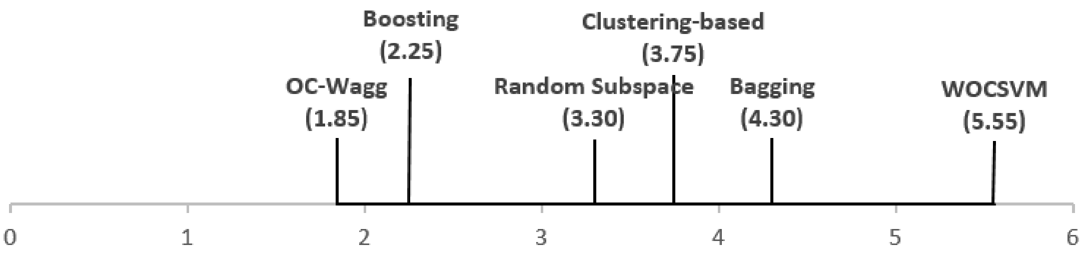


Figure 2.23: The results of Friedman ranking test for OC-Wagg and reference methods over 20 benchmark datasets.

To provide an additional verification of this claim, we present the results of post-hoc Shaffer test over multiple datasets for comparison between OC-Wagg and reference one-class classification methods in Table 2.7.

From the experimental results we may see that Wagging-based ensemble proves highly competitive to other methods. In most of the cases the obtained accuracy was highest and the differences were statistically significant.

Only in few cases Bagging and Boosting-based ensembles of WOCSVM outperformed Wagging. This proves that Wagging is a worthwhile choice for combining weighted one-class classifiers, as changing weights in each classifier has a more positive influence on the quality of the committee than using permutation of objects or simple boosting over a single class. This increases the diversity and extends the competence of the ensemble, making it more robust to potential outliers.

Interestingly, Wagging performs at best for small datasets (such as Delft Pump, Hepatitis or Sonar), significantly outperforming standard Bagging. This can be explained by a limited availability of samples for training. In small datasets, excluding a part of objects from the training procedure can significantly damper the quality of classifier (as it cannot capture the decision space properties properly). At the same time it is easy to create similar bags of objects over a small dataset. Wagging lifts those limitations, as it uses a full training set. It enforces diversity by manipulating weights assigned to objects, not objects themselves.

2.5 Comments and Recommendations for the Proposed One-Class Ensemble Forming Methods

In the previous four sections of this thesis we have introduced four novel algorithms dedicated to forming one-class classification ensembles. Each of them was based on different principles and possessed unique properties. It is worth to notice that introduced methods were statistically significantly superior to state-of-the-art one-class committees over a set of diverse benchmarks. It is important to establish under what circumstances should the end-user select specific ensemble from the proposed ones.

This section aims at emphasizing the strong points of each methods, as well as identifying their drawbacks and possible shortcomings. Such a discussion will be beneficial to the user, by offering a critical analysis of the introduced ensembles and identifying the potential areas of applicability of each method.

We carried out comparative tests on 20 benchmarks to asses the differences between the proposed ensemble forming methods. WOCSVM was used as a base classifier. OCClustE and OC-EvoClust automatically selected the number of classifiers, while OC-RotF and OC-Wagg were tuned independently for each dataset with the number of base models in range og [5;50]. Detailed results are given in Appendix C.

Here, we present results for four selected databases, in order to emphasize the advantages and drawbacks of the proposed classifiers. The comparison of G-mean results for four datasets over 5x2 CV for examined one-class ensemble methods is presented in Figure 3.10.

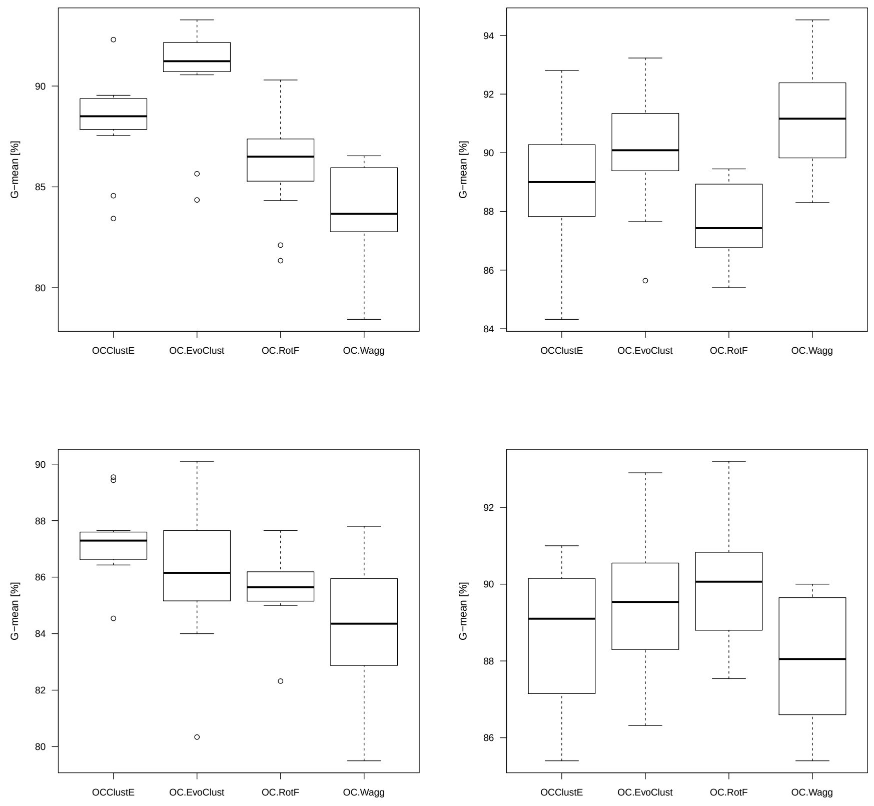


Figure 2.24: The comparison of G-mean values returned by four proposed ensemble classifiers for (*top left*) p53 Mutants, (*top right*) Delft Pump 2x2 noisy, (*bottom left*) Pima and (*bottom right*) Voting Records datasets over 5x2 CV.

Let us firstly concentrate on the useful properties offered by each of the algorithms.

- OCClustE can efficiently detect the areas of competence (or sub-concepts) hidden in the target class. By using proposed statistical indexes, it automatically select the most promising number of competence areas to train classifier on them. Thus, OCClustE automatically establishes the number of classifiers in the committee, removing this free parameter from the classifier tuning procedure. This is a significant ease for the end-user. The detected areas of competence allow for training locally specialized and compact classifiers, thus reducing the risk of overfitting or outputting too large data description volume. Additionally, OCClustE offers a fast and efficient method for establishing weights assigned to each object for soft one-class classifiers (such as WOCSVM). OCClustE requires minimal number of parameters to be set by the user and in case when the user follows our recommendations from Section 2.1.4 there is no need to specify any free parameter for the method. Therefore, it can be seen as a off-the-shelf solution. The overall complexity of the ensemble is reasonably low. The main computational effort lies in establishing the number of competence areas. Then, the training time of each WOCSVM is significantly reduced due to eliminating the computation of weights and training each model on a compact and reduced dataset. Each of such local WOCSVM is characterized by a reduced number of support vectors. Finally, OCClustE can be efficiently used for multi-class problem decomposition and has a highly parallel structure that can be straightforwardly applied in a distributed computing environment.

- OC-EvoClust is an efficient hybrid classifier, using a compound evolutionary training procedure. It is based on detecting the best local areas of competence for base classifiers. This method creates a feedback between the area detection and classifier training, allowing for an evolutionary adaptation of the positions and shapes of clusters to exploit the data characteristics. This is especially useful in case of datasets with difficult and complex distributions that require highly atypical shapes of decision boundaries. There is no need to specify the number of competence areas as the training procedure will automatically establish the set of prototypes. This is not done via model selection (computing a set of possible clustering schemes and selecting the best one like in OCClustE), but by evolutionary addition of new competence area whenever there is a chance that it will contribute to the performance of the ensemble. This is a much more flexible solution. OC-EvoClust can deal with massive datasets by training classifiers on significantly reduced subsets. Additionally, for each detected competence region an area ensemble is being trained. It consists of classifiers using different subsets of features that are selected via OC-EvoClust training procedure. This allows to efficiently process high dimensional datasets. OC-EvoClust offers a trained fusion mechanism that so far has not been applied to one-class problems. The training algorithm can calculate such weights assigned to each base learner that will allow to exploit its competencies. OC-EvoClust uses a combined criterion function (consistency and diversity) to create robust and mutually complementary base classifiers. This ensemble imposes no limitations on the type of one-class classifier that can be used as a base learner.

- OC-RotF allows to create diverse ensembles on the basis of artificially created feature spaces. It takes advantage of unsupervised feature extraction methods to prepare a set of rotation matrices that are used to train base classifiers. Such rotated spaces allow to exploit different properties of data and create different projections of the same dataset. OC-RotF combines the advantages of Random Subspaces and Bagging, by working on a reduced subset of features and objects in each rotation matrix. Contrary to its counterpart One-Class Random Forest it does not impose a requirement of generating artificial examples. Thus, OC-RotF can be viewed as a more robust method, as it does not make any assumptions about the nature of outliers. It can work with any type of one-class classifier. The ensembles outputted are usually much smaller than in case of Random Subspaces, Bagging or One-Class Random Forests. Finally, the training time of OC-RotF is very small in comparison with other methods.

- OC-Wagg was designed to alleviate the disadvantage of Bagging algorithm that does not preserve the spatial relations between data in each bag. OC-Wagg creates subsets of randomly drawn weights assigned to objects. This allows to train a pool of soft one-class classifiers (as WOCSVMs) using such information instead of computing the weights individually for each classifier. It also ensures diversity among the committee members. OC-Wagg is highly efficient for smaller dataset, where creating sub-samples of objects would lead to a very low number of training objects assigned to each base learner. One-class methods are prone to this problem and it may significantly influence its robustness to outliers. OC-Wagg does not influence the number of training objects, but the weights assigned to them thus being suitable for cases where target class representatives are not abundant.

However, every classification method is somewhat biased. Desirable properties are often introduced at the cost of some drawbacks in other areas. To allow for a fair evaluation of the proposed one-class ensemble system, we will now present a critical discussion of their potential shortcomings.

- OCClustE can suffer from the lack of interconnection between the competence area detection step and classifier training step. For difficult, complex distributions it is possible that local areas will have incorrect structure, mixing objects from different sub-concepts. There is no way to improve the partitioning once the clustering procedure is conducted. OCClustE detects the number of competence areas with the usage of model selection indexes. Such methods are very attractive due to their automatic mode of work, but cannot guarantee finding the best possible number of clusters. Usually, they are just indicators of favourable properties of a given model. So manual tuning could theoretically further improve the quality of the ensemble, but would deprive OCClustE from its automatic operation mode. Finally, OCClustE imposes no restrictions on its components, but (due to its local characteristics) favours fuzzy space partitioning algorithms and WOCSVMs.
- The main advantage of OC-EvoClust (hybrid evolutionary training procedure) can be at the same time its possible weak point. One can see that the proposed optimization problem is compound and relies on a very large chromosome. Thus, it is reasonable to assume the presence of many local minima that could stop the search for the best ensemble structure. The parameters of evolutionary procedure have a significant impact on the quality of the outputted ensemble - thus an end-user with little experience in evolutionary computation may easily select unfavorable parameter settings. Finally, the hybrid training procedure of OC-EvoClust has a high computational complexity, originating from the compound evolutionary process and training ensembles independently in each iteration. This may become prohibitive in multi-class scenarios, as one would need to train an independent OC-EvoClust ensemble for each class.
- Different base classifiers may display different behaviour when being trained on rotated feature spaces outputted by OC-RotF. Rotation Forest algorithm was originally designed for decision trees that corresponded well with axis rotations. One-class classifiers may react differently, especially in case of distance-based methods

that require proper feature scaling. Additionally, there is no clear indicator on how large the pool of classifiers should be, so the end-user need to conduct a time-consuming grid search tuning procedure.

- OC-Wagg can deliver reduced recognition accuracy over large datasets. Here, space partitioning is not a problem due to the abundance of training examples. In large datasets there is an increased chance of occurrence of noisy objects (so-called internal outliers). OC-Wagg will assign them random weights that may boost their importance for a given base classifier. OC-Wagg may assign a high degree of importance to noisy objects in bigger data collections. Finally, OC-Wagg is restricted to only soft one-class classifiers, i.e., ones that can directly process the weights assigned to each object.

Taking under consideration both strong points and shortcomings of each of the presented one-class ensemble forming algorithms, we may propose a set of suggestions on the areas of applicability of each algorithm:

- OCClustE is especially useful for big data sets, where there is a high chance of sub-concept presence. It can efficiently deal with both one-class and multi-class problems. It is a powerful classifier for general-purpose machine learning and is an off-the-shelf ensemble system. It is an attractive solution for scenarios in which we require our ensemble to be trained in reduced computational time, but not in near-real time. Should be avoided for very small datasets, as partitioning usually would not contribute to the quality of the ensemble.

- OC-EvoClust in a hybrid and compound classifier, characterized by an excellent recognition accuracy and flexible adaptation properties. It can be efficiently used in big data analytics, to deal with massive or multi-dimensional datasets. OC-EvoClust will create the most accurate ensembles for very complex datasets by evolving its structure. This framework can be adapted to work with different base classifiers. However, users should be aware of its high computational cost. Should be avoided for very small datasets, as partitioning usually would not contribute to the quality of the ensemble.

- OC-RotF is an attractive solution for standard one-class problems. It can deal with reasonably large datasets, but in case of big data analytics the feature extraction process may have a prohibitive complexity. It outputs a very competent pool of learners, requiring only the tuning of the number of classifiers in the committee. The computational complexity of this method is allows to use it for most of the problems.
- OC-Wagg is an attractive proposal for smaller datasets, being more robust to overfitting in such scenarios. It is a straightforward method that requires only tuning the number of base classifiers in the committee. It has the lowest computational complexity from all four introduced ensemble methods.

3

Pruning One-Class Ensembles

In this chapter, we will discuss the pruning methods dedicated to OCC. Due to the lack of counterexamples during the training process, one cannot apply canonical classifier selection and ensemble pruning methods. Therefore there is a need to introduce dedicated methods, tailored to the specific nature of OCC.

We introduce diversity measures for one-class ensembles, a concept so far has not been discussed in the literature.Three measures are proposed that work under different principles of evaluating one-class classifies. We show that such measures can be efficiently applied to the process of one-class ensemble pruning.

We propose a pruning mechanism based on optimization approach. It applies evolutionary algorithms to select a subset of classifiers from the pool. We investigate several different optimization approach in order to select the most robust one. We suggest to use a multi-criteria optimization that utilizes both the diversity and consistency of the ensemble.

Next, we introduce clustering-based one-class ensemble pruning. We aim at finding groups of similar classifiers and reduce the pool by selecting the representative from each of these groups. The proposed method automatically establishes the optimal number of groups. Additionally, it does not require any specific measure of performance as it uses directly the support values outputted by classifiers for each object.

Finally, we present a hybrid nature-inspired one-class ensemble pruning with firefly algorithm. It is a combination between the swarm intelligence optimization and clustering. We propose a modification of firefly method that embeds both diversity and consistency measures, thus indirectly realizing multi-criteria optimization. This method is augmented with a module for fast and efficient weight calculation for each selected classifier for the combination process.

3.1 Diversity Measures for One-Class Classifier Ensembles

Usually we have more than one classifier for a given task at our disposal. Yet they display different properties, have different areas of competence and may prove different contribution to the committee. Therefore a careful classifier selection must be conducted in order to chose the most valuable individual models. There are several ways on how to perform this operation and what should be the factor affecting the model selection. One of the most popular criteria is the ensemble diversity, which aims at choosing predictors that are as different from each other as possible. This is motivated by the fact that adding similar classifiers to the committee does not improve its quality - only increases its complexity. One chooses diverse models in hope that they will be mutually supplementary and allow to exploit different areas of competence given by each of them. So far the concept of diversity has not been used in the context of OCC. One of the aims of this thesis is to propose the carefully designed diversity measures in order to design more accurate and robust OCC ensembles. Canonical measures of diversity for multi-class problems cannot be used, as they are usually based on difference in outputs of classifiers and measuring their correct decision for each class. As we are dealing with a single-class problem, we do not have enough information to compute these measures.

Let us analyze the idea of diversity in OCC. As we deal only with objects from one class we cannot simply measure it as a difference in decisions of base classifiers. Here diversity could be understood as a specialization in different decision regions or alternative approaches to data structure description. One cannot mistake diversity with different volumes of the data description. We are interested in obtaining classifiers with highest possible generalization and discrimination abilities. Diversity measure for OCC should be designed to promote different classifiers' specializations, not simply varying volumes of decision boundaries. An example of good and bad diversity in OCC is presented in Figure 3.1.

In this section three diversity measures dedicated to one-class classifier ensembles are proposed [166]. All of them rank the diversity of a given pool of classifiers [164]. Such types of measures are more useful for one-class classifier ensembles, as a classifier that did not received a good score in *pairwise* test may still contribute to the overall ensemble. Therefore $non-pairwise$ measures return more global outlook on the ensemble performance. All presented diversity measures may take values from the interval

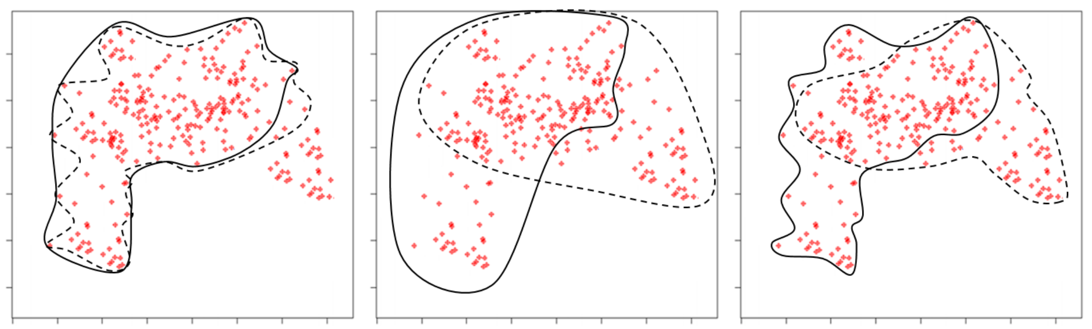


Figure 3.1: Three examples of combining one-class classifiers: (*left*) one-class classifiers with well-computed boundaries but similar competence areas; (*centre*) classifiers with different competence areas but incompact boundaries; (*right*) mutually complementary one-class classifiers. Second and third example show bad and good diversity in OCC.

$[0, 1]$, where 0 corresponds to identical ensemble and 1 to the highest possible diversity respectively.

The main contribution related to this method is as follows:

- Introduction of the concept of diversity measures applied to the one-class ensembles [166] and identification of specific constraints that are connected with them.
- Proposal of three efficient diversity measures that are tailored for the nature of OCC problems.

3.1.1 One-Class Shanon Measure

We introduce a one-class modification of Shanon diversity measure [100, 164]. Let us assume that G out of L classifiers can correctly classify a given training object x_i to ω_T.

Therefore one may propose a membership function $\mu_{x_i} = (\frac{G}{L})$ for a given object, where $0 \leq (\frac{G}{L}) \leq 1$. This is given to Shanon function, which acts as a diversity measure:

$$DIV_S(\Pi) = \sum_{i=1}^{N} \{-\mu_{x_i} \log_2(\mu_{x_i}) - (1 - \mu_{x_i}) \log_2(1 - \mu_{x_i})\}. \qquad (3.1)$$

3.1.2 One-Class Energy Measure

Energy approach is an effective measure of fuzziness, successfully implemented in many practical applications such as ECG analysis [60]. It uses a threshold $\zeta \in [0, 1]$. Its role is to filter insignificant degrees of membership that may otherwise contribute to decreasing the stability of the proposed measure. The energy measure is described as follows:

$$DIV_{EN}(\Pi) = \int_X \sum_{l=1}^{L} f_\zeta(x)dx, \tag{3.2}$$

where

$$f_\zeta(x) = \frac{\sum_{l=1}^{L} \delta(\Psi_l(x), \Psi^*(x))}{L} > \zeta, \tag{3.3}$$

and $\Psi^*(x)$ denotes a classifier correctly classifying the object x and $f(x) : X \to [0, 1]$.

If for a given set of classifiers condition form Eq. (3.3) is not meet (the function returns lower value than threshold ζ) then we automatically discard this subset of classifiers. This speeds-up the computation of the diversity measure, especially for a large number of possible classifier combinations.

3.1.3 Sphere Intersection Measure

Intuitively a high diversity of an ensemble may be achieved when each of the classifiers have a different area of competence. From this one may easily see that two classifiers with similar areas of competence will not contribute much to the quality of the committee. Taking into consideration the specific nature of boundary one-class classifiers we may assume that two predictors with high overlap of decision boundaries may be deemed as ones with a low diversity. We propose a diversity measure designed specifically for spherical one-class classifiers (such as considered in this thesis OCSVM, WOCSVM, and SVDD) based on a degree of overlap among individual classifiers [136]. In case of classifier overlapping there may be two situations - where two or more classifiers overlap. Examples of such situations are presented in Fig. 3.2.

We propose to measure the diversity of the ensemble by measuring the overall degree of overlap between all classifiers in the pool. Firstly, we need to calculate the volume of a single spherical classifier:

$$V_{\mathbb{S}}(a, R) = \frac{2\pi^{\frac{d}{2}} R^d}{\Gamma(d/2 + 1)}, \tag{3.4}$$

where a is the center of the sphere, R is the radius the sphere and Γ is the gamma function.

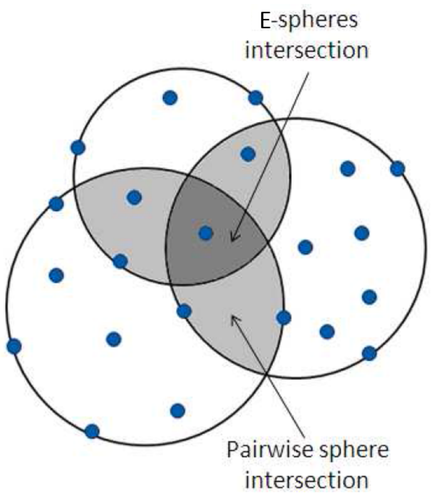


Figure 3.2: Example of possible sphere intersections - two spheres overlapping (pairwise intersection) and more than two spheres overlapping (E-spheres intersection).

Therefore the volume of all L classifiers from the pool is equal to the sum of their individual volumes:

$$V_{sum}(\Pi) = \textstyle\sum_{l=1}^{L} V_{\mathcal{S}}(a_l, R_l). \tag{3.5}$$

In case of a lack of overlap between the spheres in the committee the volume of an ensemble is equal to the sum of the volumes of all individual classifiers:

$$V_{ens}(\Pi) = V_{sum}(\Pi) \Leftrightarrow \forall_{l \neq j \quad l,j=1,\ldots,L} \quad V_{\mathcal{S}}(a_l, R_l) \cap V_{\mathcal{S}}(a_j, R_j) = \emptyset. \tag{3.6}$$

We assume that maximum diversity is achieved for the situation presented in Eq. (3.6), i.e., when no overlap between individual classifiers exist.

In case of E pairwise overlaps the volume of an ensemble is given by the following formulae:

$$V_{ens}(\Pi) = \sum_{l=1}^{L} V_{\mathcal{S}}(a_l, R_l) - \sum_{e=1}^{E} V_{\mathcal{O}_e}(a_l, a_j, R_l, R_j) \Leftrightarrow \exists_{l \neq j \quad l,j=1,\ldots,L} \quad V_{\mathcal{S}}(a_l, R_l) \cap V_{\mathcal{S}}(a_j, R_j) \neq \emptyset. \tag{3.7}$$

where $V_{\mathcal{O}_e}$ is the volume of e-th overlap.

The spherical cap [112] is a part of a hypersphere defined by its height $h_c \in [0, 2R]$ and its radius $r_c \in [0, 2R]$. In the [136] a following equation for d dimensional spherical cap has been proposed:

$$V_{cap}(R, h_c, d) = \frac{\pi^{(d-1)/2} R^{d-1}}{\Gamma((d-1)/2+1)} \int\limits_{0}^{\beta_{max}(R,h_c)} sin^{d-1}(\beta) d\beta, \tag{3.8}$$

where

$$\beta_{max}(R, h_c) = arcsin(\sqrt{(2R-h_c)(h_c/R^2)}). \tag{3.9}$$

From this we may write the volume of a single pairwise overlap $V_{\mathcal{O}_e}$ as a sum of their spherical caps:

$$V_{\mathcal{O}_e}(a_1, a_2, R_1, R_2) = V_{cap1}(R_1, h_{c1}) + V_{cap2}(R_2, h_{c2}). \tag{3.10}$$

In case when more than two classifiers overlap the computation of the volume of the overlap becomes more complex. One may simplify this problem by counting only pairwise overlaps, but this would lead to counting some parts of the overlapping region E times for E intersecting spherical classifiers.

The volume of non-pairwise ovelap can be bounded between the volume of classifiers with only pairwise overlaps and the volume of classifiers with no overlapping:

$$\sum_{l=1}^{L} V_{\mathcal{S}}(a_l, R_l) - \sum_{e=1}^{E} V_{\mathcal{O}_e}(\Pi) \leq V_{ens}(\Pi) \leq \sum_{l=1}^{L} V_{\mathcal{S}}(a_l, R_l). \tag{3.11}$$

With this, we have derived upper and lower bounds of the volume of the ensemble with non-pairwise overlapping:

$$V_{ens}(\Pi) \approx \sum_{l=1}^{L} V_{\mathcal{S}}(a_l, R_l) - \frac{1}{2} \sum_{e=1}^{E} V_{\mathcal{O}_e}(\Pi). \tag{3.12}$$

The maximum error of such an approximation is no greater than $\frac{1}{2}\sum_{e=1}^{E} V_{\mathcal{O}_e}(\Pi)$.

This is of course a naive approximation assuming that the volume of overlap is an average of lower and upper bounds. One cannot use neither of these bounds (as one states for complete lack of overlapping and second for a complete overlap), therefore this approximation seems as a reasonable solution.

With presented above equations we propose to measure the diversity by comparing the volume of the ensemble to the sum of all individual classifiers, assuming that with increase of the intersection degree the diversity falls down:

$$DIV_{SI}(\Pi) = \frac{V_{ens}(\Pi)}{V_{sum}(\Pi)}. \tag{3.13}$$

3.2 Optimization-Based One-Class Ensemble Pruning

Having a pool of classifiers one needs to select a given subset of valuable models in order to form an efficient ensemble. As we require to find a combination of learners that will maximize the quality of the committee, one may see this as a search problem. Using an exhaustive search is often impossible, due to its enormous computational cost (especially in case of large pool of classifiers to examine). Therefore, one needs to apply more efficient algorithms. This can be achieved by solving an optimization problem, stated as finding the best subset of classifiers from the pool. One should note that the optimization algorithm being used may have a crucial impact on the pruning process [170].

We propose a weighted multi-criteria classifier selection model that will take into account the diversity and individual quality of the considered classifiers. We examine several different popular optimization algorithms. We check the behavior of genetic algorithm, simulated annealing, tabu search and hybrid methods, combining the mentioned approaches in the form of memetic algorithms.

Using optimization-based ensemble pruning has the following desirable properties:

- We may take advantage of efficient methods already proposed for solving standard optimization tasks. They offer efficient search engines that are robust to local minima and can efficiently explore the potential search space.
- An optimization criterion can be adjusted by the user for the specific problem.
- Such methods allow to efficiently process large pool of classifiers in a reduced time.

We will present the details of the proposed optimization-based pruning schemes in the following subsections.

3.2.1 Pruning with Multi-Criteria Optimization

During the ensemble pruning process, one should take two main criteria under consideration [160]:

- accuracy - adding weak classifiers with a low competence will decrease the overall quality of the MCS;

- diversity - adding similar classifiers will contribute almost nothing to the ensemble apart from increasing the computational cost.

Each of these criteria has its drawbacks - highly accurate classifiers may not be diverse, while diverse classifiers may not be accurate on their own. Both of these criteria are popular in ensemble pruning for multi-class cases. Yet for the specific problem of OCC there is still little work done so far on how to efficiently evaluate the given pool of classifiers. In [49] it is suggested to prune the ensemble according to the individual performance of classifiers, while our works explored the usage of diversity measures for this task [147, 161].

We propose to select classifiers to the OCC committee according to the combination of both of these criteria, hoping that this will allow to combine their strong points while becoming more robust to flaws exhibited by each of them. With this, we are able to embed both individual quality and mutual complementarity of ensemble members into the pruning scheme [162, 165].

Let us formulate the multi-objective optimization criterion as:

$$Q(\Pi) = \beta \mathrm{CONS}(\Pi) + (1 - \beta)\mathrm{DIV}(\Pi). \tag{3.14}$$

where CONS(Π) is the consistency measure of the given l-th pool of classifiers (presented in Eq.(1.33)), DIV(Π) is any diversity measure suitable for one-class ensembles (discussed in Section 3.1) and β is the weighting factor that allows us to control the importance of each factor in the proposed multi-criteria optimization procedure. With this parameter user may declare if consistency or diversity should have a higher priority when forming an ensemble. In case of $\beta = 0.5$ the formed ensemble will assign an equal importance to both of these factors (being an equivalent to a non-weighted optimization criterion).

A subset of classifiers which maximizes the proposed criterion should be selected, as we are looking for an ensemble with the highest consistency that displays the best diversity at the same time:

$$\Pi = \Pi^* \Leftrightarrow Q(\Pi^*) = \max_{\pi \subseteq \Pi} Q(\pi). \tag{3.15}$$

Please note that such a criterion can be easily inputted into any kind of optimization procedure.

3.2.2 Encoding the One-Class Ensemble

In order to conduct a search over possible subsets of classifiers, one needs to have a common representation of classifiers in the decision space. This will allow optimization algorithms to transit from one point of search space to another, aiming at maximization of the proposed criterion.

The selected pool can be encoded as a constituent **B** in a form of a binary mask:

$$\mathbf{B} = [B_1, B_2, ..., B_L], \tag{3.16}$$

with 1s indicating the chosen individual classifiers (i.e., if we have 10 classifiers then 0010110010 would indicate that classifiers 3, 5, 6 and 9 are chosen for the ensemble).

3.2.3 Overview of the Used Optimization Methods

In this section, we present a short overview of the used optimization algorithms for one-class ensemble pruning task.

Apart from individual setting, each of the examined algorithms uses the following parameters:

- N_c - the upper limit of algorithm cycles,
- O - the upper limit of algorithm iterations without quality improvement.

We have examined the following five optimization procedures:

- *Genetic Algorithm* (GA) - a popular nature-inspired search heuristic. It is based on a population of candidates, which is evolved toward better solutions. Each candidate solution has a set of properties which can be mutated and altered. Previous works showed that GA are effective for OCC classifier selection [161]. The control parameters of the GA algorithm are as follows:
  - N_p - the population quantity,
  - η - the mutation probability,
  - γ - the crossover probability,
  - Δ_m - the mutation range factor,

- *Simulated Annealing* (SA) - a probabilistic metaheuristic for the global optimization. At each step it considers neighboring state of the current state and probabilistically decides between moving the system from state to state or remaining in the present location. These probabilities ultimately lead the system to move to states of lower energy. The control parameters of the SA algorithm are as follows:
  - T - the temperature
  - ϱ - the cooling factor
- *Tabu Search* (TS) - uses a local or neighborhood search procedure to iteratively move towards an improved solution. It uses memory structures form, known as the tabu list. It is a set of rules and banned solutions used to filter which solutions will be admitted to the neighborhood to be explored by the search. The control parameters of the TS algorithm are as follows:
  - ς - the tabu list size
- *Memetic Algorithm* (MA) - be seen as a hybrid solution that fuses together different metaheuristics in hope to use gain advantage from combining their strong points [111]. The idea of MAs is based on the individual improvement plus population cooperation. Unlike traditional Evolutionary Algorithms (EA), MAs are biased towards exploiting all the knowledge about the problem under study. By this they may be seen as less random and more directed search method. In this work we examine two types of memetic algorithms:
  - using GA for exploration and SA for exploitation,
  - using GA for exploration and TS for exploitation.

3.2.4 Experimental Study and Discussion

In this section, we carry an experimental analysis of the proposed methods. We analyze their performance over a set of 20 benchmarks, presented in Appendix A in Table A.1. We follow an experimental framework described in Appendix B. Detailed parameter setting and results of experiments are given in Appendix C.

The aims of the experiment were as follows:

- To check the level of importance of selecting a proper search algorithm to prune the one-class classifier ensembles - to how much the choice of the optimization approach determines the quality of the ensemble.
- To examine the effectiveness of the proposed multi-criteria approach and the influence of the weighting parameter β on the pruning procedure.
- To see if using a hybrid optimization (memetic algorithms) leads to a significantly better performance in comparison with standard solutions.

3.2.4.1 Assessing the Weights of Components within Multi-Criteria Optimization

The proposed multi-criteria optimization procedure can be directly controlled by the weighting factor β. It allows to shift a trade-off between the consistency and diversity of ensemble members. This is motivated by some recent works that report a low efficiency of the diversity embedded in the ensemble creation schemes [206, 294]. We wanted to examine this property in case of one-class ensembles.

We have constructed a pool of 100 SVDD classifiers on the basis of Bagging algorithm. In Appendix C results for each of 20 datasets are presented. In the main body of the thesis, we present only results for an exemplary datasets (to give insight in details of the weight assessment).

The comparison of G-mean results for an exemplary dataset (Pima) over 5x2 CV is presented in Figure 3.3. Here, we use memetic algorithm (GA for exploration and TS for exploitation) as a pruning engine.

One may see some interesting properties of the proposed multi-criteria ensemble pruning. For all of three examined methods setting equal weights to consistency and diversity never returned the best results. This prove that controlling the level of influence of each of these factors on the pruning procedure is worthwhile. One-Class Shannon measure has the lowest impact on the ensemble creation, as the best results are obtained for small weights assigned to this diversity measure. For One-Class Energy measure, we obtain the best results when consistency has larger weight than diversity, but in this case diversity has a higher weight assigned. This confirms the observations from multi-class problem that diversity itself is not sufficient to form an efficient ensemble.

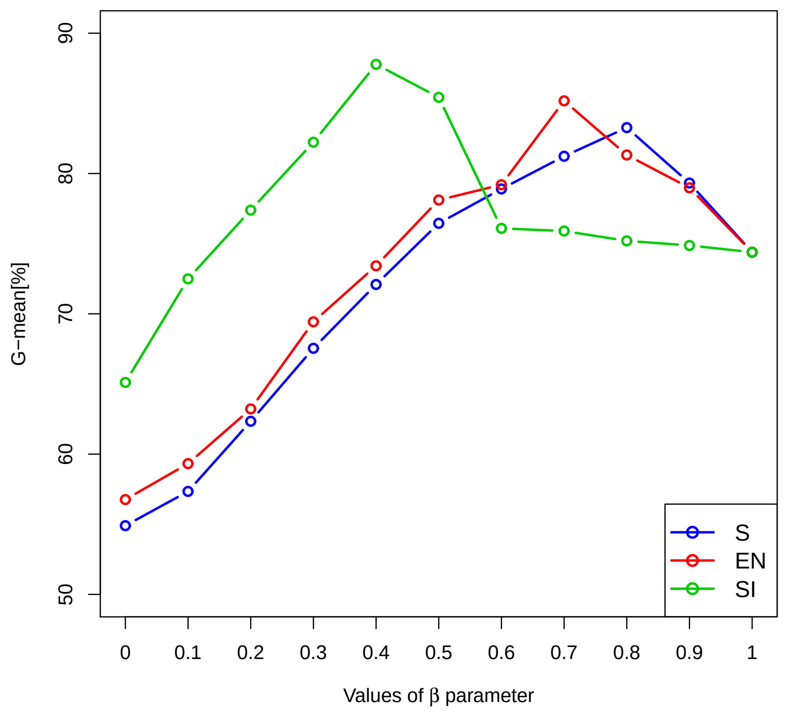


Figure 3.3: The relationship between the weighting parameter β in multi-criteria pruning and obtained G-mean values for all of three introduced diversity measures over 5x2 CV for Pima dataset.

Please note however that we get significantly better ensembles when both consistency and diversity are used (even, if the weight assigned to diversity measure is small).

However, for the Sphere Intersection measure we can observe a contradictory situation. Here, we obtain the highest G-mean when the impact of diversity is more significant than that of consistency. This can be explained by the nature of SI measure - it geometrically controls the intersections between classifiers, thus promoting locally specialized decision boundaries. This can lead on its own to selecting efficient classifiers for the ensemble (as seen from the obtained highest G-mean from all of three measures when only a diversity criterion is used). But when enhanced by the consistency measure (with assigned lower level of importance than SI diversity measure), we are able to obtain the most efficient ensemble for this problem.

This is further confirmed by trends observed over 20 datasets (see Appendix C for details).

3.2.4.2 Comparison of Different Optimization-Based Pruning Schemes

Having selected the best settings of β parameter for each dataset, we can use them to compare different optimization-based one-class pruning schemes. A pool of 100 SVDD classifiers had been constructed on the basis of Bagging algorithm. We compare five pruning engines described in Section 3.2.3. Additionally, we present the results for unpruned pool of classifiers.

In Appendix C results for each of 20 datasets are presented. In the main body of the thesis, we present only results for an exemplary datasets (to give insight in details of the method) and outcomes of statistical analysis over multiple datasets (Friedman ranking and Shaffer post-hoc), to give an outlook on the global performance of this method.

Settings and parameters of used methods are described in Appendix C

The comparison of G-mean results for an exemplary dataset (Pima) over 5x2 CV is presented in Figure 3.4.

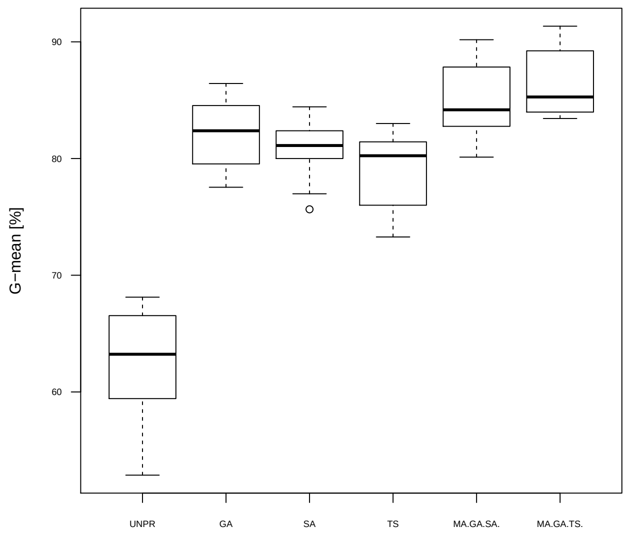


Figure 3.4: G-mean values for five optimization-based one-class pruning schemes and a full, unpruned pool of classifiers.

From the results we can see that the memetic algorithms are superior to other optimization schemes. Memetic methods are hybrid approaches that take advantage of genetic methods (good exploration of the search space) and directed optimization (good exploitation of detected area). A combination of GA and TS is slightly better than a combination of GA and SA. This can be explained by the nature of TS. In memetic algorithms we use them not to probe the entire search space, but to quickly exploit the neighborhood of the promising part of a search space. Tabu algorithm can do this effectively by guiding its performance with the list of banned steps.

To be able to give a more global conclusions, we need to check the performance of the proposed optimization-based pruning schemes over a wider set of one-class benchmarks. We present the results of Friedman ranking test, in order to check the quality of the methods over multiple datasets. The results of this ranking test are given in Figure 3.5.

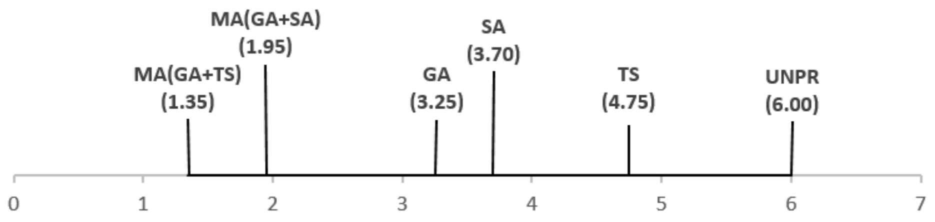


Figure 3.5: The results of Friedman ranking test for examined one-class optimization-based pruning algorithms over 20 benchmark datasets.

The results of the ranking test point out that when considering a wide spectrum of datasets memetic-based pruning methods can outperform remaining ones. We may see that for the considered overproduce-and-select strategy any kind of pruning methods is always better than combining all of available classifiers. *Bagging* or *Random Subspace* methods can often output many incompetent or similar classifiers that can reduce the quality o formed ensemble if nor removed properly. To provide an additional verification of this claim, we present the results of post-hoc Shaffer test over multiple datasets for comparison between memetic-based pruning methods and reference schemes in Table 3.1.

The experimental investigations clearly prove that the selected search method has a crucial influence on the quality of the formed ensemble. Only for a single case all the optimization method returned similar results.

Table 3.1: Shaffer test for comparison between memetic-based pruning schemes and reference pruning methods. The test is carried over 20 benchmark datasets. Symbol '+' stands for a situation in which the proposed method is superior, '-' for vice versa and '=' represents a lack of statistically significant differences.

hypothesis	p-value
MA (GA + TS) vs GA	+ (0.0274)
MA (GA + TS) vs SA	+ (0.0218)
MA (GA + TS) vs TS	+ (0.0157)
MA (GA + TS) vs UNPR	+ (0.0068)
MA (GA + SA) vs GA	+ (0.0305)
MA (GA + SA) vs SA	+ (0.0266)
MA (GA + SA) vs TS	+ (0.0198)
MA (GA + SA) vs UNPR	+ (0.0082)
MA (GA + SA) vs MA (GA + TS)	- (0.0428)

Out of all tested methods, the weakest performance was returned by Tabu Search. This may be explained by its inability to efficiently explore the search space and tendency towards exploiting local neighborhood.

Genetic algorithm and simulated annealing returned similar results, as they both put an emphasis on efficient exploration of the set of states (which is achieved in GA with cross-over and mutation and in SA with rapid decrease of energy).

Interestingly, the simpler approaches were in most cases outperformed by hybrid memetic algorithm. This results prove that population implementation allows for an efficient exploration, while local improvement is more stable than in traditional GA.

Better results were returned by the combination of GA and TS, which is quite interesting considering the unsatisfactory performance of single TS. This can be explained by the fact that GA deals with finding local minima and requires an efficient local search to fully exploit them. In such situations TS can more easily find good solutions, as the search space is reduced and it may effectively use its tabu list.

3.3 Clustering-Based One-Class Ensemble Pruning

The optimization-based pruning methods presented in the Section 3.2 are an efficient search engine for selecting valuable members to one-class committees. However, they have two main drawbacks: high computational complexity and dependency on the proper setting of parameters.

In this section, we propose an alternative approach for pruning OCC ensembles based on clustering [172]. As a search engine, we apply an clustering method that allows to detect a number of groups in data. These groups correspond to the number of classifiers with different competencies. If classifiers are within a single group, one may deem them as similar. This approach allows to prune the ensemble with a very low computational cost. We use an information-based criterion for an automatic selection of a number of clusters. Furthermore, we propose to cluster the support functions outputted by individual classifiers from the pool. With this we search for some emerging patterns in their decisions. This allows us to abstain from using any direct measure of one-class classifier's performance. We show that the proposed scheme is superior to state-of-the art one-class pruning methods.

In the following subsection, we will present the underlying mechanisms behind the clustering-based ensemble pruning for OCC ensembles.

3.3.1 Preliminaries for One-Class Cluster-Based Pruning

To apply the pruning step, we need to have separate training set $\mathcal{TS}$ and validation set $\mathcal{VS}$. The former is used for constructing the pool of classifiers, while the latter is utilized to evaluate the classifiers and conduct the pruning step.

For the clustering algorithm, we need to form a data matrix on which the group discovery will be performed. It is straightforward to apply clustering on datasets, as we look for dependencies between objects and can use their features to measure the distance between samples. In this work, we aim at clustering classifiers. Therefore, we need some kind of classifier's features description in order to form a matrix. One can use several different measures specific for OCC - such as consistency, one-class AUC or diversity. As mentioned before, they are just approximations of the potential performance on unseen data - and may be misleading. We propose to work on support function values outputted by one-class classifiers. We are interested in support of a given classifier Ψ for object x belonging to the target concept ω_T.

Working on direct supports outputted by one-class classifiers offers following benefits:

- We alleviate the problem of selecting of a proper metric for one-class classifier evaluation.
- It can work efficiently for both homogeneous and heterogeneous ensembles.
- Classifiers can form support templates for given objects. Therefore, classifiers outputting similar supports for training samples can be deemed as similar and all but one can be easily discarded. This may be seen as indirect assurance of diversity in the ensemble pool.

According to these assumptions, we propose to form a new data matrix consisting of support functions outputted by available classifiers for objects from $\mathcal{VS}$. We assume that we have L classifiers in the pool. The formed classifier support matrix **CSM** can be defined as follows:

$$\mathbf{CSM} = \begin{pmatrix} F_{\omega_T}^{(1)}(x_1) & F_{\omega_T}^{(1)}(x_2) & \dots & F_{\omega_T}^{(1)}(x_v) \\ F_{\omega_T}^{(2)}(x_1) & F_{\omega_T}^{(2)}(x_2) & \dots & F_{\omega_T}^{(2)}(x_v) \\ \vdots & \vdots & \ddots & \vdots \\ F_{\omega_T}^{(L)}(x_1) & F_{\omega_T}^{(L)}(x_2) & \dots & F_{\omega_T}^{(L)}(x_v) \end{pmatrix}, \tag{3.17}$$

,where $v = |\mathcal{VS}|$.

The proposed method uses the **CSM** for conducting the clustering step. It aims at finding the atomic groups of classifiers formed within the pool of available models. Such groups should have lowest inter-group and highest intra-group variances. With this assumptions, classifiers within a single group should have similar competencies to each other and low diversity. Therefore, using all of them would not contribute to the formed ensemble but only increase its computational complexity. On the other hand, different clusters of classifiers should be formed in complementary competence spaces - thus covering the entire decision space. Selecting a single most representative classifier from each of the established clusters will be similar to choosing classifiers with best individual quality and high diversity. Remaining classifiers in each cluster can be discarded, thus achieving the pruning effect.

The clustering-based approach for pruning has few advantages over the standard methods used in multi-class and one-class problems:

- It requires just a single run of the clustering algorithm, while evolutionary-based schemes need a high number of iterations and schemes based on ranking or performance evaluation require a full-search over the possible combinations of classifiers. This property is especially beneficial for cases, in which the pool of available models is very large.
- It uses a relatively low number of parameters to be tuned and in special cases these parameters can be automatically selected by the proper clustering algorithm - as in the proposed pruning scheme, where the number of clusters is established automatically.
- Evolutionary and ranking methods need a specific metric to be used. This is problematic in OCC due to the lack of counterexamples at the training phase. Several proposed measures, such as consistency or one-class AUC are only approximations of the performance. Diversity measures for one-class classifiers tend to work well, but work under some assumptions about the structure of outlier class. Therefore, methods that are independent from such metrics / assumptions are of a high value to this problem. We show that one do not need such metrics in clustering-based scheme.

3.3.2 X-means Clustering Scheme

From the preliminaries, one may easily see that the number of clusters will have a crucial impact on the quality of the pruning scheme. The detected number of groups will be directly translated to the number of selected classifiers. Therefore, we need to use an efficient clustering algorithm that will be able to select the optimal number of competence areas.

From a plethora of clustering algorithms, we decided to apply an efficient X-means algorithm [222]. It is a modification of k-means algorithm and can be used for our pruning task in a very straightforward way. Let us present below the main advantages of X-mean method:

- Selection of the parameter responsible for the number of clusters for many algorithms is very difficult and time consuming. X-means offer an efficient and fully automatic procedure for establishing the optimal number of clusters based on computation of the Bayesian Information Criterion (BIC).

- X-means offers efficient speed-up for large datasets, by improving the model selection scheme [223]. It calculates the membership to a given centroid not for a single point, but for a subset of points in the constructed *kd*-tree. This is done by considering the geometry of the bounding box for each centroid and their current location, in order to eliminate some of the centroids from the list of potential candidates by proving that they cannot own any point from the current node of the *kd*-tree.

- By noticing that some centroids do not change the location during evaluating different clustering models and no new centroid move into their positions, X-means reduces the time spend on traversing the $kd-tree$ in subsequent iterations.

Let us now describe the automatic selection of the number of clusters implemented in X-means.

We assume that we have at our disposal data in form of **CSM** and a family of alternative clustering models C_j. In our case, different models from this family correspond to the different number of clusters C.

To select the number of partitions X-means uses the *a posteriori* probabilities $P[C_j|\mathbf{CSM}]$ to score the models. We assume that models are spherical Gaussians. One may approximate the posterior probabilities with the use of following formula, commonly known as Schwarz criterion:

$$BIC(C_j) = l_j(\mathbf{CSM}) - \frac{N_f}{2} \cdot \log(L|\mathcal{VS}|), \tag{3.18}$$

where $l_j(\mathbf{CSM})$ is the log-likelihood of the data according to the j-th model and taken at the maximum likelihood point, N_f is the number of free parameters in C_j and $L|\mathcal{VS}|$ denotes the number of objects in **CSM**.

One should notice that the number of free parameters N_f is simply the sum of $C - 1$ class probabilities, centroid coordinates and one variance estimate. This can be extended to more than one centroid by using the fact that log-likelihood of the points belonging to all of the analyzed centroids is the sum of the log-likelihoods of the individual centroids. With this, we can replace $L|\mathcal{VS}|$ from above equations with the number of points that belong to the analyzed centroid.

3.3.3 Selecting Representative Classifiers

Having established the number of clusters with the x-means algorithm, one need to conduct the pruning procedure. There are three main ways, in which one can select the representative classifier for each cluster:

- Select the classifier that is the closest one to the corresponding centroid.
- Select the classifier that is the furthest one from all of the remaining centroids.
- Train a new classifier for each cluster.

There is no clear indication in the literature on how to train a new classifier on the basis of a cluster of classifiers. One may perceive this as a some merging procedure. This cannot be done straightforwardly for one-class methods. That is why in this thesis we examine only the first two approaches.

We assume that we have K classifiers assigned to each cluster.

In the first one, we select a classifier that lies closest to the respective centroid in hope that it will be the best representation of the established support template:

$$\Psi_h = \arg \min_{k \in \{1, \cdots, K\}} dst(\Psi_k, C_h), \tag{3.19}$$

where Ψ^h is the classifier selected for h-th cluster, C_h is the location of the centroid of h-th cluster. As each classifier is characterized by $|\mathcal{VS}|$ values of support functions (see Eq. 3.17) and each centroid C_k is described in a $|\mathcal{VS}|$ dimensional space, we can calculate the distance between them using the Euclidean metric.

In the second one, we select a classifier that lies furthest from all other centroids in hope that it will be the most diverse representation of the established support template:

$$\Psi_h = \arg \max_{k \in \{1, \cdots, K\}} \sum_{s=1, s \neq h}^{C} dst(\Psi_k, C_s), \tag{3.20}$$

where C is the total number of clusters and C_s is the location of the centroid of s-th cluster (other cluster than the one to which this classifier belong).

3.3.4 Experimental Study and Discussion

In this section, we carry an experimental analysis of the proposed methods. We analyze their performance over a set of 20 benchmarks, presented in Appendix A in Table A.1. We follow an experimental framework described in Appendix B. Detailed settings of parameters and results of experiments are given in Appendix C.

The aims of the experiment were as follows:

- To establish if it is possible to carry out an efficient pruning procedure with the usage of support templates.
- To check, which method of selecting a representative classifier from cluster returns better results.

We have constructed a pool of 100 SVDD classifiers on the basis of Bagging algorithm.

We compare two clustering-based one-class pruning methods described in Section 3.3.3 with genetic-based pruning according to a diversity-based optimization (using Sphere Intersection measure), consistency-based pruning [49] and one-class AUC-based pruning [49]. Additionally, we present the results for unpruned pool of classifiers.

In the main body of the thesis, we present only results for an exemplary datasets (to give insight in details of the method) and outcomes of statistical analysis over multiple datasets (Friedman ranking and Shaffer post-hoc), to give an outlook on the global performance of this method.

The comparison of G-mean results for an exemplary dataset (Yeast3) over 5x2 CV is presented in Figure 3.6.

From the results we can see that the clustering-based algorithms can output more efficient pruning performance than their counterparts. This is especially interesting, as our proposed method does not use any kind of specific measure (not consistency nor diversity). This shows that proposed support templates can effectively discriminate mutually complementary classifiers from a pool of redundant ones. Two prosed schemes for selecting classifiers from each cluster return similar performance. The method working on the basis of choosing the classifiers lying closest to the centroid gives a slightly better performance, but the difference is very small for Yeast3 dataset.

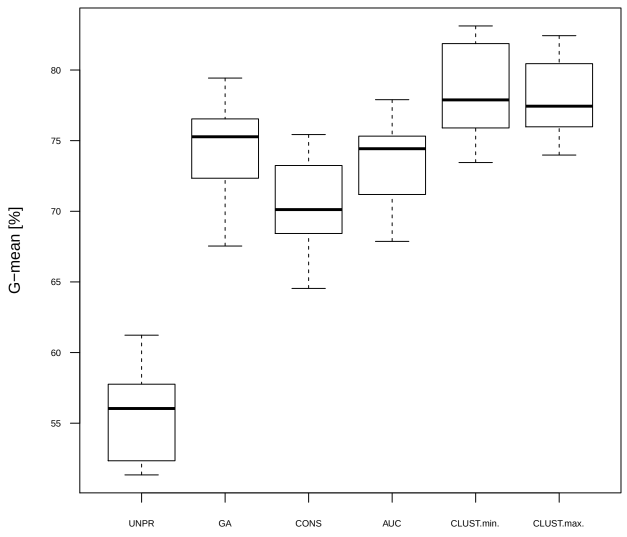


Figure 3.6: G-mean values for two clustering-based ensemble pruning schemes and reference methods for Yeast3 dataset.

To be able to give a more global conclusions, we need to check the performance of the proposed clustering-based pruning schemes over a wider set of one-class benchmarks. We present the results of Friedman ranking test, in order to check the quality of the methods over multiple datasets. The results of this ranking test are given in Figure 3.7.

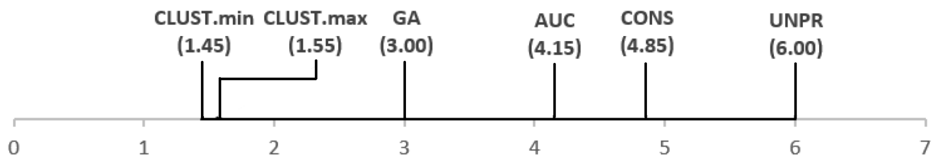


Figure 3.7: The results of Friedman ranking test for examined one-class clustering-based pruning algorithms and reference methods over 20 benchmark datasets.

The results of the ranking test point out that when considering a wide spectrum of datasets clustering-based ensembles display significantly better performance than reference methods used so far in the literature and simpler genetic-based pruning schemes. There is a very small difference between the two proposed methods for selecting representative classifiers for each cluster. To provide an additional verification of this claim, we present the results of post-hoc Shaffer test over multiple datasets for comparison between one-class clustering-based pruning algorithms and reference pruning methods in Table 3.3.

Table 3.2: Shaffer test for comparison between clustering-based pruning schemes and reference pruning methods. The test is carried over 20 benchmark datasets. Symbol '+' stands for a situation in which the proposed method is superior, '-' for vice versa and '=' represents a lack of statistically significant differences.

hypothesis	p-value
CLUST$_{min}$ vs GA	+ (0.0352)
CLUST$_{min}$ vs CONS	+ (0.0204)
CLUST$_{min}$ vs AUC	+ (0.0288)
CLUST$_{min}$ vs UNPR	+ (0.0037)
CLUST$_{max}$ vs GA	+ (0.0358)
CLUST$_{max}$ vs CONS	+ (0.0211)
CLUST$_{max}$ vs AUC	+ (0.0291)
CLUST$_{max}$ vs UNPR	+ (0.0044)
CLUST$_{max}$ vs CLUST$_{min}$	= (0.2984)

Experimental analysis has assessed a high quality of the proposed pruning method. For most of the considered benchmarks the proposed clustering-based pruning was significantly better than all of the reference methods, which was confirmed by the statistical pairwise test. Additionally using two statistical tests for multiple comparison, we were able to prove that our method is statistically superior to all other pruning approaches when considering its performance over a set of benchmarks.

The high quality of our method can be explained by an efficient combination of automatic clustering algorithm (x-means) and using support functions outputted by individual classifiers as input for pruning step. Bayesian Information Criterion allows to quickly and automatically select an optimal number of clusters for each considered model. This is of crucial importance to the pruning step (as the number of clusters will results in the final size of the pruned pool of classifiers). This allows to pre-select the ensemble size beforehand, while other methods use grid-search schemes to choose the number of classifiers. By applying clustering over the support functions, we are able to detect templates in decisions of classifiers. This allows us to chose similar groups of learners and quickly discard irrelevant or similar models. Other methods rely directly on some metrics and as they require some approximations of the outlier class they may mislead the pruning algorithm.

No trend was observed on which method for selecting classifiers from each cluster is the superior one. Statistical test show that the differences are very small, but originate from different datasets. Therefore, one should try both of methods when trying to apply this efficient pruning technique.

3.4 Hybrid One-Class Ensemble Pruning with Firefly Algorithm

In this section, we propose a novel hybrid method for pruning OCC ensembles that combines the nature-inspired optimization-based pruning with clustering-based pruning [148]. It conducts simultaneously a classifier selection and weighted combination to fully exploit the potential of given pool of classifiers. As we aim at finding the best possible subset of classifiers, we propose to apply a metaheuristic optimization procedure to find a good solution in a reasonable amount of time - an important problem for large pool of available learners. We implement the optimization task as a swarm-based search method by using *firefly algorithm* (FA). Reduced population of fireflies represent the pruned ensemble.

Below, let us list the main contributions:

- We propose a new approach for pruning ensembles of one-class classifiers. We utilize the swarm intelligence approach, implemented as firefly algorithm. We encode a given initial pool of classifiers as a population of fireflies. The fireflies form groups, based on their initial brightness. By applying a combined criterion that checks both accuracy and diversity, we ensure that formed groups are mutually diverse and complementary. We select single representatives for each group, thus reducing the number of classifiers in the committee.

- We modify the firefly algorithm, in order to adapt it to the specific nature of forming one-class classifier committees. The brightness of fireflies is expressed by the consistency measure. To describe the location of a single firefly, we do not use distance measures, as they do not give any insight on the performance of the ensemble. Instead we suggest to use the diversity measure based on geometric intersection between decision boundaries (see Eq. 3.13) that is dedicated to one-class problems. As we show further in this section, it satisfies all the conditions imposed upon distance measures, while giving an valuable information about the classifiers in the pool.

- We introduce a weighting scheme for modifying the influence of the selected classifiers during the combination phase. Weights are calculated according to the average lightness of fireflies in the given group. Then, the averaged value is assigned to the representative of this group and used as its weight in the fusion process. This way, we ensure that the final weight does not reflect a single classifier, but the entire group. With this we do not lose the information from the remaining classifiers, while reducing the quantity of learners in the ensemble.

3.4.1 Basics of Firefly Algorithm

Firefly algorithm is a kind of stochastic, nature-inspired, meta-heuristic algorithm that can be applied for solving the most difficult optimization tasks (such as NP-hard problems) [306]. As it is a stochastic approach, it uses a randomization in searching for a set of solutions. It is inspired by the flashing lights of fireflies.

Each firefly in the population is characterized by its brightness and position. Lightness represents the individual fitness of the firefly - 'stronger' lightness corresponds to the better fitness of the individual. Position determines the current location of the firefly in the search space and is used for checking the distance between two fireflies.

The FA procedure is based on interactions between individual fireflies. The level of interaction is modeled by the strength of this event. Each firefly has its attractiveness, which is used to attract other fireflies to it. Attractiveness depends on the light intensity, so each firefly is attracted to the neighbor that glows brighter. The light intensity $\Upsilon(r)$ is dependent on the environment in which the fireflies are located and is modeled according to the inverse square law:

$$\Upsilon(r) = \frac{\Upsilon_s}{r^2}, \tag{3.21}$$

where Υ_s is the light intensity exhibited by the source (the attracting firefly) and r is the distance between the considered two fireflies (the attracting firefly and attracted firefly).

Light is absorbed in the firefly's environment according to the given absorption coefficient τ. The absorption level is used to control the influence of distance between fireflies on the level of their interaction. With the increase of the absorption level, the interactions strength decreases over the same distance.

To properly model the interactions between the population members, one need to use both the distance and absorption of the environment. The combination of the inverse square law and the absorption level can be calculated according to the following solution (its worth noticing that such approach allows to avoid singularity at $r = 0$ in $\frac{\Upsilon_s}{r^2}$):

$$\Upsilon(r) = \Upsilon_o e^{-\tau r^2}, \tag{3.22}$$

where Υ_o is the original light intensity.

The attractiveness of each firefly is proportional to its brightness, as seen by its neighbor. Therefore, we may calculate the attractiveness value as follows:

$$\varkappa(r) = \varkappa_o e^{-\tau r^2}, \tag{3.23}$$

where $\varkappa_o$ is the attractiveness at $r = 0$.

From the above, one may see that the distance between the two fireflies plays a major role on the performance of the FA. To calculate the location of i-th and j-th fireflies, the Euclidean distance is used. The movement of the i-th population member from location x_i towards the j-th member at location x_j, is expressed by:

$$x_i(t) = x_i(t-1) + \varkappa_o e^{-\tau r^2}(x_i(t) - x_i(t-1)), \tag{3.24}$$

where t stands for the number of current FA iteration.

Iterations of FA are conducted until one of the two termination conditions are met:

- the maximum number of iterations N_c have been performed;
- the maximum movement of fireflies in the population is lower than given stop parameter ϵ.

The result of the FA are fireflies gathered around the points of interest. Finally, the fireflies that lay closer to each other than a given parameter Θ are merged together. Remaining population members indicate the found solutions.

3.4.2 Encoding the Ensemble as a Firefly Population

We need to adjust it for the one-class ensemble pruning procedure. It is proposed that each firefly in the population will represent a one-class classifier from the pool. Hence, if we have a given pool of L one-class classifiers assigned to the target class, then our FA population is represented by L individual fireflies.

As for the pruning procedure, we associate it with the process of merging fireflies. We get in result C groups of fireflies (clusters), located closer than Θ to each other. As our population members represent classifiers, we propose to select a single representative for each group. The criteria for selection is based on choosing a member with highest value of lightness. Therefore, we reduce the number of classifiers in the pool and select the most valuable representatives.

Finally, we need to establish how to measure the lightness of each firefly (as to represent the quality of each classifier) and how to measure the distance between the fireflies (in order to check the differences between two classifiers).

3.4.2.1 Measuring the Lightness of Fireflies

The brightness of fireflies represent their individual fitness and influences the interaction level between the population members. Therefore, it is crucial to chose a proper representation for the brightness function.

As our FA represents the classification problem, we should chose such a function that will allow to evaluate the individual quality of analyzed classifiers in the pool. There are a plethora of performance measures used in machine learning such as accuracy, F-measure, AUC or G-mean [132]. These measures require information about both positive and negative examples. As our pruning procedure will be conducted only with the use of the target class ω_T, we need an alternative measure for ranking the quality of one-class classifiers.

In the proposed approach, we associate the consistency of a given classifier (see Eq. (1.33) for details) with the lightness of a l-th firefly that represents it:

$$\Upsilon_o^{(l)} = \text{CONS}(\Psi^{(l)}). \tag{3.25}$$

3.4.2.2 Measuring the Distance Between Fireflies

To properly model the interactions between the classifiers, one need to select a proper environment, in which the distance between the fireflies will be measured.

Here FA is used for pruning one-class ensembles and fireflies represent classifiers in the pool. Hence, it is not trivial on how to describe the 'location' of a classifier. Additionally, we would like that the used measure will be meaningful and give an outlook on the differences between two classifiers (classifiers that are located closely to each other should be more similar).

We propose to use the diversity of classifiers to measure their distance from each other. Classifiers that are located further from each other are at the same time more diverse. Diversity measure fulfils all the requirements of the FA environment and gives us a meaningful information about the classifier location in the decision space.

At the same time it allows us to transform our FA-based pruning algorithm into a kind of multi-objective optimization. The lightness of fireflies will reflect their individual quality, while the distance between them their diversity in the pool. This will lead to a selection of mutually accurate and diverse ensemble members.

In FA, we need to calculate the distance between two fireflies (classifiers). Therefore, we must use a pairwise diversity measure that will give an information about the differences between a pair of one-class learners. We apply a pairwise version of the introduced Sphere Intersection Measure (see Section 3.1.3 for details). We use it as the distance between the fireflies representing the two considered classifiers:

$$dst(x_i, x_j) = DIV_{SI}(\Psi^{(i)}, \Psi^{(j)}). \tag{3.26}$$

3.4.3 Calculating Weights Assigned to Selected Classifiers

As an output of FA, we receive a pruned pool of classifiers. From groups of fireflies that are close to each other a single representative will be selected. Reduced pool of classifiers will consist of one-class learners that display high individual quality (due to their highest consistency) and that are diverse to each other (due to the used way of measuring their location).

We further extend this concept by introducing a novel weighting scheme for controlling the combination process.

To combine the outputs of individual one-class classifiers, we use a following weighted combination method:

$$F_{\omega_T}(x) = \frac{1}{L}\sum_{l=1}^{L} w^{(l)} I(F_{\omega_T}^{(l)}(x)) \geq \theta^{(l)}) + (1 - w^{(l)}) I(F_{\omega_T}^{(l)}(x)) \leq \theta^{(l)}), \tag{3.27}$$

where $w^{(l)}$ is weight assigned to the l-th one-class classifier.

As one may see, weights play an important role in the classifier combination process. They control how strongly the given individual classifier affects the collective decision.

We propose to use the lightness function as a basis for weight calculation. As our ensemble will select a single representative for each of C groups of fireflies, we are interested in calculating weights only for the selected classifiers.

We introduce a novel weighting scheme that assigns a weight for a selected classifier equal to the average lightness of fireflies in its c-th group:

$$w^{(c)} = \frac{\sum_{k=1}^{K} \Upsilon_o^k}{K}, \tag{3.28}$$

where $w^{(c)}$ is the weight assigned to the representative classifier from c-th group of fireflies and K is the number of fireflies (classifiers) in this group. Weights are then normalized to the interval [0;1].

By this each representative classifier will store some information about the quality of all classifiers in its group encoded in its weight. This approach allows to maintain some of the information from discarded classifiers (as they may give an outlook on the competence space of the problem).

3.4.4 Firefly-based One-Class Ensemble Pruning Summary

To give an outlook of the firefly-based pruning algorithm the pseudocode of the proposed method is given in Algorithm 5.

The proposed method may be seen as one originating in the idea of clustering-based ensemble pruning [314]. It aims at discovering groups of similar classifiers and replacing them with a single relevant representative. It should be noted that there are many methods on how such a grouping can be performed - k-means or self-organizing neural networks to name a few [277]. Using firefly algorithm has two additional benefits. Firstly, it does not require a pre-defined number of clusters, as fireflies self-organize

Algorithm 5 Firefly-based One-Class Ensemble Pruning and Weighting

```
Require: pool of classifiers $\Pi$,
    max. number of iterations $N_c$,
    absorption coefficient $\tau$
    movement threshold $\epsilon$
    distance threshold $\Theta$
1: $i \leftarrow 0$
2: encode $\Pi$ as a population of fireflies
3: repeat
4:     calculate the original light intensity of each firefly(according to Eq. (3.25))
5:     calculate the brightness of each firefly (according to Eq. (3.22))
6:     calculate the attractiveness of each firefly (according to Eq. (3.23))
7:     calculate the movements of each firefly (according to Eq. (3.24) and Eq. (3.26))
8: until $i = N_c$ or max movement $< \epsilon$
9: create $C$ groups of fireflies (distance between fireflies in each group lower than $\Theta$)
10: for all $c \in C$ do
11:     for $c$-th group select a single firefly with highest brightness
12:     calculate a weight assigned to the selected firefly (classifier) according to Eq. (3.28)
13: end for
14: normalize weights of $C$ classifiers.
```

themselves. Secondly, it offers the proposed weighted fusion mechanism on the basis of fireflies lightness.

3.4.5 Experimental Study and Discussion

In this section, we carry an experimental analysis of the proposed methods. We analyze their performance over a set of 20 benchmarks, presented in Appendix A in Table A.1.

We follow an experimental framework described in Appendix B. Settings of parameters for used methods, as well as detailed results are described in Appendix C

The aims of the experiment were as follows:

- To establish if the proposed hybrid one-class ensemble pruning allows to detect the most useful subset of classifiers.
- To check, if the proposed classifier weighting procedure contributes to the formed ensemble.

We compare the proposed firefly pruning with its simplified version without the classifier weighting procedure (thus using simple majority voting), genetic-based pruning according to a diversity-based optimization (using Sphere Intersection measure), consistency-based pruning [49] and one-class AUC-based pruning [49]. Additionally, we present the results for unpruned pool of classifiers. We have constructed a pool of 100 SVDD classifiers on the basis of Bagging algorithm.

In the main body of the thesis, we present only results for an exemplary datasets (to give insight in details of the method) and outcomes of statistical analysis over multiple datasets (Friedman ranking and Shaffer post-hoc), to give an outlook on the global performance of this method.

The comparison of G-mean results for an exemplary dataset (Colic) over 5x2 CV is presented in Figure 3.8.

From the results we can see that the firefly pruning algorithm can output more efficient pruning performance than their counterparts, achieving a significant gain in G-mean performance. This can be contributed to the effective combination of clustering-based pruning with a multi-criteria optimization search applied through swarm intelligence. We can also clearly see a significant improvement in the formed ensemble when the proposed classifier weighting scheme is applied.

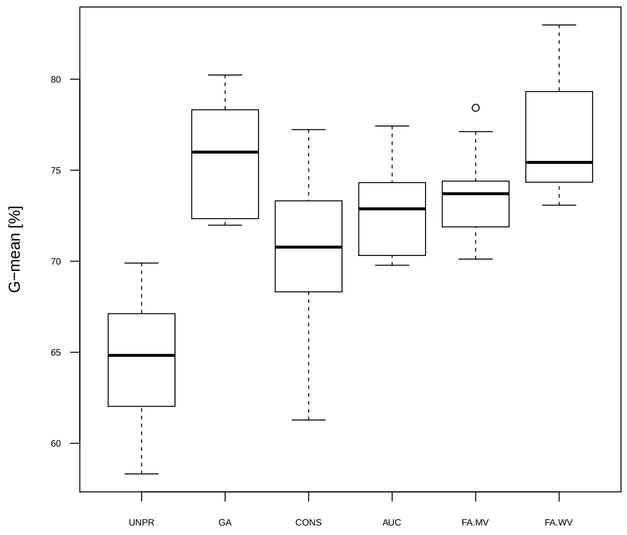


Figure 3.8: G-mean values for hybrid firefly pruning algorithm and reference methods for Colic dataset.

To be able to give a more global conclusions, we need to check the performance of the proposed hybrid firefly pruning algorithm over a wider set of one-class benchmarks. We present the results of Friedman ranking test, in order to check the quality of the methods over multiple datasets. The results of this ranking test are given in Figure 3.9.

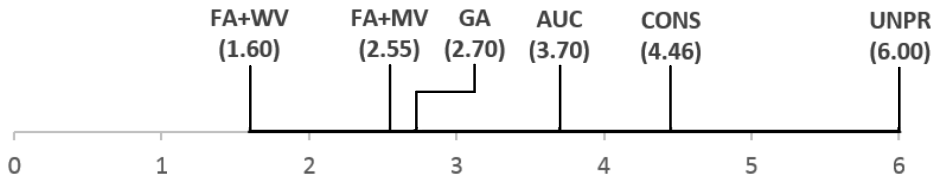


Figure 3.9: The results of Friedman ranking test for examined hybrid firefly pruning algorithm and reference methods over 20 benchmark datasets.

The results of the ranking test point out that when considering a wide spectrum of datasets hybrid firefly pruning algorithm display significantly better performance than reference methods used so far in the literature and simpler genetic-based pruning schemes. The proposed method achieves higher rank than reference methods for both fusion strategies. Weighted classifier combination proposed in this thesis is able to improve in general the performance of the pruned ensemble, thus achieving a superior rank to voting-based version of this method. To provide an additional verification of this claim, we present the results of post-hoc Shaffer test over multiple datasets for comparison between hybrid firefly pruning algorithm and reference OCC methods in Table 3.3.

Table 3.3: Shaffer test for comparison between the hybrid firefly pruning algorithm and reference pruning methods. The test is carried over 20 benchmark datasets. Symbol '+' stands for a situation in which the proposed method is superior, '-' for vice versa and '=' represents a lack of statistically significant differences.

hypothesis	*p*-value
FA+WV vs GA	+ (0.0221)
FA+WV vs CONS	+ (0.0099)
FA+WV vs AUC	+ (0.0173)
FA+WV vs UNPR	+ (0.0032)
FA+WV vs FA+MV	+ (0.0258)

The experimental analysis allowed us to shed light on the performance of the presented method. What is most interesting is the fact that for 16 out of 20 cases the proposed one-class ensemble was statistically superior to all of the reference methods. This proves the usefulness of our FA-based method for simultaneous pruning and weighting one-class committees. Let us take a closer look on the experimental findings.

All of the pruning methods were superior to the unpruned pool of classifiers. This proves our previous statements that careful selection of models in forming one-class ensembles is of a crucial importance. Of course the experiments were biased towards creating a pool with many weak or similar classifiers in it - in order to test the effectiveness of selection mechanisms embedded in each of the tested methods. However, in real-life application we often need to work with models that we have at our disposal and we have no assurance about their quality.

Consistency-based pruning method returned the worst results from all of the pruning procedures tested. This can be explained by the fact that consistency gives us some outlook on the stability of the one-class classifier - but sometimes this do not translates directly into its discriminative abilities. In our method, we use this measure for selecting the final representatives and calculating weights, but the fireflies movement is controlled by a different measure. That is why in our approach, we use the advantages of consistency measure, but at the same time create a diverse ensemble in order to have a complementary models at our disposal. Using consistency-based pruning does not guarantee that.

AUC-based pruning was considered as a good method for choosing one-class classifiers. We should note that calculating AUC requires information about the counterexamples. This is impossible to obtain in real one-class problems, therefore limits significantly the applicability of this algorithm. Authors of this method propose to generate artificial counterexamples and use them for measuring AUC, but we cannot be sure that artificially created data reflects the nature of outliers that are about to appear. Therefore, the ensemble that has a good performance on the training set (with artificial outliers), may fail when 'true' outliers appear. As our method do not require any counterexamples, it is robust to such scenarios - and this advantage in many cases resulted in higher final accuracy of the presented FA-based algorithm.

Genetic algorithm was proven to deliver a satisfactory performance of forming one-class committees, as its fitness function was based on a dedicated diversity measure. However, we have identified several of its drawbacks. It did not take into account the individual quality of each one-class classifiers. In required a significant number of iterations (several hundreds) to find the solution. For larger pool of classifiers, its computational time has risen exponentially and it tended to get stuck in local minima. The FA-based pruning method was designed to cope with this problems. Due to the swarm representation, it is able to efficiently work with large populations. It requires significantly lower number of iterations to find the solution. Finally, it allowed to easily embed the information about the individual performances of classifiers in the pool (as lightness function). Experiments show that for most datasets, our new algorithm was superior to its predecessor.

Comparison with simplified version of our FA-based pruning scheme (which used majority voting) showed that a weighted fusion is a promising research direction for one-class ensembles. In 14 out of 20 datasets, the weights assigned to classifiers allowed to further boost the recognition rate of constructed compound system. The proposed method for weight calculation is simple, yet effective. As it is embedded in the pruning algorithm, it requires no additional parameters, nor does it increases the complexity of the system.

Shaffer test for multiple comparisons showed that the proposed method is statistically better than all of the reference algorithms, when considering its performance over all of the datasets.

3.5 Comments and Recommendations for the Proposed One-Class Ensemble Pruning Methods

In the previous four sections of this thesis we have introduced the concept of diversity measures for OCC ensembles and on that basis we have proposed a set of different pruning techniques for such ensembles. These pruning schemes originated from three different families: optimization-based, clustering-based and a hybrid one utilizing a swarm intelligence approach. Each of them was based on different principles and possessed unique properties. What is worth noticing, each of the introduced methods were statistically significantly superior to state-of-the-art methods for selecting valuable one-class classifiers over a set of diverse benchmarks. It is important to establish under what circumstances should the end-user select specific pruning method from the proposed ones.

This section aims at emphasizing the strong points of each methods, as well as identifying their drawbacks and possible shortcomings. Such a discussion will be beneficial by offering a critical analysis of the introduced pruning frameworks and identifying the potential areas of applicability of each method.

We carried out comparative tests on 20 benchmarks to asses the differences between the proposed ensemble pruning methods. SVDD was used as a base classifier. We generated 100 classifiers with the usage of Bagging. Detailed results are given in Appendix C. We present results for four selected databases, in order to emphasize the advantages and drawbacks of the proposed classifiers.

The comparison of G-mean results for four datasets over 5x2 CV for examined one-class ensemble methods is presented in Figure 3.10. To gain additional insight into the performance of the proposed pruning methods we propose to investigate their running times and the size of the committee after pruning. The results for four selected databases are presented in Table 3.4, while the detailed results are given in Appendix C.

Let us firstly concentrate on the three proposed diversity measures:

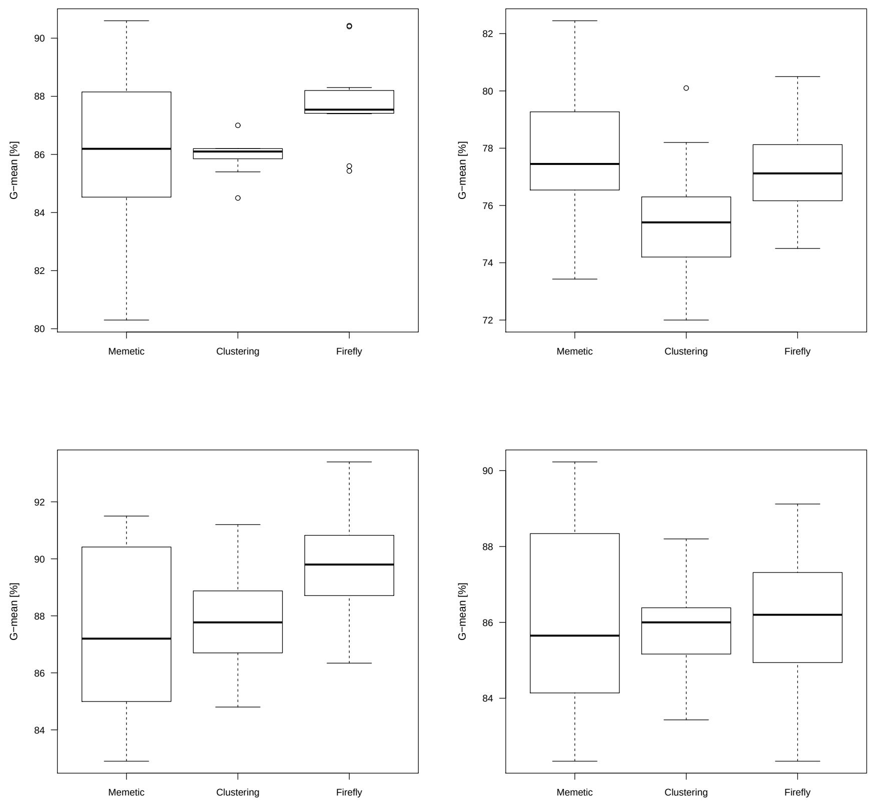


Figure 3.10: The comparison of G-mean values returned by four proposed pruning methods for (*top left*) Spambase, (*top right*) Colic, (*bottom left*) CYP2C19 isoform and (*bottom right*) Breast-Wisconsin datasets over 5x2 CV.

Table 3.4: Running time [s.] and ensemble size (rounded up) for three proposed pruning methods.

Dataset	Memetic-based		Clustering-based		Firefly-based	
	Running time	Ensemble size	Running time	Ensemble size	Running time	Ensemble size
Spambase	136.32 ± 9.45	33 ± 7	64.87 ± 3.97	27 ± 11	81.87 ± 7.82	31 ± 9
Colic	76.84 ± 17.14	19 ± 8	44.87 ± 11.26	13 ± 4	57.93 ± 12.58	14 ± 4
CYP2C19 isoform	167.43 ± 15.32	43 ± 6	93.11 ± 13.94	42 ± 6	112.06 ± 9.61	40 ± 4
Breast-Wisconsin	64.18 ± 4.08	11 ± 3	54.12 ± 2.99	9 ± 1	67.10 ± 5.17	10 ± 3

- One-Class Shanon Measure is an entropy-based measure. It can efficiently take use of a soft decision making process, allowing for a smoother transition between the satisfactory levels of diversity. This method relies on the label outputs of base one-class classifiers and requires no input parameters. Experimental results show that it has quite a little effect on the formed ensemble - it should be used with low weight assigned to it.

- One-Class Energy Measure relies on the label outputs of base one-class classifiers. It utilizes a threshold to discard such combinations of classifiers that will have a very low diversity from further computation. This improve the quality of this measure, at the cost of having higher computational complexity. Additionally, the threshold parameter must be tuned by the user, thus making this measure parameter-dependent.

- Sphere Intersection Measure is a specific method designed for the OCC. It concentrates on geometric properties of the decision boundaries. Experiments indicate that this is the most efficient diversity measure for learning in the absence of counterexamples and that it has a high impact on the quality of formed ensemble. It requires no input parameters. The only drawback of this method is the limitation of use - it can be applied only to spherical-based classifiers such as OCSVM, WOCSVM or SVDD.

Let us now present the advantages of each of three types of pruning methods for one-class ensembles introduced in this thesis:

- Optimization-based pruning takes advantage of efficient search engines. It is straightforward to implement a pruning mechanism, by encoding a pool of classifiers as binary entities and using a dedicated criterion function. Additionally, one may easily improve this scheme by using more than one criterion. We have shown that a weighted combination of consistency and diversity is never inferior to a single-criterion pruning and in many cases can significantly outperform it. User may incorporate different criteria and their combinations according to a specific need.

- Clustering-based pruning relies on detecting groups of similar classifiers and selecting a single representative from each of them. In the proposed methodology, we used support functions and clustered classifiers according to their support for each of the validation objects. By these, we are able to detect diverse classifiers without a need for an explicit measure of diversity or individual quality. This makes clustering-based pruning more flexible and suitable for one-class scenarios (where measures are only estimations, as we do not have an access to counterexamples). Additionally, by using efficient X-means clustering scheme, we are able to significantly speed-up the computation and automatically select the most promising number of clusters.

- Hybrid pruning based on fireflies offers a combination of both optimization-based and clustering-based schemes. It works significantly faster than any genetic or memetic algorithm, requiring a significantly lower number of iterations to reach stopping criteria. We do not need to set the number of groups beforehand, as the movements of fireflies will automatically determine the detected clusters. This method incorporates an efficient method for weighted combination of selected classifiers that further contributes to the high quality of this algorithm.

Every pruning method is somewhat biased. Desirable properties are often introduced at the cost of some drawbacks in other areas. To allow for a fair evaluation of the proposed one-class ensemble pruning schemes, we will now present a critical discussion of their potential shortcomings:

- Optimization-based pruning highly depends on the selected evaluation metrics. Therefore it is possible that some of them may not reflect the true performance of one-class classifiers and misguide the classifier selection process. They require a significant number of parameters to be tuned, thus requiring a time-consuming trail-and-error parameter setting step. The processing time of these methods is usually very long.

- Clustering-based pruning is dependent on the number of detected clusters. X-means cannot guarantee an optimal selection of the number of clusters, thus adding an uncertainty to the pruning step. Additionally, there are no clear indicators how one should chose classifiers from a cluster of models. This method cannot directly apply multi-criteria selection, thus reducing its elasticity.

- Hybrid pruning based on fireflies may be affected by random nature of fireflies movement and improper estimation of their lightness based on consistency (as this is only an estimation of potential performance on outlier class).

3.5 Comments and Recommendations for the Proposed One-Class Ensemble Pruning Methods

Taking under consideration both strong points and shortcomings of each of the presented one-class ensemble pruning algorithms, we may propose a set of suggestions on the areas of applicability of each algorithm:

- When working with spherical one-class classifier the Sphere Intersection Measure is the best choice for diversity estimation. In case of different types of classifiers the One-Class Energy Measure is highly efficient, provided that the threshold value has been properly tuned.

- Optimization-based pruning methods are suitable for complex selection of one-class models, when we want to take several criteria into account (e.g., stability, diversity, execution time, memory consumption). Additionally, it is a worthwhile choice when we have no time constraints imposed on the pruning step. From the tested set of optimization methods memetic algorithms proven themselves to be most efficient.

- Clustering-based pruning methods are an ideal choice for working with large collections of classifiers and for situations where we require a fast pruning algorithm, or when we have no indication which one-class measure of performance to use. These methods are able to efficiently prune the committee with the usage of only support functions outputted by base classifiers.

- Hybrid pruning based on fireflies can be seen as a bridge between these two groups. It offers a significantly reduced pruning time in comparison to examined genetic or memetic methods, while being able to take advantage of multi-criteria selection. Additionally, it can further boost the outputted ensemble by using a weighted combination scheme. This algorithm is a flexible solution for a broad range of one-class problems.

4

Practical Applications of One-Class Ensembles

In this chapter, we will discuss three applications of introduced one-class classification ensembles in real-life problems.

Two previous chapters focused on describing novel proposals for general methods dedicated to forming and pruning one-class ensembles. The experimental evaluation was performed with popular benchmarks extracted from online repositories. Such benchmarks are widely used in evaluating machine learning algorithmsand by using a diverse selection of them we can examine the behavior of given methods over a series of well-documented pattern classification problems.

Results on these benchmarks cannot be often translated directly onto the performance on new, real-life applications. Such practical implementations are often much more challenging, requiring modifications of methods, incorporation of some given background knowledge, or working under given constraints (e.g., limited computational time or memory requirements).

That is why we present an additional evaluation of the proposed methods over three real-life problems that emerged during the collaboration with external research groups. They originate from three domains: environmental engineering, computer vision and medicine. Let us present shortly each of considered problems in this chapter:

- *Recognition of a specific volatile organic compound with the use of a gas sensor array known as the artificial nose.*
- *Multi-dimensional computer vision data analysis for hyperspectral image processing in remote sensing application.*

- *Breast cancer malignancy grading on the basis of fine needle biopsy images and imbalanced data.*

The following sections will give details of each of these real-life applications.

4.1 Volatile Organic Compound Classification with Gas Sensor Array

Measurement systems based on a sensor array have attracted widespread attention in the past few decades. They have several advantages over conventional analytical instruments which use methods, such as infrared spectroscopy and chromatography, because of possible miniaturization, low data acquisition and maintenance costs [260]. These devices consist of three fundamental components: (1) a sampling system, (2) an array of gas sensors with partial or overlapping sensitivities and (3) a pattern recognition machine. The gas sensor array combined with pattern recognition algorithms have been traditionally used to solve problems of non-selectivity, nonlinearities of the sensors' responseand long-term drift [208].

Gas identification is one of the most important functions of sensor array systems [262]. By this term we mean recognizing the identity of the test gas. This function is to a large extent realized by the pattern recognition system coupled with sensor array, in particular, by the classifier. Gas classification, means categorization of the sensor outputs in classes containing similar chemical patterns, during the analysis of a test sample. In other words, the task of a classifier is to use the feature vector provided by the feature extractor to assign the gas it represents to an appropriate category. Different approaches are proposed to classify gas under test using the sensor array system. Gas identification is one of the most important functions of sensor array systems [262]. By this term we mean recognizing the identity of the test gas. This function is to a large extent realized by the pattern recognition system coupled with sensor array, in particular, by the classifier. Gas classification, means categorization of the sensor outputs in classes containing similar chemical patterns, during the analysis of a test sample. In other words, the task of a classifier is to use the feature vector provided by the feature extractor to assign the gas it represents to an appropriate category. Different approaches are proposed to classify gas under test using the sensor array system.

However, most of the proposed approaches concentrate on a scenario in which the number of possible classes (compounds) is given beforehand. Thus this can be viewed as a canonical multi-class classification task. This assumption fails from a practical point of view [261]. While in controlled laboratory environment it is possible to control the examined compounds, in normal environment we deal with highly uncertain situation. We will encounter different mixtures of compounds and it can be impossible to predict which of them will appear. Additionally, there is a high chance that some new, unseen compound will appear in the analyzed mixture. Let us consider an alcohol detector used by police to determine the condition of a driver. Here, we look for ethanol compounds and can identify a number of standard compounds that may appear in human breath. There is still a high chance of other compounds to appear, as the composition of human breath depends on many factors (e.g., last meal, mouth hygiene, or the surrounding environment). That is why standard multi-class methods will fail for such scenarios and there is a need to propose new, efficient models robust to the appearance of unknown classes.

We propose to use one-class classification for a robust detection of specific compound from a larger mixture. By this, we can deal with the appearance of any number of unexpected classes by considering all of them as outliers. Thus we achieve a more flexible pattern classification system for volatile organic compound classification. We propose to use ensemble methods for one-class classification, due to excellent results returned by classifier committees in multi-class problem applied in our earlier work [259].

Additionally, it is important to detect which sensors are the most useful for a given recognition problem. This way, we can simplify the structure of our array and reduce the cost connected with the exploitation of a large number of sensors.

This research was done with cooperation with prof. Andrzej Szczurek and prof. Monika Maciejewska from Faculty of Environmental Engineering, Wroclaw University of Technology, Poland. They contributed with measurements and data acquisition.

In the following section, we will present the details of the proposed solution.

4.1.1 Details of the Gas Sensor Array

The measurements were performed with an experimental setup composed of the following functional blocks: pure air generator, apparatus for the preparation and delivery of gaseous samples, sensor array, voltage supplier, interface circuits containing load resistors, digital multimeter with multiplexer module and data acquisition card, personal computer with HP BenchLink Datalogger software, as shown in Figure 4.1 The system was dedicated to prepare gaseous samples and to perform dynamic sensor measurements.

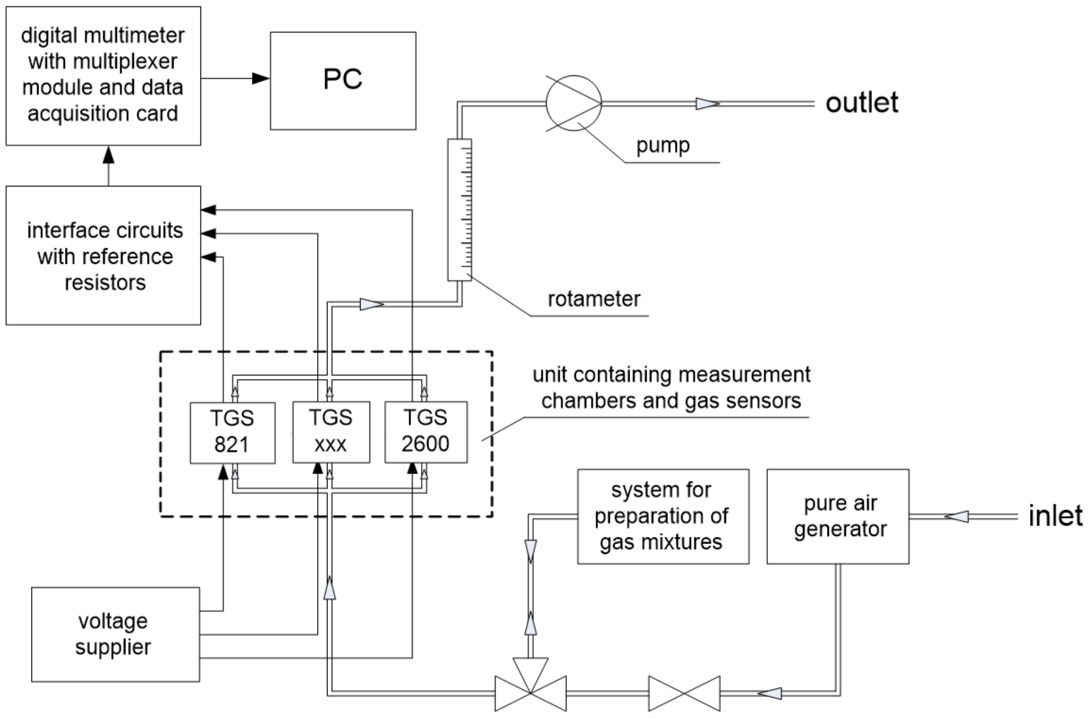

digital multimeter with multiplexer module and data acquisition card
PC
outlet
pump
interface circuits with reference resistors
rotameter
unit containing measurement chambers and gas sensors
TGS 821
TGS xxx
TGS 2600
system for preparation of gas mixtures
pure air generator
inlet
voltage supplier

Figure 4.1: Scheme of experimental setup together with gas sensor array details used for data collection [259].

The function of pure air generator (Horiba) was to purify ambient air. It was achieved by passing the air through cartridges filled with silica gel, activated carbon, soda lime and molecular sieve. The evaporation method was applied for preparing gas mixtures of predefined composition in the dedicated apparatus. The desired amounts of liquid analytes were injected into the heated vessel and then vaporized in a stream of purified air. The concentrations of the substances in air were determined by the dosage, the airflow and the dilution rate. Both, pure air generator and gas mixture preparation system were designed for the dynamic and continuous supply of gases. There was a possibility to connect the unit containing gas sensors with the pure air generator as well as with the system for preparation of the test gas, by Teflon tubes. The unit consisted of fifteen sensor chambers with sensors inside. Those small, airtight, flow-type test chambers made of aluminum were specially designed. Each sensor was mounted inside its own chamber. As a result of parallel chamber connection, by means of Teflon tubes, sensors were simultaneously exposed to the gas mixture of the same composition. Moreover, sensor cross-talk was eliminated, which could result from the exposure to the same gas. Chambers were fitted with necessary electrical connections. Each sensor was connected to the voltage supplier and to the measuring unit. The working temperature of the sensors was approximately 350° Cand it was controlled by applying a constant voltage to their heaters. The voltage variations measured on the load resistor, connected to the sensor, constituted the sensor output signal. The experiments were carried out at a constant bias voltage. The digital multimeter with the data acquisition card was used to transfer the output voltage of each sensor to the PC. The signal recording was performed every 1 s.

Fifteen commercially available Taguchi Gas Sensors made by Figaro Engineering Japan were applied in this work: TGS 821, TGS 822, TGS 824, TGS 825, TGS 826, TGS 880, TGS 883, TGS 800, TGS 2201 (gasoline), TGS 2201 (diesel), TGS 2106, TGS 2104, TGS 2602, TGS 2620, TGS 2600. These sensors were chosen because of their satisfactory performance e.g. sensitivity, response time, robustness, low price and simplicity of use in many applications involving measurements of volatile compounds at different concentrations.

In this work we have focused on the dynamic response of the sensors when they were exposed to the test gas in various physical and chemical conditions. This kind of exposure was realized by applying the mode of operation called stop flow [207]. The measuring procedure in this mode consists of three, sequentially performed stages. They are shown in Figure 4.2 together with the typical corresponding gas sensor signal.

The duration of each stage is fixed. The exposure in stop flow mode is preceded by attaining steady state of sensor in the stream of pure air.

The first stage of operation in stop flow mode is the dynamic exposure of sensors to the stream of air containing the volatile organic compound of interest. The test sample is delivered to sensor chambers and it is allowed to continuously flow through. The gas flow rate in the sensor system is kept constant. The sensor output signal during this stage results from the kinetics of the processes which evoke sensor response. Initially, the concentration of the test gas quickly increases in sensor chambers. This causes gas concentration change at the sensing surface. The conductivity of semiconductor is

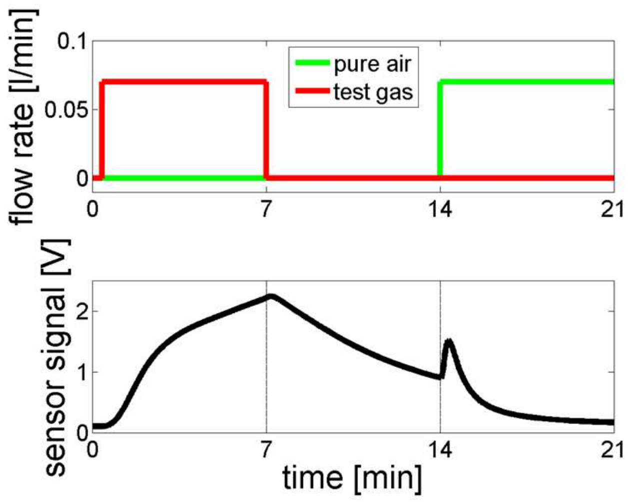


Figure 4.2: The measuring procedure in stop flow mode of gas sensor operation and the associated sensor signal [259].

additionally affected by a number of time dependent processes such as: the transport of the reactive species into the sensor, the diffusion of gas molecules inside pores of the sensing material, adsorption and desorption, the catalyzed red-ox reactions on the surface of the sensing layer (mainly their kinetics) and the electrical/electronic effects in the semiconductor. The recorded sensor signal is characterized by a considerable dynamics, in particular at the very beginning of test gas delivery. Later, the atmosphere surrounding sensor stabilizes, which allows for attaining the dynamic equilibrium and a quasi-steady state of the sensing material. During second phase of operation, the gas flow is stopped and the sample remains in sensor chambers. In this period of time, sensor temperature and partial pressure of the volatile organic compounds in sensor chambers are continuously changed due to the oxidation reaction taking place at the sensor surface. The associated sensor output signals usually exhibit slow decay in time at an approximately constant rate. The third stage of stop flow mode of operation functions both as the gas sensor recovery process and as a source of analytical information. The gas line and chambers are cleaned with a stream of pure air, which is delivered to sensor system at a constant flow rate. Sensors are again in dynamic conditions. The processes, which influence output signals, are similar as during the first stage. However, the chemical composition of gas stream is changed due to the test compound removal from the sensor chambers. The dynamics of sensor signal is initially high, followed by an asymptotic decrease towards the sensor baseline.

Each sensor may contribute different information about the analyzed compounds. That is why an array of them is being used. However, in many cases some of the sensors are redundant or noisy, thus reducing the discriminative power of the entire array. An example of such a situation, where selective use of sensors may bring a better recognition accuracy than using an entire array is given in Figure 4.3.

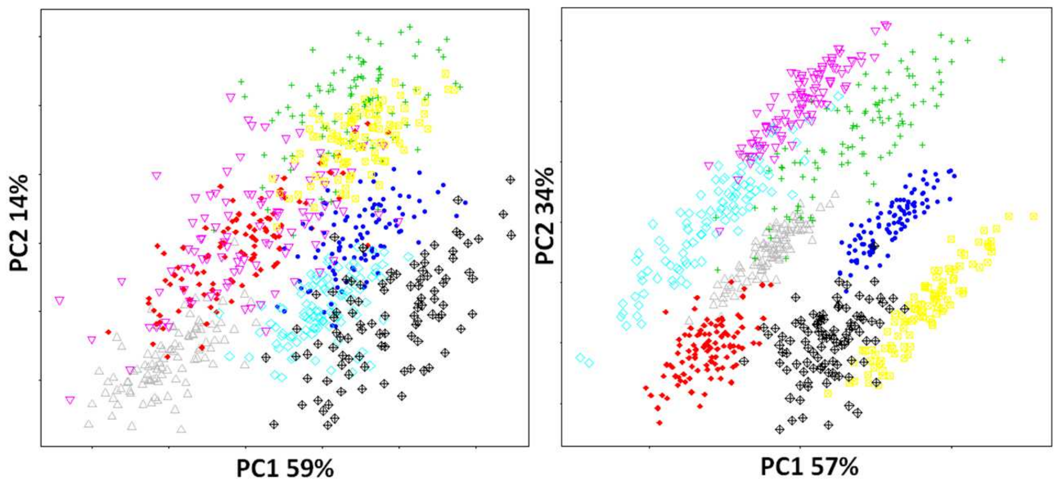


Figure 4.3: PCA plot for the data based on: (a) the response of the entire sensor array, (b) the dynamic signal of single senor, TGS 2201 (diesel). Following color scheme was used for examined compounds: hexane (blue), heptane (black), octane (yellow), cyclohexane (gray), benzene (aquamarine), toluene (green), xylene (red), ethylobenzene (pink).

That is why sensor selection is an important step in VOC classification process.

4.1.2 Feature Extraction

We utilised the sensor array response to the test gas, obtained in the stop flow mode of operation as the basis for gas identification. The distinct character of our approach consists in focussing on the analysis of the response signal of each individual sensor in the array. The stop flow mode of operation provides a set of time-dependent transient sensor signals in response to a target gas. On one hand their transient character guarantees the increased amount of information, but large number of data in the time series which form the signal is a serious disadvantage especially if part of the data is irrelevant for the classification task. In order to address this problem sensor signal pre-processing is usually applied, aimed at data compression without a significant loss of information which characterizes particular gas mixture.

Most frequently versatile parameters of dynamic sensor signals are considered instead of the original signals in order to achieve the compression. These are for example: the max-min difference, the slope, the derivative, the integral of the signal calculated for a selected time interval. Also parameters of mathematical models of sensor signal are utilized for the purpose of compression. The most popular are the parameters of Fourier and wavelet transform which are well applicable to sensor signals obtained by means of modulation. These two strategies although proved successful on a number of occasions, retract from utilizing the actual values of sensor signal as the classifier input. In that sense, their use increases the computational complexity of the gas identification task at the stage of sensor signal processing. We proposed a reduction of sensor signal dimensionality by a granulation approach.

The original sensor signal obtained in our experiments was composed of 1260 data points, each them representing discrete sensor measurement performed every 1 s. We used a window of a width equal to ten time units and the sensor signal was divided into fragments of this size. Every ten values in a window were then considered a single data granule and they were replaced with a single value equal to the average of ten points. This means that the original feature space was transformed into the ten times smaller.

A distinct feature vector, which consisted of 126 elements, was obtained based on the signal of each single sensor. Therefore, after granulation operation, there were available fifteen feature vectors. Each of them was an alternative fingerprint corresponding to a given gas mixture. These signatures were exploited by a pattern recognition machine for the classification task.

4.1.3 Proposed One-Class Ensemble Approach

To solve the problem of detecting of a specific compound in an unknown input mixture, we propose to utilize one-class classifiers. We delegate One-Class Rotation Forest (described in the Section 2.3) for this task. We want to detect which sensors have a high impact on the classification step. In our previous works we have shown that associating a classifier with each of the sensors improve the diversity in the pool and leads to locally specialized classifiers [259].

In order to select the most important sensors, we propose to apply an ensemble pruning step. As each base classifier is assigned to a single sensor, thus discarding redundant classifier will lead to removal of an associated sensor. For this task, we propose to apply hybrid one-class ensemble pruning with firefly algorithm described in Section 3.4.

The details of the proposed approach are given in the pseudocode form in Algorithm 6.

Algorithm 6 One-class learning for volatile organic compound classification with the use of a gas sensor array

```
Require: pool of classifiers $\Pi$,
    One-Class Rotation Forest training procedure (),
    hybrid one-class ensemble pruning procedure (),
    number of sensors $S$
1: $i \leftarrow 0$
2: repeat
3:     Train $s$-th OC-RotF on the data from $s$-th sensor (as in Algorithm 3)
4:     Add $s$-th classifier to $\Pi$
5:     $i \leftarrow i + 1$
6: until $i = S$
7: $\Pi^* \leftarrow$ hybrid one-class ensemble pruning ($\Pi$) (as in Algorithm 5)
8: combine classifiers from $\Pi^*$ using weighted combination
```

4.1.4 Experimental Analysis and Discussion

We have two aims of the experimental study:

- To asses the quality of OC-RotF in real environmental engineering application
- To see if it is possible to identify the most relevant sensors using an ensemble pruning technique.

We decided to model the recognition scenario as benzene detection problem. Benzene is a natural constituent of crude oiland is one of the most elementary petrochemicals. Benzene is an aromatic hydrocarbon and the second [n]-annulene ([6]-annulene), a cyclic hydrocarbon with a continuous pi bond. It is sometimes abbreviated Ph–H.

Benzene is a colorless and highly flammable liquid with a sweet smell. It is mainly used as a precursor to heavy chemicals, such as ethylbenzene and cumene, which are produced on a billion kilogram scale. Because it has a high octane number, it is an important component of gasoline, comprising a few percent of its mass. However most non-industrial applications have been limited by benzene's carcinogenicity. Benzene increases the risk of cancer and other illnesses. It is a notorious cause of bone marrow failure. Substantial quantities of epidemiologic, clinicaland laboratory data link benzene to aplastic anemia, acute leukemia and bone marrow abnormalities. Therefore it is highly important to early detect the leaks of benzene in industrial buildings where people operate.

To create a mixture of gases to be analyzed, we have mixed benzene with the following ones: hexane, heptane, octane, cyclohexane, toluene, xyleneand ethylobenzene. Chemicals were purchased from Sigma–Aldrich. There were examined gas mixtures composed of pure, dry air and one volatile organic compound (VOC). The following concentration ranges were considered: hexane (17-204 ppm), heptane (15-183 ppm), octane (14-165 ppm), cyclohexane (21-249 ppm), benzene (25-302 ppm), toluene (21-255 ppm), xylene (18-222 ppm), ethylobenzene (18-220 ppm).

Each signal was characterized by 126 features. We have obtained 100 measurements for each compound, thus creating 100 objects for each class.

We have created a difficult pattern classification scenario, where a specific class must be detected from a mixture of eight classes. We assume that during the training step we have at our disposal only objects from benzene class.

Our OC-RotF uses SVDD as a base classifier with RBF kernel, $\sigma = 0.3$and $C = 8$. We have compared this method with single SVDD, SVDD with Bagging, SVDD with Boostingand SVDD with Random Subspace. Each ensemble consisted of 25 base classifiers. In order to present a fair comparison, each examined method used the specialization on sensors (one classifier per sensor) and hybrid firefly pruning. Details of these methods and their parameters are given in Appendix C. The results of 5x2 combined F-test are given in Table 4.1, the G-mean results from a 5x2 CV for analyzed methods are given in Figure 4.4, while the frequency of each sensor being selected to the final ensemble over 5x2 CV is presented in Figure 4.5.

From the results, we can see that our OC-RotF has returned superior performance of G-mean around 97.5%. This is more than 5% better than the second-best approach,

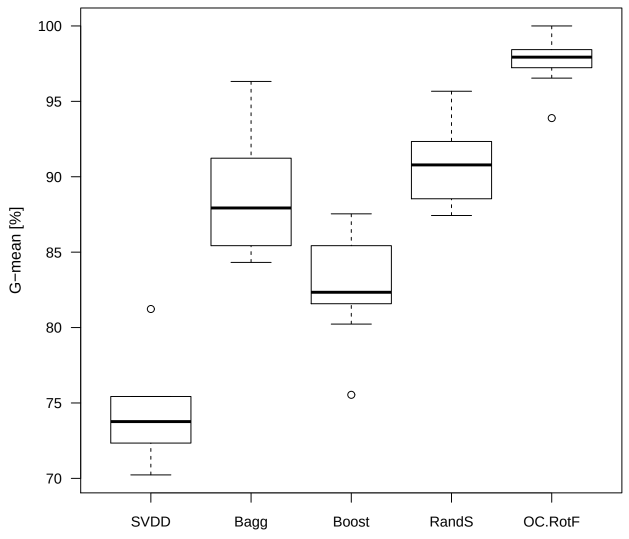


Figure 4.4: G-mean values for OC-RotF and reference methods for volatile organic compound classification.

Table 4.1: 5x2 combined F-test for comparison between the OC-RotF and reference methods over the analyzed VOC dataset. Symbol '+' stands for a situation in which the proposed method is superior, '-' for vice versaand '=' represents a lack of statistically significant differences.

hypothesis	p-value
OC-Rotf vs SVDD	+ (0.0041)
OC-Rotf vs Bagg	+ (0.0296)
OC-Rotf vs Boost	+ (0.0103)
OC-Rotf vs RandS	+ (0.0371)

based on Random Subspace. Boosting and Bagging returned unsatisfactory performance, not being able to capture the properties of the target class.

Considering various gas sensing properties of sensors used in this work it was expected that their relative importance as the sources of classification useful information could be different. This fact was analyzed based on a pruning procedure, by discarding classifiers associated with sensors outputting data with lower discriminative power. The weights shown in Figure 4.5 indicate that the sensor array used in our study contained

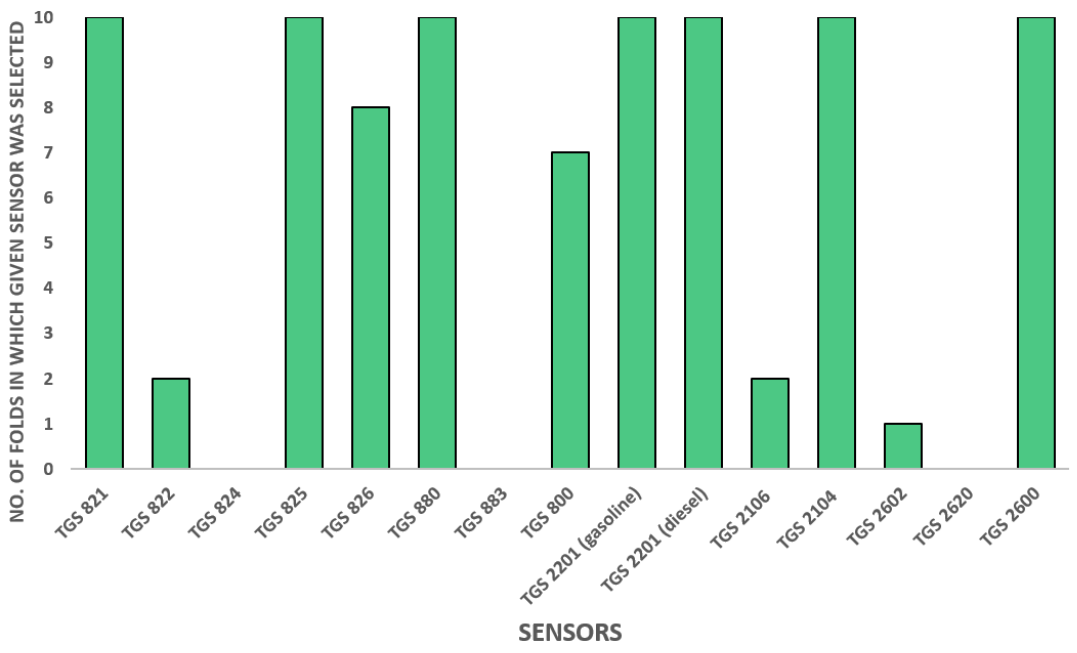


Figure 4.5: The number of iterations of 5x2CV in which the given sensor was selected for the final ensemble. One may easily see that some sensors do not contribute nothing to the classification process.

elements which were highly influential for the classification results and others which had a relatively small contribution to the final decision. The following sensors were selected in all of iterations: TGS 2104, TGS 2201 (gasoline), TGS 2201 (diesel), TGS 2600 and TGS 825.

We have shown that it is possible to create a highly robust pattern classification system for gas sensor array analysis with the use of proposed one-class ensemble forming and pruning schemes. Additionally, we have showed that it is possible to obtain excellent recognition accuracy with a limited number of classifiers which contributes to a significant reduction in the exploitation costs of the sensor array.

4.2 Multi-Dimensional Data Classification for Hyperspectral Image Analysis

Classical pattern recognition methods work on vector spaces. This reflects basic properties of simple measurements which stack different values (features) into 1D vectors which are assigned to classes. Many phenomena lead to measurements which change specifically depending on a chosen dimension or a coordinate. Well known examples are video sequences which are composed of two-dimensional frames containing three-valued pixels, displayed a number of times per second .

Naturally they are four-dimensional data which become even five-dimensional considering sound. Such examples arise in many domains when measuring signals under different settings of an experiment.

Another example are hyperspectral images. Such images contain hundreds of spectral channel, each one covering a small portion of electromagnetic spectrum. This spectral high-resolution is expected to allow making detailed thematic maps of remote sensing data by means of spectral classification of different materials expected in the sensed scene. This richness of information may even allow efficient content based exploration of remote sensing databases. The classification of hyperspectral data is a challenging task due to the high dimension of the data.

Such data are called multidimensional or tensor like signals. They do not fit well into the classical one-dimensional vector based framework. Although there are many ways to vectorize multidimensional data, it has been observed that such operation usually leads to significant loss of important information, since some values which were close in terms of a chosen coordinate become differently separated if data are arbitrarily linearized into a vector. Therefore in recent years much attention gained development of pattern recognition methods which inherently consider multidimensionality of the classified data.

In order to deal with the multi-dimensional and complex nature of the data, we propose to enrich standard one-class classifiers with a method to process complex images directly as a tensor. We do this by applying the recently proposed chordal kernel for tensor data [249]. It allows to transform and manage data in form of tensors and can be directly implemented into kernel-based one-class classifiers such as OCSVM, WOCSVMand SVDD.

This research was done in cooperation with prof. Bogusł aw Cyganek from Faculty of Electronics, AGH University of Technology, Poland, who contributed with the tensor representation.

4.2.1 Chordal Distance Between Pattern Tensors

In this Section, we will discuss the basis of tensors applied in pattern recognition and machine learning, as well as methodology for computing the chordal distance for tensor-based kernels.

4.2.1.1 Tensors for Pattern Recognition

Tensors play an important role in physics, especially in mechanics and relativistic physics. They are used to describe relations among physical values, which follow changes of the coordinate systems change in accordance with strict rules called tensor transformation laws. The other definition of tensors can be constructed with help of the multi-linear functions operating on a vector field and its dual. However, in data analysis tensors are limited to represent multi-dimensional cubes of data. In other words, data that depends on multiple factors, or degree of freedom, can be grouped into such a multi-dimensional array. An example can be measurements of groups of clients buying specific groups of merchandise at certain days, prices, etc. Similarly, a color video signal can be seen as a four dimensional cube of values changing in accordance with the x-y spatial, c colorand t time dimensions. Thus, frequently patterns which are expressed as multi-dimensional cubes of data need to be analyzed with mathematical tools relevant to tensor analysis. We focus on tensor representation of different image types. Theretofore, a brief introduction to tensor representation for data analysis with special stress on image processing is presented. A more detailed description with further explanations can be found in literature [55]. A *tensor*

$$\mathcal{A} \in \Re^{N_1 \times N_2 \times \ldots N_L} \tag{4.1}$$

is a L-dimensional "cube" of real valued data, in which each dimension corresponds to a different factor of the input data space. For further discussion, scalars are denoted with small letters, such as a, column vectors with bold $\mathbf{a}$, matrices with the bold capitals, such as $\mathbf{A}$and higher order structures - tensors - with bold calligraphic letters, such as $\mathcal{A}$.

With the above definition of a tensor, the j-mode vector of the K-th order tensor is a vector obtained from elements of $\mathcal{A}$ by varying only one its index N_j while keeping all other indices fixed. Further, if from the tensor $\mathcal{A}$ the matrix

$$\mathbf{A}_{(j)} \in \Re^{N_j \times (N_1 N_2 \ldots N_{j-1} N_{j+1} \ldots N_L)} \tag{4.2}$$

is created, then the columns of $\mathbf{A}_{(j)}$ are j-mode vectors of $\mathcal{A}$. Also $A_{(j)}$ is a matrix representation of the tensor $\mathcal{A}$, called a j-mode tensor flattening (known also as tensor matricization). The j-th index becomes a row index of $\mathbf{A}_{(j)}$, while its column index is a product of all the rest L-1 indices. Figure 4.6 shows three flattenings of a 3D tensor.

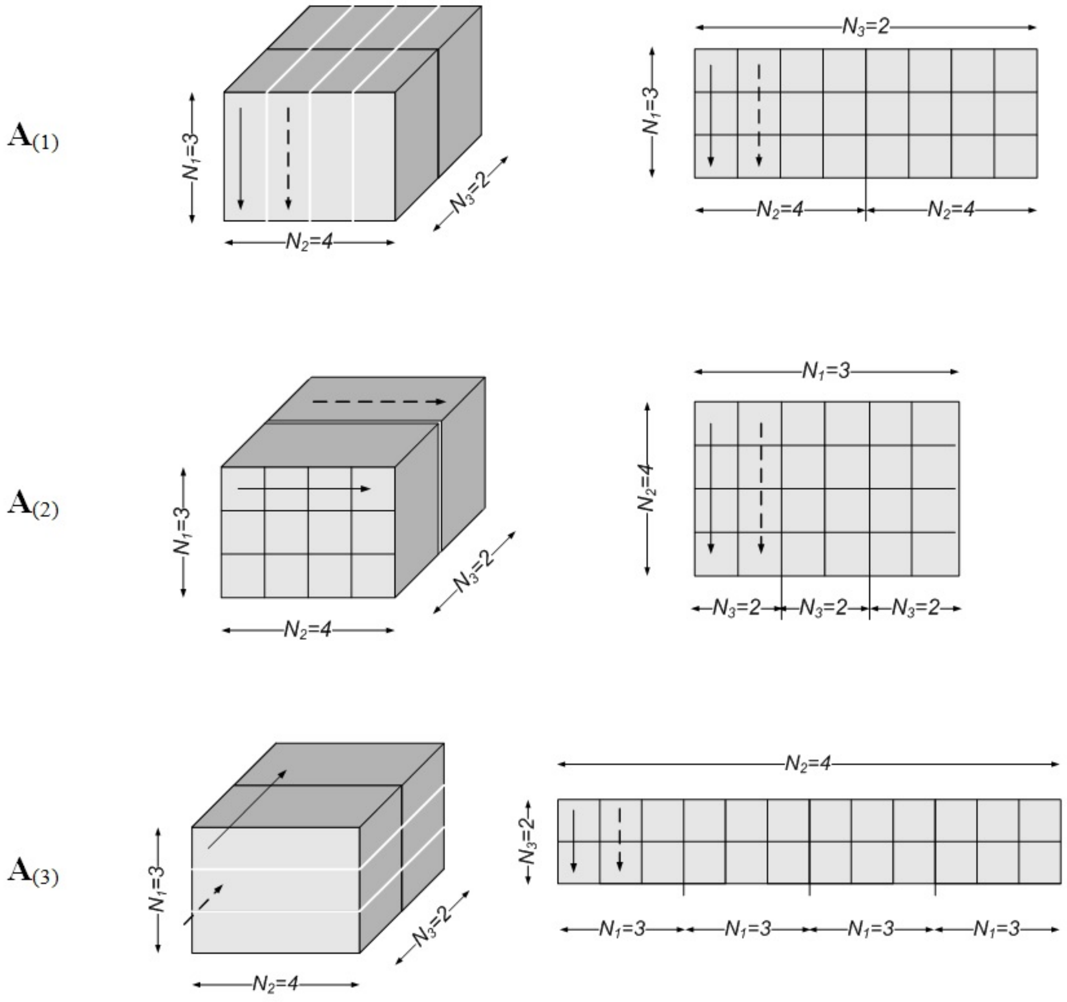


Figure 4.6: Examples of 3D tensor flattening in the forward index permutation mode.

The three distinct flattenings of a $N_1 \times N_2 \times N_3$ (3×4×2) tensor shown in Figure 4.6 assume a forward mode of index permutations which is more suitable for video processing [56]. The matrix $\mathbf{A}_{(1)}$, which directly reflects video location in memory, has dimensions $N_1{\times}N_2N_3$, $\mathbf{A}_{(2)}$ – $N_2{\times}N_3N_1$and $\mathbf{A}_{(3)}$ – $N_3{\times}N_1N_2$, respectively.

A useful concept of tensor algebra is a p-mode product of a tensor $\mathcal{A} \in \Re^{N_1 \times N_2 \times \ldots N_L}$ with a matrix $\mathbf{M} \in \Re^{Q \times N_p}$. A result of this operation is the tensor $\mathcal{B} \in \Re^{N_1 \times N_2 \times \ldots N_{p-1} \times Q \times N_{p+1} \times \ldots N_L}$ whose elements are as follows:

$$\mathcal{B}_{n_1 n_2 \ldots n_{p-1} q n_{p+1} \ldots n_L} = (\mathcal{A} \times_p \mathbf{M})_{n_1 n_2 \ldots n_{p-1} q n_{p+1} \ldots n_L} = \sum_{n_p=1}^{N_p} a_{n_1 n_2 \ldots n_{p-1} n_p n_{p+1} \ldots n_L} m_{q n_p}. \tag{4.3}$$

where $a_{n_1 n_2 \ldots n_L}$ stands for an element of the L-th dimensional tensor $\mathcal{A}$ at (position) index $(n_1, n_2, \ldots, n_L)$, similarly m_{qn_p} is an element of the matrix $\mathbf{M}$ at index (q, n_p).

As was shown, the *p-mode* product can be equivalently represented in terms of the flattened versions of the tensors $\mathbf{A}_{(p)}$ and $\mathbf{B}_{(p)}$. That is, if the following holds

$$\mathcal{B} = \mathcal{A} \times_p \mathbf{M} \tag{4.4}$$

then

$$\mathbf{B}_{(p)} = \mathbf{M}\,\mathbf{A}_{(p)} \tag{4.5}$$

An important property of the tensor flattening is that each gives rise to a

different matrix with specific properties. Thus, an analysis of the space properties spanned by each flattening matrix $\mathbf{A}_{(j)}$, gives unique information of data cube seen from the j-th dimension. This property is used to build the higher-order singular value decomposition (HOSVD), as well as will be used to construct a suitable kernel for data analysis, as will be discussed in the next section.

To analyze properties of a space spanned by each matrix $\mathbf{A}_{(j)}$, it is decomposed with the Singular Value Decomposition (SVD) decomposition, as follows:

$$\mathbf{A}^{(j)} = \mathbf{S}^{(j)}\mathbf{V}^{(j)}\mathbf{D}^{T(j)} = \sum_{i=1}^{R_{A_{(j)}}} v_i^{(j)} \mathbf{s}_i^{(j)} \mathbf{d}_i^{T(j)} = \begin{bmatrix} \mathbf{S}_{\mathbf{A},1}^{(j)} & \mathbf{S}_{\mathbf{A},2}^{(j)} \end{bmatrix} \begin{bmatrix} \mathbf{V}_{\mathbf{A},1}^{(j)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{D}_{\mathbf{A},1}^{T(j)} \\ \mathbf{D}_{\mathbf{A},2}^{T(j)} \end{bmatrix}. \tag{4.6}$$

In the above, $_{\mathbf{A},1}$ and $_{\mathbf{A},2}$ denote indices of block matrices related to the kernel and null spaces of $\mathbf{A}_{(j)}$, respectively. It holds also that $\mathbf{S}_{\mathbf{A},1}^{(j)}$ and $\mathbf{D}_{\mathbf{A},1}^{T(j)}$ are unitary matrices of the kernel of $\mathbf{A}_{(j)}$. Finally,$\mathbf{V}_{\mathbf{A},1}^{(j)}$ is a diagonal matrix with R_A non-zero elements, whose size determines rank of the matrix $\mathbf{A}_{(j)}$. It immediately follows also that

$$\mathbf{D}_{\mathbf{A},1}^{T(j)}\mathbf{D}_{\mathbf{A},1}^{(j)} = \mathbf{I}_{R_{\mathbf{A}}\times R_{\mathbf{A}}} \tag{4.7}$$

The analogous conditions hold for the j-th mode flattening of the tensor $\mathcal{B}$. However, in this case, its rank can be different which is further denoted by R_B.

In pattern recognition one of the most important concepts is a distance between patterns. In the well know vector space, frequently used is the Euclidean distance. It can be also directly applied to the patterns represented as tensors. However, such simplistic approach disregards important information hidden behind the spatial composition and interrelations among data. In this case a useful concept is to consider distances of the subspaces spanned by the flattening matrices of pattern tensors. In this formulation a more appropriate is a distance among principal angles, called *projection Frobenius norm* or *a chordal distance* [107]. For two tensors in their *j-th* flattened mode matrices $\mathbf{A}_{(j)}$ and $\mathbf{B}_{(j)}$, their chordal distance is defined as follows [249]:

$$D_{ch}^2\left(\mathbf{A}_{(j)},\mathbf{B}_{(j)}\right) = D_F^2\left(\Pi_{\mathbf{A}_{(j)}},\Pi_{\mathbf{B}_{(j)}}\right) = \left\|\Pi_{\mathbf{A}_{(j)}} - \Pi_{\mathbf{B}_{(j)}}\right\|_F^2 \tag{4.8}$$

where $\Pi_{\mathbf{A}_{(j)}}$denotes a projector matrix of $\mathbf{A}_{(j)}$, defined as follows:

$$\Pi_{\mathbf{A}_{(j)}} = \mathbf{D}_{\mathbf{A},1}^{(j)}\mathbf{D}_{\mathbf{A},1}^{T(j)} \tag{4.9}$$

Inserting Eq. (4.9) into Eq. (4.8) yields

$$D_{ch}^2\left(\mathbf{A}_{(j)},\mathbf{B}_{(j)}\right) = \left\|\mathbf{D}_{\mathbf{A},1}^{(j)}\mathbf{D}_{\mathbf{A},1}^{T(j)} - \mathbf{D}_{\mathbf{B},1}^{(j)}\mathbf{D}_{\mathbf{B},1}^{T(j)}\right\|_F^2 \tag{4.10}$$

Based on D_{ch}^2 a tensor kernel can be defined as follows [249]:

$$\begin{aligned} &K_j\left(\mathcal{A},\mathcal{B}\right) = \exp\left(-\tfrac{1}{2\sigma^2}D_{ch}^2\left(\mathbf{A}_{(j)},\mathbf{B}_{(j)}\right)\right) \\ &= \exp\left(-\tfrac{1}{2\sigma^2}\left\|\mathbf{D}_{\mathbf{A},1}^{(j)}\mathbf{D}_{\mathbf{A},1}^{T(j)} - \mathbf{D}_{\mathbf{B},1}^{(j)}\mathbf{D}_{\mathbf{B},1}^{T(j)}\right\|_F^2\right). \end{aligned} \tag{4.11}$$

Thus, for a L-dimensional tensor a product kernel can be defined as follows

$$K(\mathcal{A},\mathcal{B}) = \prod_{j=1}^{L} K_j(\mathcal{A},\mathcal{B}) = \prod_{j=1}^{L} \exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{D}_{\mathbf{A},1}^{(j)}\mathbf{D}_{\mathbf{A},1}^{T(j)} - \mathbf{D}_{\mathbf{B},1}^{(j)}\mathbf{D}_{\mathbf{B},1}^{T(j)}\right\|_F^2\right) \tag{4.12}$$

Evidently, computation of Eq. (4.12) requires prior computations of $2{\cdot}L$ SVD decompositions, after which the Frobenius norm needs to be computed out of the kernel space matrices $\mathbf{D}$. However, in the case of large tensors this might require a prohibitive time of computations and the expression can be simplified, as will be discussed.

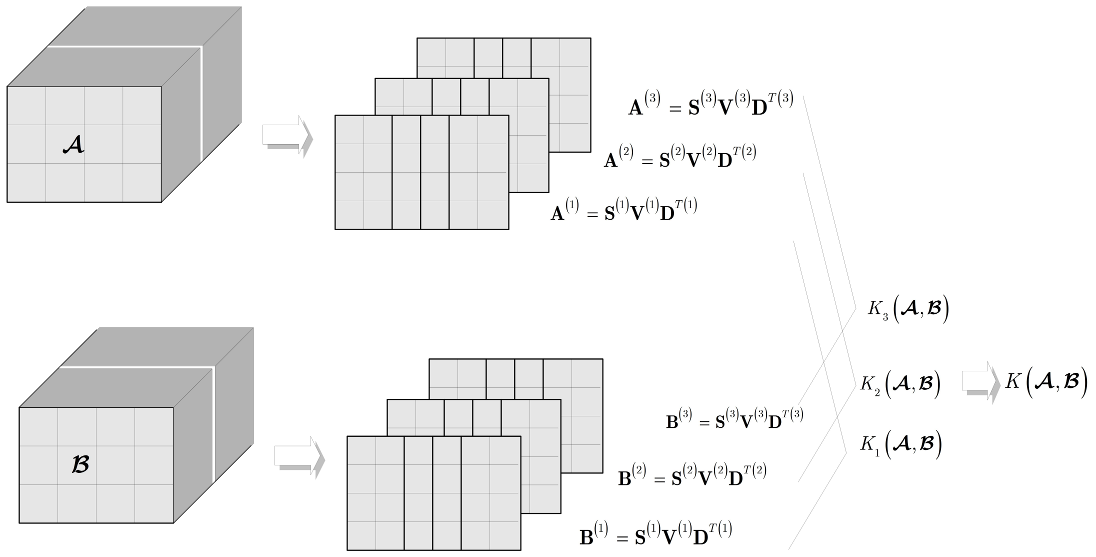

$\mathcal{A}$
$\mathbf{A}^{(3)} = \mathbf{S}^{(3)}\mathbf{V}^{(3)}\mathbf{D}^{T(3)}$
$\mathbf{A}^{(2)} = \mathbf{S}^{(2)}\mathbf{V}^{(2)}\mathbf{D}^{T(2)}$
$\mathbf{A}^{(1)} = \mathbf{S}^{(1)}\mathbf{V}^{(1)}\mathbf{D}^{T(1)}$
$K_3(\mathcal{A},\mathcal{B})$
$K_2(\mathcal{A},\mathcal{B})$
$K(\mathcal{A},\mathcal{B})$
$K_1(\mathcal{A},\mathcal{B})$
$\mathcal{B}$
$\mathbf{B}^{(3)} = \mathbf{S}^{(3)}\mathbf{V}^{(3)}\mathbf{D}^{T(3)}$
$\mathbf{B}^{(2)} = \mathbf{S}^{(2)}\mathbf{V}^{(2)}\mathbf{D}^{T(2)}$
$\mathbf{B}^{(1)} = \mathbf{S}^{(1)}\mathbf{V}^{(1)}\mathbf{D}^{T(1)}$

Figure 4.7: Visualization of the computation of the chordal distance between two tensors.

4.2.1.2 Computation of the Chordal Distance

Let us denote the squared norm in Eq. (4.10) as follows:

$$\|\mathbf{P}-\mathbf{Q}\|^2 = Tr\left(\mathbf{P}^T\mathbf{P}\right) - 2Tr\left(\mathbf{P}^T\mathbf{Q}\right) + Tr\left(\mathbf{Q}^T\mathbf{Q}\right) \tag{4.13}$$

where $Tr(.)$ denotes matrix traceand the two matrices $\mathbf{P}$ and $\mathbf{Q}$ in the above are defined as follows:

$$\mathbf{P} = \mathbf{D}_{\mathbf{A},1}^{(j)}\mathbf{D}_{\mathbf{A},1}^{T(j)}\ , \quad \mathbf{Q} = \mathbf{D}_{\mathbf{B},1}^{(j)}\mathbf{D}_{\mathbf{B},1}^{T(j)} \tag{4.14}$$

Let us notice that these two matrices $\mathbf{P}$ and $\mathbf{Q}$ are of the same size. Now, for the consecutive terms in Eq. (4.13), the following is obtained (we skip the indices 1 and 2 for simplicity)

$$Tr\left(\mathbf{P}^T\mathbf{P}\right) = Tr\left(\left(\mathbf{D_A}\mathbf{D_A}^T\right)^T\left(\mathbf{D_A}\mathbf{D_A}^T\right)\right) = Tr\left(\mathbf{D_A}\underbrace{\mathbf{D_A}^T\mathbf{D_A}}_{\mathbf{I}}\mathbf{D_A}^T\right) = \\ Tr\left(\mathbf{D_A}\mathbf{D_A}^T\right) = Tr\left(\mathbf{D_A}^T\mathbf{D_A}\right) = R_{\mathbf{A}}. \tag{4.15}$$

Similarly, it holds that

$$Tr\left(\mathbf{Q}^T\mathbf{Q}\right) = R_{\mathbf{B}} \tag{4.16}$$

On the other hand, the middle term in Eq. (4.13) can be expressed as follows:

$$Tr\left(\mathbf{P}^T\mathbf{Q}\right) = Tr\left(\left(\mathbf{D_A}\mathbf{D_A}^T\right)^T\mathbf{D_B}\mathbf{D_B}^T\right) = Tr\left(\mathbf{D_A}\mathbf{D_A}^T\mathbf{D_B}\mathbf{D_B}^T\right) = \\ Tr\left(\mathbf{D_A}^T\mathbf{D_B}\mathbf{D_B}^T\mathbf{D_A}\right) = Tr\left(\left(\underbrace{\mathbf{D_B}^T\mathbf{D_A}}_{\mathbf{G}}\right)^T\underbrace{\mathbf{D_B}^T\mathbf{D_A}}_{\mathbf{G}}\right) = Tr\left(\mathbf{G}^T\mathbf{G}\right). \tag{4.17}$$

Thus, Eq. (4.13) can be written as follows:

$$D_{ch}^2\left(\mathbf{A}_{(j)},\mathbf{B}_{(j)}\right) = R_{\mathbf{A}} + R_{\mathbf{B}} - 2\,Tr\left(\mathbf{G}_{(j)}^T\mathbf{G}_{(j)}\right) \tag{4.18}$$

where

$$\mathbf{G}_{(j)} = \mathbf{D}_{\mathbf{B},1}^{T(j)}\mathbf{D}_{\mathbf{A},1}^{(j)} \tag{4.19}$$

Expressions from Eq. (4.18) and Eq. (4.19) are easier for computation than Eq. (4.10) since only the matrix $\mathbf{G}_{(j)}$ needs to be computed after computation of the SVD decompositions of j-th mode flattened versions $\mathbf{A}_{(j)}$ and $\mathbf{B}_{(j)}$ of the tensors $\mathcal{A}$ and $\mathcal{B}$, respectively. For the computation of the chordal kernel distance, such computations need to be repeated L times, which is a dimensionality of the two tensors.

In practice, choice of the numbers of the non-zero singular values, denoted by R_A and R_B in the matrices $\mathbf{V}_A$ as well as $\mathbf{V}_B$, respectively, is based on experimental results with particular datasets or some heuristic methods. This is an analogous problem to determining subspace dimensionality for the PCA decomposition. For the latter, many methods were proposed in literature. An overview of recent approaches can be found e.g. in [56]. In our experiments a simple threshold value was used, i.e., all singular values falling below this threshold are assumed to be 0. Thus, for each $\mathbf{A}_{(j)}$, its $R_{\mathbf{A}(j)}$ is a number of singular values above the experimentally chosen threshold. A similar strategy is used for $\mathbf{B}_{(j)}$.

When comparing computations of D_{ch} with Eq. (4.10) and Eq. (4.18) we see a significant difference. The first is that the matrix $\mathbf{G}_{(j)}$ is of dimensions $R_{\mathbf{B}(j)}$ x $R_{\mathbf{A}(j)}$, whereas each product in the difference in (10) is of dimensions N_j x N_j which is always larger. The second benefit of using Eq. (4.18) is that Tr can be directly computed from the matrix $\mathbf{G}$ given in Eq. (4.19) with no further matrix multiplications.

For a matrix of dimensions r x c, the computational complexity of determining only the matrices $\mathbf{V}$ and $\mathbf{D}$ from the SVD decomposition is of order $4c^2(r + 2c)$ [55]. This can seem prohibitive for large tensors. However, in many pattern recognition tasks, such as computation of the nearest neighbors, SVD for the prototype patterns can be computed beforehand and stored in a database. This greatly simplifies computations since once a test pattern is SVD decomposed. Then only Eq. (4.18) needs to be determined which requires one matrix multiplication given in Eq. (4.19), as well as one inner product to determine the third term in Eq. (4.18).

4.2.2 Proposed One-Class Ensemble Approach

We propose to embed the chordal distance-based kernel directly into the WOCSVM learning procedure. This way that classifier will be able to efficiently process multi-dimensional data, while preserving their spatial properties and relations. Additionally, one should notice that this procedure changes the behavior of WOCSVM. Originally it calculated weights assigned to each object. Here, it calculates weights assigned to tensors, thus being able to measure the importance of a given tensor for the pattern classification procedure.

To further boost the quality of the proposed tensor WOCSVM (T-WOCSVM), we propose to combine it with One-Class Weighted Bagging approach (described in Section 2.4). This way we are able to assure better diversity for soft tensor-based one-class classifiers and reduce the computational complexity of the method (as standard approach for weight calculation would now be based on tensor distance and be connected with very long processing time).

As OC-Wagg does not guarantee to output an optimal collection of classifiers, we will use it in a overproduce-and-select mode. We will generate higher number of one-class classifiers and then select the best subset from them. For this task, we apply X-means clustering-based pruning (see Section 3.3).

4.2.3 Experimental Analysis and Discussion

We have two aims of the experimental study:

- To asses the quality of tensor-based WOCSVM for multi-dimensional hyperspectral image analysis.
- To see if combining tensor-based one-class classifiers using weighted baggingand then pruning them with clustering-based method can benefit to the considered problem.

For our experiments we use a real-life Pavia University dataset [7], which is a scene acquired by the ROSIS sensor during a flight campaign over Pavia, Northern Italy. The number of spectral bands is 103 for Pavia University. Pavia University is 610 x 610 pixels image. The geometric resolution is 1.3 meters. Image ground truth differentiate 9 classes. Pavia scenes were provided by Prof. Paolo Gamba from the Telecommunications and Remote Sensing Laboratory , Pavia university (Italy). Details of this dataset are given in Table 4.2, while Pavia University ground truth picture in Figure 4.8. Two sample bands from Pavia University dataset are given in Figure 4.9.

We define the classification problem as detection of asphalt in the image. Asphalt class is responsible for roadsand road monitoring is one of the big challenges for remote sensing methods. By being able to precisely detect roads, we can perform an object recognition and tracking procedure, e.g., for traffic monitoring, surveillance or recognition of certain vehicles.

Our OC-Wagg uses WOCSVM as a base classifier with RBF kernel, $\sigma = 0.1$and $C = 5$. Ensemble consist of 30 classifiers. A chordal distance-based kernel is embedded within this classifiers. X-means pruning selects the classifier that lies closest to the centroid of

Table 4.2: Ground truth classes for the Pavia University scene and their respective samples number.

	Class	Samples
1	Asphalt	6631
2	Meadows	18649
3	Gravel	2099
4	Trees	3064
5	Painted metal sheets	1345
6	Bare Soil	5029
7	Bitumen	1330
8	Self-Blocking Bricks	3682
9	Shadows	947

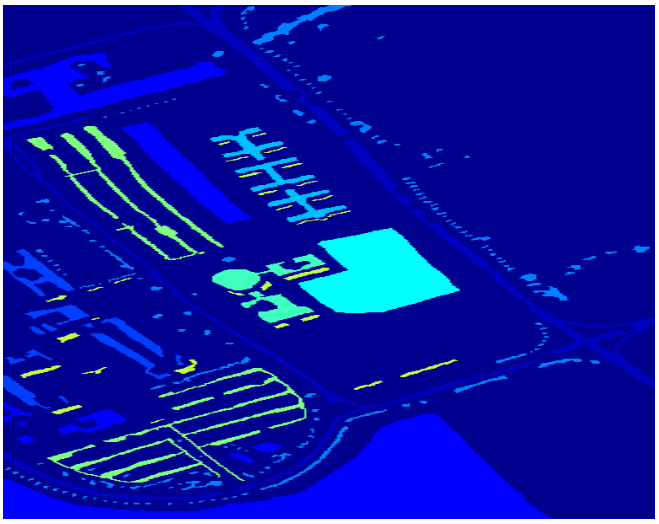


Figure 4.8: Pavia University ground truth.

the considered cluster. We have compared this method with single WOCSVM working on raw pixels,WOCSVM with PCA dimensionality reduction (50 components are used), single tensor-based WOCSVMand OC-Wagg without the pruning procedure. Details of these methods and their parameters are given in Appendix C. The G-mean results from a 5x2 CV for analyzed methods are given in Figure 4.10 and the results of 5x2 combined F-test are given in Table 4.3.

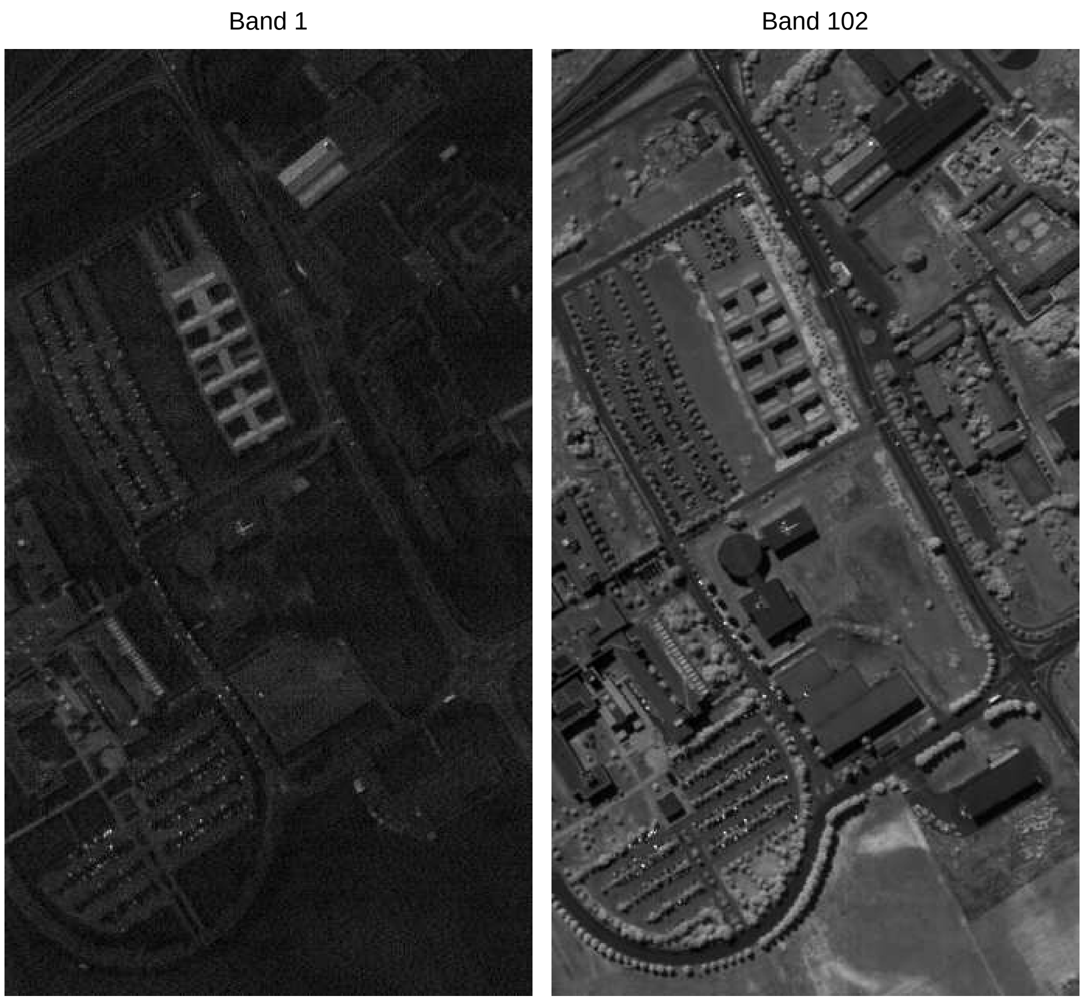


Figure 4.9: Two sample bands from Pavia University dataset: (*left*) band 1; (*right*) band 102.

From the results, we can see that with the usage of proposed tensor-based ensemble of one-class classifiers a highly accurate detection of asphalt from hyperspectral images is possible. This shows a high potential of one-class classifier ensembles with chordal distance kernel to be applied in real-life remote sensing and object tracking applications.

Using canonical WOCSVM is not efficient for hyperspectral data processing. It works on a multi-dimensional objects, without considering spatial relations between them. Training a one-class classifier on such a highly dimensional input space leads to a significant increase in computational complexity of the proposed systemand may become prohibitive if we require an adaptive and on-line learning.

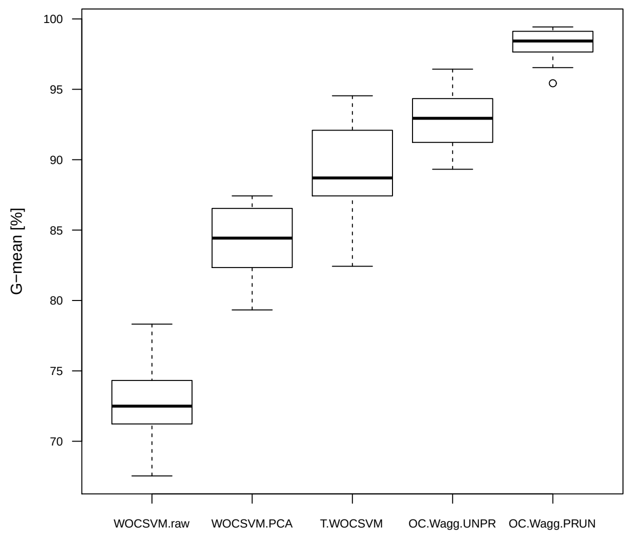


Figure 4.10: G-mean values for pruned tensor-based OC-Wagg and reference methods for Pavia University dataset.

Table 4.3: 5x2 combined F-test for comparison between the pruned tensor-based OC-Wagg and reference methods over the analyzed hyperspectral dataset. Symbol '+' stands for a situation in which the proposed method is superior, '-' for vice versaand '=' represents a lack of statistically significant differences.

hypothesis	p-value
OC-Wagg$_{pruned}$ vs WOCSVM$_{raw}$	+ (0.0057)
OC-Wagg$_{pruned}$ vs WOCSVM$_{PCA}$	+ (0.0169)
OC-Wagg$_{pruned}$ vs T-WOCSVM	+ (0.0308)
OC-Wagg$_{pruned}$ vs OC-Wagg$_{unpruned}$	+ (0.0362)

Comparing a PCA feature extraction (a popular method for handling multi-dimensional images) with chordal distance-based tensor kernel, we may observe that the proposed tensor approach is able to significantly boost the recognition accuracy by offering a more efficient representation of complex hyperspectral data.

The proposed T-WOCSVM can be further improved by creating an ensemble of local models. We improve the diversity within the pool with the usage of OC-Wagg approach. As this is a randomized approach, it cannot guarantee that all of outputted classifiers will be of the same quality. Embedding a pruning step allows us to reduce the size of the ensemble, while improving its robustness to outliers and discriminative power. We can conclude that a pruned ensemble of tensor one-class classifiers is a highly efficient tool for the analyzed problem.

Finally, we must report that the proposed T-WOCSVM version requires a longer training time than its PCA-based counterpart. However, for most of the classification task this is not a problem, as we may train the classifier beforehand without any time constraints. In such scenarios, the gain in accuracy is far more important than increase of the training time. We should note that the response times of both RBF-based and chordal distance-based WOCSVMs are almost identical - so it is suitable for real-time operation. For cases, in which one would require a re-trainable or adaptive classifier (e.g., data streams with concept drift), one of our future goals is to introduce a distributed version of our WOCSVM classifier suitable to be run on CUDA architecture.

4.3 Breast Cancer Malignancy Grading

Based on the data provided by the National Cancer Registry, there was 16534 diagnosed cases of breast cancer in Poland[1]. This statistics makes the breast cancer most often diagnosed type of cancer among middle–age women. The number of diagnosed cases is still increasing. From 2009 to 2011 there was an increase of 782 cases. Unfortunately this also suggests a large death rate, which was recorded to be 5437 deaths in 2011 and was larger than in 2009 by 195 cases. Most of these cases could be fully recovered if the diagnosis would be made in the early stage of the disease. This is because cancers in their early stages are vulnerable to treatment while cancers in their most advanced stages are usually almost impossible to treat. During the diagnosis process, the cancer is assigned a grade that is used to determine the appropriate treatment. Successful treatment is a key to reduce the high death rate. For this purpose a screening mammographic tests are performed and when a suspicious region is found the fine needle aspiration biopsy (FNA) is taken. This is an invasive method to extract a small sample of the questionable breast tissue that allows the pathologist to describe the type of the cancer in detail. Malignancy grading allows doctors to precisely estimate cancer behavior with or without undertaking treatment and therefore is called a prognostic factor. It plays

[1]National Cancer Registry, http://85.128.14.124/krn/

an important role in breast cancer diagnosis and the appropriate treatment is chosen accordingly to this factor.

The determination of malignancy is performed by assigning a malignancy grade to the case. To help in this very difficult task, a grading scale was proposed by Bloom and Richardson in 1957 [28]. The grading scheme proposed by the authors was derived to assess malignancy from histological slides and is now widely used among pathologists to grade not only histological but also cytological tissue.

In this section, we discuss the application of one-class ensembles and image processing methods to extract the information from the FNA slides and automatically assign a malignancy grade [149, 151]. Automatic detection of pathologies from cytological images is currently very active and important area of research [79, 150, 152, 153, 154, 156] and the automatic cancer grading is a very challenging task due to large variation in cancer imaging and analysis. As our dataset is imbalanced, there is a need for an effective classification system that can handle difficulties embedded in the analyzed objects. For this purpose, we propose to apply our developed OCClustE ensemble locally specialized on the majority class to achieve a very high accuracy on both minority and majority classes, while outperforming several state-of-the-art methods.

This research was done in cooperation with dr Ł ukasz Jeleń from Faculty of Electronics, Wroclaw University of Technology, Poland, who contributed with image processing and feature extraction steps.

4.3.1 Medical Background

Breast cancer malignancy grading is an integral procedure in the diagnosis process of this disease. It not only allows for the determination of the treatment, but also for the prediction if the undertaken treatment is going to be successful. The malignancy grading is performed according to the well defined scheme called the Bloom–Richardson grading system. The Bloom–Richardson grading system (BR) was originally proposed by Bloom and Richardson [28], later modified by Scarff and known as modified Scarff–Bloom–Richardson system, for grading breast cancer malignancy is one of the best known prognostic factors for this type of cancer. These systems are based on grading of cells' polymorphy, ability to reform histoformative structuresand mitotic index. All of these features are described by the Bloom-Richardson scheme as three factors that use a point based scale for assessing each feature according to the following description:

1. **Degree of structural differentiation (SD)** - In histopathological slides this is also described as tubule formation, which reflects cell tendency to form tubules. Since in cytological smears tubules are not preserved, the scoring given below for this factor is based on the classification of cell groupings within a smear, see Figure 4.11 for example. In the fig. 4.11a only one group is visible, which indicates lower malignancy than the case in the fig. 4.11b where dispersed cells are visible.

    - one point - cells are grouped and spread regularly.
    - two points - both grouped and single cells found.
    - three points - cells are spread irregularly.

2. **Pleomorphism (P)** - This factor takes into consideration differences in size, shape and staining of the nuclei. This scoring is fairly straightforward because with the growth of irregularity of the nuclei the prognosis becomes worse. Figure 4.12 shows an example of these variations. Arrows in the image indicate cells with visible variations in shape and color.

    - one point - nuclei with uniform size, shape and staining.
    - two points - moderate variations are found.
    - three points - very significant variations (see Figure 4.12).

3. **Frequency of hyperchromatic and mitotic figures (HMF)** - Mitosis is a process in the cell life cycle in which a mother cell divides into two identical cells. Main objective of this factor is to assess the number of mitosis in the field of view. Several fields of view on the same slide are taken into account because this step is done in a large magnification. The more cases of mitosis found, the worse the prognosis is. During the staining process, mitotic cells stain the most intensively providing the darkest areas in the nucleus as shown in Figure 4.13.

    - one point - occasional figures per field are found.
    - two points - smears with two or three figures in most fields.
    - three points - more than three figures per fields are found.

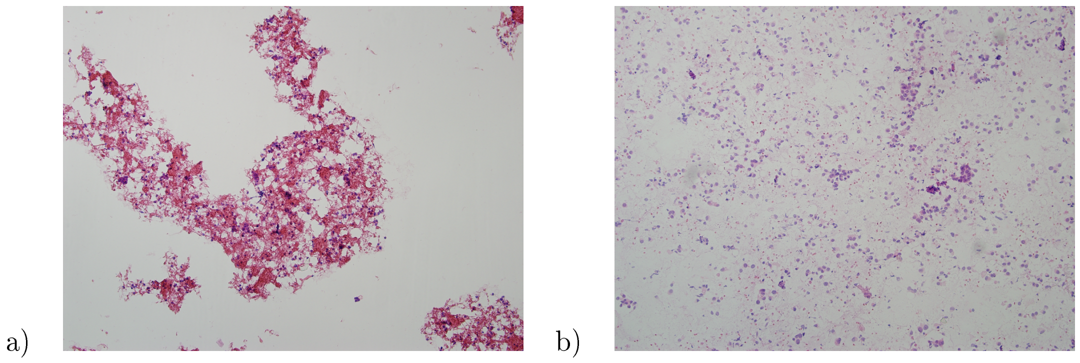


Figure 4.11: Illustration of the Structural differentiation. a) Intermediate malignancy case b) High malignancy case.

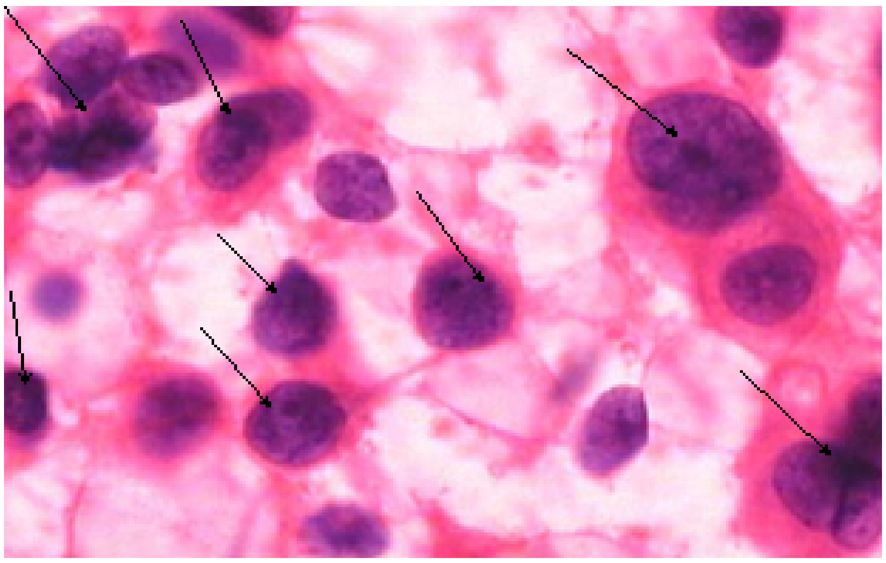


Figure 4.12: Illustration of the Pleomorphism feature.

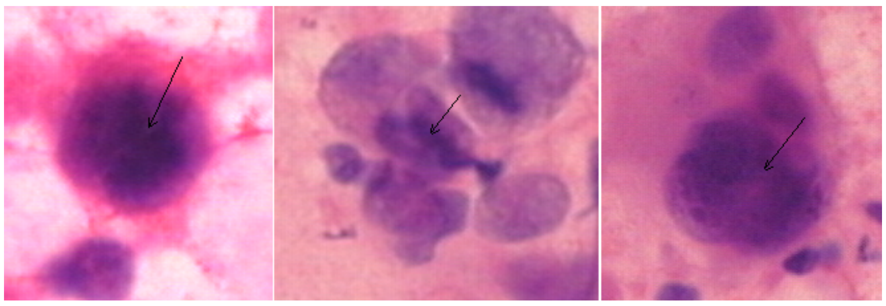


Figure 4.13: Illustration of the mitosis.

$$\overbrace{\underbrace{3\ 4\ 5}_{GradeI}\qquad\quad \underbrace{6\ 7}_{GradeII}\qquad\quad \underbrace{8\ 9}_{GradeIII}}^{Points}$$

$$\begin{array}{ccc} Low & Intermediate & High \\ malignancy & malignancy & malignancy \end{array}$$

Figure 4.14: Grade determination for the Bloom – Richardson scheme.

According to the BR scheme, the malignancy of the tumor is assigned a grade that depends on the quantitative values of the above factors and is determined by the following equation:

$$G = SD + P + HMF. \tag{4.20}$$

The final grade is obtained by the summation of all the awarded points for each factor described earlier. Depending on the value of G, the tumor is assigned one of three grades according to the chart shown in Figure 4.14.

Assigning a malignancy to a case is a very difficult task and is dependent on the experience of the pathologist. More experienced pathologists that have seen more cases are more reliable in their diagnosis. On the other hand, due to overwork and fatigue, seeing more similar cases may lead to misclassification of the malignancy. To address this problem we present an automated grading approach that is able to evaluate and assign a grade to Fine Needle aspiration biopsy (FNA) tissue. To achieve this we convert the Bloom–Richardson grading scheme into a classification problem.

4.3.2 Image Segmentation and Feature Extraction

Segmentation is a task where the object of interest are localized and extracted from the image. In our research we have divided this task according to the magnification of the image. Due to the fact that features are calculated separately for the low and high magnification images, we have applied different techniques for their segmentation. The task of low magnification image segmentation requires to extract groups of cells and therefore the segmentation procedure was reduced only to the thresholding of the image. The threshold level was automatically determined with the algorithm described by Riddler and Calvard [234]. This methods determines the threshold T based on the

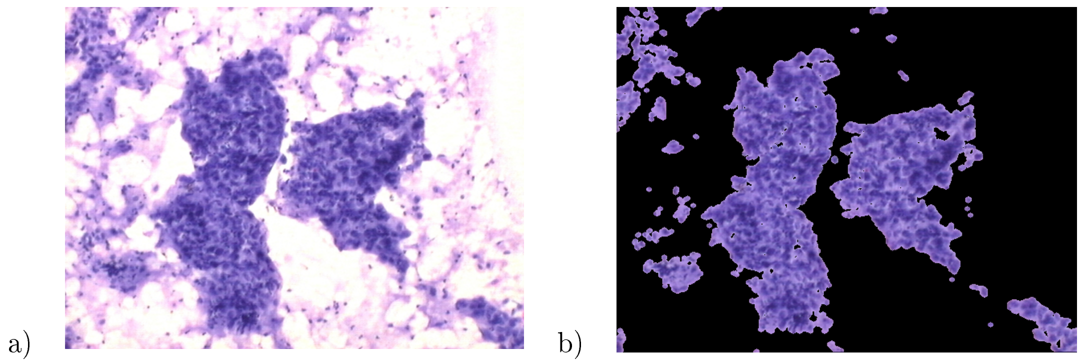


Figure 4.15: Segmentation of the 100x magnification case. a) Original image b) Segmented image.

bimodal histogram of the image based on Eq. 4.21.

$$T = \frac{\mu_1 + \mu_2}{2}, \tag{4.21}$$

where μ_1 and μ_2 are the means of the components separated by T.

In Figure 4.15 a segmentation of the 100x magnification case is shown. From the image one can see that the method described here is suitable for segmentation of cells groupings. In this study we have used image red channel for thresholding and for further feature extraction. This is because during the staining process of FNA images, nuclei stains with shades purple and when red channel is extracted all the nuclear features are preserved while the background information is lost, therefore providing the best information about nuclear structures out of the three RGB channels.

The high magnification images are used to extract features of cell' polimorphy and therefore require more sophisticated method for the representation of the shape of the nucleus. To be able to precisely represent the nuclei we have adopted an active contour technique. Active contours are able to represent the boundary well because they change their shape according to the information in the image which makes them a very good choice in biomedical applications. For this purpose, in this study, we have applied a level set method described by Li *et al.* [195].

Historically, level sets were first described in 1988 by Osher and Sethian [219] as a method for capturing moving fronts. In the level set formulation, the segmentation problem is equivalent to the computation of a surface $\Lambda(t)$ that propagates in time along its normal direction. The Λ surface is also called a propagating front, which according to authors [219] is embedded as a zero level of a time–varying higher dimensional function $\phi(x,t)$. An evolution equation for ϕ, from which we can determine Λ, where Λ is a closed curve in R^2, can be written in a general form as:

$$\frac{\partial \phi}{\partial t} + F|\nabla \phi| = 0, \tag{4.22}$$

The function ϕ describes a curve defined by $\phi(x,t) = d$, where d is a signed distance between x and the surface $\Lambda(t)$. If x is inside (outside) of $\Lambda(t)$ then d is negative (positive). Function F is a scalar speed function that depends on image data and the function ϕ.

The main drawback of this procedure is that during the evolution, ϕ can assume sharp or flat shapes. To overcome this problem ϕ is initialized as a signed distance function before evolution. Later, during evolution, it is periodically reshaped to be a signed distance function. We us a modification of the traditional variational level sets to overcome the problem of reshaping function ϕ to be a distance function within the evolution cycle. We use an evolution equation of the form:

$$\frac{\partial \phi}{\partial t} = -\frac{\partial \mathcal{E}}{\partial \phi} \tag{4.23}$$

where $\frac{\partial \mathcal{E}}{\partial \phi}$ is a Gateaux derivative of the energy function $\mathcal{E}$ and is represented by:

$$\frac{\partial \mathcal{E}}{\partial \phi} = -\mu[\Delta\phi - div(\frac{\nabla\phi}{|\nabla\phi|})] - \lambda\delta(\phi)div(g\frac{\nabla\phi}{|\nabla\phi|}) - \nu g\delta(\phi), \tag{4.24}$$

where Δ is the Laplacian operator, div is the divergence operator, $\mu > 0$ is a parameter controlling the effect of penalizing the deviation of ϕ from a signed distance function, g is an edge indicator function, $\lambda > 0$ and ν are constants.

An example of the segmented image of the nuclei is shown in the Figure 4.16

Feature extraction is an important part of each classification task. Poor definition of features can lead to a high error rate of a classification system. Each classification system takes a feature vector as an input and responds with a category to which the object belongs. A feature vector is a set of features extracted from the input data.

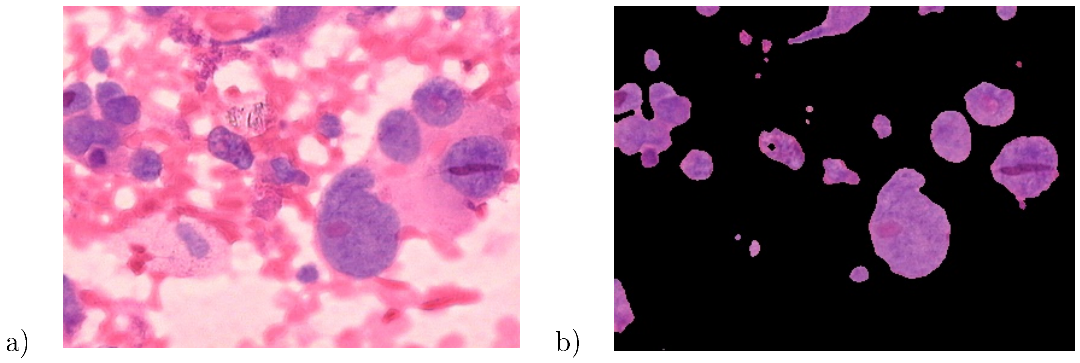


Figure 4.16: Segmentation of the 400x magnification case. a) Original image b) Segmented image with nuclear contour.

Based on these segmentations a 32 element feature vector was constructed that consisted of two types of features depending on the type of the image. In this study we extracted the following features:

1. **a) 100x magnification images**
    - area of the groups, number of groups, dispersion
2. **b) 400x magnification images**
    - binary features:
        - area of the nucleus, nucleus perimeter, convexity, eccentricity, coordinates of the nucleus centroid called x–centroid and y–centroid respectively, nucleus orientation, horizontal and vertical projections,
    - 7 momentum-based features:
        - $\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6, \varphi_7$
    - 5 histogram based features:
        - mean, standard deviation, skew, energy and entropy
    - 2 textural features calculated from the gray level co–occurrence matrix:
        - inverse difference and correlation
    - 5 color–based features calculated from the red channel histogram:
        - mean, standard deviation, skew, energy and width
    - average gray level

4.3.3 Imbalanced Learning with One-Class Classifiers

Canonical machine learning algorithms assume that the distribution of objects between the classes in the training set is roughly equal. However, in many real-life applications it is impossible to gather equal samples of objects from all classes, as some may appear less frequently or be more costly to gather. In case where one of the classes is represented by a significantly greater number of examples than other, we deal with a problem known as the imbalanced classification. Such an uneven distribution tends to result in a bias of the decision boundary produced by classifiers towards the majority class. This leads to a significant drop in the minority class recognition rate. Therefore, there is a need for using dedicated algorithms that can alleviate this difficult data distribution [177].

Recent reports state that the disproportion in the number of examples is not the major source of problem [258]. In case, when the classes are imbalanced, but the minority class is well-represented by a large sample, even standard algorithms can achieve good recognition rate [47]. The true difficulty lies in additional data properties that often accompany imbalanced datasets:

- **Class overlapping** occurs, when part or entire distribution of the minority class is located within the majority one. As typical discriminative rules are constructed in such a way as to minimize the number of misclassified instances, this may lead to poor performance for minority objects in the overlap area [91].

- **Small sample size** is a problem connected with the difficulty of obtaining minority class samples. In this situation, during the training phase we have a limited access to minority objects and this concept is not properly represented. Therefore, the number of examples is insufficient for a classifier to be properly trained. This results in the overfitting for minority class and poor generalization abilities of the newly created predictor.

- **Small disjuncts** is a problem connected to the previous one. It often happens that minority class objects do not form an uniform structure. Instead, they form a number of subconcepts. This means that minority examples are distributed among several chunks of data. This causes significant problems for classifiers and leads to an overly complex decision boundary, due to the lack of compact structure and small number of objects in each subconcept.

These data properties are the major reason why canonical machine learning algorithms cannot be effectively applied for imbalanced datasets.

There is a number of reports that discuss the usefulness of OCC approach in cases, where objects from all of classes are at disposal [155, 175]. This is explained by several attractive properties of OCC, resulting from their different learning procedure. They do not minimize the classification error, but adapt to the features of the target class. They do not use all of the available knowledge from the training set, which results in worse recognition accuracy than multi-class methods for standard problems. However, in case of difficult datasets, OCC can outperform multi-class algorithms, as single-class classifiers are robust to many difficulties embedded in the nature of data.

This observation about the specific behavior of OCC is appealing in the context of imbalanced learning. As most of the problems in this field are caused by single class that often has a complex distribution, OCC seems to be an attractive solution. Let us discuss shortly the idea of imbalanced learning with one-class classifiers.

As it was mentioned earlier, OCC is designed for cases with a lack of access to counterexamples. We may define the task in a more broad perspective. Let us assume that in some cases the available counterexamples do not sufficiently capture the properties of the represented class. This happens quite often in real-life applications, where we cannot influence the source of data in any way. Therefore, we do not have any control over the quality of data in the training set. In such cases, the counterexamples may misdirect the classifier and lead to a creation of wrong decision boundary. In such cases, OCC allows to focus on relevant class, without taking into consideration the quality of counterexamples. Let us extend this concept on the imbalanced classification. Here, minority class is underrepresented in comparison with the majority class and various difficulties, embedded in the nature of the data, limit the relevance of minority samples. In such case, we may transform the binary imbalanced problem into an OCC task.

The intuitive approach for transforming an imbalanced problem into one-class task is to use the majority class as target concept and train an one-class classifier with the use of the majority objects. Consequently, minority class examples are considered as outliers and are not used during the classifier training phase. By this, we discard information about the minority class and form the decision boundary on the basis of majority samples. This is an useful approach in case of unrepresentative objects from minority class. One-class classifier adjust itself to the properties of majority class and can capture its structure. This allows to have a good majority class detection rate without sacrificing the generalization abilities. However, the main attention should be paid to the minority class. When OCC is properly trained on the majority class, it can detect objects that deviate from the target distribution. By using only majority samples for training, OCC alleviates the classification bias that penalizes minority samples. OCC approach can handle most of the difficulties that accompany imbalanced datasets. Minority class objects will be recognized as outliers, even if they are underrepresented, distributed in chunks or small disjuncts. Experimental analysis carried so far prove that this approach is highly competitive to other methods, dedicated to imbalanced data. We should note that using majority class as the target concept in OCC may result in some problems with overlapping minority examples.

4.3.4 Experimental Analysis and Discussion

We have two aims of the experimental study:

- To propose an efficient and fully automatic system for breast cancer malignancy grading.
- To asses the quality of one-class ensembles in a binary imbalanced scenario.

For the purpose of this study we have collected a database of fine neelde aspirates that where used for the breast cancer diagnosis. Fine Needle Aspirates of the breast tissue are a courtesy of prof. Michał Jeleń from the Department of Pathology and Clinical Cytology, Medical University of Wrocław, Poland. All of the slides were stained with the Haematoxylin and Eosin technique (HE) which yielded purple and black stain for nuclei, shades of pink for cytoplasm and orange/red for red blood cells. These slides where digitalized with Olympus BX 50 microscope with mounted CCD–IRIS camera connected to a PC computer with MultiScan Base 08.98 software at the Department of Pathology and Clinical Cytology, Medical University of Wrocław, Poland. The resolution of the recorded images was 96 dots per inch (dpi) and a size of 764x572 pixels.

The database is constantly growing and at as of today it consist of 341 images. There are two types of images each recorded at different magnification (see Figure 4.17). Images recorded in low magnification (100x) are used to define features related to the

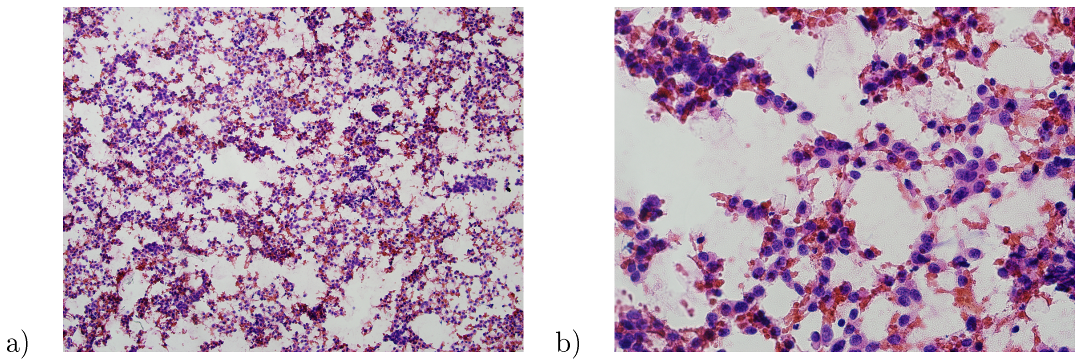


Figure 4.17: Example of a case images in the database. a) Low magnification. b) High magnification.

degree of structural differentiation and images recorded in high magnification (400x) are used to calculate the features that reflect cell' polimorphy and mitotic count.

There are 167 low magnification images and 174 images for high magnification. The uneven number of slides is caused by the need of having two or three high magnification images for one low magnification image. From the diagnostic point of view this is caused by the fact that there are more that one suspicious region in the 100x image. In this study we have treated these cases separately. The images in the database can also be divided depending on the malignancy grade they represent. In this case, we have collected slides of intermediate (G2) and high (G3) malignancy grades. There are 268 images belonging to the G2 class and 73 to the G3 class. The number of images for each malignancy grade shows us the tendency of occurrence of each class. This unbalanced number of cases makes the classification scheme more difficult and was a motivation to perform these studies.

As our base classifier, we apply OCClustE method (see Section 2.1 for details). In this experiment the majority class was used as the target class for OCC.

To put the obtained results into a context, we compare our method with several state-of-the-art algorithms dedicated to binary and imbalanced classification [85]. The list of used models is given in Table 4.4, while their parameters are given in Appendix C. The parameter values were established through a grid-search procedure.

Firstly, we need to establish the proper number of competence areas for our OC-ClustE ensemble. For this, we applied Akaike Information Criterion (see Section 2.1.5 for details). The results of AIC criterion averaged over 5x2 CV are presented in Figure 4.18. The G-mean results from a 5x2 CV for analyzed methods are given in Figure 4.19 and results of 5x2 combined F-test are given in Table 4.5.

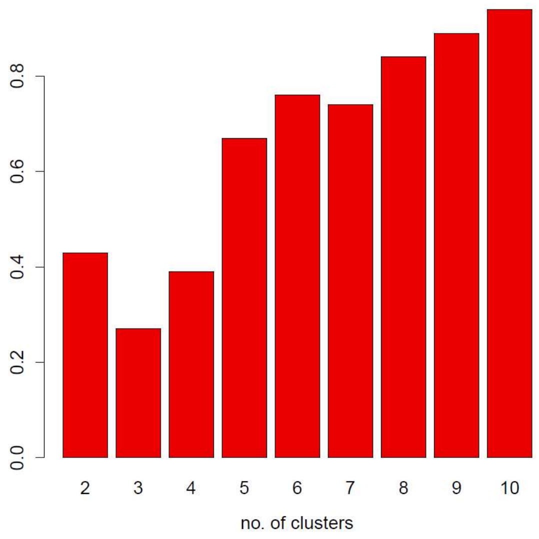


Figure 4.18: Mean AIC values over analyzed breast malignancy dataset for fuzzy *c*-means with the number of clusters $\in$ [2;10].

Out of a plethora of approaches, we have selected five ensemble methods, dedicated for imbalanced learning. Our choice was dictated by recent survey on imbalanced classification [85], in which the selected ensembles have achieved best performance over a varied set of benchmarks. They represent all major approaches to imbalanced learning: undersampling, artificial objects oversampling and classifier-level approaches. Out of these methods, Roughly Balanced Bagging achieved the best performance both in terms of sensitivity and specificity. This proves the claims from previous work on undersampling ensembles that they can outperform all other methods using relatively simple and easy to implement technique. To our surprise, EasyEnsemble and SMOTEBagging returned worst performance from all of ensemble methods. This is contrary to some literature findings, which reported high quality of these methods.

The first thing to do was to establish the number of clusters (classifiers) in the ensemble. This was done with the usage of automatic Akaike Information Criterion. It was run separately for each fold of CVand mean results are presented in Figure 4.19. From it, we can see that the minimal entropy was returned for 3 clusters. This shows that our ensemble does not need a big number of base classifiers to achieve good recog-

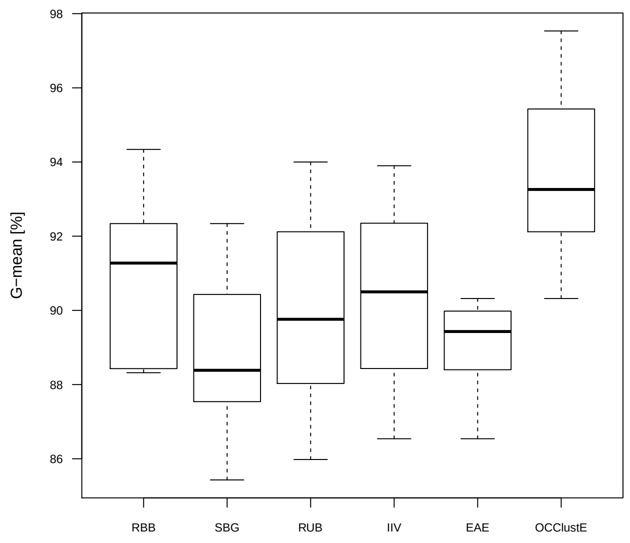


Figure 4.19: G-mean values for OCClustE and reference methods for imbalanced classification of breast cancer malignancy.

nition accuracy. As these 3 sub-concepts were detected, delegating a locally specialized one-class classifier to each of them resulted in good generalization abilities over both classes.

Our proposed ensemble outperformed all other reference methods, achieving excellent sensitivity while managing high specificity. This is very important, as many methods sacrifice specificity for minority class recognition. As we are dealing with breast cancer malignancy grading, we need to have high accuracy on both of the considered classes. This prove claims that OCC is a valuable tool for problems with uneven distributions. Additionally, OCClustE was able to efficiently detect local areas of competence and train compact, diverse and competent classifiers on their basis.

Table 4.4: Short description of classification models used in experiments.

Abbr.	**Model**	**Short Description**
		Ensembles for Imbalanced Problems
RBB	Roughly Balanced Bagging [118]	Bagging with undersampling of the majority class, number of negative examples varies in each bag
SBG	SMOTEBagging [284]	Bagging with varied SMOTE quantity in each bag
RUB	RUSBoost [245]	AdaBoost.M2 with random undersampling in each iteration
IIV	IIVotes [26]	IVotes with SPIDER (strong) oversampling in each iteration
EAE	EasyEnsemble [203]	UnderBagging learned with AdaBoost in each bag
		One-Class Ensemble
OCE	One-Class Clustering-Based Ensemble	Uses WOCSVM and kernel k-means

Table 4.5: 5x2 combined F-test for comparison between the OCClustE and reference methods over the analyzed breast cancer malignancy dataset. Symbol '+' stands for a situation in which the proposed method is superior, '-' for vice versaand '=' represents a lack of statistically significant differences.

hypothesis	p-value
OCClustE vs RBB	+ (0.0317)
OCClustE vs SBG	+ (0.0116)
OCClustE vs RUB	+ (0.0218)
OCClustE vs IIV	+ (0.0242)
OCClustE vs EAE	+ (0.0201)

5

Conclusions and Future Works

This thesis has focused on the topic of one-class classification, a specific area of machine learning where one does not have an access to counterexamples during the classifier training step. Therefore one is forced to estimate such a decision boundary that will encompass all of the relevant target class samples with the assumption that some new, unseen outliers may appear in the exploitation phase. Two key factors to a good performance of one-class classifiers are generalization ability over the target class and robustness to outliers. There is a need for novel one-class approaches, as massive, multidimensional and complex data may prove challenging to canonical one-class classifiers.

This thesis had explored using ensemble learning principles for one-class classification. Combining classifiers often lead to a significantly better classification model and thus seemed as a useful approach for OCC problems. One could not directly use most of the standard multi-class methods for forming one-class ensembles due to the lack of examples from any other class than the target one. As most of canonical methods relied on accuracy or diversity of the classifiers output, one cannot use them when no counterexamples are known during the training phase. So far there were several proposals for designing ensembles for one-class classification but majority of them were focused on applications, not on proposing universal and flexible frameworks for ensemble learning with one-class classifiers.

This thesis has aimed at presenting a set of universal algorithms for designing efficient combined one-class classifiers and tackled the two important issues in one-class committee design:

- How to form a pool of one-class classifiers in such a way that base learners will have high individual quality and be complimentary to each other.
- How to design a pruning mechanisms for discarding irrelevant classifiers when one has only access to objects from a single class.

For the one-class ensemble forming step we have identified several drawbacks of currently used methods, like lack of preservation of spatial relations within data, high variance in quality of outputted classifiers, high chance of overfitting or low adaptability to complex and multi-dimensional data. In order to overcome these limitations four new ensemble models were introduced:

- One-Class Clustering-Based Ensemble (OCClustE), which used a fuzzy kernel clustering to detect atomic competence areas and train on their basis locally specialized one-class classifiers. OCClustE is a very flexible framework that put almost no limitations on the user regarding how to select components for it. We have examined a number of methods dedicated to automatic detection of the number of competence areas for a given dataset. We have presented an efficient mechanism for extracting importance level associated to each object from the clustering step and using it to initialize soft one-class classifiers. We have presented its usefulness for dealing with both one-class and multi-class problems.
- One-Class Evolutionary Clustering Ensemble (OC-EvoClust), which utilized an evolutionary algorithm to change the shape of competence areas in order to better comply with the nature of one-class classification. We have presented a compound and hybrid training procedure that enclosed cluster reshaping, feature selection and weighted fusion into one optimization task. Additionally, this training method is able to automatically establish a proper number of clusters by incremental addition of new ones during training. We have shown how to use a multi-criteria optimization in order to maximize both the individual stability and diversity of base classifiers.
- One-Class Rotation Forest (OC-RotF), which used a space rotation via given pre-processing method in order to improve the diversity of base classifiers. This method is a direct counterpart to the recently proposed One-Class Random Forest

(OC-RandF). OC-RandF was based on artificial generation of outliers and using binary trees - which is not in accordance with the principles of OCC. Thus, our proposal aims at lifting these drawbacks. OC-RotF can work with any type of one-class classifier and does not require artificial counterexamples, as the rotation is done with unsupervised feature extraction methods. We have shown that OC-RotF often outperforms OC-RandF, while being a native one-class method.

- One-Class Weighted Bagging (OC-Wagg), which used a random distribution of weights assigned to the training objects. It can be seen as an alternative to Bagging, as it draws a random importance level for each object. These weights can be directly used for soft one-class classifiers (like WOCSVM) in order for training a diverse ensemble with low computational cost. Additionally, OC-Wagg is highly suitable for small datasets, as it always uses a full training set for each classifier while still being able to introduce significant diversity among them.

For the one-class ensemble pruning step there was little work done so far. This originated from the difficulty on how to evaluate the quality of one-class classifier. Model selection for single-classifier approach was difficult, all the more for choosing a subset of complementary classifiers. This thesis presented developments in this area by formulating the concept of diversity measures suitable for one-class classification, as well as proposing three mechanisms for discarding irrelevant classifiers from the pool:

- We have introduced novel diversity measures suitable for handling one-class classifiers. As we do not have an access to counterexamples, we needed to look for a different way of evaluating how different classifiers are from each other. An exemplary introduced measure was based on the geometrical properties of sphere-based one-class classifiers. Here it was possible to calculate their degree of overlapping in kernel spaces (where they formed spheres). We showed that with the increase in overlapping area decreases the diversity of the ensemble. Such approach allowed for an efficient evaluation of one-class classifiers without a need for an access to outliers.

- We have proposed several methods based on heuristic optimization. This allowed us to use an efficient search engine for browsing through possible combinations of classifiers in the pool. We have presented an efficient methodology based on memetic algorithms that combined evolutionary global search with local exploitation. We showed how to formulate a weighted multi-criteria pruning method that allowed to manipulate the degree of importance of each criteria in the ensemble pruning.

- We have proposed a clustering-based pruning approach. It uses a cluster detection technique that grouped similar classifiers together. Then from each group a single most representative classifier is selected, thus achieving the pruning step. We showed how to formulate support matrix on the basis of continuous outputs of one-class classifiers. Working directly on the discriminants alleviated the need for dedicated measures of performance for one-class classifiers. We had presented the application of X-means clustering for fast selection of a proper clustering model.
- We have introduced a hybrid pruning approach using firefly algorithm. Here we encoded the pool of classifiers as a population and used a swarm intelligence approach to process it. This is a hybrid method, as it used optimization tools to form compact clusters. Then for each cluster a representative classifier was selected. We showed how to embed a multi-criteria selection into the firefly optimization. Additionally, we introduced a weighting scheme for deriving weights assigned to each selected classifier in the fusion process.

All of our methods were evaluated on a set of 20 diverse benchmarks and analyzed with statistical tests of significance (pairwise, ranking and post-hoc).

To further show the quality flexibility and usefulness of the proposed methods, we presented three real-life applications done in collaboration with other research teams. We showed that our methods deliver excellent performance in real-life scenarios from environmental engineering (artificial nose sniffing), computer vision (hyperspectral image processing) and medicine (automatic breast cancer malignancy grading).

Results obtained for a set of proposed algorithms on the basis of benchmark and real-life data allow us to conclude that the proposed research hypothesis (see Section 1.4) has been verified positively. Indeed it is possible to propose novel methods for forming and pruning one-class classifier ensembles that can outperform canonical algorithms from OCC domain proposed in the literature.

Let us now present several main conclusions that can be derived from this thesis:

- Canonical ensemble forming methods must cannot be directly translated into one-class domain. Bagging does not preserve the spatial relations between data. As most of one-class classifiers aims at creating a single enclosing boundary such an approach may lead to creation of a too large classifier volume - and this in turn reduces significantly the robustness to outliers. On the other hand Boosting can be easily overfitted to data, as it will correct the mistakes only on a single class. One may easily see that a combined classifier that will always point out at the target class will achieve a minimal error on the boosting training. The one-class version of Random Forest cannot be treated as a native one-class classifier, as it requires an artificial generation of outliers and uses binary trees. This may decrease the adaptive properties of a given classifier. Only Random Subspace can be easily used for one-class learning, especially that it reduces the dimensionality for each model (and one-class classifiers work better in lower dimensions).

- Highly effective methods for one-class ensembles are based on object space partitioning. By finding local atomic subsets of objects (areas of competence), we are able to train sparse one-class classifiers that are able to capture complex internal structures in the target class. We reduce the number of input objects, thus speeding-up the computation. We assure diversity by using different clusters of training samples for each classifier. Individual accuracies of classifiers are maintained by allowing them to adapt to a smaller sample.

- One-class classifiers work very well when combined with unsupervised feature extraction methods. Therefore an ensemble utilizing different rotations of the feature space can efficiently introduce needed diversity for the pool without the need for creating artificial objects.

- In case of soft one-class classifiers, one may achieve good and mutually complementary classifiers with the usage of weighted bagging. By drawing weights assigned to objects at random, we are able to create a pool of classifiers where each model puts emphasis on different training samples.

- In many cases (especially for Random Subspace and Weighted Bagging) ensembles should work in the overproduce-and-select mode. As we have no assurance on the quality of each outputted classifier, it is better to generate a larger pool of models (in hope that they will cover different areas of competence) and then discard weak or similar models.

- It is worthwhile to exploit some specific properties of one-class classifiers (such as the geometric properties of their boundaries) in order to design novel diversity measures tuned to the OCC problems, as they tend to return excellent performance. Therefore when designing new diversity measures for OCC one should explore different directions than just the outputs of classifiers.

- Using multi-criteria approaches can benefit both one-class ensemble forming step (where we can directly influence the way how classifiers are being trained) and pruning step (where we can evaluate available models from different points of view). Using weighted criteria allows to control which of them is more important for the ensemble. It is worthwhile to include a diversity measure into this process, but it is important to control its level of influence.

- It is possible to conduct an efficient one-class classifier selection without the need for any specialized measure. When using support values outputted by each base learner, one may look for templates in individual decisions. By clustering outputs one may detect which classifiers give similar decisions.

- Training weights for one-class classifiers can lead to a better combination step than when using canonical methods from the literature. This can be especially useful if the weight training procedure is embedded in the ensemble design step and can interact with the training or pruning steps in order to produce more specialized models.

Although this thesis has explored different areas and possible applications, it at the same time opened new possible research tracks for future works. Let us present some main directions that may be explored in the future:

- One may further improve OCClustE and OC-EvoClust by adding a clustering pruning scheme. Such a mechanism will be responsible for merging similar clusters in case when their number was not properly detected. This could increase the robustness of these methods to overfitting in certain points of the decision space.

- One can propose a dynamic classifier selection and a dynamic ensemble selection systems in order to provide a more flexible ensemble structure. This seems highly attractive as in one-class classification we must be prepared for versatile possible scenarios.

- A fully unexplored area are the ranking-based pruning methods. They allow for an efficient ordering of classifiers and fast pruning by thresholding. This can be useful if we need to rebuild our ensemble on-the-fly, thus working within time constraints.

- One-class classifiers can also be used for decomposition of multi-class problems as an extension of one-versus-all approaches. Therefore one should investigate mechanisms for selecting classifiers for each individual class and methods for combining them in order to reconstruct the original multi-class problem.

- It seems attractive to use one-class classifiers and ensembles for data stream analytics. One can create ensembles of batch and on-line one-class learners in order to iteratively process massive data. One-class classifiers seem also attractive for concept drift, as they adapt to the structure of the target class and may easily detect any changes in it.

- Another possible direction is to combine one-class classifiers with MapReduce and Spark architectures in order to process big data. Current methods can become computationally expensive for massive volumes of information, thus there is a need for creating efficient computational architectures to deal with ever-occurring data growth.

Research presented in Sections 2.1, 2.4, 3.1, 3.2, 3.3, 3.4, and 4.3 was partially supported by the Polish National Science Centre under the grant PRELUDIUM number DEC-2013/09/N/ST6/03504 realized in years 2014-2016.

Research presented in Sections 2.2, 2.3, 4.1, and 4.2 was partially supported by Polish National Science Center under the grant no. DEC-2013/09/B/ST6/02264.

Appendices

Appendix A

Overview of Dataset Benchmarks

In this Appendix, we present the details of benchmark datasets that were used thorough this thesis for an experimental evaluation of the proposed one-class ensemble forming and pruning algorithms.

In order to present an unified experimental framework, ensure the repeatability of experiments and ensure the ease of comparison between different methods, we have run all of the experiments on the same set of datasets. Additionally, each dataset was firstly partitioned into folds, which ensured that each method is trained and tested on exactly the same setting of objects.

Due to the lack of one-class benchmarks we use the binary ones. The training set was composed from the part of objects from the target class (according to cross-validation rules), while the testing set consisted of the remaining objects from the target class and outliers (to check both the false acceptance and false rejection rates). Majority class was used as the target concept and minority class as outliers. This scheme is popularly used in works on one-class classifiers.

We have chosen 20 binary datasets in total, originating from four different repositories: UCI[1], KEEL[2] [2], TUDelft[3] [271] and TunedIT[4]. Details of the chosen datasets are given in Tab. A.1.

Additionally, we applied our OCClustE algorithm for decomposition of multi-class problems. In total we selected 20 multi-class datasets. Here, we treated them as nor-

[1]https://archive.ics.uci.edu/ml/index.html
[2]http://sci2s.ugr.es/keel/datasets.php
[3]http://homepage.tudelft.nl/n9d04/occ/index.html
[4]http://tunedit.org/challenge/QSAR

Table A.1: Details of datasets used in the experimental investigation. Indexes stands for a database from which given benchmark is taken. Numbers in parentheses indicates the number of objects in the minority class.

No.	Name	Objects	Features	Classes
1.	Arrhythmia[3]	420 (183)	278	2
2.	Biomed[3]	194 (67)	5	2
3.	Breast-cancer[1]	286 (85)	9	2
4.	Breast-Wisconsin[1]	699 (241)	9	2
5.	Colic[1]	368 (191)	22	2
6.	Diabetes[1]	768 (268)	8	2
7.	Delft Pump 2x2 noisy[3]	240 (64)	64	2
8.	Glass0123vs456[2]	214 (51)	9	2
9.	Heart-statlog[1]	270 (120)	13	2
10.	Hepatitis[3]	155 (32)	19	2
11.	Ionosphere[1]	351(124)	34	2
12.	Liver[1]	345 (145)	6	2
13.	New-thyroid2[2]	215 (37)	5	2
14.	p53 Mutants[1]	16772 (3354)	5409	2
15.	Pima[1]	768 (268)	8	2
16.	Spambase[3]	4601 (1813)	57	2
17.	Sonar[1]	208 (97)	60	2
18.	Yeast3[2]	1484 (163)	8	2
19.	Voting records[1]	435 (168)	16	2
20.	CYP2C19 isoform[4]	837 (181)	242	2

mal recognition scenario, where we have examples from each class available during the learning procedure. Details of the chosen datasets are given in Table A.2.

Table A.2: Details of multi-class datasets used in the experiments with OCClustE applied to multi-class problem decomposition.

No.	Name	Objects	Features	Classes
1	Audiology	226	69	24
2	Balance	625	4	3
3	Breast-cancer	286 (85)	9	2
4	Breast-Wisconsin	699 (241)	9	2
5	Colic	368 (191)	22	2
6	Credit-rating	690	15	6
7	Diabetes	768 (268)	8	2
8	Glass	214	9	6
9	Heart-c	303	13	5
10	Heart-h	294	13	5
11	Heart-statlog	270 (120)	13	2
12	Hepatitis	155 (32)	19	2
13	Ionosphere	351(124)	34	2
14	Iris	150	4	3
15	Lymphography	148	18	4
16	Primary tumor	339	17	21
17	Sonar	208 (97)	60	2
18	Voting records	435 (168)	16	2
19	Wine	178	13	3
20	CYP2C19 isoform	837 (181)	242	2

Appendix B

General Set-up Framework for Experimental Studies

B.1 Measures of Performance

To compare the learning algorithms, one needs a specified measure of their recognition quality. The evaluation of the performance of a classifier is a key issue both to guide its modeling and to properly assess its quality with respect to other classifiers dealing with the same problem.

In this thesis, we dealt with the one-class classification problem. However, in order to analyze the quality of the proposed methods we run experiments in a controlled environment with the usage of binary benchmarks. We used a fold of target class objects for training, and a fold of both target class and outlier class objects for testing. With this, we preserved the paradigm of learning in the absence of counterexamples (as we used only single class for training), while getting a good estimation of the method's performance (having both classes for testing). The training procedures were conducted only with the usage of measures suitable for one-class learning (such as consistency measure or dedicated diversity measures), but for evaluation of classifier's performance we can use well-defined measures for binary problems.

When addressing a two-class problem, the results of the correctly and incorrectly classified examples of each class can be stored in a confusion matrix (Table B.1).

Although historically accuracy rate has been the most commonly used measure to evaluate the performance of classifiers (Eq. B.1), it is not suitable when we want to

Table B.1: Confusion matrix for a two-class problem.

	Positive prediction	Negative prediction
Positive class	True Positive (TP)	False Negative (FN)
Negative class	False Positive (FP)	True Negative (TN)

gain a deeper insight into the results of the experiment regarding each of the classes.

$$\mathrm{Acc} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{FN}+\mathrm{FP}+\mathrm{TN}} \tag{B.1}$$

We want to evaluate one-class classifiers with respect to their robustness to outliers (error on outlier class) and generalization properties (error on the target class). Additionally, in our framework usually we have a higher number of objects from the target class than outliers (to reflect real-world scenarios), which can be related to imbalanced classification.

On this account, other measures need to be considered in this framework, which take the into account the performance for each class independently. From the confusion matrix (Table B.1), different measures evaluating the performance over each class independently can be deduced:

- *True positive rate* (also known as *Sensitivity*) $\mathrm{TP}_{rate} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$.
- *True negative rate* $\mathrm{TN}_{rate} = \frac{\mathrm{TN}}{\mathrm{FP}+\mathrm{TN}}$.
- *False positive rate* $\mathrm{FP}_{rate} = \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}}$.
- *False negative rate* $\mathrm{FN}_{rate} = \frac{\mathrm{FN}}{\mathrm{TP}+\mathrm{FN}}$.

Nonetheless, these measures on their own are still inadequate, because they only consider one of the classes. To present a combined performance on both classes, with the respect to the number of objects in each of them, we propose to use a Geometric Mean (GM) [15] for each of the one-class experiments in this thesis.

GM considers a balancing between the accuracy over the instances of the minority and majority classes at the same time (Eq. (B.2)) being appropriate to deal with the class imbalance problem. Hence, in order to achieve a good performance the accuracy of both classes should be maximized at the same time.

$$\mathrm{GM} = \sqrt{\mathrm{TP}_{rate} \cdot \mathrm{TN}_{rate}}. \tag{B.2}$$

B.2 Statistical Analysis

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 [90]. 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.

- We use a 5x2 CV combined F-test [3] for simultaneous training/testing and pairwise statistical analysis. It repeats five-time two fold cross-validation. The pseudocode of this method is given in Algorithm 7.

Algorithm 7 5×2 Cross-Validation

```
Require: learning_set $\mathcal{LS}$
Ensure: 10 validation errors
1: for $i := 1$ to 5 do
2:     Divide $\mathcal{LS}$ randomly into 2 equal subsets $\mathcal{TS}_1, \mathcal{TS}_2$ where $\mathcal{TS}_1 \cap \mathcal{TS}_2 = \varnothing$ and $\mathcal{LS} = \mathcal{TS}_1 \cup \mathcal{TS}_2$
3:     train classifier using $\mathcal{TS}_1$
4:     error_2i-1 ← evaluate validation error using $\mathcal{TS}_2$
5:     train classifier using $TS_2$
6:     error_2i ← evaluate validation error using $\mathcal{TS}_1$
7: end for
```

The combined F-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.

Let us present how it conducts a pairwise comparison between classifiers:

1. Obtain 10 differences between accuracies (or any other measures) dif_i^j of two classifiers, where i stands for the repetition number and j for the subset number.
2. Calculate an average for each repetition:

$$\bar{dif}_i = (dif_i^{(1)} + dif_i^{(2)})/2. \tag{B.3}$$

3. Calculate the variance:

$$var_i = (dif_i^{(1)} - \bar{dif}_i)^2 + (dif_i^{(2)} - \bar{dif}_i)^2. \tag{B.4}$$

4. Calculate the F-statistic that uses all of ten differences:

$$f = \frac{\sum_{i=1}^{5}\sum_{j=1}^{2}(dif_i^{(j)})^2}{var} \sim F_{10,5}. \tag{B.5}$$

This 5×2 CV combined F-test rejects the hypothesis of the lack of statistical differences between classifiers for a given significance level α fir cases, where value obtained from Eq. (B.5) is greater than $F_{\alpha/10,5}$. Usually the significance level is set to $\alpha = 0,05$ ($F_{0,05/10,5} = 4,74$).

- For assessing the ranks of classifiers over all examined benchmarks, we use a Friedman ranking test [65]. It checks, if the assigned ranks are significantly different from assigning to each classifier an average rank.

- We use the Shaffer post-hoc test [90] to find out which of the tested methods are distinctive among an n x n comparison. The post-hoc procedure is based on a specific value of the significance level α. 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.

Appendix C

Detailed Settings and Results of Experiments

Due to the excessive number of conducted experiments, obtained results, tables and figures, the detailed settings and results from this thesis can be found online at:

http://www.kssk.pwr.edu.pl/krawczyk/thesis/

Bibliography

[1] N.M. Adams and D.J. Hand. **Improving the Practice of Classifier Performance Assessment**. *Neural Computation*, **12**(2):305–311, 2000. 25

[2] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, and S. García. **KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework**. *Multiple-Valued Logic and Soft Computing*, **17**(2-3):255–287, 2011. 215

[3] E. Alpaydin. **Combined 5 x 2 cv F Test for Comparing Supervised Classification Learning Algorithms**. *Neural Computation*, **11**(8):1885–1892, 1999. 221

[4] E. Alpaydin and E. Mayoraz. **Learning error-correcting output codes from data**. In *IEE Conference Publication*, **2**, pages 743–748, 1999. 54, 83

[5] Ethen Alpaydin. *Introduction to Machine Learning, Second Edition.* The MIT Press, 2010. 7, 99

[6] I. Arel, D. C. Rose, and T. P. Karnowski. **Deep machine learning-a new frontier in artificial intelligence research [research frontier]**. *Computational Intelligence Magazine, IEEE*, **5**(4):13–18, 2010. 9

[7] B. Ayerdi, I. Marqués, and M. Graña. **Spatially regularized semisupervised Ensembles of Extreme Learning Machines for hyperspectral image segmentation**. *Neurocomputing*, **149**:373–386, 2015. 185

[8] T. Back, D. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation.* Oxford Univ. Press, 1997. 95, 98

[9] M. A. Bagheri, G. A. Montazer, and E. Kabir. **A subspace approach to error correcting output codes**. *Pattern Recognition Letters*, **34**(2):176–184, 2013. 54

[10] M. Ballings and D. Van den Poel. **Kernel Factory: An ensemble of kernel machines**. *Expert Systems with Applications*, **40**(8):2904–2913, 2013. 46

[11] A. Banerjee, P. Burlina, and C. Diehl. **A support vector method for anomaly detection in hyperspectral imagery**. *IEEE Transactions on Geoscience and Remote Sensing*, **44**(8):2282–2291, 2006. 42

[12] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer. **Ensemble diversity measures and their application to thinning**. *Information Fusion*, **6**(1):49–62, 2005. 49

[13] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer. **A comparison of decision tree ensemble creation techniques**. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **29**(1):173–180, 2007. 45

[14] Y. Baram. **Partial classification: the benefit of deferred decision**. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **20**(8):769 –776, aug 1998. 51

[15] R. Barandela, J. S. Sánchez, V. García, and E. Rangel. **Strategies for learning in class imbalance problems**. *Pattern Recognition*, **36**(3):849–851, 2003. 220

[16] V. Barnett and T. Lewis. *Outliers in Statistical Data*. Wiley Series in Probability and Mathematical Statistics. Wiley; 3rd edition, 1994. 32

[17] B. Baruque, S. Porras, and E. Corchado. **Hybrid Classification Ensemble Using Topology-preserving Clustering**. *New Generation Computing*, **29**:329–344, 2011. 51

[18] E. Bauer and R. Kohavi. **An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants**. *Machine Learning*, **36**(1-2):105–139, 1999. 114

[19] R.E. Bellman. *Dynamic Programming*. Princeton University Press, 1957. 10, 33

[20] M. Bicego and M. A. T. Figueiredo. **Soft clustering using weighted one-class support vector machines**. *Pattern Recognition*, **42**(1):27–32, 2009. 31, 38, 39, 114

[21] A. Bifet and G. D. F. Morales. **Big Data Stream Learning with SAMOA**. In *Data Mining Workshop (ICDMW), 2014 IEEE International Conference on*, pages 1199–1202. IEEE, 2014. 46

[22] B. Biggio, G. Fumera, and F. Roli. **Bayesian analysis of linear combiners**. In *Proceedings of the 7th international conference on Multiple classifier systems*, MCS'07, pages 292–301, Berlin, Heidelberg, 2007. Springer-Verlag. 51

[23] J. Bilmes. **A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models**. *International Computer Science Institute*, **4**(510):126, 1998. 33

[24] C. M. Bishop. **Novelty detection and neural network validation**. *IEE Proceedings: Vision, Image and Signal Processing*, **141**(4):217–222, 1994. 19

[25] C. M. Bishop. *Pattern recognition and machine learning*. springer New York, 2006. 33

[26] J. Blaszczynski, M. Deckert, J. Stefanowski, and S. Wilk. **Integrating selective preprocessing of imbalanced data with Ivotes ensemble**. **6086 LNAI** of *Lecture Notes in Computer Science*, pages 148–157. 2010. 203

[27] J. Błaszczyński and J. Stefanowski. **Neighbourhood sampling in bagging for imbalanced data**. *Neurocomputing*, **150**:529–542, 2015. 46

[28] H.J.G. Bloom and W.W. Richardson. **Histological Grading and Prognosis in Breast Cancer**. *British Journal of Cancer*, **11**:359–377, 1957. 190

[29] Piero Bonissone, José M Cadenas, M Carmen Garrido, and R Andrés Díaz-Valladares. **A fuzzy random forest**. *International Journal of Approximate Reasoning*, **51**(7):729–747, 2010. 47

[30] L. Bottou and Y. Bengio. **Convergence Properties of the K-Means Algorithms.** In *Advances in Neural Information Processing Systems 7,[NIPS Conference, Denver, Colorado, USA, 1994]*, pages 585–592, 1994. 35

[31] Ch. Boutsidis, A. Zouzias, M. W. Mahoney, and P. Drineas. **Randomized dimensionality reduction for k-means clustering**. *IEEE TRANSACTIONS ON INFORMATION THEORY*, **61**(2):1045, 2015. 34

[32] A.P. Bradley. **The use of the area under the {ROC} curve in the evaluation of machine learning algorithms**. *Pattern Recognition*, **30**(7):1145 – 1159, 1997. 25

[33] L. Breiman. **Bagging predictors**. *Machine learning*, **24**(2):123–140, 1996. 45

[34] L. Breiman. **Random forests**. *Machine learning*, **45**(1):5–32, 2001. 46, 105

[35] J. S. Bridle. **Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition**. In *Neurocomputing*, pages 227–236. Springer, 1990. 11

[36] G. Brown and L. I. Kuncheva. **"Good" and "Bad" Diversity in Majority Vote Ensembles**. In *MCS*, pages 124–133, 2010. 49

[37] P. Büchlmann and B. Yu. **Analyzing bagging**. *Annals of Statistics*, pages 927–961, 2002. 46

[38] C. J. C. Burges. **A tutorial on support vector machines for pattern recognition**. *Data Mining and Knowledge Discovery*, **2**(2):121–167, 1998. 40

[39] G.G. Cabral and A.L.I. Oliveira. **One-Class Classification based on searching for the problem features limits**. *Expert Systems with Applications*, **41**(16):7182–7190, 2014. 29

[40] G.A. Carpenter, S. Grossberg, and D. B. Rosen. **ART 2-A: An adaptive resonance algorithm for rapid category learning and recognition**. *Neural Networks*, **4**(4):493–504, 1991. 31

[41] P. Casale, O. Pujol, and P. Radeva. **Approximate polytope ensemble for one-class classification**. *Pattern Recognition*, **47**(2):854–864, 2014. 56

[42] M. Cha, J.S. Kim, and J.-G. Baek. **Density weighted support vector data description**. *Expert Systems with Applications*, **41**(7):3343–3350, 2014. 42

[43] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer. **SMOTEBoost: Improving prediction of the minority class in boosting**. In *Lecture Notes in Artificial Intelligence*, **2838**, pages 107–119, 2003. 46

[44] B. Chen, A. . Feng, S. . Chen, and B. Li. **One-cluster clustering based data description**. *Jisuanji Xuebao/Chinese Journal of Computers*, **30**(8):1325–1332, 2007. 30, 31

[45] H. Chen, P. Tino, and X. Yao. **A probabilistic ensemble pruning algorithm**. pages 878–882, 2006. 48

[46] S. Chen, H. He, and E. A. Garcia. **RAMOBoost: Ranked minority oversampling in boosting**. *IEEE Transactions on Neural Networks*, **21**(10):1624–1642, 2010. 46

[47] X. . Chen and M. Wasikowski. **FAST: A roc-based feature selection metric for small samples and imbalanced data classification problems**. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 124–132, 2008. 197

[48] Y. Chen, X. S. Zhou, and T. S. Huang. **One-class SVM for learning in image retrieval**. In *IEEE International Conference on Image Processing*, **1**, pages 34–37, 2001. 30

[49] V. Cheplygina and D. M. J. Tax. **Pruned random subspace method for one-class classifiers**. In *Multiple Classifier Systems*, **6713 LNCS** of *Lecture Notes in Computer Science*, pages 96–105, 2011. 57, 132, 145, 155

[50] Y.-S. Choi. **Least squares one-class support vector machine**. *Pattern Recognition Letters*, **30**(13):1236–1240, 2009. 39

[51] C. K. Chow. **Statistical Independence and Threshold Functions**. *Electronic Computers, IEEE Transactions on*, **EC-14**(1):66 –68, feb. 1965. 43

[52] D. Chung and H. Kim. **Accurate ensemble pruning with PL-bagging**. *Computational Statistics and Data Analysis*, **83**:1–13, 2015. 50

[53] G. Cohen, H. Sax, and A. Geissbuhler. **Novelty detection using one-class parzen density estimator. An application to surveillance of nosocomial infections**. In *Studies in Health Technology and Informatics*, **136**, pages 21–26, 2008. 30, 31

[54] L. Cordella, P. Foggia, C. Sansone, F. Tortorella, and M. Vento. **A Cascaded Multiple Expert System for Verification**. In *Multiple Classifier Systems*, **1857** of *Lecture Notes in Computer Science*, pages 330–339. Springer Berlin / Heidelberg, 2000. 51

[55] B. Cyganek. *Object Detection and Recognition in Digital Images: Theory and Practice*. Wiley, New York, 2013. 178, 184

[56] B. Cyganek. **Pattern recognition framework based on the best rank-(R1,R2,...,RK) tensor approximation**. In *Computational vision and medical image processing IV: Proceedings of VipIMAGE 2013 and IV ECCOMAS thematic conference on Computational vision and medical image processing*, pages 301–206, 2013. 180, 184

[57] B. Cyganek and B. Krawczyk. **Data Classification with Ensembles of One-Class Support Vector Machines and Sparse Nonnegative Matrix Factorization**. In *Intelligent Information and Database Systems - 7th Asian Conference, ACIIDS 2015, Bali, Indonesia, March 23-25, 2015, Proceedings, Part I*, pages 526–535, 2015. 62

[58] I. Czarnowski and P. Jedrzejowicz. **Ensemble Classifier for Mining Data Streams**. In *18th International Conference in Knowledge Based and Intelligent Information and Engineering Systems, KES 2014, Gdynia, Poland, 15-17 September 2014*, pages 397–406, 2014. 57

[59] I. Czarnowski and P. Jedrzejowicz. **Ensemble Online Classifier Based on the One-Class Base Classifiers for Mining Data Streams**. *Cybernetics and Systems*, **46**(1-2):51–68, 2015. 57

[60] E. Czogala and J. Leski. **Application of entropy and energy measures of fuzziness to processing of ECG signal**. *Fuzzy Sets and Systems*, **97**(1):9–18, 1998. 127

[61] Q. Dai, T. Zhang, and N. Liu. **A new reverse reduce-error ensemble pruning algorithm**. *Applied Soft Computing Journal*, **28**:237–249, 2015. 50

[62] E. N. de Souza and S. Matwin. **Improvements to Boosting with Data Streams**. In *Advances in Artificial Intelligence*, pages 248–255. Springer, 2013. 46

[63] O. Dekel, O. Shamir, and L. Xiao. **Learning to classify with missing and corrupted features**. *Machine Learning*, **81**(2):149–178, 2010. 11

[64] S. del Río, V. López, J. M. Benítez, and F. Herrera. **On the use of MapReduce for imbalanced big data using Random Forest**. *Information Sciences*, **285**:112–137, 2014. 47

[65] J. Demsar. **Statistical comparisons of classifiers over multiple data sets**. *Journal of Machine Learning Research*, **7**:1–30, 2006. 109, 222

[66] C. Desir, S. Bernard, C. Petitjean, and L. Heutte. **One class random forests**. *Pattern Recognition*, **46**(12):3490–3506, 2013. 57, 105

[67] L. Devroye. *A probabilistic theory of pattern recognition*, **31**. Springer Science & Business Media, 1996. 11

[68] V. Di Gesù and G. Lo Bosco. **Combining One Class Fuzzy KNN's**. In *Applications of Fuzzy Sets Theory, 7th International Workshop on Fuzzy Logic and Applications, WILF 2007, Camogli, Italy, July 7-10, 2007, Proceedings*, pages 152–160, 2007. 31

[69] R. Díaz-Uriarte and S. A. De Andres. **Gene selection and classification of microarray data using random forest**. *BMC bioinformatics*, **7**(1):3, 2006. 47

[70] T. G Dietterich. **An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization**. *Machine learning*, **40**(2):139–157, 2000. 46

[71] T. G. Dietterich and G. Bakiri. **Solving multiclass learning problems via error-correcting output codes**. *J. Artif. Int. Res.*, **2**:263–286, January 1995. 54

[72] T.G. Dietterich. **Approximate Statistical Test For Comparing Supervised Classification Learning Algorithms**. *Neural Computation*, **10**(7):1895–1923, 1998. 18

[73] P. Domingos. **A few useful things to know about machine learning**. *Communications of the ACM*, **55**(10):78–87, 2012. 10, 15

[74] E.M. Dos Santos, R. Sabourin, and P. Maupin. **Single and multi-objective genetic algorithms for the selection of ensemble of classifiers**. pages 3070–3077, 2006. 50

[75] E.M. Dos Santos, R. Sabourin, and P. Maupin. **A dynamic overproduce-and-choose strategy for the selection of classifier ensembles**. *Pattern Recognition*, **41**(10):2993–3009, 2008. 48

[76] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* Wiley, New York, 2. edition, 2001. 8, 11, 16

[77] R.P.W. Duin. **The combining classifier: to train or not to train?** In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, **2**, pages 765 – 770 vol.2, 2002. 52

[78] B. Fei and J. Liu. **Binary tree of SVM: A new fast multiclass training and classification algorithm**. *IEEE Transactions on Neural Networks*, **17**(3):696–704, 2006. 54

[79] P. Filipczuk, B. Krawczyk, and M. Woźniak. **Classifier ensemble for an effective cytological image analysis**. *Pattern Recognition Letters*, **34**(14):1748–1757, 2013. 190

[80] P.A. Flach. **The Geometry of ROC Space: Understanding Machine Learning Metrics through ROC Isometrics**. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 194–201, 2003. 24

[81] Y. Freund and R. E. Schapire. **Experiments with a new boosting algorithm**. In *ICML*, **96**, pages 148–156, 1996. 45, 46

[82] J. H. Friedman. **On bias, variance, 0/1—loss, and the curse-of-dimensionality**. *Data mining and knowledge discovery*, **1**(1):55–77, 1997. 33

[83] B. Gabrys and D. Ruta. **Genetic algorithms in classifier fusion**. *Applied Soft Computing Journal*, **6**(4):337–347, 2006. 50

[84] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and 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 Recognition*, **44**(8):1761–1776, 2011. 28, 46, 54

[85] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera. **A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches**. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, **42**(4):463–484, 2012. 46, 200, 201

[86] M. Galar, A. Fernández, E. Barrenechea, and F. Herrera. **EUSBoost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling**. *Pattern Recognition*, **46**(12):3460–3471, 2013. 46

[87] M. Galar, A. Fernández, E. Barrenechea, and F. Herrera. **DRCW-OVO: Distance-based relative competence weighting combination for One-vs-One strategy in multi-class problems**. *Pattern Recognition*, **48**(1):28–42, 2015. 54

[88] M. Galar, A. Fernández, E. Barrenechea Tartas, H. Bustince Sola, and F. Herrera. **Dynamic classifier selection for One-vs-One strategy: Avoiding non-competent classifiers**. *Pattern Recognition*, **46**(12):3412–3424, 2013. 54

[89] H. Gao, C. Zhao, and H. Zhang. **A traversability detection method based on combining one-class SVM**. *Journal of Information and Computational Science*, **9**(17):5509–5516, 2012. 55

[90] S. García, A. Fernández, J. Luengo, and 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):2044–2064, 2010. 221, 222

[91] V. Garcia, R. A. Mollineda, and J. S. Sanchez. **On the k-NN performance in a challenging scenario of imbalance and overlapping**. *Pattern Analysis and Applications*, **11**(3-4):269–280, 2008. 197

[92] A.B. Gardner, A.M. Krieger, G. Vachtsevanos, and B. Litt. **One-class novelty detection for seizure analysis from intracranial EEG**. *Journal of Machine Learning Research*, **7**:1025–1044, 2006. 40

[93] M. Gashler, Ch. Giraud-Carrier, and T. Martinez. **Decision tree ensemble: Small heterogeneous is better than large homogeneous**. In *Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on*, pages 900–905. IEEE, 2008. 46

[94] S. Geman, E. Bienenstock, and R. Doursat. **Neural networks and the bias/variance dilemma**. *Neural computation*, **4**(1):1–58, 1992. 16

[95] G. Giacinto, R. Perdisci, M. Del Rio, and F. Roli. **Intrusion detection in computer networks by a modular ensemble of one-class classifiers**. *Inf. Fusion*, **9**:69–82, 2008. 55

[96] G. Giacinto, F. Roli, and L. Didaci. **Fusion of multiple classifiers for intrusion detection in computer networks**. *Pattern Recognition Letters*, **24**(12):1795–1803, 2003. 44

[97] G. Giacinto, F. Roli, and G. Fumera. **Design of effective multiple classifier systems by clustering of classifiers**. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, **2**, pages 160 –163 vol.2, 2000. 43, 51

[98] K. Goebel and W. Yan. **Choosing Classifiers for Decision Fusion**. In *Proceedings of the Seventh International Conference on Information Fusion*, pages 563–568, 2004. 48

[99] K.-S. Goh, E.Y. Chang, and B. Li. **Using one-class and two-class SVMs for multiclass image annotation**. *IEEE Transactions on Knowledge and Data Engineering*, **17**(10):1333–1346, 2005. 40

[100] A. Golestani, J. Azimi, M. Analoui, and M. Kangavari. **A new efficient fuzzy diversity measure in classifier fusion**. In *Proceedings of the IADIS Conference*, pages 722–726, 2007. 127

[101] L. Guo and S. Boukir. **Margin-based ordered aggregation for ensemble pruning**. *Pattern Recognition Letters*, **34**(6):603–609, 2013. 51

[102] I. Guyon. *Feature extraction: foundations and applications*, **207**. Springer Science & Business Media, 2006. 10

[103] I. Guyon and A. Elisseeff. **An introduction to variable and feature selection**. *The Journal of Machine Learning Research*, **3**:1157–1182, 2003. 48

[104] B. Hadjadji, Y. Chibani, and Y. Guerbai. **Multiple one-class classifier combination for multi-class classification**. pages 2832–2837, 2014. 56

[105] B. Hadjadji, Y. Chibani, and H. Nemmour. **Fuzzy integral combination of one-class classifiers designed for multi-class classification**. *Lecture Notes in Computer Science*, **8814**:320–328, 2014. 57

[106] M.S. Haghighi, A. Vahedian, and H.S. Yazdi. **Creating and measuring diversity in multiple classifier systems using support vector data description**. *Applied Soft Computing Journal*, **11**(8):4931–4942, 2011. 57

[107] Lee D. Hamm, J. **Grassmann discriminant analysis: a unifying view on subspace-based learning**. In *In Proceedings of the 25th international conference on machine learning*, pages 376–383, 2008. 181

[108] B. Hanczar and M. Sebag. **Combination of one-class support vector machines for classification with reject option**. *Lecture Notes in Artificial Intelligence*, **8724 LNAI**(PART 1):547–562, 2014. 56

[109] L.K. Hansen and P. Salamon. **Neural network ensembles**. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **12**(10):993 –1001, oct 1990. 51

[110] P.-Y. Hao. **Fuzzy one-class support vector machines**. *Fuzzy Sets and Systems*, **159**(18):2317 – 2336, 2008. 31

[111] M. Harman and P. McMinn. **A theoretical and empirical study of search-based testing: Local, global, and hybrid search**. *IEEE Transactions on Software Engineering*, **36**(2):226–247, 2010. 134

[112] J. W. Harris and H. Stocker. *Handbook of mathematics and computational science.* Springer-Verlag, New York, 1998. 129

[113] S. Hashemi, Y. Yang, Z. Mirzamomen, and M. Kangavari. **Adapted one-versus-all decision trees for data stream classification**. *Knowledge and Data Engineering, IEEE Transactions on*, **21**(5):624–637, 2009. 54

[114] T. Hastie and R. Tibshirani. **Classification by pairwise coupling**. *Annals of Statistics*, **26**(2):451–471, 1998. 54

[115] X. He, G. Mourot, D. Maquin, J. Ragot, P. Beauseroy, A. Smolarz, and E. Grall-Maes. **Multi-task learning with one-class SVM**. *Neurocomputing*, **133**:416–426, 2014. 40

[116] K. Hempstalk, E. Frank, and I.H. Witten. **One-Class Classification by Combining Density and Class Probability Estimation**. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML/PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part I*, pages 505–519, 2008. 33

[117] Á. Herrero, U. Zurutuza, and E. Corchado. **A Neural-Visualization IDS for Honeynet Data**. *Int. J. Neural Syst.*, **22**(2), 2012. 35

[118] S. Hido, H. Kashima, and Y. Takahashi. **Roughly balanced Bagging for Imbalanced data**. *Statistical Analysis and Data Mining*, **2**(5-6):412–426, 2009. 46, 203

[119] T. K. Ho. **The Random Subspace Method for Constructing Decision Forests**. *IEEE Trans. Pattern Anal. Mach. Intell.*, **20**:832–844, 1998. 45

[120] H. Hoffmann. **Kernel {PCA} for novelty detection**. *Pattern Recognition*, **40**(3):863 – 874, 2007. 31

[121] G. Huang, H. Chen, Z. Zhou, F. Yin, and K. Guo. **Two-class support vector data description**. *Pattern Recognition*, **44**(2):320–329, 2011. 28

[122] A. Hyvärinen and E. Oja. **A fast fixed-point algorithm for independent component analysis**. *Neural computation*, **9**(7):1483–1492, 1997. 107

[123] E. Ikonomovska, J. Gama, and S. Džeroski. **Online tree-based ensembles and option trees for regression on evolving data streams**. *Neurocomputing*, **150**:458–470, 2015. 8, 46

[124] K. Jackowski, B. Krawczyk, and M. Woźniak. **Cost-Sensitive Splitting and Selection Method for Medical Decision Support System**. In *Intelligent Data Engineering and Automated Learning - IDEAL 2012 - 13th International Conference, Natal, Brazil, August 29-31, 2012. Proceedings*, pages 850–857, 2012. 50, 89

[125] K. Jackowski, B. Krawczyk, and M. Woźniak. **Application of Adaptive Splitting and Selection Classifier to the Spam Filtering Problem**. *Cybernetics and Systems*, **44**(6-7):569–588, 2013. 89

[126] K. Jackowski, B. Krawczyk, and M. Woźniak. **Improved Adaptive Splitting and Selection: the Hybrid Training Method of a Classifier Based on a Feature Space Partitioning**. *Int. J. Neural Syst.*, **24**(3), 2014. 89, 94

[127] K. Jackowski and M. Woźniak. **Algorithm of designing compound recognition system on the basis of combining classifiers with simultaneous splitting feature space into competence areas**. *Pattern Analysis and Applications*, **12**(4):415–425, 2009. 88, 94

[128] K. Jackowski and M. Woźniak. **Method of classifier selection using the genetic approach**. *Expert Systems*, **27**(2):114–128, 2010. 44, 89

[129] R. A. Jacobs. **Methods For Combining Experts' Probability Assessments**. *Neural Computation*, **7**(5):867–888, 1995. 52

[130] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. **Adaptive mixtures of local experts**. *Neural Comput.*, **3**:79–87, 1991. 53

[131] A. K. Jain. **Data clustering: 50 years beyond K-means**. *Pattern Recognition Letters*, **31**(8):651–666, 2010. 8, 34

[132] A.K. Jain, R.P.W. Duin, and M. Jianchang. **Statistical pattern recognition: a review**. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **22**(1):4 –37, jan 2000. 7, 14, 28, 42, 151

[133] N. Japkowicz, C. Myers, and M.A. Gluck. **A Novelty Detection Approach to Classification**. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes*, pages 518–523, 1995. 19, 21, 31, 35

[134] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2002. 107

[135] V. Jumutc and J.A.K. Suykens. **Multi-class supervised novelty detection**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **36**(12):2510–2523, 2014. 29

[136] P. Juszczak. *Learning to recognise. A study on one-class classification and active learning*. PhD thesis, Delft University of Technology, 2006. 64, 128, 129

[137] P. Juszczak and R.P.W. Duin. **Combining one-class classifiers to classify missing data**. *Lecture Notes in Computer Science*, **3077**:92–101, 2004. 56

[138] P. Juszczak, D. M. J. Tax, E. Pekalska, and R. P. W. Duin. **Minimum spanning tree based one-class classifier**. *Neurocomputing*, **72**(7-9):1859–1869, 2009. 30, 31

[139] T. Kacprzak, K. Walkowiak, and M. Woźniak. **Optimization of overlay distributed computing systems for multiple classifier system - heuristic approach**. *Logic Journal of the IGPL*, **20**(4):677–688, 2012. 44

[140] I. Kang, M.K. Jeong, and D. Kong. **A differentiated one-class classification method with applications to intrusion detection**. *Expert Systems with Applications*, **39**(4):3899–3905, 2012. 19

[141] S. Katagiri and S. Abe. **Incremental training of support vector machines using hyperspheres**. *Pattern Recognition Letters*, **27**(13):1495–1507, 2006. 40

[142] N.M. Khan, R. Ksantini, I. Shafiq Ahmad, and L. Guan. **Covariance-guided One-Class Support Vector Machine**. *Pattern Recognition*, **47**(6):2165–2177, 2014. 39

[143] T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano. **Comparing boosting and bagging techniques with noisy and imbalanced data**. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, **41**(3):552–568, 2011. 46

[144] G. Kim, S. Lee, and S. Kim. **A novel hybrid intrusion detection method integrating anomaly detection with misuse detection**. *Expert Systems with Applications*, **41**(4 PART 2):1690–1700, 2014. 19

[145] R. Kohavi and D. H. Wolpert. **Bias plus variance decomposition for zero-one loss functions**. In *ICML*, pages 275–283, 1996. 15

[146] T. Kohonen. **Essentials of the self-organizing map**. *Neural Networks*, **37**:52–65, 2013. 35

[147] B. Krawczyk. **Diversity in Ensembles for One-Class Classification**. In M. Pechenizkiy and M. Wojciechowski, editors, *New Trends in Databases and Information Systems*, **185** of *Advances in Intelligent Systems and Computing*, pages 119–129. Springer Berlin Heidelberg, 2012. 132

[148] B. Krawczyk. **One-class classifier ensemble pruning and weighting with firefly algorithm**. *Neurocomputing*, **150**:490–500, 2015. 148

[149] B. Krawczyk and P. Filipczuk. **Cytological image analysis with firefly nuclei detection and hybrid one-class classification decomposition**. *Eng. Appl. of AI*, **31**:126–135, 2014. 190

[150] B. Krawczyk, P. Filipczuk, and M. Woźniak. **Adaptive Splitting and Selection Algorithm for Classification of Breast Cytology Images**. In *Computational Collective Intelligence. Technologies and Applications - 4th International Conference, ICCCI 2012, Ho Chi Minh City, Vietnam, November 28-30, 2012, Proceedings, Part I*, pages 475–484, 2012. 190

[151] B. Krawczyk, L. Jelen, A. Krzyzak, and T. Fevens. **One-Class Classification Decomposition for Imbalanced Classification of Breast Cancer Malignancy Data**. In *Artificial Intelligence and Soft Computing - 13th International Conference, ICAISC 2014, Zakopane, Poland, June 1-5, 2014, Proceedings, Part I*, pages 539–550, 2014. 190

[152] B. Krawczyk and G. Schaefer. **Effective multiple classifier systems for breast thermogram analysis**. In *Proceedings of the 21st International Conference on Pattern Recognition, ICPR 2012, Tsukuba, Japan, November 11-15, 2012*, pages 3345–3348, 2012. 190

[153] B. Krawczyk and G. Schaefer. **Breast Thermogram Analysis Using Classifier Ensembles and Image Symmetry Features**. *IEEE Systems Journal*, **8**(3):921–928, 2014. 190

[154] B. Krawczyk and G. Schaefer. **A hybrid classifier committee for analysing asymmetry features in breast thermograms**. *Appl. Soft Comput.*, **20**:112–118, 2014. 190

[155] B. Krawczyk, G. Schaefer, and M. Woźniak. **Combining one-class classifiers for imbalanced classification of breast thermogram features**. In *2013 Fourth International Workshop on Computational Intelligence in Medical Imaging, CIMI 2013, Singapore, April 16-19, 2013*, pages 36–41, 2013. 198

[156] B. Krawczyk, G. Schaefer, and M. Woźniak. **A cost-sensitive ensemble classifier for breast cancer classification**. In *IEEE 8th International Symposium on Applied Computational Intelligence and Informatics, SACI 2013, Timisoara, Romania, May 23-25, 2013*, pages 427–430, 2013. 12, 190

[157] B. Krawczyk, J. Stefanowski, and M. Woźniak. **Data stream classification and big data analytics**. *Neurocomputing*, **150**:238–239, 2015. 9

[158] B. Krawczyk and M. Woźniak. **Designing Cost-Sensitive Ensemble – Genetic Approach**. In R. Choras, editor, *Image Processing and Communications Challenges 3*, **102** of *Advances in Intelligent and Soft Computing*, pages 227–234. Springer Berlin / Heidelberg, 2011. 50

[159] B. Krawczyk and M. Woźniak. **Privacy Preserving Models of k-NN Algorithm**. In R. Burduk, M. Kurzynski, M. Woźniak, and A. Zołnierek, editors, *Computer Recognition Systems 4*, **95** of *Advances in Intelligent and Soft Computing*, pages 207–217. Springer Berlin Heidelberg, 2011. 44

[160] B. Krawczyk and M. Woźniak. **Analysis of Diversity Assurance Methods for Combined Classifiers**. In Ryszard S. Choras, editor, *Image Processing and Communications Challenges 4*, **184** of *Advances in Intelligent Systems and Computing*, pages 179–186. Springer Berlin Heidelberg, 2012. 131

[161] B. Krawczyk and M. Woźniak. **Combining Diverse One-Class Classifiers**. In *Hybrid Artificial Intelligent Systems*, **7209** of *Lecture Notes in Computer Science*, pages 590–601. Springer Berlin / Heidelberg, 2012. 132, 133

[162] B. Krawczyk and M. Woźniak. **Accuracy and diversity in classifier selection for one-class classification ensembles**. In *Proceedings of the IEEE Symposium on Computational Intelligence and Ensemble Learning, CIEL 2013, IEEE Symposium Series on Computational Intelligence (SSCI), 16-19 April 2013, Singapore*, pages 46–51, 2013. 132

[163] B. Krawczyk and M. Woźniak. **Distributed Privacy-Preserving Minimal Distance Classification**. In *Hybrid Artificial Intelligent Systems - 8th International Conference, HAIS 2013, Salamanca, Spain, September 11-13, 2013. Proceedings*, pages 462–471, 2013. 44

[164] B. Krawczyk and M. Woźniak. **On diversity measures for fuzzy one-class classifier ensembles**. In *Proceedings of the IEEE Symposium on Computational Intelligence and Ensemble Learning, CIEL 2013, IEEE Symposium Series on Computational Intelligence (SSCI), 16-19 April 2013, Singapore*, pages 60–65, 2013. 126, 127

[165] B. Krawczyk and M. Woźniak. **Pruning One-Class Classifier Ensembles by Combining Sphere Intersection and Consistency Measures**. In *Artificial Intelligence and Soft*

Computing - 12th International Conference, ICAISC 2013, Zakopane, Poland, June 9-13, 2013, Proceedings, Part I, pages 426–436, 2013. 132

[166] B. Krawczyk and M. Woźniak. **Diversity measures for one-class classifier ensembles**. *Neurocomputing*, **126**:36 – 44, 2014. 126, 127

[167] B. Krawczyk and M. Woźniak. **Experiments on simultaneous combination rule training and ensemble pruning algorithm**. In *2014 IEEE Symposium on Computational Intelligence in Ensemble Learning, CIEL 2014, Orlando, FL, USA, December 9-12, 2014*, pages 1–6, 2014. 54

[168] B. Krawczyk and M. Woźniak. **Influence of Distance Measures on the Effectiveness of One-Class Classification Ensembles**. *Applied Artificial Intelligence*, **28**(3):258–271, 2014. 30, 36

[169] B. Krawczyk and M. Woźniak. **New untrained aggregation methods for classifier combination**. In *2014 International Joint Conference on Neural Networks, IJCNN 2014, Beijing, China, July 6-11, 2014*, pages 617–622, 2014. 52

[170] B. Krawczyk and M. Woźniak. **Optimization Algorithms for One-Class Classification Ensemble Pruning**. In *Intelligent Information and Database Systems - 6th Asian Conference, ACIIDS 2014, Bangkok, Thailand, April 7-9, 2014, Proceedings, Part II*, pages 127–136, 2014. 131

[171] B. Krawczyk and M. Woźniak. **Untrained Method for Ensemble Pruning and Weighted Combination**. In *Advances in Neural Networks - ISNN 2014 - 11th International Symposium on Neural Networks, ISNN 2014, Hong Kong and Macao, China, November 28- December 1, 2014. Proceedings*, pages 358–365, 2014. 53

[172] B. Krawczyk and M. Woźniak. **Pruning Ensembles of One-Class Classifiers with X-means Clustering**. In *Intelligent Information and Database Systems - 7th Asian Conference, ACIIDS 2015, Bali, Indonesia, March 23-25, 2015, Proceedings, Part I*, pages 484–493, 2015. 140

[173] B. Krawczyk, M. Woźniak, and B. Cyganek. **Clustering-based ensembles for one-class classification**. *Information Sciences*, **264**:182–195, 2014. 61, 67, 114

[174] B. Krawczyk, M. Woźniak, and B. Cyganek. **Weighted One-Class Classifier Ensemble Based on Fuzzy Feature Space Partitioning**. In *22nd International Conference on Pattern Recognition, ICPR 2014, Stockholm, Sweden, August 24-28, 2014*, pages 2838–2843, 2014. 62, 67

[175] B. Krawczyk, M. Woźniak, and F. Herrera. **Weighted one-class classification for different types of minority class examples in imbalanced data**. In *2014 IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2014, Orlando, FL, USA, December 9-12, 2014*, pages 337–344, 2014. 198

[176] B. Krawczyk, M. Woźniak, and F. Herrera. **On the Usefulness of One-Class Classifier Ensembles for Decomposition of Multi-Class Problems**. *Pattern Recognition*, 2015. Available from: `http://www.sciencedirect.com/science/article/pii/S0031320315002216`. 28, 66

[177] B. Krawczyk, M. Woźniak, and G. Schaefer. **Cost-sensitive decision tree ensembles for effective imbalanced classification**. *Appl. Soft Comput.*, **14**:554–562, 2014. 197

[178] M.M. Krell and H. Wöhrle. **New one-class classifiers based on the origin separation approach**. *Pattern Recognition Letters*, **53**:93–99, 2015. 30

[179] A. Krogh and J. Vedelsby. **Neural network ensembles, cross validation, and active learning**. *Advances in neural information processing systems*, pages 231–238, 1995. 45

[180] L. Kuncheva, J. C. Bezdek, and R. P. W. Duin. **Decision templates for multiple classifier fusion: an experimental comparison**. *Pattern Recognition*, **34**(2):299–314, 2001. 54

[181] L. I. Kuncheva. **Using measures of similarity and inclusion for multiple classifier fusion by decision templates**. *Fuzzy Sets and Systems*, **122**(3):401–407, 2001. 54

[182] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004. 43, 45

[183] L. I. Kuncheva and Ch. J. Whitaker. **Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy**. *Machine learning*, **51**(2):181–207, 2003. 49

[184] L.I. Kuncheva. **Clustering-and-selection model for classifier combination**. In *Knowledge-Based Intelligent Engineering Systems and Allied Technologies, 2000. Proceedings. Fourth International Conference on*, **1**, pages 185 –188 vol.1, 2000. 50, 51, 88

[185] L.I. Kuncheva. **Switching between selection and fusion in combining classifiers: An experiment**. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, **32**(2):146–156, 2002. 48

[186] L.I. Kuncheva and C.J. Whitaker. **Ten measures of diversity in classifier ensembles: limits for two classifiers**. In *Intelligent Sensor Processing, A DERA/IEE Workshop on*, pages 1001 – 1010, 2001. 49

[187] M.B. Kursa. **rFerns: An implementation of the random ferns method for general-purpose machine learning**. *Journal of Statistical Software*, **61**(10):1–13, 2014. 47

[188] M. Kurzynski and M. Woźniak. **Combining classifiers under probabilistic models: experimental comparative analysis of methods**. *Expert Systems*, **29**(4):374–393, 2012. 43

[189] A. Lazarevic and Z. Obradovic. **Effective pruning of neural network classifier ensembles**. In *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on*, **2**, pages 796–801. IEEE, 2001. 50

[190] F. Lecron, J. Boisvert, S. Mahmoudi, H. Labelle, and M. Benjelloun. **Three-dimensional spine model reconstruction using one-class SVM regularization**. *IEEE Transactions on Biomedical Engineering*, **60**(11):3256–3264, 2013. 40

[191] K. Lee, D.-W. Kim, D. Lee, and K.H. Lee. **Improving support vector data description using local density degree**. *Pattern Recognition*, **38**(10):1768–1771, 2005. 42

[192] K.Y. Lee, D.-W. Kim, K.H. Lee, and D. Lee. **Density-induced support vector data description**. *IEEE Transactions on Neural Networks*, **18**(1):284–289, 2007. 42

[193] S.-W. Lee, J. Park, and S.-W. Lee. **Low resolution face recognition based on support vector data description**. *Pattern Recognition*, **39**(9):1809–1812, 2006. 42

[194] L. Lei, W. Xiao-dan, L. Xi, and S. Ya-fei. **Hierarchical error-correcting output codes based on SVDD**. *Pattern Analysis and Applications*, 2015. 57

[195] C. Li, C. Xu, C. Gui, and M.D. Fox. **Level Set Evolution Without Re-initialization: A New Variational Formulation.** In *IEEE Proceedings of Conference on Computer Vision and Pattern Recognition 2005*, pages 430–436, 2005. 194

[196] L. Li, Q. Hu, X. Wu, and D. Yu. **Exploration of classification confidence in ensemble learning**. *Pattern Recognition*, **47**(9):3120–3131, 2014. 48

[197] C. Lin, W. Chen, C. Qiu, Y. Wu, S. Krishnan, and Q. Zou. **LibD3C: Ensemble classifiers with a clustering and dynamic selection strategy**. *Neurocomputing*, **123**:424–435, 2014. 50

[198] N. Lipka, B. Stein, and M. Anderka. **Cluster-based one-class ensemble for classification problems in information retrieval**. pages 1041–1042, 2012. 55

[199] B. Liu, Y. Xiao, L. Cao, Z. Hao, and F. Deng. **SVDD-based outlier detection on uncertain data**. *Knowledge and Information Systems*, **34**(3):597–618, 2013. 42

[200] B. Liu, Y. Xiao, P.S. Yu, L. Cao, Y. Zhang, and Z. Hao. **Uncertain one-class learning and concept summarization learning on uncertain data streams**. *IEEE Transactions on Knowledge and Data Engineering*, **26**(2):468–484, 2014. 19

[201] J. Liu, J. Song, Q. Miao, and Y. Cao. **FENOC: An ensemble one-class learning framework for malware detection**. pages 523–527, 2013. 55

[202] R. Liu and B. Yuan. **Multiple classifiers combination by clustering and selection**. *Information Fusion*, **2**(3):163–168, 2001. 50

[203] X. Liu, J. Wu, and Z. Zhou. **Exploratory undersampling for class-imbalance learning**. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, **39**(2):539–550, 2009. 203

[204] Y.-H. Liu, Y.-C. Liu, and Y.-J. Chen. **Fast support vector data descriptions for novelty detection**. *IEEE Transactions on Neural Networks*, **21**(8):1296–1313, 2010. 42

[205] A.C. Lozano, S.R. Kulkarni, and R.E. Schapire. **Convergence and Consistency of Regularized Boosting With Weakly Dependent Observations**. *IEEE Transactions on Information Theory*, **60**(1):651–660, 2014. 46

[206] R. Lysiak, M. Kurzynski, and T. Woloszynski. **Optimal selection of ensemble classifiers using measures of competence and diversity of base classifiers**. *Neurocomputing*, **126**:29–35, 2014. 49, 135

[207] M. Maciejewska and A. Szczurek. **Representativeness of shorter measurement sessions in long-term indoor air monitoring**. *Environ. Sci.: Processes Impacts*, **17**:381–388, 2015. 169

[208] M. Maciejewska, A. Szczurek, G. Sikora, and A. Wyłomańska. **Diffusive and subdiffusive dynamics of indoor microclimate: A time series modeling**. *Phys. Rev. E*, **86**:031128, 2012. 166

[209] S. Maldonado and C. Montecinos. **Robust classification of imbalanced data using one-class and two-class SVM-based multiclassifiers**. *Intelligent Data Analysis*, **18**(1):95–112, 2014. 19

[210] L. Manevitz and M. Yousef. **One-class document classification via Neural Networks**. *Neurocomputing*, **70**(7-9):1466–1481, 2007. 30, 31, 35

[211] G. Martinez-Munoz, D. Hernandez-Lobato, and A. Suarez. **An analysis of ensemble pruning techniques based on ordered aggregation**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **31**(2):245–259, 2009. 50

[212] F. Masulli and G. Valentini. **Comparing decomposition methods for classification**. *International Conference on Knowledge-Based Intelligent Electronic Systems, Proceedings, KES*, **2**:788–791, 2000. 54

[213] T. M. Mitchell. *Machine learning.* McGraw-Hill Boston, MA:, 1997. 15

[214] M. Moya and D. Hush. **Network constraints and multi-objective optimization for one-class classification**. *Neural Networks*, **9**(3):463–474, 1996. 19

[215] T. Mu and A.K. Nandi. **Multiclass classification based on extended support vector data description**. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, **39**(5):1206–1216, 2009. 28

[216] I. Mukherjee, C. Rudin, and R.E. Schapire. **The rate of convergence of AdaBoost**. *Journal of Machine Learning Research*, **14**(1):2315–2347, 2013. 46

[217] I. Mukherjee and R.E. Schapire. **A theory of multiclass boosting**. *Journal of Machine Learning Research*, **14**(1):437–497, 2013. 46

[218] J. Munoz-Mari, F. Bovolo, L. Gomez-Chova, L. Bruzzone, and G. Camp-Valls. **Semisupervised one-class support vector machines for classification of remote sensing data**. *IEEE Transactions on Geoscience and Remote Sensing*, **48**(8):3188–3197, 2010. 40

[219] S. Osher and J.A. Sethian. **Fronts Propagating with Curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations**. *J Comp. Phys.*, **79**:12–49, 1988. 194, 195

[220] M. Özuysal, M. Calonder, V. Lepetit, and P. Fua. **Fast Keypoint Recognition Using Random Ferns**. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **32**(3):448–461, 2010. 47

[221] D. Partridge and W. Krzanowski. **Software diversity: practical statistics for its measurement and exploitation**. *Information and Software Technology*, **39**(10):707 – 717, 1997. 49

[222] D. Pelleg and A. W. Moore. **X-means: Extending K-means with Efficient Estimation of the Number of Clusters**. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, pages 727–734, 2000. 142

[223] D. Pelleg and A.W. Moore. **Accelerating Exact k-means Algorithms with Geometric Reasoning**. In *KDD*, pages 277–281, 1999. 143

[224] X. Peng and D. Xu. **Efficient support vector data descriptions for novelty detection**. *Neural Computing and Applications*, **21**(8):2023–2032, 2012. 42

[225] X. Peng and D. Xu. **Twin support vector hypersphere (TSVH) classifier for pattern recognition**. *Neural Computing and Applications*, **24**(5):1207–1220, 2014. 30

[226] I.T. Podolak and Roman A. **Theoretical Foundations and Experimental Results for a Hierarchical Classifier with Overlapping Clusters**. *Computational Intelligence*, **29**(2):357–388, 2013. 51

[227] R. Polikar. **Ensemble Based Systems in Decision Making**. *IEEE Circuits and Systems Magazine*, **6**(3):21–45, 2006. 42

[228] J.R. Quinlan. **Simplifying decision trees**. *International Journal of Man-Machine Studies*, **27**(3):221–234, 1987. 48

[229] L.A. Rastrigin and R. H. Erenstein. *Method of Collective Recognition.* Energoizdat, Moscow, 1981. 51

[230] G. Ratsch, S. Mika, B. Scholkopf, and K. . Muller. **Constructing boosting algorithms from SVMs: An application to one-class classification**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**(9):1184–1199, 2002. 56

[231] J. Read, B. Pfahringer, G. Holmes, and E. Frank. **Classifier chains for multi-label classification**. *Machine learning*, **85**(3):333–359, 2011. 9

[232] R. Reed. **Pruning algorithms - a survey**. *IEEE Transactions on Neural Networks*, **4**(5):740–747, 1993. 48

[233] J.A. Reyes and D. Gilbert. **Combining one-class classification models based on diverse biological data for prediction of protein-protein interactions**. *Lecture Notes in Bioinformatics*, **5109 LNBI**:177–191, 2008. 56

[234] T.W. Ridler and S. Calvard. **Picture Thresholding Using an Iterative Selection.** *IEEE Trans. System, Man and Cybernetics*, **8**:630–632, 1978. 193

[235] J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso. **Rotation forest: A new classifier ensemble method**. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **28**(10):1619–1630, 2006. 47, 105

[236] F. Roli, G. Giacinto, and G. Vernazza. **Methods for designing multiple classifier systems**. In *Multiple Classifier Systems*, pages 78–87. Springer, 2001. 45

[237] D. Ruta and B. Gabrys. **Classifier selection for majority voting**. *Information Fusion*, **6**(1):63–81, 2005. 50

[238] M. J. Saberian and N. Vasconcelos. **Multiclass boosting: Theory and algorithms**. In *Advances in Neural Information Processing Systems*, pages 2124–2132, 2011. 46

[239] J.A. Sáez, M. Galar, J. Luengo, and F. Herrera. **Analyzing the presence of noise in multi-class problems: alleviating its influence with the One-vs-One decomposition**. *Knowl. Inf. Syst.*, **38**(1):179–206, 2014. 11, 51

[240] C. Sanchez-Hernandez, D.S. Boyd, and G.M. Foody. **One-class classification for mapping a specific land-cover class: SVDD classification of fenland**. *IEEE Transactions on Geoscience and Remote Sensing*, **45**(4):1061–1072, 2007. 42

[241] T.H. Sarma, P. Viswanath, and B.E. Reddy. **A hybrid approach to speed-up the k-means clustering method**. *International Journal of Machine Learning and Cybernetics*, **4**(2):107–117, 2013. 64

[242] R. E. Schapire and Y. Freund. *Boosting: Foundations and algorithms*. MIT press, 2012. 46

[243] R. E. Schapire and Y. Singer. **Improved boosting algorithms using confidence-rated predictions**. *Machine learning*, **37**(3):297–336, 1999. 46

[244] B. Schölkopf and A.J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. Adaptive computation and machine learning. MIT Press, 2002. 31, 36, 37

[245] C. Seiffert, T.M. Khoshgoftaar, J. Van Hulse, and A. Napolitano. **RUSBoost: A hybrid approach to alleviating class imbalance**. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, **40**(1):185–197, 2010. 46, 203

[246] G. Sewell. *Computational Methods of Linear Algebra*. World Scientific Publishing Company; 3rd Revised edition edition, 2014. 33

[247] A. D. Shieh and D. F. Kamm. **Ensembles of One Class Support Vector Machines**. In *Multiple Classifier Systems, 8th International Workshop, MCS 2009*, **5519** of *Lecture Notes in Computer Science*, pages 181–190. Springer, 2009. 55, 56

[248] H.W. Shin and S.Y. Sohn. **Selected tree classifier combination based on both accuracy and error diversity**. *Pattern Recognition*, **38**(2):191–197, 2005. 49

[249] M. Signoretto, L. De Lathauwer, and J.A.K. Suykens. **A kernel-based framework to tensorial data analysis**. *Neural Networks*, **24**(0):861–874, 2011. 177, 181

[250] M. Skurichina and R. P.W. Duin. **Bagging for linear classifiers**. *Pattern Recognition*, **31**(7):909–930, 1998. 46

[251] S. Sonnenburg, G. Ratsch, C. Schafer, and B. Scholkopf. **Large scale multiple kernel learning**. *Journal of Machine Learning Research*, **7**:1531–1565, 2006. 30

[252] V. Soto, S. Garcia-Moratilla, G. Martinez-Munoz, D. Hernandez-Lobato, and A. Suarez. **A double pruning scheme for boosting ensembles**. *IEEE Transactions on Cybernetics*, **44**(12):2682–2695, 2014. 48

[253] E.J. Spinosa and A.C.P.L.F. De Carvalho. **Combining one-class classifiers for robust novelty detection in gene expression data**. **3594**, pages 54–64, 2005. 56

[254] D. Steinley and M. J. Brusco. **Initializing K-means batch clustering: a critical evaluation of several techniques**. *Journal of Classification*, **24**(1):99–121, 2007. 34

[255] M. Sugiyama. **Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis**. *The Journal of Machine Learning Research*, **8**:1027–1061, 2007. 107

[256] H. Sun, S. Wang, and Q. Jiang. **FCM-Based Model Selection Algorithms for Determining the Number of Clusters**. *Pattern Recognition*, **37**(10):2027 – 2037, 2004. 69

[257] Y. Sun, M. S. Kamel, A. K. C. Wong, and Y. Wang. **Cost-sensitive boosting for classification of imbalanced data**. *Pattern Recognition*, **40**(12):3358–3378, 2007. 46

[258] Y. Sun, A. K. C. Wong, and M. S. Kamel. **Classification of imbalanced data: A review**. *International Journal of Pattern Recognition and Artificial Intelligence*, **23**(4):687–719, 2009. 9, 197

[259] A. Szczurek, B. Krawczyk, and M. Maciejewska. **VOCs classification based on the committee of classifiers coupled with single sensor signals**. *Chemometrics and Intelligent Laboratory Systems*, **125**(0):1 – 10, 2013. 167, 168, 170, 172

[260] A. Szczurek and M. Maciejewska. **“Artificial sniffing” based on induced temporary disturbance of gas sensor response**. *Sensors and Actuators B: Chemical*, **186**(0):109 – 116, 2013. 166

[261] A. Szczurek and M. Maciejewska. **Gas sensing method applicable to real conditions**. *Measurement Science and Technology*, **24**(4):045103, 2013. 167

[262] A. Szczurek, M. Maciejewska, and B. BFlisowska-Wiercik. **Method of gas mixtures discrimination based on sensor array, temporal response and data driven approach**. *Talanta*, **83**(3):916 – 923, 2011. 166

[263] L. Tarassenko, P. Hayton, N. Cerneaz, and M. Brady. **Novelty detection for the identification of masses in mammograms**. In *Fourth International Conference on Artificial Neural Networks*, pages 442–447, 1995. 31

[264] D. M. J. Tax and R. P. W. Duin. **Combining One-Class Classifiers**. In *Proceedings of the Second International Workshop on Multiple Classifier Systems*, MCS '01, pages 299–308, 2001. 55

[265] D. M. J. Tax and R. P. W. Duin. **Support Vector Data Description**. *Machine Learning*, **54**(1):45–66, 2004. 30, 31, 40, 64

[266] D. M. J. Tax and K. . Müller. **A consistency-based model selection for one-class classification**. In *Proceedings - International Conference on Pattern Recognition*, **3**, pages 363–366, 2004. 26

[267] D. M.J. Tax. *One-class classification.* TU Delft, Delft University of Technology, 2001. 19, 23, 26

[268] D.M.J. Tax and R. P. W. Duin. **Outlier Detection Using Classifier Instability**. In *Advances in Pattern Recognition, Joint IAPR International Workshops SSPR '98 and SPR '98, Sydney, NSW, Australia, August 11-13, 1998, Proceedings*, pages 593–601, 1998. 21

[269] D.M.J. Tax and R. P. W. Duin. **Support vector domain description**. *Pattern Recognition Letters*, **20**(11-13):1191–1199, 1999. 22, 36

[270] D.M.J. Tax and R. P. W. Duin. **Uniform Object Generation for Optimizing One-class Classifiers**. *Journal of Machine Learning Research*, **2**:155–173, 2001. 22

[271] D.M.J. Tax and R. P. W. Duin. **Characterizing one–class datasets**. In *Proceedings of the Sixteenth Annual Symposium of the Pattern Recognition Association of South Africa*, pages 21–26, 2005. 21, 215

[272] D.M.J. Tax, P. Juszczak, E. Pekalska, and R.P.W. Duin. **Outlier Detection Using Ball Descriptions with Adjustable Metric**. In *Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshops, SSPR 2006 and SPR 2006, Hong Kong, China, August 17-19, 2006, Proceedings*, pages 587–595, 2006. 30, 31

[273] O. Taylor and J. MacIntyre. **Adaptive local fusion systems for novelty detection and diagnostics in condition monitoring**. In *Proceedings of SPIE - The International Society for Optical Engineering*, **3376**, pages 210–218, 1998. 30, 31

[274] K. Ting, J. Wells, S. Tan, S. Teng, and G. Webb. **Feature-subspace aggregating: ensembles for stable and unstable learners**. *Machine Learning*, **82**:375–397, 2011. 45

[275] W. G. Touw, J. R. Bayjanov, L. Overmars, L. Backus, J. Boekhorst, M. Wels, and S. A.F.T. van Hijum. **Data mining in the Life Sciences with Random Forest: a walk in the park or lost in the jungle?** *Briefings in bioinformatics*, page bbs034, 2012. 47

[276] G. Tsoumakas, I. Katakis, and I. Vlahavas. **Effective voting of heterogeneous classifiers**. In *Machine Learning: ECML 2004*, pages 465–476. Springer, 2004. 46

[277] G. Tsoumakas, I. Partalas, and I. Vlahavas. **An ensemble pruning primer**. *Studies in Computational Intelligence*, **245**:1–13, 2009. 48, 153

[278] K. Tumer and J. Ghosh. **Analysis of decision boundaries in linearly combined neural classifiers**. *Pattern Recognition*, **29**(2):341 – 348, 1996. 44

[279] M. van Erp, L. Vuurpijl, and L. Schomaker. **An overview and comparison of voting methods for pattern recognition**. In *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on*, pages 195 – 200, 2002. 51

[280] V. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998. 16

[281] A. Verikas, A. Gelzinis, and M. Bacauskiene. **Mining data with random forests: A survey and results of new tests**. *Pattern Recognition*, **44**(2):330–349, 2011. 47

[282] K. Walkowiak, Sz. Sztajer, and M. Woźniak. **Decentralized Distributed Computing System for Privacy-Preserving Combined Classifiers - Modeling and Optimization**. In *Computational Science and Its Applications - ICCSA 2011 - International Conference, Santander, Spain, June 20-23, 2011. Proceedings, Part I*, pages 512–525, 2011. 44

[283] A. Wang, G. Wan, Z. Cheng, and S. Li. **An incremental extremely random forest classifier for online learning and tracking**. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 1449–1452. IEEE, 2009. 47

[284] S. Wang and X. Yao. **Diversity analysis on imbalanced data sets by using ensemble models**. In *2009 IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2009 - Proceedings*, pages 324–331, 2009. 203

[285] X. Wang, F.-L. Chung, and S. Wang. **Theoretical analysis for solution of support vector data description**. *Neural Networks*, **24**(4):360–369, 2011. 42

[286] T. Wilk and M. Woźniak. **Complexity and Multithreaded Implementation Analysis of One Class-Classifiers Fuzzy Combiner**. In E. Corchado, M. Kurzynski, and M. Woźniak, editors, *Hybrid Artificial Intelligent Systems*, **6679** of *Lecture Notes in Computer Science*, pages 237–244. Springer Berlin / Heidelberg, 2011. 44

[287] T. Wilk and M. Woźniak. **Comparison of Fuzzy Combiner Training Methods**. In *Computational Collective Intelligence. Technologies and Applications - 4th International Conference, ICCCI 2012, Ho Chi Minh City, Vietnam, November 28-30, 2012, Proceedings, Part I*, pages 166–173, 2012. 57

[288] T. Wilk and M. Woźniak. **Soft computing methods applied to combination of one-class classifiers**. *Neurocomput.*, **75**:185–193, January 2012. 57

[289] T. Windeatt and R. Ghaderi. **Coding and decoding strategies for multi-class learning problems**. *Information Fusion*, **4**(1):11–21, 2003. 54

[290] T. Windeatt and C. Zor. **Ensemble pruning using spectral coefficients**. *IEEE Transactions on Neural Networks and Learning Systems*, **24**(4):673–678, 2013. 50

[291] D.H. Wolpert. **The supervised learning no-free-lunch Theorems**. In *In Proc. 6th Online World Conference on Soft Computing in Industrial Applications*, pages 25–42, 2001. 14, 15, 16, 42

[292] K. Woods, Jr. Kegelmeyer, W.P., and K. Bowyer. **Combination of multiple classifiers using local accuracy estimates**. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **19**(4):405 –410, 1997. 52

[293] M. Wo zniak. **Experiments on Linear Combiners**. In E. Pietka and J. Kawa, editors, *Information Technologies in Biomedicine*, **47** of *Advances in Soft Computing*, pages 445–452. Springer Berlin / Heidelberg, 2008. 53

[294] M. Woźniak, P. Cal, and B. Cyganek. **The Influence of a Classifiers' Diversity on the Quality of Weighted Aging Ensemble**. In *Intelligent Information and Database Systems - 6th Asian Conference, ACIIDS 2014, Bangkok, Thailand, April 7-9, 2014, Proceedings, Part II*, pages 90–99, 2014. 49, 135

[295] M. Woźniak, M. Grana, and E. Corchado. **A survey of multiple classifier systems as hybrid systems**. *Information Fusion*, **16**(1):3–17, 2014. 42, 94

[296] M. Woźniak and K. Jackowski. **Some Remarks on Chosen Methods of Classifier Fusion Based on Weighted Voting**. In E. Corchado, X. Wu, E. Oja, A. Herrero, and B. Baruque, editors, *Hybrid Artificial Intelligence Systems*, **5572** of *Lecture Notes in Computer Science*, pages 541–548. Springer Berlin / Heidelberg, 2009. 52

[297] M. Woźniak and B. Krawczyk. **Combined classifier based on feature space partitioning**. *Journal of Applied Mathematics and Computer Science*, **22**(4):855–866, 2012. 89

[298] M. Woźniak and M. Zmyslony. **Designing combining classifier with trained fuser - analytical and experimental evaluation**. *Neural Network World*, **20**(7):925–934, 2010. 52

[299] Z.-D. Wu, W.-X. Xie, and J.-P. Yu. **Fuzzy c-means clustering algorithm based on kernel method**. In *Computational Intelligence and Multimedia Applications, 2003. ICCIMA 2003. Proceedings. Fifth International Conference on*, pages 49–54. IEEE, 2003. 68

[300] J. Xia, P. Du, X. He, and J. Chanussot. **Hyperspectral remote sensing image classification based on rotation forest**. *IEEE Geoscience and Remote Sensing Letters*, **11**(1):239–243, 2014. 47

[301] Y. Xiao, H. Wang, and W. Xu. **Parameter Selection of Gaussian Kernel for One-Class SVM**. *IEEE Transactions on Cybernetics*, 2014. 39

[302] Y. Xiao, H. Wang, L. Zhang, and W. Xu. **Two methods of selecting Gaussian kernel parameters for one-class SVM and their application to fault detection**. *Knowledge-Based Systems*, **59**:75–84, 2014. 39

[303] W. Xie, S. Uhlmann, S. Kiranyaz, and M. Gabbouj. **Incremental learning with support vector data description**. pages 3904–3909, 2014. 42

[304] L. Xu, A. Krzyzak, and C.Y. Suen. **Methods of combining multiple classifiers and their applications to handwriting recognition**. *Systems, Man and Cybernetics, IEEE Transactions on*, **22**(3):418 –435, may/jun 1992. 51

[305] Y. Xu and C. Liu. **A rough margin-based one class support vector machine**. *Neural Computing and Applications*, **22**(6):1077–1084, 2013. 39

[306] X. . Yang. **Engineering Optimization: An Introduction with Metaheuristic Applications**, 2010. 149

[307] C.-Y. Yeh, Z.-Y. Lee, and S.-J. Lee. **Boosting one-class support vector machines for multi-class classification**. *Applied Artificial Intelligence*, **23**(4):297–315, 2009. 56, 83, 87

[308] A. Ypma and R.P.W. Duin. **Support objects for domain approximation**. In L. Niklasson, M. Boden, and T. Ziemke, editors, *ICANN 98*, Perspectives in Neural Computing, pages 719–724. 1998. 31

[309] G. Yu, J. Yuan, and Z. Liu. **Unsupervised random forest indexing for fast action search**. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 865–872. IEEE, 2011. 47

[310] E.A. Zanaty. **Determining the number of clusters for kernelized fuzzy C-means algorithms for automatic medical image segmentation**. *Egyptian Informatics Journal*, **13**(1):39 – 58, 2012. 69, 72

[311] B. Zenko, L. Todorovski, and S. Dzeroski. **A comparison of stacking with meta decision trees to bagging, boosting, and stacking with other methods**. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 669–670. IEEE, 2001. 53

[312] C. Zhang and S. Zhang. *Association rule mining: models and algorithms*. Springer-Verlag, 2002. 8

[313] D. Zhang, Y. Wang, L. Cai, and L. Zhang. **A learning algorithm for one-class data stream classification based on ensemble classifier**. **2**, pages V2596–V2600, 2010. 56

[314] H. Zhang and L. Cao. **A spectral clustering based ensemble pruning approach**. *Neurocomputing*, **139**:289–297, 2014. 50, 153

[315] J. Zhang and K.-W. Chau. **Multilayer ensemble pruning via novel multi-sub-swarm particle swarm optimization**. *Journal of Universal Computer Science*, **15**(4):840–858, 2009. 50

[316] J. Zhang, J. Lu, and G. Zhang. **Combining one class classification models for avian influenza outbreaks**. pages 190–196, 2011. 55

[317] K. Zhang. **Four Changes of Modern Universities From the Perspective of “4V” of Big Data**. *Canadian Social Science*, **11**(3), 2015. 9

[318] L. Zhang, W. . Zhou, and L. . Jiao. **Kernel clustering algorithm**. *Jisuanji Xuebao/Chinese Journal of Computers*, **25**(6):587–590, 2002. 68

[319] Y. Zhang, S. Burer, and W.N. Street. **Ensemble pruning via semi-definite programming**. *Journal of Machine Learning Research*, **7**:1315–1338, 2006. 50

[320] Y. Zhang, X.-D. Liu, F.-D. Xie, and K.-Q. Li. **Fault classifier of rotating machinery based on weighted support vector data description**. *Expert Systems with Applications*, **36**(4):7928–7932, 2009. 42

[321] Y. Zhang, N. Meratnia, and P. Havinga. **Adaptive and online one-class support vector machine-based outlier detection techniques for wireless sensor networks**. pages 990–995, 2009. 40

[322] Z.-H. Zhou and M.-L. Zhang. **Solving multi-instance problems with classifier ensemble based on constructive clustering**. *Knowledge and Information Systems*, **11**(2):155–170, 2007. 45

[323] F. Zhu, N. Ye, W. Yu, S. Xu, and G. Li. **Boundary detection and sample reduction for one-class Support Vector Machines**. *Neurocomputing*, **123**:166–173, 2014. 30, 39

[324] H. Zuo, O. Wu, W. Hu, and B. Xu. **Recognition of blue movies by fusion of audio and video**. In *2008 IEEE International Conference on Multimedia and Expo, ICME 2008 - Proceedings*, pages 37–40, 2008. 30, 31

Index