One class random forests

Chesner Désir, Simon Bernard, Caroline Petitjean, Heutte Laurent

One Class Random Forests

Chesner Désir, Simon Bernard, Caroline Petitjean, Laurent Heutte

Université de Rouen, LITIS EA 4108
BP 12 - 76801 Saint-Etienne du Rouvray, France
firstname.lastname@univ-rouen.fr

Abstract

One class classification is a binary classification task for which only one class of samples is available for learning. In some preliminary works, we have proposed *One Class Random Forests* (OCRF), a method based on a random forest algorithm and an original outlier generation procedure that makes use of classifier ensemble randomization principles. In this paper, we propose an extensive study of the behavior of OCRF, that includes experiments on various UCI public datasets and comparison to reference one class algorithms – namely, gaussian density models, Parzen estimators, gaussian mixture models and One Class SVMs – with statistical significance. Our aim is to show that the randomization principles embedded in a random forest algorithm make the outlier generation process more efficient, and allow in particular to break the curse of dimensionality. *One Class Random Forests* are shown to perform well in comparison to other methods, and in particular to maintain stable performance in higher dimension, while the other algorithms may fail.

Keywords:
One class classification, supervised learning, decision trees, ensemble methods, random forests, outlier generation, outlier detection

1. Introduction

One class classification (OCC) is a binary classification task for which only one class of objects, the target class or positive class, is available for learning. Little knowledge or even no prior information about the other class, the outlier class or negative class, is available during the learning stage, most of time because these data are either difficult or impossible to collect [1]. However, such data may occur during the prediction phase. Application examples include authorship verification [2], typist or speaker recognition [3, 4], mobile-masquerader detection [5], intrusion detection [6, 7], medical diagnosis [8]. We refer the reader to [9, 10] for a more exhaustive list of OCC applicative fields.

In the OCC literature, two main types of approaches are usually proposed: (i) methods using only positive samples to learn the target concept; (ii) methods generating or

simulating negative samples so that existing multi-class classification methods may be used. The first type of approaches aims at estimating the probability density function by fitting a statistical distribution, such as a Gaussian, to the target data, and predicting as outlier any instance that exhibits a low probability of appearing. However, these methods are sensitive to an increasing number of features: an intractable amount of training samples is required in order to provide a good estimate of the distribution, even in reasonably sized feature spaces [11, 12]. The second type of approaches consists in extrapolating the missing samples so that the resulting binary classification problem can be learnt with standard (discriminative) classifiers. This extrapolation may be ensured by artificially generating outliers during training [3]. In this case, outliers are often assumed to be uniformly distributed, so as to cover the whole domain of variation of the feature space. This implies to generate an exponential and thus expensive amount of outliers with respect to the dimension of the feature space.

One solution to tackle this issue may be to use randomization principles offered by classifier ensemble approaches. These approaches, although popular, have not been exploited very much for OCC problems [13]. In some preliminary works [8, 14], we have shown that randomization principles may be used in a one-class classification task for generating outliers quite efficiently. This first solution for OCC, called One Class Random Forests (OCRF), is based on a random forest (RF) algorithm [15] and an original outlier generation procedure that makes use of the ensemble learning mechanisms offered by RF algorithms to reduce both the number of artificial outliers to generate and the size of the feature space in which they are generated. Promising but preliminary results obtained on a real-world medical problem [8] and on a few UCI datasets [14] have led us to investigate more deeply this new OCC method. We thus propose in this paper an extensive study of the behavior of OCRF, that includes experiments on various UCI public datasets and comparison to reference one class algorithms – namely, gaussian density models, Parzen estimators, gaussian mixture models and One Class SVMs – with statistical significance. Our aim is to confirm that the randomization principles embedded in the random forest algorithm make the outlier generation process more efficient, and allow in particular to break the curse of dimensionality. *One Class Random Forests* are shown to perform equally well or better than the state-of-the-art OCC methods, and in particular to maintain stable performance in higher dimension feature spaces, while the other algorithms may fail.

The remainder of the paper is organized as follows. In Section 2, we present related works on OCC. In Section 3, the One Class Random Forest method is detailed. Section 4 is devoted to the experimental protocol while results are reported in Section 5. Conclusions and future works are drawn in Section 6.

2. Related works in one class classification

Numerous reviews presenting OCC state-of-the-art have been proposed in the past decade [13, 16, 5]. Some of them specifically address OCC variants such as outlier detection [17, 9], anomaly detection [10] and novelty detection [18, 19]. In the following, we divide existing OCC methods into methods learning from available target samples only, and methods requiring the extrapolation of outlier samples. We finally conclude this brief overview by focusing on one class classifier ensemble based approaches in order to introduce and justify our contribution.

2.1. Learning from available target samples only

Learning from the available target samples only means that the classifier does not require any hypothesis on the outlier data to estimate the decision boundary. Generative methods are straightforwardly applicable to OCC as the target class may directly be modeled from the available training instances, by formulating some hypothesis on the underlying target distribution. For this reason, generative methods are the most used methods for OCC [20], even though they generally require the estimation of a large number of parameters. Generative methods include: (i) density-based methods, such as gaussian and Mixture of Gaussians estimators, (ii) distance-based approaches, such as Nearest Neighbor density estimator and k-means clustering, (iii) reconstruction approaches that encode the target data, and (iv) SVM-based data description (SVDD) [16, 19, 21].

Density-based methods aim at estimating the probability density function of the underlying distribution of the target data. The main difficulties reside in finding an appropriate model for the distribution of the training data and providing an accurate and adapted threshold on the output probability for deciding to accept or reject an input sample. Furthermore, density-based approaches require a large number of training data to obtain a reliable estimate of the probability model, in particular when the data dimensionality is high [19, 22, 12]. Well-known density-based approaches are Parzen windows and Mixture of Gaussians [23, 24, 25, 16, 26, 27]. A Parzen classifier is a non-parametric density estimator that consists in computing an identical kernel for each example of the training set and then defining a linear combination of these kernels to estimate the probability density function of the data. A gaussian kernel is generally used and the width parameter can be estimated with a leave-one-out procedure [28, 29]. As Parzen is a non parametric density estimator, the output score of a test input is better predicted when the training set is large, but high computational resources are then required. Because of the large computational cost of Parzen estimator, Mixture of Gaussians (MoG) are generally preferred. The MoG approach consists in building several density functions (kernels) to model the entire available target data set. Parameters of the

mixture are estimated on the training data: the number of gaussian kernels and the standard parameters of each kernel are estimated by maximizing the log-likelihood of the training data for the model, using standard techniques like Expectation-Maximization algorithm [30, 31, 32, 33]. However, when a small amount of data is available, the choice of the number of kernels for the MoG classifier becomes critical and a unimodal normal distribution is often used [16, 19].

Distance-based approaches, such as Nearest Neighbor (NN) methods, have also been proposed for non-parametric density estimation. The one class NN, a modified version of the classical NN, consists in computing the distance of an input $\mathbf{x}$ to the nearest neighbor $NN(\mathbf{x})$ and comparing to the distance of the nearest neighbor to its nearest neighbor ($NN(NN(\mathbf{x}))$) [19, 34]. If the first distance is larger than the second, then the input example is considered as outlier. A *k*-NN approach is also proposed, where a new data is considered as an outlier data if the average distance of its *k* nearest neighbors is above a predefined threshold. The main difficulty of the NN approach lies in its high computational cost for large sized datasets as the whole training set has to be stored and entirely evaluated. Furthermore, it has theoretically and empirically been shown in [35] that in a broad set of conditions such as i.i.d. assumption, the distance to the nearest neighbor of an input becomes closer to the distance to the farthest neighbor as dimensionality increases (beyond roughly 10-15 dimensions). These observations limit the use of these methods in high dimensional problems. Other approaches include also clustering [36, 37]. For example, in the one class *k*-means algorithm, *k* clusters are first computed from the target data, and the minimum distance of a test input to the nearest cluster is compared to a predefined threshold in order to decide whether or not the test input is rejected.

Reconstruction methods aim at encoding the target data, i.e. mapping the input data onto the output of the classifier by learning a more compact representation of the target data. The optimization routine aims at minimizing the reconstruction error on the training target data. Thus, at prediction time, an example having high reconstruction error is likely to be an outlier instance. Auto-encoder networks are one of the most used reconstruction methods [38, 39, 27, 40]. In [40], the authors propose a Diabolo network where a hidden layer is composed of a very low number of units, creating a bottleneck that is expected to compress the available information by mapping the target class in the hidden layer. Thus, inputs that have low projection on this layer will produce high reconstruction error.

Finally SVM-based approaches have also been proposed for OCC. Support Vector Data Description (SVDD) [34] is a generative approach derived from the Support Vector Machine classifier (SVM) [41]. It aims at minimizing the volume (i.e. its radius) of

an hypersphere covering the target data. The data description can be made more flexible by applying the kernel trick instead of the rigid hypersphere. Results show that SVDD performance are comparable to Gaussian, Parzen density estimators and Nearest Neighbor method.

Note that density-based methods and SVDD may assume that a fraction of legitimate target data are outlier data. This allows to automatically set a threshold on the probability density function for density-based methods and it makes the data description more flexible by optimizing the regularization of the cost parameter for SVDD [34, 19].

2.2. Learning from both target samples and artificial outliers

In this category of methods, the aim is to learn directly from the training data set the decision boundary to support both the target and the outlier classes. Therefore, these methods require either the presence of outlier data in the training set or a strong hypothesis on their distribution so that the outlier data can be taken into account during the learning phase [5]. Heuristics have been proposed in order to adapt standard multi-class discriminative methods to the one class problem: (i) generating outlier data based on hypothesis concerning their distribution, their quantity, and their location [3, 6], (ii) considering strong assumptions on the outlier data distribution without generating them in the training set [42, 43], or (iii) modifying the inner workings of existing standard multi-class boundary estimators in order to adapt them to OCC without generating outlier data [44]. We now review these three possibilities.

The first approach consists in augmenting the training set with outlier data, that are generated according to a predefined distribution. The outlier data are commonly assumed to be either uniformly distributed in the entire feature space, or located in sparse regions of the target domain, i.e regions where the target data are either absent or isolated from the rest of the data. Note that any standard multi-class method can be used, since the one class problem has been turned into a classical two-class classification problem, i.e. target versus outlier. In [3] for example, the authors combine such an outlier generation method with a tree-based class probability estimation to obtain a model of the target distribution. Firstly, the outlier data are generated following a normal distribution estimated directly from the target data. Secondly, class probability estimates are induced with the decision tree learner, with the training set composed of generated outlier data and target data. Lastly, by using the Bayes' rule, the authors combine the class probability estimates with the outlier density function to obtain an estimate of the target density function. In [6], the authors propose to generate outliers close to the target data by constraining the learning algorithm to form an accurate boundary between known classes and anomalies. To generate an outlier data, the authors randomly change the value of one feature of a target instance while leaving other features unchanged. One property of this approach is that the authors identify locations in the feature space that

are poorly populated with target data. Indeed, by analyzing the frequencies of target data values for each dimension of the feature space, sparse regions are found and consequently more outlier data are generated in these regions. Major drawbacks of most of outlier generation approaches are the impossibility to generate a sufficient amount of outlier data in high dimensional situations due to the curse of dimensionality, and the fact that the strong assumptions about the outlier data distribution may be violated in real datasets [20].

The second approach consists in taking into account the possible presence of outlier data during the training phase while these data are not physically present in the training set, i.e. these data are not generated. Thus, strong hypotheses have to be stated, such as a uniform distribution of outliers in the entire feature space or in some identified and delimited sub-regions of the target region. In [43] for example, the authors have proposed to identify specific subspaces of the target domain using a sparsity coefficient that measures how the target data populate the selected regions, under the strong assumption of uniform distribution of outlier data. The main difficulty of the algorithm resides in the search and selection of these valuable regions For this purpose, the authors present an evolutionary search algorithm that is able to quickly find hidden combinations of dimensions resulting in sparsely populated regions. The sparsely populated subsets can be seen as a partition of the data that highlight possible outlier patterns, relaxing the need for a classifier. In [42], the authors propose a decision tree induction procedure to perform clustering tasks. In order to define clusters of target data points, outlier data points are simulated and not generated, as they are not needed physically to compute the partitioning criterion at each node of the decision tree. The method has initially been proposed to tackle clustering tasks with a supervised learner, but it can be easily shifted to a one class classification task by labeling initial clustered data as target and identified empty regions as outlier.

The third approach consists in modifying a standard two-class or multi-class classifier to make it learn from the target training set only, i.e. without generating outlier data. An example of such method has been proposed in [44] as the One Class SVM (OCSVM) or ν-SVM. OCSVM is derived from the traditional SVM algorithm [41] with a modified objective function and may be trained with a unique class. Its main principle is to separate the target class from the origin, considered as the unique instance of the outlier class, with an hyperplane. The algorithm maximizes the margin between the hyperplane and the origin. The frontier separating the target data from the origin can be made more flexible using the kernel trick. The authors show that SVDD and OCSVM coincide in their decision function for a particular choice of kernel functions such as the gaussian kernel but differ for other choices. However, as the underlying principle of these two approaches is different, these two methods are categorized differently in this paper.

2.3. Classifier ensemble based methods

Ensemble methods have been poorly exploited for the design of OCC methods [13, 12, 45, 46]. Yet, ensemble methods offer more versatility to learn from the available data than a single algorithm and have been shown to outperform individual classifiers [47]. Examples of ensembles of one class classifiers are presented in [12], where the authors' goal is to propose some guidelines for the induction of one class classifier combination systems. They have studied various multiple classifier systems, several combination rules with distance-based learners (k-Means, k-Center), reconstruction-based learners (auto-encoder network), generative model SVDD and density-based one class classifiers (Gauss, MoG, Parzen), using different feature sets. In [45], the authors present a bagging OCSVM in which a pool of OCSVM classifiers are combined. Each OCSVM classifier is constructed from a bootstrap sample of the available target data. The authors show that the ensemble method improves performance compared to the individual and rather unstable OCSVM, but requires higher computational resources than the OCSVM alone.

Although these methods adopt a multiple classifier architecture and apply it to an OCC task, they use existing OCC approaches instead of fully exploiting all the mechanisms offered by classifier ensemble theory to build an OCC ensemble. Yet, this family of learning methods embed some interesting randomization principles [48], like bagging, random feature selection and random subspaces, that can be used to tackle issues specific to one class classification. In particular, these three methods can be used to overcome the exponential amount of outlier data to generate, by reducing both the number of artificial outliers to generate and the size of the feature space in which they are generated. We propose to tackle the one class classification task with a Random Forest (RF) based method, as (i) it allows to benefit from several of the aforementioned randomization principles that are naturally embedded in RF algorithms, and (ii) it uses tree-based classifiers that have shown to perform well with these randomization principles [15]. In the following section, we present such a one class random forest method.

3. One class random forests

The One Class Random Forest is an ensemble learning approach based on a random forest algorithm. Let us recall that the RF principle is one of the most successful and general purpose ensemble techniques [49, 50], and has been shown to be competitive with state-of-the-art classifiers like SVM and Adaboost [15, 51, 52]. It uses randomization to produce a diverse pool of individual tree-based classifiers. Particularly, it has been shown in [15, 53] that the forest error rate depends on the correlation between any pair of trees in the forest and on the strength (or performance) of individual trees: minimizing the correlation between trees and maximizing their individual accuracy both contribute to decreasing the forest error rate. In the standard RF learning algorithm [15],

two powerful randomization processes are used: bagging [54] and random feature selection (RFS). The first principle, bagging, consists in training each individual tree on a bootstrap replica of the training set. It is typically used to create the expected diversity among the individual classifiers and is particularly effective with unstable classifiers, like decision trees, in which small changes in the training set result in large changes in predictions. The second principle, RFS, is a randomization principle specifically used in tree induction algorithms. It consists, when growing the tree, in randomly selecting at each node of the tree a subset of features from which the splitting test is chosen. RFS contributes to the reduction of the dimensionality and has been shown to significantly improve RF accuracy over bagging alone [55, 56].

3.1. Artificial outlier generation and related issues

Our OCRF algorithm integrates an artificial outlier generation process in order to transform the OCC task into a binary classification problem. When generating such outliers, one faces the difficulty to generate both enough and representative outliers to obtain quite good performance. These artificial outliers need to cover the "entire" feature space and are expected to be sufficiently dense for being well separated from target data during training. This implies to determine:

- the outlier distribution: the distribution of the outlier data is unknown a priori. Generally, outlier data are supposed to be uniformly distributed in the entire feature space [57, 27].
- the outlier sampling: the number of outliers and their range of values must be defined according to the available target samples; in practice, the domain may be set as an hyperbox or an hypersphere surrounding the target data [57, 43].

Once a distribution is chosen, the number of outliers to generate in order to keep the same sparsity among the outlier data increases drastically according to the dimension of the feature space [43]. Indeed, considering a uniform distribution of outlier data in a rectangular domain of the whole feature space (hypercube), the volume of the outlier domain is:

$$V_{hypercube}(c) = c^M \tag{1}$$

where c is the side of the hypercube and M the dimension of the feature space. If we consider a rectangular grid covering the hypercube domain in which we generate exactly one outlier data in each cell, the amount of outlier data is given by:

$$N_{outliers}(c) = \frac{V_{hypercube}}{V_{outlier}} = \frac{c^M}{(10^{-p})^M} \tag{2}$$

where 10^{-p} is the value of the side of the individual rectangular cells of the grid or, in other words, the desired precision on the values of the outlier data. Considering a unite

hypercube for example, we have $N_{outliers} = \frac{1}{10^{-pM}} = 10^{pM}$, e.g tens of billions of outliers have to be generated for reasonable values of precision $p = 2$ and dimension $M = 5$. This phenomenon thus makes it almost impossible to adopt the uniform distribution strategy, even for low-sized feature spaces such as $M = 10$.

3.2. Our solution for efficient outlier generation

The rationale behind OCRF is to generate outliers, without following a uniform distribution. The first idea of OCRF is to use some randomization principles of ensemble learning methods to subsample the number of features and the number of training target instances in order to make possible the generation of outliers from a computation point of view. To subsample the number of features, we use the random subspace method (RSM) [58]. This well-known randomization principle of ensemble learning consists in training each individual classifier of the ensemble on a random subspace of features: K unique features are randomly selected from the initial feature set; the training samples are then projected in the subspace formed by these K features, and a component tree classifier is trained on the new resulting set. This process is repeated L times to form an ensemble of L component classifiers, each one trained on a different, K-sized feature space. RSM may be used to subsample the feature space, while controlling its dimension through the parameter K, making thus possible to reduce the amount of outliers to generate as desired. To subsample the training set and therefore the number of outliers to generate, we use bagging [54]. It consists in training each individual tree classifier on a bootstrap replica of the training dataset. These bootstrap replicas are formed by randomly selecting with replacement a subset of training samples. These two randomization principles combined together provide a natural solution to overcome the exponential amount of outliers that would be needed otherwise to reach good performance.

The second idea of the OCRF method is to make use of the information given by the target samples in order to adapt accordingly the outlier distribution. As mentioned above, if outliers were generated following a uniform distribution throughout the entire domain of definition, the amount to generate would be exponential regarding the number of features. Additionally, it would mean that outliers could be generated in areas where target instances are already densely located, leading to produce useless outliers that may introduce confusion in the learning process. One way to avoid generating these useless outliers is to generate more outliers where the target data points are sparsely located in the feature space, and conversely to generate fewer outliers in areas containing a lot of target samples. The distribution of outliers is thus designed to be complementary to the distribution of targets.

To summarize, the outlier generation process of OCRF is based on three key mechanisms: (i) the RSM process allows to reduce the dimension of the feature space in which outliers are generated; (ii) the bagging principle allows to subsample the target data so that less outliers are needed; (iii) the outlier distribution estimation is complementary

to the target distribution so that it avoids the generation of confusing and useless outliers. These three mechanisms altogether allow to transform one complex (and often impossible to solve) outlier generation problem into several easier and more efficient ones.

3.3. The OCRF algorithm

The whole OCRF procedure, illustrated in Figure 1, is made of the following steps. First, we build as many bootstrap replicas of the training set as the number L of trees of the random forest, each component tree being trained in a randomly selected subspace. Second, using sparsity information extracted from the initial training set T, we generate for each component tree the artificial outliers according to a distribution designed to be complementary to the distribution of the target samples. Third, each component tree is trained on the binary dataset made up of the projected target samples and the artificial outliers, and is then added to the ensemble. The OCRF learning procedure is detailed in Algorithm 1.

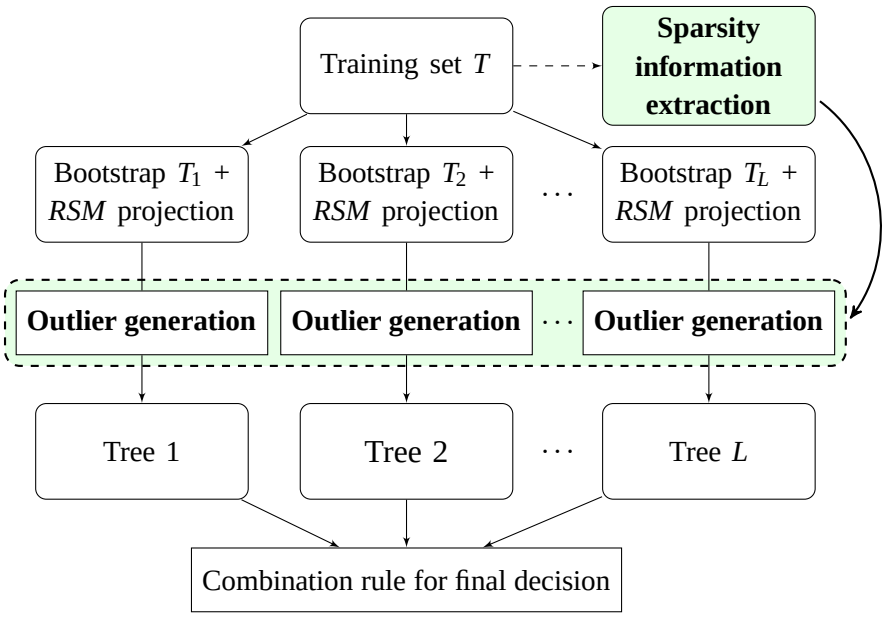


Figure 1: Overview of the OCRF induction. Additional procedures, in comparison to a traditional RF, are highlighted (in green and boldface).

In summary, the OCRF method takes advantage of: (i) combining a diverse ensemble of weak and unstable classifiers, which is known to be accurate and to increase the generalization performance over single classifiers, and (ii) subsampling the training dataset, in terms of training samples and features, in order to efficiently generate outliers by controlling their location and their number.

Algorithm 1 OCRF training algorithm

Require: a training set T, the number of outliers to be generated $N_{outlier}$, the domain of definition for the generation of outliers $\Omega_{outlier}$, the number of trees in the forest L, the parameter of RSM K_{RSM}

Ensure: a one class random forest classifier

(A) ***Prior information extraction***

Compute H_{target} the normalized histogram of the target data

Compute $H_{outlier}$ the normalized histogram of the outlier data, so that $H_{outlier}$ is the complementary of H_{target}, i.e. $H_{outlier} = 1 - H_{target}$

(B) ***Outlier generation and forest induction***

for $l = 1$ to L **do**

  (i) Draw a bootstrap sample T_l from the training set

  (ii) Project this bootstrap sample onto a random subspace of dimension K_{RSM}

  (iii) Generate $N_{outlier}$ outliers according to the complementary histogram $H_{outlier}$ in the domain $\Omega_{outlier}$, so that the probability that a generated outlier falls in a bin of the histogram $H_{outlier}$ is proportional to the value associated to that bin

  (iv) Train a random tree on the augmented dataset composed of the target data and the newly generated outlier data

end for

return a one class random forest model

4. Experimental protocol

In this section, we present the public datasets, the evaluation metrics, the one class methods used in our comparison study, and the parameters fixed for the experiments.

4.1. Datasets

By definition, negative instances for OCC applications are rare and/or unevenly spread in the feature space for being correctly sampled. Therefore, genuine one class datasets, with representative positive and negative samples, are also rare. In order to test OCC methods, authors generally transform multi-class problems into several binary "target versus outlier" classification tasks and adopt a "one versus rest" strategy, for each class of the dataset. Some authors select one class as target and label the remaining classes as outliers [57, 3, 44], while some other do the opposite, i.e. select one class as outlier and consider the remaining classes as one single target class [59, 37]. As a consequence, elaborating fair comparisons with other works based on such datasets is difficult as there is no consensus on a clear and single protocol. We will use in our experiment the first approach that is the most frequently used in the literature, with one class as target and the others as outliers.

We tackle in our experiments several problems of the literature, taken from the recognized UC Irvine Machine Learning public repository (see Table 1). We have selected these 13 datasets as they are often used for OCC comparison. The *mfeat* problem (*multiple feature dataset*) [12] is computed for 5 different feature spaces extracted from scanned handwritten numerals, resulting in five different datasets; these feature spaces are Fourier coefficients, Karhunen-Loeve coefficients, morphological features, raw pixels values, Zernike moments and factor correlations. Two datasets from the *mfeat* problem, namely *mfeat-pixel* and *mfeat-fourier* are not included in this paper due to the parameterization optimization of standard density estimators that have failed on these datasets. Dataset *glass* includes originally 7 classes, describing different kinds of glasses commonly found on criminal scenes. Since *glass* type 4 is not represented at all and type 6 has only 9 elements, these two classes have not been taken into account and type 6 data have been merged into the outlier cases.

Datasets have feature space sizes ranging from 4 to 216, number of classes ranging from 2 to 10 and number of instances from 150 to 11000. Our experiments thus cover a wide range of conditions. As one class is selected in turn for the target class and the others gathered for the outlier class, we have conducted experiments on 78 datasets in total, according to Table 1. In our experiments, the data were not preprocessed, i.e. there was no normalization, nor principal component reduction.

Table 1: Description of the datasets taken from the UC Irvine repository [60]

	Total number of		
Datasets	attributes	classes	instances
sonar	60	2	208
glass	9	5	214
ionosphere	34	2	351
optdigits	64	10	5620
iris	4	3	150
musk	166	2	6598
breast cancer wisconsin (bcw)	9	2	699
pendigits	16	10	10994
diabetes	8	2	768
mfeat-factors	216	10	2000
mfeat-karhunen	64	10	2000
mfeat-zernike	47	10	2000
mfeat-morphological	6	10	2000
Total number of one class datasets		**78**	

4.2. Evaluation criteria

It is difficult to fairly compare between OCC methods as classical evaluation measures may not be adapted nor accurate enough for the proposed task, influenced by the nature of the dataset sample and/or the nature of the domain studied. Indeed, a wide range of measures has been proposed in the literature among which global accuracy, sensitivity, specificity, precision and recall, ROC curves, Area Under the ROC curve (AUC), weighted AUC or other averaging methods that aim at summarizing the performance of a given classifier [61]. But there is no consensus for the performance computation of one class algorithms nor for their comparison because each one of these measures is more or less biased by the imbalanced ratio between the two classes.

In spite of this bias, results of our experiments will be presented in terms of global accuracy, target and outlier recognition rates, as these measures will allow for an analysis of the "target vs outlier performance" trade-off. However, as these evaluation measures do not take into account the imbalanced nature of OCC datasets [61], we will use an additional measurement, *i.e.* the Matthews correlation coefficient (MCC) or "phi coefficient" [62]. It is often used in combination with precision and sensitivity measures in biomedical applications, where datasets are known to be particularly imbalanced. As we will show in Section 5, this coefficient is more suitable for one class studies than standard accuracy measures [61, 36].

MCC uses the contingency table from the confusion matrix and is given by:

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{3}$$

where TP (true positive) is the number of correctly identified target data, TN (true negative) the number of correctly identified outlier data, FN (false negative or non-detection) the number of legitimate target data that have been misclassified and FP (false positive or false alarm) the number of misclassified legitimate outlier data. MCC measures the degree of correlation between the observed classes and the outputs of the classifier. It ranges from -1 if all predictions are wrong to +1 for perfect classification. Null values indicate that either predictions are completely random or one of the two classes has not been correctly classified at all, i.e. the classifier always predicts only one of the two classes.

Results will thus be presented in terms of MCC, global accuracy, target and outlier recognition rates. For the evaluation process, we computed for each dataset a 10-fold stratified cross validation repeated 5 times. The 10-fold cross-validation method is commonly considered as a good estimate of the mean of the classification error and is a good compromise in case of small datasets [63, 64]. The classifier performance are then averaged over the different runs.

4.3. Statistical comparison

It is not a straightforward task to evaluate and compare multiple classifiers on multiple datasets as pointed out in [65, 66] and developed in a comprehensive study of this issue [67, 68, 69]. Several techniques have been proposed to compare classifiers like statistical tests [70] for pairwise comparisons (pairwise t-test with re-sampling evaluation schemes, t-test with 10-folds, corrected t-test [71] taking into account overlapping issues, 5x2CV, McNemar test) or ranking methods like average ranks [72]. There have been many discussions about the right tests to apply for general comparison purposes [70, 73, 74]. But methods for fairly comparing multiple classifiers on multiple datasets are rare and often totally ignored [69]. Two approaches are suggested in [69]: the well-known ANOVA [75] and the non-parametric Friedman statistical test [76, 77] associated to the Nemenyi post-hoc test [78]. ANOVA is based on assumptions that are not always granted in typical machine learning studies, i.e. the performance samples must be drawn from a normal distribution and the random variables must have equal variance. Therefore we eschewed the ANOVA test in favor of the Friedman test, that better suits the characteristics of our experimental protocol (as suggested in [69]).

The Friedman test is a statistical test that uses the rank of each algorithm on each dataset. The null hypothesis states that the compared algorithms are not significantly different. For k classifiers and N datasets, if R_j is the mean rank of classifier j among all datasets, the Friedman statistic is given by:

$$\chi_F^2 = \frac{12N}{k(k+1)} \left(\sum_{j=1}^{k} R_j^2 - \frac{k(k+1)^2}{4} \right) \tag{4}$$

with $k-1$ degrees of freedom. An improved version of this test has been proposed by Iman and Davenport [79] with a less conservative correction of χ_F^2 :

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1) - \chi_F^2} \tag{5}$$

distributed according to the F-distribution with $(k-1)$ and $(k-1)(N-1)$ degrees of freedom.

If the null hypothesis is rejected, a post-hoc test is carried out like the Nemenyi test [78], which is used when all classifiers are compared to each other or the Bonferroni Dunn test [80], which is used when comparing one control algorithm to the other ones. We use in this study the Nemenyi statistical test given by the Critical Difference (CD):

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{6}$$

where q_α is the critical tabulated value for this test [78].

4.4. State-of-the-art OCC methods and parameterization

The OCRF algorithm is compared to four state-of-the-art OCC algorithms, namely the One Class SVM [44] taken from the LibSVM toolbox and three density estimators, Gaussian estimator (Gauss), Parzen windowing (Parzen) and Mixture of Gaussians models (MoG) taken from the Pattern Recognition Toolbox (PRTools) [81, 82].

The OCSVM algorithm is computed using the default parameter settings taken from the LibSVM toolbox (i.e. cost coefficient $C = 1$, radial basis function for the kernel choice, $\gamma = \frac{1}{dimension}$ for the kernel bandwidth), except for the ν coefficient, a lower bound on the fraction of support vectors that we set to a more frequently cited value $\nu = 0.1$ (instead of $\nu = 0.5$ in LibSVM). Similarly, the three other algorithms (Gauss, Parzen and MoG) are run with their default parameters defined in PRTools. In particular, MoG is computed by default with 5 clusters, the position, size and priors of each of the clusters being optimized using the conventional Expectation-Maximization (EM) algorithm. Likewise, the bandwidth parameter h in Parzen is by default optimized by maximizing the likelihood on the training data using leave-one-out [81]. Note also that for these three density estimators PRTools defines the parameter $fracrej = 0.05$, corresponding to the fraction of legitimate target cases that will be considered as outliers in the training phase of the algorithm. This trick allows in particular to compute a threshold on the outputs of the density estimator. All these default parameter settings are discussed in [3, 83, 57, 44].

Regarding the OCRF parameterization, standard values for the parameters are also used:

- the number of trees in the random forest is $L = 200$, a value commonly considered as sufficient in practice to ensure statistical convergence of the algorithm [55, 53];
- trees are fully developed as it is proven to be more efficient in RF [15].
- the number of attributes for the Random Subspace Method is empirically set to $K_{RSM} = 10$ or $K_{RSM} = M$ if $M < 10$, where M is the dimension of the feature space [58];
- the number of attributes for the Random Feature Selection is $K_{RFS} = \sqrt{K_{RSM}}$, as suggested in [55, 53].

Finally, the generation of outliers during training requires to define their number and the range of their values. We have chosen the generation domain of outliers to be 1.2 times greater than the target domain estimated through the training set, assuming that the outlier domain needs to cover the whole target domain. The number of outliers to generate is empirically set to $N_{outlier} = 10 \cdot N_{target}$ where N_{target} is the sample size of the bootstrap replica (see Algorithm 1).

5. Experimental results

In this section, we report the results obtained according to the experimental protocol described in the preceding section. We first discuss the overall performance of each OCC method mainly according to the ”outlier versus target” trade-off. We then show that OCRF method compares favorably to standard OCC algorithms (OCSVM, Gauss, Parzen, MoG), according to the test of statistical significance.

5.1. Analysis of classifier performance

In order to show the interest of using the MCC for performance analysis, we discuss the results of Table 2. In this table, results for three OCC classifiers (namely OCRF, OCSVM and Gauss) are presented on two illustrative datasets (namely *opdigits* and *mfeat-factors*) in terms of averaged MCC, accuracy, target and outlier recognition rates. We recall that this coefficient enables to take into account the imbalanced nature of OCC datasets: the closer to +1 the MCC, the better the performance of the classifier; a null value of MCC usually indicates that the classifier predicts only one of the two classes, the other class is never classified correctly; the closer to -1 the MCC, the worst the performance of the classifier. Results show that MCC better predicts the behavior of an OCC classifier than the accuracy rate, as shown in Table 2 where three different cases can be identified:

(i) MCC and accuracy both have a high value (e.g. OCRF on *optdigits_0*)

(ii) MCC and accuracy both have a low value (e.g. OCRF on *optdigits_1*)

(iii) MCC value is null or very low while accuracy is high (e.g. OCSVM on all datasets in this table)

In the latter case, one can see that accuracy rates fail to indicate that either target or outlier data are poorly, or even not at all, recognized by the classifier whereas MCC better highlights this phenomenon by providing a value close to zero. Thus the remainder of the result analysis will be conducted using the MCC indicator.

For sake of clarity, the averaged MCC values are presented for all datasets and all classifiers in Appendix A. A synthesis of these results is presented in Table 3. In the forthcoming analysis, our aim is (i) to focus on the global performance of OCRF on all datasets and (ii) to give some insights on local behaviors in comparison with the four other state-of-the-art OCC methods.

Table A.7 and Table 3 show that OCRF performs generally well on most of the datasets, as more than half of its MCC values are high (typically above 0.5 for 45 datasets over 78). This illustrates the ability of the method to correctly handle the ”target vs outlier” trade-off for a large range of OCC problems, in spite of their imbalanced

		OCRF	OCSVM	Gauss
opt_0	MCC	0,776	0,165	**0,954**
	Acc	0,94	0,90	0,99
	T	0,99	0,04	0,92
	O	0,94	1,00	1,00
opt_1	MCC	0,147	0,054	**0,937**
	Acc	0,26	0,90	0,99
	T	1,00	0,01	0,90
	O	0,18	1,00	1,00
opt_2	MCC	0,143	0,000	**0,953**
	Acc	0,26	0,90	0,99
	T	1,00	0,00	0,92
	O	0,18	1,00	1,00
opt_3	MCC	0,121	0,000	**0,914**
	Acc	0,22	0,90	0,98
	T	1,00	0,00	0,92
	O	0,13	1,00	0,99
opt_4	MCC	0,077	0,000	**0,905**
	Acc	0,16	0,90	0,98
	T	1,00	0,00	0,92
	O	0,06	1,00	0,99
opt_5	MCC	0,041	0,000	**0,954**
	Acc	0,12	0,90	0,99
	T	1,00	0,00	0,92
	O	0,02	1,00	1,00
opt_6	MCC	0,410	0,026	**0,956**
	Acc	0,70	0,90	0,99
	T	1,00	0,00	0,92
	O	0,67	1,00	1,00
opt_7	MCC	0,264	0,000	**0,933**
	Acc	0,48	0,90	0,99
	T	1,00	0,00	0,91
	O	0,42	1,00	1,00
opt_8	MCC	0,043	0,000	**0,719**
	Acc	0,12	0,90	0,94
	T	1,00	0,00	0,91
	O	0,02	1,00	0,94
opt_9	MCC	0,077	0,000	**0,860**
	Acc	0,15	0,90	0,97
	T	1,00	0,00	0,90
	O	0,06	1,00	0,98

(a)

		OCRF	OCSVM	Gauss
fact_0	MCC	**0,844**	0,000	0,737
	Acc	0,97	0,90	0,96
	T	0,86	0,00	0,58
	O	0,98	1,00	1,00
fact_1	MCC	**0,873**	0,000	0,712
	Acc	0,98	0,90	0,95
	T	0,79	0,00	0,54
	O	1,00	1,00	1,00
fact_2	MCC	**0,879**	0,000	0,740
	Acc	0,98	0,90	0,96
	T	0,85	0,00	0,58
	O	0,99	1,00	1,00
fact_3	MCC	**0,887**	0,017	0,695
	Acc	0,98	0,90	0,95
	T	0,87	0,00	0,51
	O	0,99	1,00	1,00
fact_4	MCC	**0,884**	0,000	0,743
	Acc	0,98	0,90	0,96
	T	0,82	0,00	0,58
	O	1,00	1,00	1,00
fact_5	MCC	**0,843**	0,013	0,738
	Acc	0,97	0,90	0,96
	T	0,82	0,00	0,58
	O	0,99	1,00	1,00
fact_6	MCC	**0,910**	0,068	0,770
	Acc	0,98	0,90	0,96
	T	0,85	0,02	0,62
	O	1,00	1,00	1,00
fact_7	MCC	**0,879**	0,017	0,841
	Acc	0,98	0,90	0,97
	T	0,83	0,00	0,73
	O	1,00	1,00	1,00
fact_8	MCC	0,613	0,000	**0,647**
	Acc	0,91	0,90	0,94
	T	0,83	0,00	0,45
	O	0,91	1,00	1,00
fact_9	MCC	**0,866**	0,026	0,751
	Acc	0,98	0,90	0,96
	T	0,85	0,01	0,60
	O	0,99	1,00	1,00

(b)

Table 2: Case study for results of OCRF, OCSVM and Gauss on (a) *optdigit* (*opt_N*) and (b) *mfeat-factors* (*fact_N*) datasets. Best MCC results for each dataset are indicated in bold face.

	OCRF	OCSVM	Gauss	Parzen	MoG
# of MCC negative values	0	1	2	2	1
# of MCC null values	0	32	0	37	12
# of MCC values superior to 0.5	45	12	60	8	39
# of occurences of 1st rank wrt MCC values	23	7	35	4	9

Table 3: Synthesis of the results extracted from Table A.7 in Appendix A

nature. This observation will be confirmed in the following section, with a statistical analysis based on the rank values of each classifier on all datasets. Another observation is that OCRF is the best OCC method on 23 datasets among 78, whereas MoG is the best one on only 9 datasets, OCSVM on 7 and Parzen on 4 datasets, in terms of MCC values. Gauss is the best of the five OCC methods, since it has the highest MCC values on 35 datasets over the 78. According to these results, Gauss and OCRF clearly outperform the three other OCC methods. This will be also confirmed in the next section with the statistical comparison.

Another remarkable result is that OCRF never exhibits null values of MCC, contrary to the four other state-of-the-art OCC methods: Parzen has null values on 37 datasets, OCSVM on 32, MoG on 12 and Gauss none. Let us recall that a null MCC value indicates that the classifier either predicts completely randomly or always predicts only one of the two classes. Such cases may be found in Table A.7 where OCSVM and Parzen for example both always predict the outlier class on *pendigits*, *optdigits* and *mfeat-zernike*, as 90% of the test data are outliers. Similarly, Parzen and MoG always predict the target class only on *mfeat-factor*, as 10% of the test data are targets. These latter results confirm that OCSVM, Parzen and MoG are less able to handle the "target vs outlier" trade-off than OCRF and Gauss. Note finally that a few negative values are obtained for all four state-of-the-art methods whereas OCRF does not exhibit any. Let us recall that a negative MCC value indicates that the classifier behaves worst than a random predictor. Nevertheless, these behaviors seem to be marginal.

In summary, even if these experiments highlight that OCRF is not always the best OCC method over all the datasets, they reveal that OCSVM, Parzen and sometimes MoG may have very unstable behaviors, being the best one on some datasets and the worst in some others. This is not the case of OCRF neither Gauss, which appear to be good classifiers for OCC tasks.

5.2. Classifier ranking and statistical significance

In this section, our aim is to rank the OCC algorithms to allow for a better comparison of the five OCC methods. We use the statistical test presented in section 4.3 to

evaluate the significance of these comparisons.

For each dataset, the five OCC methods have been ranked from 1 (highest MCC value) to 5 (lowest MCC value). The mean rank over the 78 datasets is provided in Table 4, with corresponding standard deviation values. One can observe that the best method in average is Gauss, as it exhibits the lowest mean rank (1.90) and that the second best one is OCRF (2.43). The MoG method is ranked in average right after OCRF, with a mean rank of 2.79, while the two remaining methods, OCSVM and Parzen, are clearly outperformed by the three others.

Additional statistics on the rank values are provided in Figure 2, under the form of boxplots. Red lines correspond to the median values, boxes to the half of the rank values and black segment to the minimum and maximum ranks of each method. One can see that OCRF is ranked in the Top 3 best methods for 75% of the datasets (i.e. 52 over 78). However, it does not outperform Gauss that is ranked either first or second on 75% of the datasets, and that exhibits ranks always inferior or equal to 3. These plots confirm that OCSVM and Parzen have the worst performance, with ranks between 3 and 5 for 75% of the datasets.

	OCRF	OCSVM	Gauss	Parzen	MoG
Mean rank	2.43 ± 1.16	4.04 ± 1.28	1.90 ± 1.15	3.83 ± 1.12	2.79 ± 1.08

Table 4: Mean rank values ($\pm$ standard deviation) of OCC methods over the 78 datasets

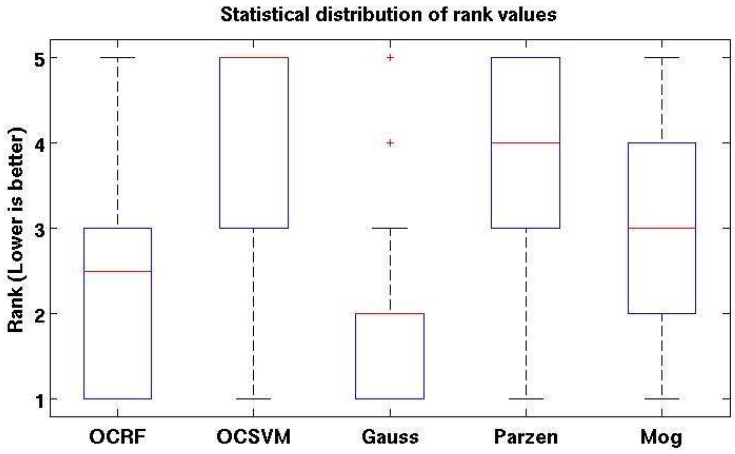


Figure 2: Boxplots of rank values of the OCC methods over the 78 datasets. Red segments correspond to the median values. The boxes indicates the repartition of half of the ranks around the median and black segments indicate minimum and maximum ranks.

Figure 3 is another way to gain some insight on the ranking of the 5 methods. It

shows the ratio of the datasets for which a given method is among the n first ranked methods, n ranging from 1 to 5. For instance, Gauss is the first ranked method for 50% of the datasets, while OCRF is ranked first for 30% of the datasets. We can clearly distinguish two groups of methods: the first one is made of Gauss, OCRF and MoG, that are in the Top 3 for more than 70% of the datasets, while the second one is made of OCSVM and Parzen, that are in the Top 3 methods for less than 30% of the datasets. We can thus assume that this gap of performance between these two groups is significant enough, but it seems less obvious how methods compare to one another inside each group. This hypothesis is now tested using the Friedman test presented in section 4.3.

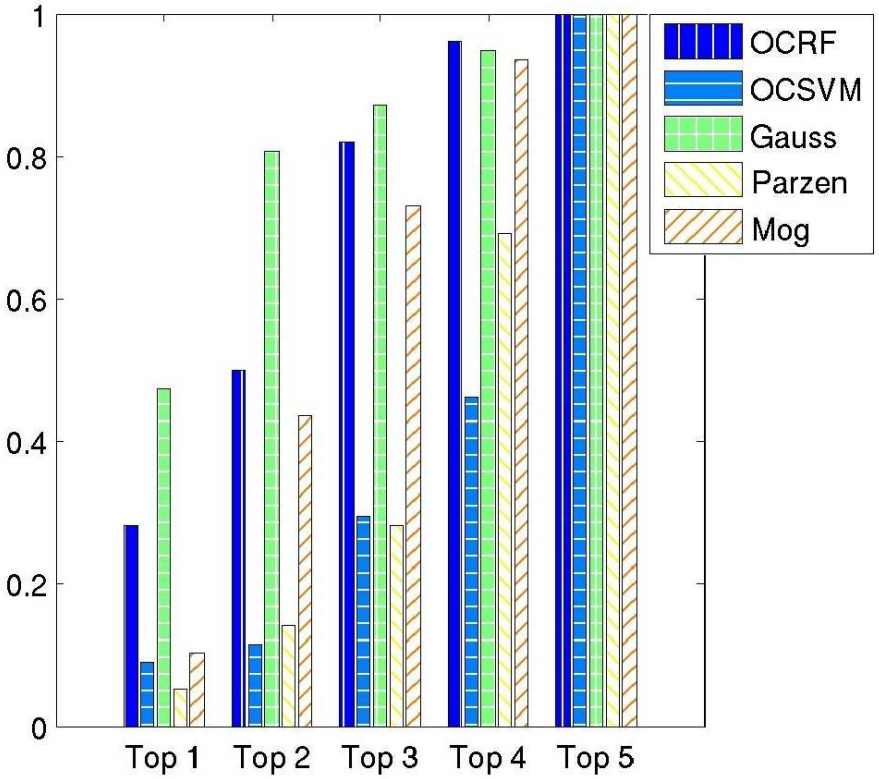


Figure 3: Statistics on rank values obtained over the 78 datasets: ratio of datasets associated to the cumulative rank values for all classifiers (the higher the ratio, the better; the lower the rank value, the better).

Let us recall that the Friedman test has been applied on MCC values. Using Equation 4 with $N = 78$ datasets and $k = 5$ classifiers, we have $\chi_F^2 = 104.48$. Applying the Iman and Davenport improvement, we obtain $F_F = 38.768$ from Equation 5. From common tabulated values, we read that the critical value for the F-distribution with $k-1 = 4$ and $(k-1)(N-1) = 308$ degrees of freedom and under the risk $\alpha = 0.05$ is $F(4,308) \approx 2.37 < 38.768$. This indicates that the null hypothesis is rejected, concluding that the

given ranks are significantly different. We can now use the post-hoc Nemenyi test for the method ranking. From Equation 6, we obtain $CD = 0.690$. From this value, we can conclude that, under the risk of 5%, OCRF performs significantly better than Parzen (as $R_{Parzen} - R_{OCRF} = 3.83 - 2.43 > CD$) and OCSVM (as $R_{OCSVM} - R_{OCRF} = 4.04 - 2.43 > CD$). On the contrary, nothing can be said about OCRF *vs* Gauss nor OCRF *vs* MoG as the differences between their mean ranks are smaller than *CD*. Using the same calculus, one can conclude that Gauss performs better than MoG, Parzen and OCSVM; MoG performs significantly better than OCSVM and Parzen. Table 5 summarizes all the duels results. It clearly shows that the two groups identified in the previous figures are indeed statistically different in terms of MCC results, i.e. Parzen and OCSVM are both statistically outperformed by the three other methods. However, these results barely allow to conclude about the differences between the methods inside these two groups: (i) Parzen and OCSVM can not be distinguished from their MCC values; (ii) even if Gauss statistically outperforms MoG, it is difficult to establish a ranking between these two methods and OCRF, since the two remaining duels are not conclusive. More data would be of course required to reliably compare these three classifiers.

Note finally that our OCRF method is quite surprisingly often outperformed by a simple parametric density estimator such as Gauss. It could be thus inferred that target samples are normally distributed in a majority of the datasets of our experiments since the Gaussian estimator would particularly suit to those cases. We report in the next subsection the results of our investigation on this issue.

	Gauss	OCRF	MoG	Parzen	OCSVM
Gauss		0	+	+	+
OCRF	0		0	+	+
MoG	-	0		+	+
Parzen	-	-	-		0
OCSVM	-	-	-	0	

Table 5: Duels between the methods, in terms of statistical significance. A '+' (resp. '-') indicates that the method in the corresponding line statistically outperforms (resp. is outperformed by) the method in the corresponding column. A '0' indicates that no conclusion may be drawn from the statistical test.

5.3. OCRF vs Gauss: multi-normality of target samples

As mentioned above, the Gaussian estimator is the classifier that most often outperforms the other methods, leading to infer that target samples may be normally distributed in the corresponding datasets. In order to test the multi-normality of these datasets, we have used the classical Mardia's test of multivariate skewness and kurtosis [84]. This test is considered as one of the best method to assess the degree to which multivariate data deviate from multi-normality [85]. The test uses two statistical moments,

the multivariate skewness (third moment) and kurtosis (fourth moment) in order to test independently if these measures are consistent with the assumption of multi-normality. Data are assumed to conform to a multi-normal distribution only if the null hypothesis of multi-normality has not been rejected for both tests, *i.e* no significant skew in the data and no significant deviation of kurtosis from expectancy.

The sample measure of multivariate skewness is given by :

$$\gamma_{1,d} = \frac{1}{N^2} \sum_{i \leq N, j \leq N} m_{ij}^3 \tag{7}$$

and the measure of kurtosis by :

$$\gamma_{2,d} = \frac{1}{N} \sum_{i \leq N} m_{ii}^2 \tag{8}$$

where N is the sample size, d the dimension of the feature space, $m_{i,j} = (x_i - \bar{x})^T S^{-1} (x_j - \bar{x})$, x_i a data vector, $\bar{x}$ the sample mean and S the sample covariance matrix.

The first part of the Mardia's test leans on the fact that, under the assumption of multi-normality, the statistic $N \cdot \gamma_{1,d}/6$ asymptotically follows a chi-square (χ^2) distribution with $d(d+1)(d+2)/6$ degrees of freedom. Hence, if the estimation given by equation 7 significantly deviates from the corresponding reference value in the χ^2 distribution table, one can conclude that the underlying data is not likely to come from a multi-normal distribution. In the same manner, the second part of the test is based on the fact that the statistic $\gamma_{2,d}$ is asymptotically normally distributed with mean $d(d+2)$ and variance $8d(d+2)/N$. The (centered reduced) $\gamma_{2,d}$ value estimated from equation 8 can thus be compared to the corresponding critical value from the normal distribution table.

All those values, obtained with equations 7 and 8, and from the normal and χ^2 distribution tables, are provided in table B.8. They indicate that among 78 datasets, the multi-normality hypothesis is rejected for 58 datasets and accepted for 3. For 17 datasets no result could be obtained due to singular variance-covariance matrix for each dataset causing computational issues when computing for instance the Mahalanobis distance or $m_{i,j}$ values aforementioned.

Nonetheless, these results show that a vast majority of the tested datasets do not match the assumption of multi-normality, in particular on those datasets for which the Gaussian estimator exhibits the best performance (Sonar and MFeat-zernike datasets for example). It seems therefore that the good results of Gauss over OCRF cannot be explained by the multi-normality of these datasets and that further investigations are needed to better understand why OCRF is often outperformed by Gauss.

5.4. Robustness with respect to dimensionality

One of the main advantages of OCRF over the other state-of-the-art OCC methods is its robustness regarding the number of dimensions. To assess the rather good behavior of OCRF with an increasing size of the feature space, we have performed the additional experiments detailed below.

Note first that it is still an issue to generate artificial datasets for "high" dimensional spaces due to the curse of dimensionality. In some preliminary experiments, we have noticed that above about 10 features, OCC methods often exhibit MCC values equal to either 1 or 0, depending on the distribution of both positive and negative samples. As shown in [86, 87], the difficulty of generating well representative distributions can be explained by empty space phenomenons or concentration of measures. On the other hand, real-world high-dimensional datasets are very often built with a lot of uninformative or sparse features, as it is the case in gene analysis or text categorization for example. Such amounts of non discriminant or non informative features may strongly bias the results and the analysis as explained in [86, 87].

Therefore, we have rather turned towards designing a dedicated experimental protocol by creating a quite high-dimensional artificial dataset from the real-world MFeat datasets: a feature space has been created with discriminant features by concatenating the different *MFeat* feature vectors. Let us recall that the four datasets *MFeat-Factors*, *MFeat-Karhunen*, *MFeat-Zernike* and *MFeat-Morphological* have been built from the same data instances representing single digits between 0 and 9, but for which different descriptors have been extracted. We have thus created a new dataset, called *MFeat-FKZM*, by concatenating factors, karhunen, zernike and morphological descriptors in the same feature vector, leading for each sample to a 333-feature vector. Then, 10 different OCC datasets, called *"digit X"* where *X* is the digit used as the target class, have been extracted following the previously used "1-versus-rest" strategy. The five OCC classifiers have been tested on the 10 *digit X* datasets several times with different sizes of the feature space: on each *digit X* dataset, m features have been randomly sampled from 333 features, for all $m \in \{2, 3, 5, 10, 15, 20, 25, 50, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300, 325, 333\}$; each time, 10 random subsets of m features have been sampled so that averaged MCC values over these 10 replicas have been obtained on each *digit X* and for each value of m, along with their standard deviations.

Full MCC results for each of the 10 datasets and for each classifier are presented in Figure 5 as curves of mean MCC values with respect to m. To give the precise values of MCC means and standard deviations, a detailed example on *digit 1* dataset is also given in Table 6. For clarity concerns, only *digit 1* dataset is presented in this table and enlarged in Figure 4, but as it can be seen on Figure 5, similar results have been obtained for the 9 other datasets.

From Figure 5, one can clearly observe that OCRF exhibits very stable behaviors

as the size of the feature space increases, contrary to the four other methods that fail to maintain their performance obtained on lower dimensions. As for previously presented MCC results, it is important to note that some classifiers like OCSVM exhibits several zero MCC values as they predict only one of the two classes for almost all samples. These stable behaviors are assessed by the evolution of MCC values, but are also confirmed by standard deviations, as shown in Table 6 and Figure 4 for the *digit 1* dataset. OCRF exhibits quite low values of standard deviation (*i.e.* lower than or equal to 0.05) for all sizes above 50, contrary to Gauss for example, that shows standard deviations up to 0.14 for the same sizes. Even if the Gauss method sometimes outperforms OCRF in terms of averaged MCC values, these results show that it may be at the expense of rather unstable performance for larger dimensions.

As a conclusion, these additional experiments give better insights on how OCRF manage to handle OCC problems, for which state-of-the-art methods often exhibit very unstable performance. They confirm as expected that randomized ensemble principles make the OCRF method more robust than the other methods to an increasing size of the feature space.

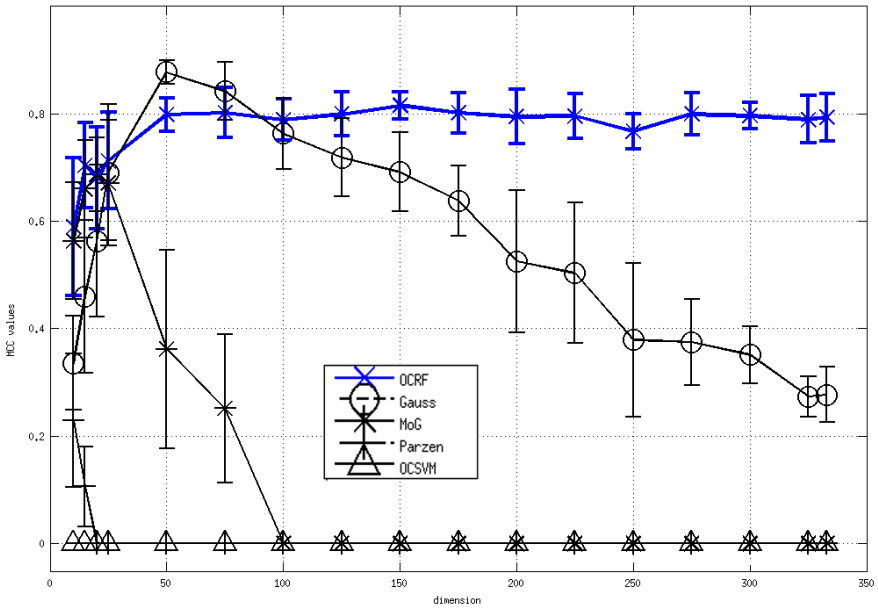
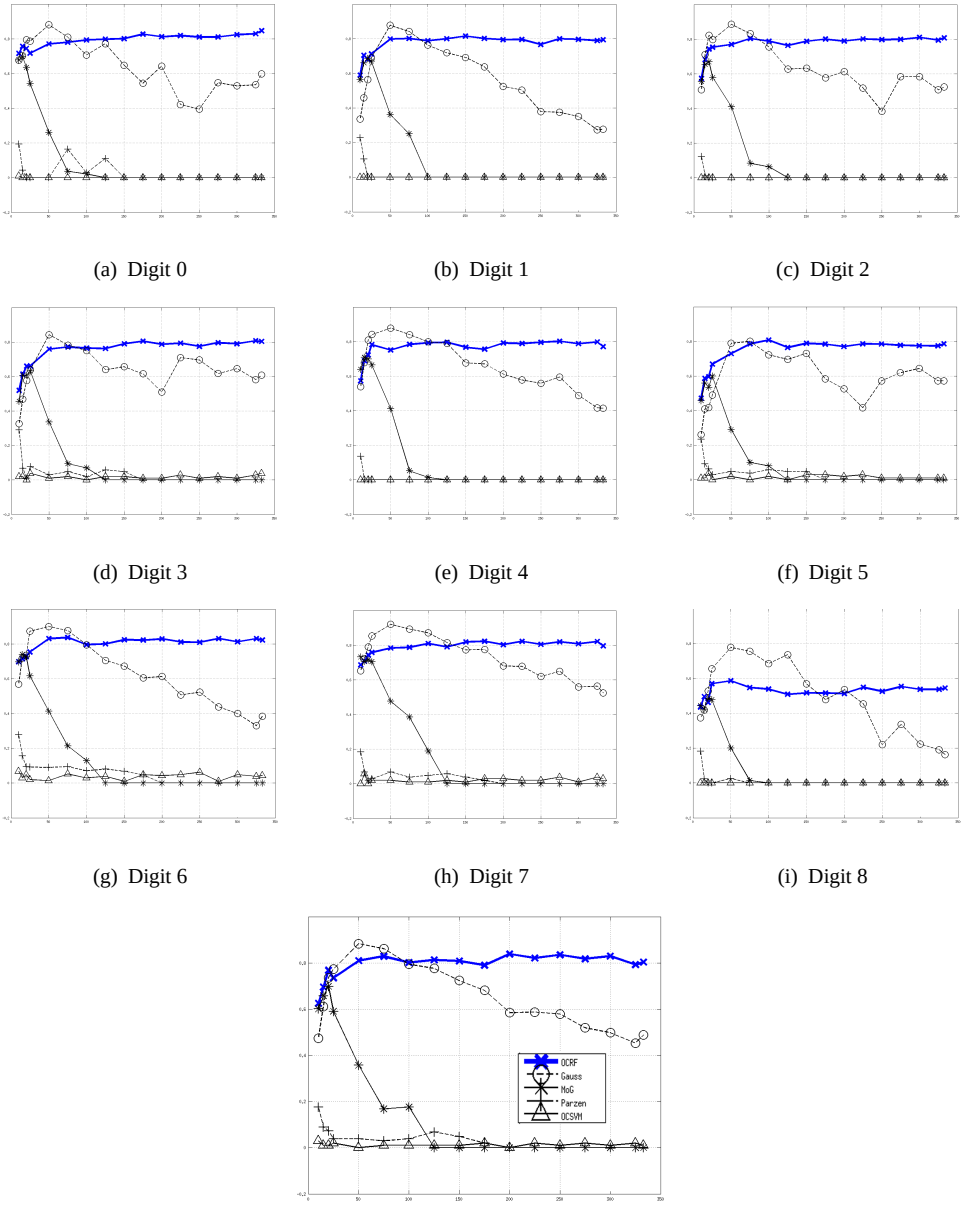


Figure 4: MCC values with respect to the number of features for OCRF, OCSVM, Gauss, Parzen and MoG classifiers, obtained on the *digit 1* dataset.

Table 6: Performances of OCRF compared with OCSVM, Gauss, Parzen, MoG classifiers on MFeat-FKZM dataset for digit 1

m	OCRF	OCSVM	Gauss	Parzen	MoG
2	0.15 (± 0.09)	**0.53** (± 0.16)	0.10 (± 0.06)	0.13 (± 0.05)	0.12 (± 0.05)
3	0.20 (± 0.13)	**0.39** (± 0.31)	0.11 (± 0.09)	0.17 (± 0.12)	0.15 (± 0.10)
5	**0.39** (± 0.09)	0.00 (± 0.00)	0.19 (± 0.07)	0.30 (± 0.10)	0.26 (± 0.06)
10	**0.59** (± 0.13)	0.00 (± 0.00)	0.34 (± 0.09)	0.23 (± 0.12)	0.56 (± 0.11)
15	**0.70** (± 0.08)	0.00 (± 0.00)	0.46 (± 0.14)	0.11 (± 0.07)	0.66 (± 0.09)
20	0.68 (± 0.09)	0.00 (± 0.00)	0.56 (± 0.14)	0.00 (± 0.00)	**0.69** (± 0.07)
25	**0.71** (± 0.09)	0.00 (± 0.00)	0.69 (± 0.13)	0.00 (± 0.00)	0.67 (± 0.12)
50	0.80 (± 0.03)	0.00 (± 0.00)	**0.88** (± 0.02)	0.00 (± 0.00)	0.36 (± 0.18)
75	0.80 (± 0.05)	0.00 (± 0.00)	**0.84** (± 0.05)	0.00 (± 0.00)	0.25 (± 0.14)
100	**0.79** (± 0.04)	0.00 (± 0.00)	0.76 (± 0.07)	0.00 (± 0.00)	0.00 (± 0.00)
125	**0.80** (± 0.04)	0.00 (± 0.00)	0.72 (± 0.07)	0.00 (± 0.00)	0.00 (± 0.00)
150	**0.82** (± 0.03)	0.00 (± 0.00)	0.69 (± 0.07)	0.00 (± 0.00)	0.00 (± 0.00)
175	**0.80** (± 0.04)	0.00 (± 0.00)	0.64 (± 0.07)	0.00 (± 0.00)	0.00 (± 0.00)
200	**0.79** (± 0.05)	0.00 (± 0.00)	0.53 (± 0.13)	0.00 (± 0.00)	0.00 (± 0.00)
225	**0.80** (± 0.04)	0.00 (± 0.00)	0.50 (± 0.13)	0.00 (± 0.00)	0.00 (± 0.00)
250	**0.77** (± 0.03)	0.00 (± 0.00)	0.38 (± 0.14)	0.00 (± 0.00)	0.00 (± 0.00)
275	**0.80** (± 0.04)	0.00 (± 0.00)	0.37 (± 0.08)	0.00 (± 0.00)	0.00 (± 0.00)
300	**0.80** (± 0.02)	0.00 (± 0.00)	0.35 (± 0.05)	0.00 (± 0.00)	0.00 (± 0.00)
325	**0.79** (± 0.04)	0.00 (± 0.00)	0.27 (± 0.04)	0.00 (± 0.00)	0.00 (± 0.00)
333	**0.79** (± 0.04)	0.00 (± 0.00)	0.28 (± 0.05)	0.00 (± 0.00)	0.00 (± 0.00)



(a) Digit 0 (b) Digit 1 (c) Digit 2

(d) Digit 3 (e) Digit 4 (f) Digit 5

(g) Digit 6 (h) Digit 7 (i) Digit 8

(j) Digit 9

Figure 5: MCC values vs. the number of features (ranging from 0 to 333) for OCRF ($\times$), OCSVM ($\triangle$), Gauss ($\circ$), Parzen ($+$) and MoG ($\star$) classifiers on MFeat-FKZM dataset for all digits

6. Conclusion and future works

One-Class Random Forest is a discriminative OCC method based on the reference random forest algorithm combined with an original procedure for generating artificial outliers. Generating outliers is often used with discriminative learning methods to counterbalance the absence of outlier data during the training phase but is difficult to implement since the number of outliers to generate for having reasonably good performance is exponential with respect to the dimension of the feature space, and may also increase as the number of available training samples increases. We have shown that the random principles used in traditional RF can be powerful tools to overcome this issue: by subsampling the training set for each component classifier of the ensemble, through the selection of both the training samples (with bagging) and the features (with random feature selection and random subspace method), and by then combining all of them, we reduce the minimum number of outliers to generate and increase the generalization accuracy of the ensemble.

To assess the efficiency of our method, experiments have been conducted on several public datasets from the UCI repository and OCRF has been statistically compared to four of the most used OCC algorithms, namely one-class SVM, Gaussian estimator, Parzen windowing and Mixture of Gaussians models. On most of these datasets and using the default parameterization of each method, results have shown that OCRF performs equally well or better than these state-of-the-art OCC algorithms. Besides, OCRF performance appears to be rather stable even in relatively high dimensional space, whereas other OCC method accuracy rates decrease.

Room for improvement is still left in our OCRF framework. Indeed, the OCRF method depends on two parameters whose values have not been investigated nor tuned for these experiments: (i) the ratio of outliers to be generated according to the number of available target samples and (ii) the range of each feature value of these artificial outliers. Although standard values for these parameters give satisfying results for most of the datasets, preliminary experiments conducted on some particular datasets have shown that these parameters could have optimal values. A deeper study on the influence of these parameters according to the target distribution could help choosing their appropriate values automatically.

Appendix A. Matthews correlation coefficient (MCC)

Dataset	OCRF	OCSVM	Gauss	Parzen	MoG
iris_versicolour	0,579 (81,5)	0,897 (95,3)	**0,903** (95,6)	0,685 (85,6)	0,607 (82,9)
iris_virginica	0,614 (82,7)	**0,900** (95,5)	0,813 (90,9)	0,716 (87,3)	0,604 (82,5)
iris_setosa	0,722 (87,1)	0,903 (95,6)	**0,921** (96,4)	0,799 (90,9)	0,643 (83,3)
bcw_benign*	**0,919** (96,2)	0,848 (92,1)	0,902 (95,3)	0,709 (83,2)	0,867 (93,3)
results continued on next page					

Dataset	OCRF	OCSVM	Gauss	Parzen	MoG
bcw_malignant*	**0,629** (81,3)	0,208 (68,2)	0,179 (46,3)	0,273 (69,1)	0,084 (49,6)
ionosphere_good	0,683 (83,3)	**0,785** (89,5)	0,781 (89,3)	0,180 (40,8)	0,584 (75,4)
ionosphere_bad	**0,169** (56,7)	-0,348 (28,2)	-0,410 (26,0)	0,106 (64,7)	-0,346 (33,2)
musk_0	0,071 (84,6)	0,103 (21,0)	**0,264** (83,6)	0,180 (30,6)	0 (84,6)
musk_1	0,306 (49,3)	0,049 (84,7)	**0,818** (95,1)	0,495 (88,9)	0 (15,4)
sonar_mines	0,048 (53,3)	**0,882** (93,6)	0,342 (65,9)	0 (46,2)	0,222 (47,8)
sonar_rocks	0,179 (59,0)	**0,889** (94,0)	0,120 (56,3)	0 (53,8)	0,274 (56,1)
diabetes_positive	0,139 (46,4)	0 (65,2)	0,147 (35,2)	0,188 (55,3)	**0,219** (39,2)
diabetes_negative	**0,241** (68,7)	0 (34,8)	-0,046 (66,5)	0,064 (53,9)	0,020 (68,3)
pendigits_0	**0,976** (99,6)	0 (89,6)	0,970 (99,4)	0,100 (89,7)	0,961 (99,3)
pendigits_1	0,585 (85,8)	0 (89,6)	0,652 (90,0)	0,212 (90,1)	**0,835** (96,6)
pendigits_2	0,835 (96,3)	0 (89,6)	**0,957** (99,2)	0 (89,6)	0,956 (99,2)
pendigits_3	0,918 (98,5)	0 (90,4)	**0,969** (99,5)	0,092 (90,4)	0,949 (99,1)
pendigits_4	0,961 (99,3)	0 (89,6)	**0,969** (99,4)	0 (89,6)	0,953 (99,1)
pendigits_5	0,756 (94,1)	0 (90,4)	0,880 (97,8)	0,092 (90,4)	**0,942** (99,0)
pendigits_6	**0,985** (99,7)	0 (90,4)	0,970 (99,5)	0 (90,4)	0,954 (99,2)
pendigits_7	0,887 (97,6)	0 (89,6)	0,887 (97,7)	0 (89,6)	**0,937** (98,8)
pendigits_8	0,634 (89,3)	0 (90,4)	0,716 (93,2)	0 (90,4)	**0,951** (99,2)
pendigits_9	0,577 (85,9)	0 (90,4)	0,577 (86,9)	0,093 (90,4)	**0,936** (98,9)
optdigits_0	0,776 (94,2)	0,165 (90,5)	**0,954** (99,2)	0 (90,1)	0,745 (95,9)
optdigits_1	0,147 (26,2)	0,054 (89,9)	**0,937** (98,9)	0 (89,8)	0,803 (96,7)
optdigits_2	0,143 (25,8)	0 (90,1)	**0,953** (99,2)	0 (90,1)	0,755 (96,0)
optdigits_3	0,121 (21,7)	0 (89,8)	**0,914** (98,4)	0 (89,8)	0,727 (95,5)
optdigits_4	0,077 (15,6)	0 (89,9)	**0,905** (98,3)	0 (89,9)	0,766 (96,1)
optdigits_5	0,041 (11,5)	0 (90,1)	**0,954** (99,2)	0 (90,1)	0,738 (95,8)
optdigits_6	0,410 (70,3)	0,026 (90,1)	**0,956** (99,2)	0 (90,1)	0,778 (96,3)
optdigits_7	0,264 (48,2)	0 (89,9)	**0,933** (98,8)	0 (89,9)	0,777 (96,3)
optdigits_8	0,043 (11,7)	0 (90,1)	**0,719** (93,6)	0 (90,1)	0,696 (95,2)
optdigits_9	0,077 (15,2)	0 (90,0)	**0,860** (97,4)	0 (90,0)	0,739 (95,7)
mfeat-factors_0	**0,844** (97,2)	0 (90,0)	0,737 (95,8)	0 (10,0)	0 (10,0)
mfeat-factors_1	**0,873** (97,8)	0 (90,0)	0,712 (95,4)	0 (10,0)	0 (10,0)
mfeat-factors_2	**0,879** (97,9)	0 (90,0)	0,740 (95,8)	0 (10,0)	0 (10,0)
mfeat-factors_3	**0,887** (98,0)	0,017 (90,0)	0,695 (95,1)	0 (10,0)	0 (10,0)
mfeat-factors_4	**0,884** (98,0)	0 (90,0)	0,743 (95,8)	0 (10,0)	0 (10,0)
mfeat-factors_5	**0,843** (97,3)	0,013 (90,0)	0,738 (95,8)	0 (10,0)	0 (10,0)
mfeat-factors_6	**0,910** (98,5)	0,068 (90,2)	0,770 (96,2)	0 (10,0)	0 (10,0)
mfeat-factors_7	**0,879** (97,9)	0,017 (90,0)	0,841 (97,3)	0 (10,0)	0 (10,0)
mfeat-factors_8	**0,613** (90,6)	0 (90,0)	0,647 (94,5)	0 (10,0)	0 (10,0)
mfeat-factors_9	**0,866** (97,6)	0,026 (90,1)	0,751 (96,0)	0 (10,0)	0 (10,0)
mfeat-karhunen_0	**0,807** (96,4)	0,363 (91,6)	0,784 (96,5)	0 (90,0)	0,302 (90,1)
mfeat-karhunen_1	0,750 (95,5)	0,248 (90,9)	**0,765** (96,1)	0 (90,0)	0,247 (90,3)
mfeat-karhunen_2	0,755 (95,5)	0,222 (90,7)	**0,776** (96,3)	0 (90,0)	0,213 (90,1)
mfeat-karhunen_3	0,703 (93,8)	0,239 (90,8)	**0,759** (96,0)	0,213 (90,1)	0,213 (90,1)

results continued on next page

Dataset	OCRF	OCSVM	Gauss	Parzen	MoG
mfeat-karhunen_4	**0,813** (96,5)	0,192 (90,6)	0,794 (96,6)	0 (90,0)	0,257 (90,1)
mfeat-karhunen_5	0,622 (91,6)	0,167 (90,5)	**0,730** (95,7)	0,213 (90,1)	0,229 (90,2)
mfeat-karhunen_6	0,684 (93,8)	0,255 (90,9)	**0,790** (96,5)	0,224 (90,2)	0,232 (90,2)
mfeat-karhunen_7	**0,864** (97,7)	0,532 (93,2)	0,849 (97,5)	0,213 (90,1)	0,257 (90,4)
mfeat-karhunen_8	0,407 (78,5)	0,030 (90,1)	**0,713** (95,4)	0 (90,0)	0,213 (90,0)
mfeat-karhunen_9	0,752 (95,4)	0,315 (91,2)	**0,770** (96,2)	0,213 (90,1)	0,220 (90,1)
mfeat-zernike_0	0,697 (93,5)	0 (90,0)	**0,944** (99,0)	0 (90,0)	0,637 (94,4)
mfeat-zernike_1	0,663 (92,7)	0 (90,0)	**0,908** (98,4)	0 (90,0)	0,686 (95,0)
mfeat-zernike_2	0,679 (93,5)	0 (90,0)	**0,903** (98,3)	0 (90,0)	0,512 (93,0)
mfeat-zernike_3	0,365 (77,3)	0,017 (90,0)	**0,674** (91,8)	0,213 (90,1)	0,617 (94,2)
mfeat-zernike_4	0,461 (84,2)	0 (90,0)	**0,908** (98,3)	0 (90,0)	0,653 (94,6)
mfeat-zernike_5	0,322 (72,4)	0,013 (90,0)	**0,721** (94,1)	0,213 (90,1)	0,535 (93,2)
mfeat-zernike_6	0,413 (79,7)	0,068 (90,2)	**0,551** (86,3)	-0,036 (86,4)	0,321 (87,4)
mfeat-zernike_7	0,796 (96,5)	0,013 (90,0)	**0,925** (98,7)	0,213 (90,1)	0,647 (94,6)
mfeat-zernike_8	0,548 (87,3)	0 (90,0)	**0,908** (98,4)	0 (90,0)	0,598 (93,9)
mfeat-zernike_9	0,455 (83,5)	0,026 (90,1)	**0,578** (87,5)	-0,045 (86,6)	0,337 (87,6)
mfeat-morph_0	0,698 (91,6)	0,136 (90,4)	0,682 (91,6)	**0,765** (94,5)	0,764 (94,5)
mfeat-morph_1	0,304 (56,5)	0 (90,0)	0,345 (65,2)	0,375 (82,2)	**0,395** (71,3)
mfeat-morph_2	0,291 (54,0)	0 (90,0)	0,400 (71,9)	**0,457** (81,2)	0,407 (72,9)
mfeat-morph_3	**0,335** (63,5)	0,030 (90,1)	0,326 (63,0)	0,298 (71,5)	0,328 (63,2)
mfeat-morph_4	0,294 (56,8)	0 (90,0)	0,432 (75,4)	**0,443** (87,2)	0,430 (75,3)
mfeat-morph_5	0,378 (67,4)	0,013 (90,0)	0,468 (78,6)	0,388 (86,5)	**0,468** (78,7)
mfeat-morph_6	**0,637** (88,7)	0,057 (90,1)	0,397 (71,7)	0,398 (75,6)	0,416 (74,0)
mfeat-morph_7	0,398 (70,0)	0,026 (90,1)	0,524 (82,8)	0,505 (88,0)	**0,540** (84,0)
mfeat-morph_8	**0,943** (98,9)	0,013 (90,0)	0,682 (91,6)	0,666 (91,7)	0,645 (89,9)
mfeat-morph_9	**0,456** (76,7)	0,013 (90,0)	0,389 (70,9)	0,395 (74,1)	0,398 (71,9)
glass_1	0,403 (66,2)	**0,896** (95,4)	0,465 (67,0)	0,484 (77,7)	0,509 (77,8)
glass_2	0,229 (56,5)	**0,880** (94,4)	0,212 (49,7)	0,322 (64,8)	0,365 (65,5)
glass_3	0,064 (69,0)	**0,908** (98,6)	0,179 (73,7)	0,145 (92,1)	0,091 (92,6)
glass_5	0,498 (90,9)	0,465 (96,6)	**0,964** (96,1)	0,307 (94,1)	0,823 (94,4)
glass_7	0,813 (95,0)	0,703 (95,4)	0,308 (67,2)	**0,877** (96,4)	0,749 (93,1)

Table A.7: Matthews correlation coefficient (MCC) obtained on all datasets for all one-class classifiers; accuracy rate is indicated in parenthesis. Best MCC results are indicated in bold face. *bcw refers to the *breast cancer wisconsin* dataset.

Appendix B. Multi-normality test results

Detailed results for the Mardia's multinormality test are given in Table B.8. The hypothesis of multinormality is accepted (A) only if both the absolute value of the statistical measure for Mardia's skewness test (Ms) and Mardia's kurtosis test (Mk) are smaller than their respective critical values (CVs and CVk). Otherwise, the hypothesis is rejected (R). We have used the publicly available implementation of A. Trujillo-Ortiz and R. Hernandez-Walls [88].

Dataset	Gauss	OCRF	M	Hs	Hk	Ms	CVs	Mk	CVk
iris_versicolour	0.903	0.579	A	A	A	23.70	31.41	-1.03	1.64
iris_virginica	0.813	0.614	A	A	A	24.73	31.41	-0.34	1.64
iris_setosa	0.921	0.722	A	A	A	24.22	31.41	0.81	1.64
bw_benign	0.902	0.919	R	R	R	17792	195.97	262.98	1.64
bw_malignant	0.179	0.629	R	R	A	379.05	195.97	0.19	1.64
ionosphere_good	0.781	0.683	-	-	-	-	7337.70	-	1.64
ionosphere_bad	-0.410	0.169	-	-	-	-	7337.70	-	1.64
musk_0	0.264	0.071	R	R	R	7427800	778270	759.52	1.64
musk_1	0.818	0.306	R	R	R	2510300	778270	400.30	1.64
sonar_mines	0.342	0.048	R	R	R	46219	38274	11.72	1.64
sonar_rocks	0.120	0.179	R	R	R	42422	38274	4.29	1.64
diabetes_positive	0.147	0.139	R	R	R	834.50	146.57	16.28	1.64
diabetes_negative	-0.046	0.241	R	R	R	2036.70	146.57	35.90	1.64
pendigits_0	0.970	0.976	R	R	R	35883	883.57	276.09	1.64
pendigits_1	0.652	0.585	R	R	R	25181	883.57	145.08	1.64
pendigits_2	0.957	0.835	R	R	R	29982	883.57	181.53	1.64
pendigits_3	0.969	0.918	R	R	R	93759	883.57	501.62	1.64
pendigits_4	0.969	0.961	-	-	-	-	883.57	-	1.64
pendigits_5	0.880	0.756	R	R	R	9777	883.57	47.19	1.64
pendigits_6	0.970	0.985	R	R	R	79284	883.57	423.84	1.64
pendigits_7	0.887	0.887	R	R	R	16781	883.57	72.32	1.64
pendigits_8	0.716	0.634	R	R	R	15720	883.57	71.33	1.64
pendigits_9	0.577	0.577	R	R	R	22974	883.57	117.02	1.64
optdigits_0	0.954	0.776	-	-	-	-	46259	-	1.64
optdigits_1	0.937	0.147	-	-	-	-	46259	-	1.64
optdigits_2	0.953	0.143	-	-	-	-	46259	-	1.64
optdigits_3	0.914	0.121	-	-	-	-	46259	-	1.64
optdigits_4	0.905	0.077	-	-	-	-	46259	-	1.64
optdigits_5	0.954	0.041	-	-	-	-	46259	-	1.64
optdigits_6	0.956	0.410	-	-	-	-	46259	-	1.64
optdigits_7	0.933	0.264	-	-	-	-	46259	-	1.64
optdigits_8	0.719	0.043	-	-	-	-	46259	-	1.64
optdigits_9	0.860	0.077	-	-	-	-	46259	-	1.64
mfeat_factors_0	0.737	0.844	R	A	R	1182300	1706100	-197.80	1.64
mfeat_factors_1	0.712	0.873	R	A	R	1294000	1706100	-181.54	1.64
mfeat_factors_2	0.740	0.879	R	A	R	1307400	1706100	-174.76	1.64
mfeat_factors_3	0.695	0.887	R	A	R	1274800	1706100	-190.40	1.64
mfeat_factors_4	0.743	0.884	R	A	R	1292000	1706100	-182.07	1.64
mfeat_factors_5	0.738	0.843	R	A	R	1403200	1706100	-122.57	1.64
mfeat_factors_6	0.770	0.910	R	A	R	1261500	1706100	-203.41	1.64
mfeat_factors_7	0.841	0.879	R	A	R	1322600	1706100	-172.74	1.64
mfeat_factors_8	0.647	0.613	R	A	R	1313900	1706100	-172.14	1.64

Mardia's results continued on next page

Dataset	Gauss	OCRF	M	Hs	Hk	Ms	CVs	Mk	CVk
mfeat_factors_9	0.751	0.866	R	A	R	1259600	1706100	-199.69	1.64
mfeat_karhunen_0	0.784	0.807	R	R	R	60984	46259	21.57	1.64
mfeat_karhunen_1	0.765	0.750	R	R	R	62413	46259	23.83	1.64
mfeat_karhunen_2	0.776	0.755	R	R	R	59144	46259	17.76	1.64
mfeat_karhunen_3	0.759	0.703	R	R	R	63683	46259	23.30	1.64
mfeat_karhunen_4	0.794	0.813	R	R	R	62594	46259	23.28	1.64
mfeat_karhunen_5	0.730	0.622	R	R	R	56194	46259	12.32	1.64
mfeat_karhunen_6	0.790	0.684	R	R	R	63072	46259	24.79	1.64
mfeat_karhunen_7	0.849	0.864	R	R	R	70505	46259	38.11	1.64
mfeat_karhunen_8	0.713	0.407	R	R	R	54754	46259	9.40	1.64
mfeat_karhunen_9	0.770	0.752	R	R	R	59354	46259	18.11	1.64
mfeat_zernike_0	0.944	0.697	R	R	R	45559	18741	65.53	1.64
mfeat_zernike_1	0.908	0.663	R	R	R	52521	18741	81.09	1.64
mfeat_zernike_2	0.903	0.679	R	R	R	31692	18741	32.08	1.64
mfeat_zernike_3	0.674	0.365	R	R	R	39827	18741	54.49	1.64
mfeat_zernike_4	0.908	0.461	R	R	R	42016	18741	59.67	1.64
mfeat_zernike_5	0.721	0.322	R	R	R	33583	18741	36.38	1.64
mfeat_zernike_6	0.551	0.413	R	R	R	42811	18741	65.64	1.64
mfeat_zernike_7	0.925	0.796	R	R	R	34880	18741	42.32	1.64
mfeat_zernike_8	0.908	0.548	R	R	R	38553	18741	52.73	1.64
mfeat_zernike_9	0.578	0.455	R	R	R	45975	18741	71.96	1.64
mfeat_morph_0	0.682	0.698	-	-	-	-	74.47	-	1.64
mfeat_morph_1	0.345	0.304	R	R	R	10122	74.47	218.95	1.64
mfeat_morph_2	0.400	0.291	R	R	R	10030	74.47	201.90	1.64
mfeat_morph_3	0.326	0.335	-	-	-	-	74.47	-	1.64
mfeat_morph_4	0.432	0.294	R	R	R	7960.40	74.47	168.16	1.64
mfeat_morph_5	0.468	0.378	-	-	-	-	74.47	-	1.64
mfeat_morph_6	0.397	0.637	R	R	R	12871.00	74.47	300.58	1.64
mfeat_morph_7	0.524	0.398	-	-	-	-	74.47	-	1.64
mfeat_morph_8	0.682	0.943	R	R	R	4434.20	74.47	90.45	1.64
mfeat_morph_9	0.389	0.456	R	R	R	10504	74.47	248.09	1.64
glass_1	0.465	0.403	R	R	R	1005.50	195.97	18.55	1.64
glass_2	0.212	0.229	R	R	R	1397.60	195.97	28.80	1.64
glass_3	0.179	0.064	R	A	R	143.56	195.97	-2.33	1.64
glass_5	0.964	0.498	R	A	R	109.14	195.97	-3.25	1.64
glass_7	0.308	0.813	R	R	R	314.42	195.97	2.74	1.64

Table B.8: Details for Mardia's multivariate skewness and kurtosis statistical test, reported with the MCC values of Gauss classifier and OCRF; M is the Mardia's test result, Hs the result for the skewness test, Hk for the kurtosis tests (the hypothesis of multi-normality is either rejected (R) or accepted (A)), Ms the statistical value for Mardia's skewness test, Mk the measure for Mardia's kurtosis test, CVs the critical value for the skewness test and CVk for the kurtosis test. Missing values are indicated with an hyphen; they are related to computational issues due to singular variance-covariance matrix.

References

[1] M. Moya, D. Hush, Network constraints and multi-objective optimization for one-class classification, Neural Networks 9 (3) (1996) 463–474.

[2] M. Koppel, J. Schler, Authorship verification as a one-class classification problem, in: International Conference on Machine Learning, ACM, 2004, p. 62.

[3] K. Hempstalk, E. Frank, I. Witten, One-class classification by combining density and class probability estimation, Machine Learning and Knowledge Discovery in Databases (2008) 505–519.

[4] A. Brew, M. Grimaldi, P. Cunningham, An evaluation of one-class classification techniques for speaker verification, Artificial Intelligence Review 27 (4) (2007) 295–307.

[5] O. Mazhelis, One-class classifiers: A review and analysis of suitability in the context of mobile-masquerader detection, South African Computer Journal (SACJ), ARIMA & SACJ Joint Special Issue on Advances in End-User Data-Mining Techniques 36 (2006) 29–48.

[6] W. Fan, M. Miller, S. Stolfo, W. Lee, P. Chan, Using artificial anomalies to detect unknown and known network intrusions, Knowledge and Information Systems 6 (5) (2004) 507–527.

[7] K. Wang, S. Stolfo, One-class training for masquerade detection, in: Workshop on Data Mining for Computer Security, 2003, pp. 19–22.

[8] C. Desir, S. Bernard, C. Petitjean, L. Heutte, A random forest based approach for one-class classification in medical imaging, 3rd MICCAI International Workshop on Machine Learning in Medical Imaging (MLMI), Nice, France 7588 (2012) 250–257.

[9] V. Hodge, J. Austin, A survey of outlier detection methodologies, Artificial Intelligence Review 22 (2) (2004) 85–126.

[10] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Computing Surveys (CSUR) 41 (3) (2009) 1–58.

[11] R. Duda, P. Hart, Pattern classification and scene analysis, John Wiley and sons, 1973.

[12] D. Tax, R. Duin, Combining one-class classifiers., in: J. Kittler, F. Roli (Eds.), Multiple Classifier Systems, Vol. 2096 of LNCS, Springer, 2001, pp. 299–308.

[13] S. Khan, M. Madden, A survey of recent trends in one class classification, Artificial Intelligence and Cognitive Science (2010) 188–197.

[14] C. Desir, S. Bernard, C. Petitjean, L. Heutte, A new random forest method for one-class classification, in: IAPR International Workshop on Statistical Techniques in Pattern Recognition (SPR), Hiroshima, Japan, Vol. 7626 of LNCS, 2012, pp. 282–290.

[15] L. Breiman, Random forests, Machine Learning Vol. 45 (1) (2001) 5–32.

[16] B. Ng, Survey of anomaly detection methods, Lawrence Livermore National Laboratory. Livermore, CA, 2006.

[17] V. Chandola, A. Banerjee, V. Kumar, Outlier detection: A survey, ACM Computing Surveys (2009) 1–72.

[18] S. Marsland, Novelty detection in learning systems, Neural computing surveys 3 (2003) 157–195.

[19] M. Markou, S. Singh, Novelty detection: a review–part 1: statistical approaches, Signal Processing 83 (12) (2003) 2481–2497.

[20] N. Abe, B. Zadrozny, J. Langford, Outlier detection by active learning, in: ACM SIGKDD International Conference on Knowledge discovery and data mining, 2006, pp. 504–509.

[21] D. Tax, R. Duin, Outlier detection using classifier instability, LNCS (1998) 593–601.

[22] A. Nairac, T. Corbett-Clark, R. Ripley, N. Townsend, L. Tarassenko, Choosing an appropriate model for novelty detection, in: Artificial Neural Networks, Fifth International Conference on (Conf. Publ. No. 440), IET, 1997, pp. 117–122.

[23] J. Toivola, M. Prada, J. Hollmén, Novelty detection in projected spaces for structural health monitoring, Advances in Intelligent Data Analysis IX (2010) 208–219.

[24] L. Tarassenko, D. Clifton, P. Bannister, S. King, D. King, Novelty detection, Encyclopedia of Structural Health Monitoring.

[25] G. Cohen, H. Sax, A. Geissbuhler, Novelty Detection using One-class Parzen Density Estimator. An Application to Surveillance of Nosocomial Infections, in: EHealth Beyond the Horizon: Get It There: Proceedings of MIE2008 the XXIst International Congress of the European Federation for Medical Informatics, Ios Pr Inc, 2008, p. 21.

[26] L. Tarassenko, P. Hayton, N. Cerneaz, M. Brady, Novelty detection for the identification of masses in mammograms, in: Fourth International Conference on Artificial Neural Networks, 1995, pp. 442–447.

[27] C. Bishop, Novelty detection and neural network validation, IEE Proceedings-Vision, Image and Signal processing 141 (4) (1994) 217–222.

[28] R. Duin, On the choice of smoothing parameters for Parzen estimators of probability density functions, Computers, IEEE Transactions on 100 (11) (1976) 1175–1179.

[29] M. Kraaijveld, R. Duin, A criterion for the smoothing parameter for parzen-estimators of probability density functions, Tech. rep., Delft University of Technology (1991).

[30] C. M. Bishop, Neural Networks for pattern recognition, Clarendon Press Oxford, 1995.

[31] T. Hastie, R. Tibshirani, J. H. Friedman, The Elements of Statistical Learning, Springer Verlag, 2001.

[32] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum likelihood from incomplete data via the em algorithm, J. Royal Statisfical Society Vol. 39(1) (1977) pp. 1–38.

[33] S. Roberts, L. Tarassenko, A probabilistic resource allocating network for novelty detection, Neural Computation 6 (2) (1994) 270–284.

[34] D. Tax, R. Duin, Support vector data description, Machine learning 54 (1) (2004) 45–66.

[35] K. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft, When is ”nearest neighbor” meaningful?, Database Theory-ICDT 99 (1999) 217–235.

[36] M. Yousef, N. Najami, W. Khalifa, A comparison study between one-class and two-class machine learning for microrna target detection, Journal of Biomedical Science and Engineering.

[37] F. Ratle, M. Kanevski, A. Terrettaz-Zufferey, P. Esseiva, O. Ribaux, A comparison of one-class classifiers for novelty detection in forensic case data, Intelligent Data Engineering and Automated Learning-IDEAL 2007 (2007) 67–76.

[38] M. Markou, S. Singh, Novelty detection: a review–part 2: Neural network based approaches, Signal Processing 83 (12) (2003) 2499–2521.

[39] M. Desforges, P. Jacob, J. Cooper, Applications of probability density estimation to the detection of abnormal conditions in engineering, Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science 212 (8) (1988) 687–703.

[40] N. Japkowicz, C. Myers, M. Gluck, A novelty detection approach to classification, in: International Joint Conference on Artificial Intelligence, Vol. 14, 1995, pp. 518–523.

[41] V. Vapnik, Statistical Learning Theory, Wiley Interscience, 1998.

[42] B. Liu, Y. Xia, P. Yu, Clustering through decision tree construction, in: Proceedings of the ninth international conference on Information and knowledge management, ACM New York, NY, USA, 2000, pp. 20–29.

[43] C. Aggarwal, P. Yu, Outlier detection for high dimensional data, in: ACM SIGMOD International Conference on Management of Data, 2001, p. 46.

[44] B. Scholkopf, J. Platt, J. Shawe-Taylor, A. Smola, R. Williamson, Estimating the support of a high-dimensional distribution, Neural computation 13 (7) (2001) 1443–1471.

[45] A. Shieh, D. Kamm, Ensembles of one class support vector machines, Multiple Classifier Systems (2009) 181–190.

[46] P. Evangelista, M. Embrechts, B. Szymanski, Taming the curse of dimensionality in kernels and novelty detection, Applied Soft Computing Technologies: The Challenge of Complexity (2006) 425–438.

[47] T. Dietterich, Ensemble methods in machine learning., in: J. Kittler, F. Roli (Eds.), Multiple Classifier Systems, Vol. 1857 of LNCS, Springer, 2000, pp. 1–15.

[48] L. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, Wiley, 2004.

[49] R. Genuer, J. Poggi, C. Tuleau, Random forests: some methodological insights, Arxiv preprint arXiv:0811.3619.

[50] G. Biau, Analysis of a random forests model, Arxiv preprint arXiv:1005.0208.

[51] M. Robnik-Sikonja, Improving random forests, Machine Learning: ECML 2004 (2004) 359–370.

[52] L. Kuncheva, J. Rodríguez, An experimental study on rotation forest ensembles, Multiple Classifier Systems (2007) 459–468.

[53] S. Bernard, L. Heutte, S. Adam, Influence of hyperparameters on random forest accuracy, Multiple Classifier Systems (2009) 171–180.

[54] L. Breiman, Bagging predictors, Machine Learning 26(2) (1996) 123–140.

[55] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Machine Learning 63 (1) (2006) 3–42.

[56] S. Bernard, L. Heutte, S. Adam, Forest-rk: A new random forest induction method, Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence (2008) 430–437.

[57] D. Tax, R. Duin, Uniform object generation for optimizing one-class classifiers, The Journal of Machine Learning Research 2 (2002) 155–173.

[58] T. Ho, The random subspace method for constructing decision forests, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (8) (1998) 832–844.

[59] D. Tax, A. Ypma, R. Duin, Support vector data description applied to machine vibration analysis, in: Proc. 5th Annual Conference of the Advanced School for Computing and Imaging (Heijen, NL), 1999.

[60] C. Blake, C. Merz, UCI Repository of machine learning databases [http://www. ics. uci. edu/˜ mlearn/MLRepository. html]. University of California, Department of Information and Computer Science 55.

[61] P. Baldi, S. Brunak, Y. Chauvin, C. Andersen, H. Nielsen, Assessing the accuracy of prediction algorithms for classification: an overview, Bioinformatics 16 (5) (2000) 412–424.

[62] B. Matthews, Comparison of the predicted and observed secondary structure of t4 phage lysozyme, Biochimica et Biophysica Acta (BBA)-Protein Structure 405 (2) (1975) 442–451.

[63] T. Hastie, R. Tibshirani, J. Friedman, J. Franklin, The elements of statistical learning: data mining, inference and prediction, The Mathematical Intelligencer 27 (2) (2005) 83–85.

[64] R. Kohavi, Wrappers for performance enhancement and oblivious decision graphs, Tech. rep., Stanford University, Stanford, CA (1996).

[65] Z. Zheng, A benchmark for classifier learning, in: Proceedings of the Sixth Australian Joint Conference on Artificial Intelligence, 1993, pp. 281–286.

[66] R. Duin, A note on comparing classifiers, Pattern Recognition Letters 17 (5) (1996) 529–536.

[67] D. Hand, Classifier technology and the illusion of progress, Statistical Science 21 (1) (2006) 1–14.

[68] A. Jamain, D. J. Hand, Mining supervised classification performance studies: A meta-analytic investigation, J. Classif. 25 (2008) 87–112.

[69] J. Demsar, Statistical comparisons of classifiers over multiple data sets, The Journal of Machine Learning Research 7 (2006) 1–30.

[70] T. G. Dietterich, Approximate statistical tests for comparing supervised classification learning algorithms, Neural Computation 10 (7) (1998) 1895–1924.

[71] C. Nadeau, Y. Bengio, Inference for the generalization error, Machine Learning 52 (3) (2003) 239–281.

[72] P. Brazdil, C. Soares, A comparison of ranking methods for classification algorithm selection, Machine Learning: ECML 2000 (2000) 63–75.

[73] J. Menke, T. Martinez, Using permutations instead of student's t distribution for p-values in paired-difference algorithm comparisons, in: IEEE International Joint Conference on Neural Networks, Vol. 2, 2004, pp. 1331–1335.

[74] R. Banfield, L. Hall, K. Bowyer, W. Kegelmeyer, A comparison of decision tree ensemble creation techniques, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (1) (2007) 173–180.

[75] S. Fisher, Statistical methods and scientific inference, Vol. 1959, Oliver and Boyd, 1959.

[76] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, Journal of the American Statistical Association 32 (200) (1937) 675–701.

[77] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, The Annals of Mathematical Statistics 11 (1) (1940) 86–92.

[78] P. Nemenyi, Distribution-free multiple comparisons, Ph.D. thesis, Princeton University (1963).

[79] R. Iman, J. Davenport, Approximations of the critical region of the friedman statistic, Tech. rep., Sandia Labs., Albuquerque, NM & Texas Tech Univ., Lubbock (USA) (1979).

[80] O. Dunn, Multiple comparisons among means, Journal of the American Statistical Association (1961) 52–64.

[81] R. Duin, PRTools version 3.0: A matlab toolbox for pattern recognition, in: Proc. of SPIE, Citeseer, 2000.

[82] D. Tax, DDtools, the Data Description Toolbox for Matlab, version 1.7.3 (Dec 2009).

[83] L. M. Manevitz, M. Yousef, One-class svms for document classification, J. Mach. Learn. Res. 2 (2002) 139–154.

[84] K. Mardia, Applications of some measures of multivariate skewness and kurtosis in testing normality and robustness studies, Sankhya: The Indian Journal of Statistics, Series B (1974) 115–128.

[85] A. Von Eye, G. Bogat, Testing the assumption of multivariate normality, Psychology Science 46 (2004) 243–258.

[86] M. Verleysen, et al., Learning high-dimensional data, Nato Science Series Sub Series III Computer And Systems Sciences 186 (2003) 141–162.

[87] D. Donoho, Aide-memoire. High-dimensional data analysis: The curses and blessings of dimensionality, American Math. Society Lecture - Math Challenges of the 21st Century.

[88] A. Trujillo-Ortiz, R. Hernandez-Walls., Mskekur: Mardia's multivariate skewness and kurtosis coefficients and its hypotheses testing. A MATLAB file (2003), `http://www.mathworks.com/matlabcentral/fileexchange/3519`.