Information Fusion 63 (2020) 30–40

Contents lists available at ScienceDirect

Information Fusion

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

A dynamic ensemble outlier detection model based on an adaptive k-nearest neighbor rule

Biao Wang[a,*], Zhizhong Mao[b]

[a] School of Automation, Shenyang Aerospace University, 110136 Shenyang, China
[b] School of Information Science and Engineering, Northeastern University, 110819 Shenyang, China

ARTICLE INFO

Keywords:
Outlier detection
Ensemble learning
Dynamic classifier selection
Adaptive k-nearest neighbor

ABSTRACT

Ensembles of outlier detectors are drawing increasing attentions recently, in spite of the difficulty on developing ensembles in the framework of unsupervised learning. We have noted that existing outlier ensembles often use certain fusion rules (e.g. majority voting) to aggregate individual learners. Theoretically, these individuals are assumed to be error-independent so that single models can be outperformed by the ensemble. However, it is of great difficulty to satisfy this assumption in practical applications. By dynamic selecting more competent individual(s) for each test pattern, this problem can be alleviated effectively. Inspired by this idea, this paper proposes a dynamic ensemble outlier detection model using one-class classifiers as base learners. As the competences of base detectors are estimated totally on data points in the validation set, its impact on the selection is significant. In order to achieve an efficient selection, we propose an adaptive k-nearest neighbor (KNN) rule, instead of traditional KNN algorithm, to constitute the validation set for each test pattern. Our adaptive KNN rule firstly uses algorithm support vector data description (SVDD) to mine the local area where class conditional probabilities are not constant in terms of the corresponding test pattern. Competences estimated with neighbor patterns in this area should thus be more accurate than that by traditional KNN rule. A probabilistic model that uses posterior probabilities of one-class classifiers is used then to estimate classifier competences. We present experimental evidence of the detection performance improvement over single models and over a variety of static ensemble models, by using data sets from UCI repository.

1. Introduction

The importance of outlier detection resides in the fact that outliers in data translate to significant, and often critical, actionable information. Outliers in computer systems may indicate underlying malicious activities, outliers in credit cards may indicate fraud behaviors, outliers in medical systems may indicate underlying diseases, and outliers in industrial systems may indicate system faults, just to name a few. In this situation, numerous outlier detection methods have been proposed in recent years. A detailed survey classifies these detection methods into five main groups, i.e. classification-based techniques, distance-based techniques, clustering-based techniques, statistics-based techniques, and information theory based techniques. Then, from the perspective of the availability of data labels, these methods can also be classified into supervised, unsupervised, and semi-supervised detection techniques [1]. We have noted that semi-supervised detection methods are also referred to as one-class classifiers from the perspective of classification, since they assume that all training patterns originate from single class.

Interest in one-class classification methods has soared in recent years due to its wide applicability in many practical problems where classification in the absence of counterexamples is needed. It works with the assumption that during the training phase it has only objects originating from a single class at our disposal [2]. This may be caused by cost restraints, difficulties or ethical implication of collecting some samples or simply complete lack of ability to access or generate objects. As we have no access to any counterexample during the training phase, constructing an efficient model and selecting optimal parameters become a very demanding task. Among them a combined classification seems a promising direction. Recently, we have seen a significant development of algorithms known as ensemble classifiers or multiple classifier systems. Their success lies in the ability to tackle complex tasks by decomposition, utilizing different properties of each model and taking advantage of collective classification abilities. Classifier ensemble can work properly only if base classifiers should be characterized by high individual quality and be at the same time mutually complementary. In a nutshell, these ensemble models assume that different base classifiers can make

* Corresponding author.
E-mail address: wangbiao_auto@sau.edu.cn (B. Wang).

https://doi.org/10.1016/j.inffus.2020.05.001
Received 5 October 2019; Received in revised form 15 April 2020; Accepted 2 May 2020
Available online 24 May 2020

independent errors. In real outlier detection applications, nevertheless, it is difficult to design a set of detection models that satisfy such an assumption.

In traditional classification problem, the idea of dynamic classifier selection (DCS) is proposed to alleviate the issue of error-independence assumption in static ensembles [3]. The rationale of DCS is based on an assumption that there is always a trade-off between the flexibility and generalization abilities of a given ensemble system [4]. By generating a larger pool of diverse base classifiers we potentially obtain improved classification accuracy and be prepared for various types of outliers to appear. However, increasing sizes of classifier pool also indicates increasing the probability of including non-competent learners. The combination of non-competent can deteriorate the overall performance of ensemble. To this end, a proper classifier selection mechanism may be effective. A critical point in DCS is that our choice of one individual classifier over the rest will depend on how much we trust the estimate of the generalization of the classifiers [5]. In addition to DCS, the idea of dynamic ensemble selection (DES) is also introduced as an alternative. The advantage of DES is that we distribute the risk of this over-generalization by choosing a group of classifiers instead of one individual classifier for a test pattern [6].

Inspired by the dynamic selection mechanisms in multi-class classification, this paper proposes a dynamic outlier ensemble in which we use one-class classifiers as base learners. Being identical with other dynamic selection mechanisms, our dynamic ensemble also have three main processes, i.e. base classifier generation, validation set determination, and competence estimates [7-9]. At the base classifier generation process, we use Bagging and Random Subspace techniques to train a pool of diverse one-class classifiers. The proposed adaptive KNN rule is used in the determination of validation set. Note that many researches have showed that the performance of DCS methods are much lower than that of the oracle, i.e. of an ideal selector that always select the optimal classifier [3]. The reason may be that some characteristics of the basic KNN rule will become less appealing on finite training samples in a high dimensional input feature space. In our method, we firstly use algorithm SVDD to find the decision boundary around which class conditional probabilities of examples are likely to be not constant. Then for test patterns close to the boundary, we can use the local information extracted by SVDD to produce highly stretched neighborhoods along boundary directions. As a result, the class conditional probabilities tend to be constant in the modified neighborhoods. Finally, with these neighbor patterns, we use a probabilistic model to estimate competences of all base classifiers. Finally, the most competent base detector(s) for corresponding test patterns can be selected.

The remaining of this paper is organized as followings. In Section 2, we review some literature concerning ensemble outlier detection. Some important concepts are introduced in Section 3. Section 4 will present the proposed method in detail. Extensive experiments will be conducted in Section 5. Finally, in Section 6, some conclusions will be concluded.

2. Literature review

Recently, some ensemble outlier detection methods have been developed in order to alleviate some problems associated with single models. In these ensemble detection models, one-class classifiers are used as base learners, which are also referred to as base classifiers accordingly. Here we category these ensemble models into three groups according to their combination mechanisms.

We refer to the first group as "*Static Ensemble* (*SE*)", in which base classifiers are aggregated by fixed or well-trained functions. Performance of averaging and multiplying rule are discussed in [10]. To cope with all possible situations with missing data, one-class classifiers trained on one-dimensional problems, *n*-dimensional problems or features are combined in dissimilarity representations in [11]. Bagging and Random Subspace (RS) are combined together to generate diverse training subsets for an improved SVDD (support vector data description) in [12] to detect outliers in process control systems. SVDD-based classifiers are combined for image database retrieval to improve the performance of single one-class classifier in [13]. Clustering algorithms like *k*-means and fuzzy *c*-means are also proposed to generate diverse training subsets in [14,15].

The second group is referred to as "*Pruned Ensemble* (*PE*)", where only part of elite classifiers are selected from the pool to construct a sub-ensemble model. In contrast to multi-class classification, to which many diversity measures can be applied [16], designing diversity measures for one-class classifiers may be nontrivial since none counterexample is available. With a combined criterion that consists of consistency and pairwise diversity, Krawczyk [17] employs firefly algorithm to complete the task of searching for the best possible subset of classifiers. Similarly, with an exponential consistency measure and a non-pairwise diversity measure as criterion, Parhizkar and Abadi [18] uses a novel binary artificial bee colony algorithm to prune the initial ensemble so that a near-optimal sub-ensemble can be found. An ensemble pruning methodology using non-monotone Simple Coalitional Games is proposed in [19]. According to their empirical results, PE methods usually outperform SE methods.

The third group is referred to as "*Dynamic Ensemble* (*DE*)" owning to their dynamic selection mechanisms. DE techniques rely on the assumption that different classifiers are competent ("experts") in different local regions of feature space. So the key issue in DE is to estimate the competence of base classifiers on any test pattern [20]. Due to the difficulty residing in estimating competence measure, researches concerning dynamic one-class classifier selection method are very limited. In [21], three competence measures for one-class classifiers are developed and one of base classifiers is delegated dynamically to the decision area where it is the most competent. In [22], a dynamic ensemble outlier detection method on the basis of one-class classification is proposed to process monitoring in industrial processes.

3. Basic concepts

In this section, we present the basic concepts regarding one-class classification and ensemble one-class classifiers. This is beneficial to understanding our dynamic ensemble outlier detection model described in Section 4.

3.1. One-class classification

Because one-class classifiers will be used as base detectors of the proposed ensemble, here we introduce the basic idea of one-class classification, followed by several commonly used one-class classifiers. One-class classifiers operate with the assumption that during the training phase only patterns originating from a single class are available. This class is often referred to as the target class or normal class. Then patterns that do not belong to this class are deemed outliers. The significant difference from binary classification is that during the training phase outlier examples are available for binary classification. A good one-class classifier will achieve both a small fraction of false negatives, as well as a small fraction of false positives theoretically. Many techniques have been developed to one-class classifiers in the past years. They are usually be categorized into three groups according to the definition of target concept.

A. The first group of one-class classifiers are developed based on density estimation technique. Given a data set, density estimation technique is used to learn the target concept. The rationale of this group is simple and the performance can be very effective when training patterns are sufficient. However, this may also be its limitation for the situation of small samples. The most widely used classifiers of this group should be Gaussian, mixture of Gaussian, and Parzen [23].
B. The second group of one-class classifiers define the target concept by constructing a boundary surrounding all target patterns with the

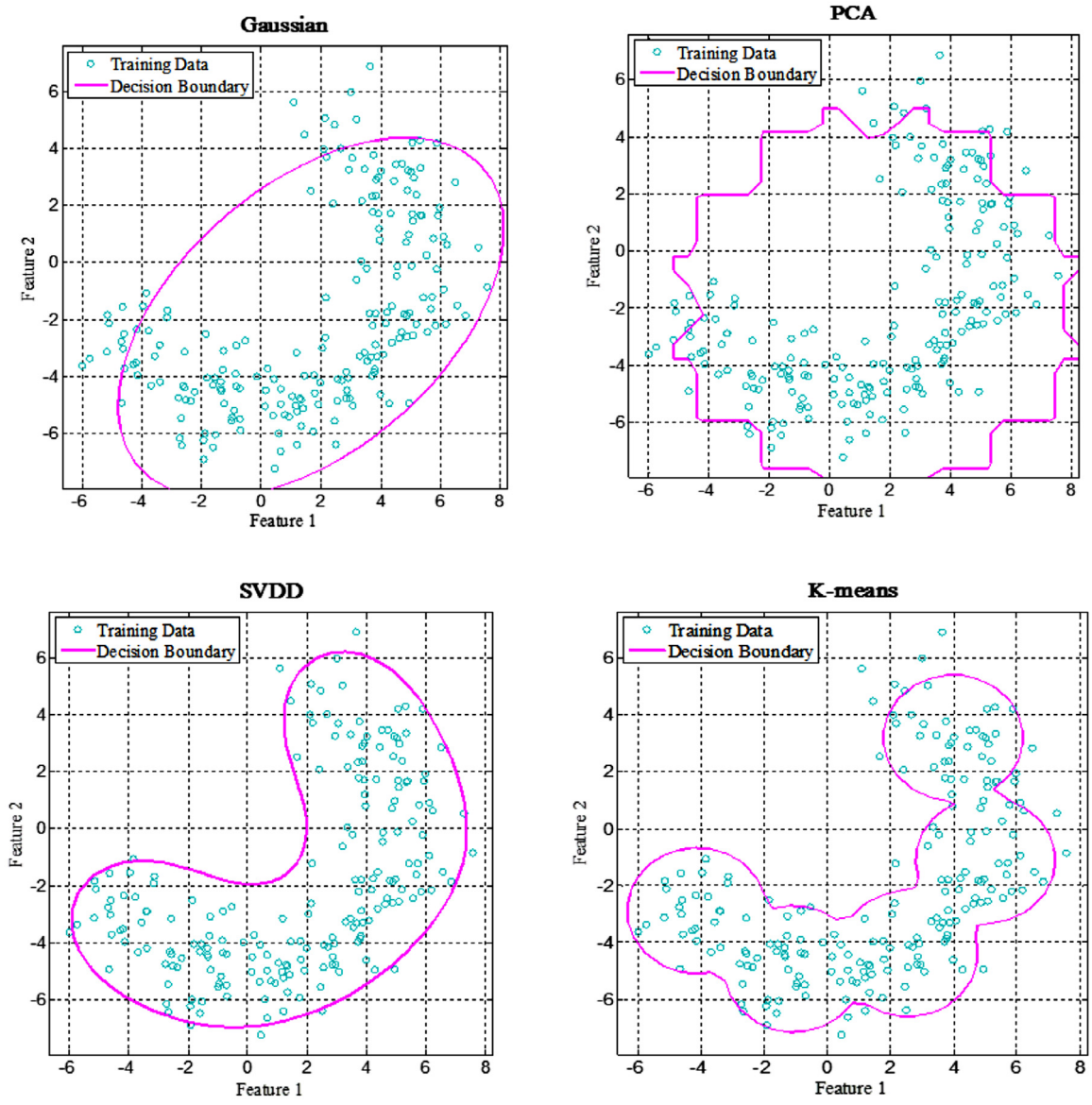

Fig. 1. Demonstration of four one-class classifiers on a 2-dimension data set.

volume as small as possible. In contrast to density estimation, the construction of a boundary has less requirement for data size. The commonly used boundary-based one-class classifiers may be support vector data description and one-class support vector machine [24,25].

C. The third group of one-class classifiers are developed from some data modeling techniques, which have not been primarily constructed for one-class classification. These methods usually make assumptions about the clustering characteristics if the data or their distribution in subspaces. A set of prototypes or subspaces is defined and a reconstruction error is minimized. This group is hence referred to as reconstruction-based classifier. Outliers are then assumed that do not satisfy the assumptions about the target distribution, leading to higher reconstruction errors. Algorithm K-means, principal component analysis, self-organizing map, and auto-encode neural network are often used in this group [26,27].

For illustration we show results of several common one-class classifiers on a two-dimensional data set in Fig. 1. We can find that different classifiers usually have different results for the same training set. This may broadcast light to the proposal of a selection scheme.

3.2. Ensemble one-class classifier

Ensemble technique has been extensively studied in the context of many data mining problems such as classification and clustering. Its application in outlier detection or one-class classification is rather limited. The main reason is the natural absence of ground truth when learning data models. Most ensemble models require explicit evaluation criteria in order to show their relative merits over base learners. For example, accuracy in multi-class classification, and precision in binary classification. Furthermore, evaluation criteria are often used in the intermediate steps of an ensemble algorithm, in order to make future decisions about the precise construction of the ensemble. Among all core data mining problems, outlier analysis is the hardest to evaluate because of a combination of its small sample space and unsupervised nature [28]. The small sample space issue make it difficult to quantify an approach in a statistically robust way. While the unsupervised nature necessitates the construction of simpler ensembles with fewer qualitative decisions about the choice of the components in the ensemble.

Even though the unsupervised nature of outlier detection makes ensemble analysis more challenging, the construction of outlier ensembles

is still very beneficial. According to the theoretical foundations provided in [29], bias-variance theory in the classification domain can also be extended to outlier ensembles. From the perspective of bias reduction or variance reduction, outlier ensembles can achieve accuracy improvements over single models. On the other hand, the construction of outlier ensembles can also alleviate the model selection problem. Note that unlike classification, where one can perform model selection and parameter tuning with the use of labels (e.g. cross validation), this is impossible in outlier detection. Therefore, outlier detection algorithms are often very subjective, and the results of different algorithms often vary more significantly over different data sets. In a nutshell, the performance gains in outlier ensembles are often inherently more useful and significant because they partially overcome the weak performances resulting from lack of supervision.

From the perspective of variance reduction, techniques such as Bagging and Random Subspace can be easily used in outlier ensembles. By generating more training subsets from the given training set, a pool of diverse base learners can be obtained and aggregated. Experimental results have proved the accuracy improvement over single models. It is noteworthy that those subspace-based ensembles are very effective for high-dimensional data sets [29,30]. From the perspective of bias reduction, many techniques in classification ensembles cannot directly applied to outlier detection, since bias-reduction algorithms often require a quantification of error in intermediate steps of the algorithm (e.g. Boosting). In spite of this, one can still perform heuristic forms of bias reduction in unsupervised settings. The key idea is that the output of base outlier detectors provided a kind of weak ground truth, which can be used in a limited way to define the steps required for bias improvement. In this case, bias-reduction outlier ensembles are always a guessing game since the above weak ground truth could be wrong. As a consequence, these heuristic improvements do not always provide better results, although they are useful in many settings [31–33].

4. Methodology

Being like dynamic selection mechanisms in multi-class classification, our dynamic outlier ensemble that uses one-class classifiers as base learners also includes three main processes, i.e. base classifier generation, validation set determination, and competence estimation. The following subsections will present these processes in detail.

4.1. Base classifier generation

Classifier ensemble can work properly only if base classifiers should be characterized by high individual quality and be at the same time mutually complementary. Actually, the requirement for mutual complementarity is very difficult to satisfy because we can hardly know the competent regions of base classifiers a priori. Therefore, most ensemble methods focus on generating more diverse base classifiers, with the aim of achieving higher performance. One efficient way is to generate diverse training subsets (e.g. Bagging) or feature subsets (e.g. Random Subspace) from original training sets and use one algorithm to train base classifiers on these subsets [34,35]. Another way is to use various parameter configurations of one algorithm to train base classifiers. Methods developed by the above two ways are usually referred to as homogeneous ensembles, since only one base algorithm is used in these methods. In contrast to homogeneous ensembles, we can also use different algorithms to constitute a pool of diverse base learners, and such models are referred to as heterogeneous ensembles.

In order to underline the proposed dynamic mechanism sufficiently, we will verify our method in both homogeneous and heterogeneous frameworks. For homogeneous ensembles, technique Bagging and Random subspace are used to generate training subsets firstly. Then we will train a boundary-based one-class classifier, i.e. SVDD model on each subset. The reason for the selection of SVDD from several one-class classifiers resides in its better performance, as proved in [36]. The basic idea of SVDD is to construct a hypersphere that encloses all training points. The volume of this hypersphere should be constrained simultaneously as small as possible so that less outliers would be accepted by the target class. Kernel trick is also used in SVDD in order to achieve a flexible description [37]. In the heterogeneous ensemble, we use ten different one-class classifiers. It is desired that more diversity can be obtained by selecting these classifiers from different groups, since similar base learners may contribute less to the ensemble. A simple description of these one-class classifiers will be listed in Section 5.4 and more details can also be found in [38].

After training base classifiers, the issue of scaling should be paid extra attention, prior to the selection phase. Because different one-class classifiers do not use the same scale of reference, it is not reasonable to compare them directly. If we combine classifiers with incomparable scales, it is likely to inadvertently weight one classifier more than the other, and this will also lead to completely unpredictable results. As a consequence, a normalization procedure is indispensable. We have observed that it is desirable to somehow convert the outlier scores into probabilities. This has two main advantages. Firstly, probability estimates provide a more systematic approach for selecting the appropriate threshold for declaring outliers, instead of an ad-hoc manner. On the other hand, the probability estimates also provide a more robust approach for developing an ensemble outlier detection framework than methods based on aggregating the relative rankings of outlier scores. Finally, the probability estimates are useful to determine the uncertainties in outlier prediction. Here we use a method based on Expectation-Maximization (EM) algorithm to transform outputs of base classifiers into probability estimates. The rationale is to treat the missing labels as hidden variables and apply the EM algorithm to maximize the expected likelihood of the data. Without loss of generality, it is assumed that the higher outlier score f_i is, the more likely x_i is an outlier. The objective is to estimate the posterior probability that x_i is an outlier given its outlier score f_i, i.e. $p_i = P(O|f_i)$. More details about this algorithm can be found in [39].

4.2. Validation set determination

We have observed that the traditional KNN rule is often used to determine validation sets for test patterns in dynamic selection methods. The DCS by local accuracy estimate assumes that it is possible to estimate the accuracy of one classifier on one test pattern through the accuracy in a particular neighborhood of this test pattern [8]. This relies on the assumption of locally constant class conditional probabilities. In traditional KNN rule where the Euclidean distance measure is used, the input space is regarded as isotropic or homogeneous implicitly. But this is often invalid and generally undesirable in a high dimensional feature space with finite samples. As a consequence, a choice of distance measure becomes crucial in determining the outcome of nearest neighbor patterns. Fig. 2 illustrate a case, where different features have different influence on different patterns. A SVDD model is trained on a 'banana-shape' data set in this case. The decision boundary provided by SVDD is shown by a carmine curve. A, B, and C are three patterns close to the decision boundary. For pattern A, dimension 2 is more relevant since a slight move along this direction may change the class label of A, while for pattern B, dimension 1 is more relevant. For pattern C, however, both dimensions are of nearly equal relevance. This implies that distance computation does not vary with equal strength or in the same proportion in all directions in the feature space.

Here we propose a locally adaptive KNN rule with the assist of algorithm SVDD. SVDD is a well-known one-class classifier with solid theoretic foundations. The solution provided by SVDD allows to determine locations where class condition probabilities are likely to be not constant, and guides the extraction of local information in such areas. Suppose $g(x) = 0$ is the decision boundary provided by SVDD. The gradient vector $\boldsymbol{n}_d = \nabla_d g$, computed at any point $\boldsymbol{x}_d$ of level curse $g(x) = 0$ gives the perpendicular direction to the decision boundary in input space

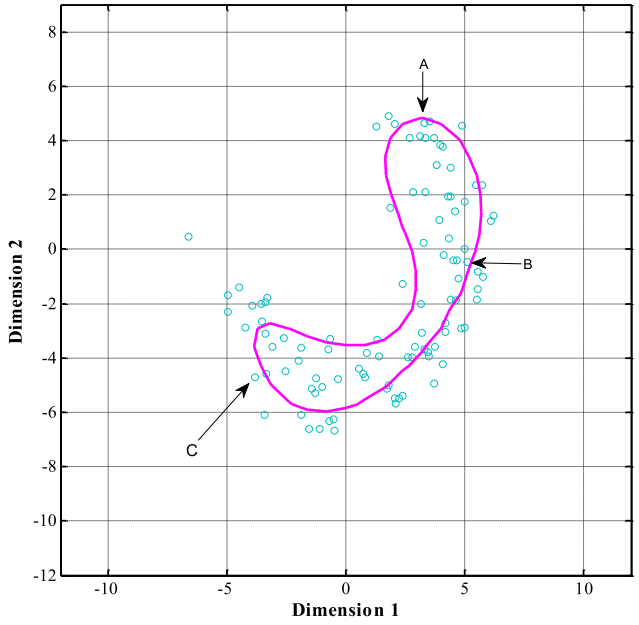

Fig. 2. Feature relevance for different patterns.

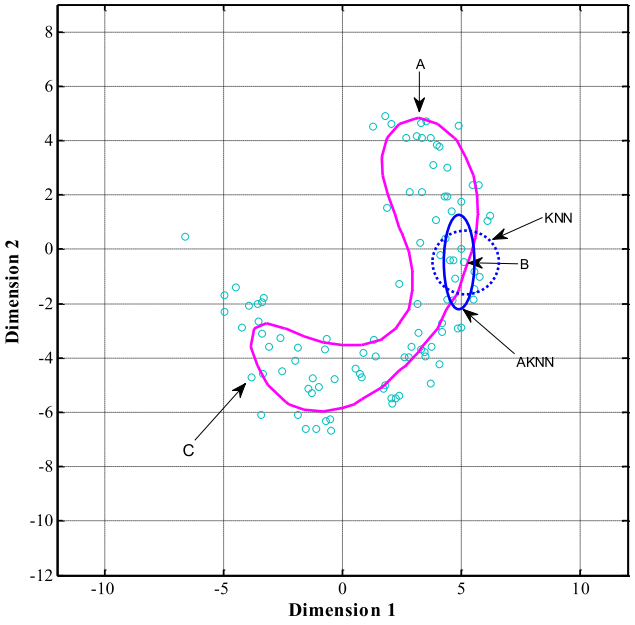

Fig. 3. Difference between KNN and adaptive KNN.

at x_d. It can be easily found that neighbor patterns of x_d projected on the orientation of $\boldsymbol{n}_d$ can be separated to the largest extent. Such information is used in our adaptive KNN rule to define a local measure of feature relevance.

Let x_t be a test pattern locating close to the boundary decided by algorithm SVDD. Let x_d be the closest point to x_t on the boundary.

$$\begin{aligned} x_d &= \arg\min_{x_i} ||x_t - x_i|| \\ s.t. \quad & g(x_i) = 0 \end{aligned} \tag{1}$$

When applying nearest neighbor rule at x_t, we desire that neighbor patterns stay close to x_t along the $\boldsymbol{n}_d$ direction, because patterns in this area are more likely to be similar with x_t in terms of class posterior probabilities. Accordingly, distances should be constricted along $\boldsymbol{n}_d$ direction and those close to it. As the direction moves farther from $\boldsymbol{n}_d$, the less discriminant the corresponding direction becomes. This implies that class labels are likely not to change along these directions. Therefore, in order to include neighbor patterns with the same class conditional probabilities as x_t in this area, distances should be elongated for these directions. As a consequence, a weighting scheme on data features seems to be reasonable. For illustration, Fig. 3 shows the difference between traditional KNN and adaptive KNN (AKNN). The used data set and SVDD are identical with those in Fig. 1. We can find that nearest neighbor patterns found by adaptive KNN have identical class posterior probabilities with pattern *B*. While the class posterior probabilities in neighborhood provided by KNN are not constant.

We desire that the above process could discover highly stretched neighborhoods along boundary directions when the test pattern locates close to the boundary. The class conditional probabilities thus tend to be constant in the modified neighborhoods. We present the procedure of the adaptive KNN rule in **Algorithm 1**.

At step 1, a SVDD model is learnt on the given training set in order to derive the decision boundary ($g(x) = 0$), with which we can find the closest point to x_t at step 2. Move from the test pattern x_t along the input feature and stop as soon as the boundary is crossed along this feature direction. Then from step 3 to step 11, we compute the weights of all features by

$$\omega_j(x_t) = \frac{exp(AR_j(x_t))}{\sum_{i=1}^{n} exp(AR_i(x_t))}, \ j = 1, \ldots, n \tag{2}$$

where parameter $A = D - B_t$ (step 7). As the distance of x_t from the boundary increases, parameter A is expected to decrease. Parameter B_t is the distance of x_t from the closest non-bounded support vector (step 7), i.e. $B_t = |x_t - x_t^{cl}|$ in which x_t^{cl} denotes the closest non-bounded support vector. Those non-bounded support vectors relate to Lagrange multipliers with $0 < \alpha_i < C$ in SVDD [24]. Parameter D is the average distance between all training patterns and their corresponding closest non-bounded support vectors (step 4–6). The term $R_j(x_t)$ is the feature relevance of feature j, locally at x_t (step 9).

$$R_j(x_t) \equiv \left|\boldsymbol{u}_j{}^T \cdot \boldsymbol{n}_d\right| = \left|n_{d,j}\right|, \ \boldsymbol{n}_d = \left(n_{d,1}, \ldots, n_{d,n}\right)^T \tag{3}$$

where $\boldsymbol{u}_j$ denote the unit vector along input feature j, for $j = 1, \ldots, n$. This exponential weighting scheme can convey stability to the method by preventing neighborhoods to extend infinitely in any direction. It can be then used as weights associated with features for weighted distance computation (step 12–14). Finally, all values of distance to training points have been sorted in ascending order and the k nearest neighbors could be singled out.

4.3. Competence estimation

Most DCS and DES methods for multi-class classification use class local accuracy (CLA) to represent competences of corresponding classifiers. In this situation, the ground truth labels of training patterns are indispensable in order to compute CLA values of different classifiers. However, this scheme is invalid in the case of outlier detection application, due to the natural absence of ground truth. Note that in one-class classification all training patterns are assumed from normal class. But some data points will be excluded in order to avoid the overfitting problem. Then we can use these points to play the role of outliers when estimating classifier competence. By averaging outputs of all classifiers, the results can be used to label neighbor patterns of any test pattern. Then we use a probabilistic model to compute competences for classifiers.

Let $V = \{X_t^1, \ldots, X_t^K\}$ denote the validation set of test pattern X_t. For any classifier C_j, the probability of correct classification of any test pattern can be estimated by (if class labels of training patterns are available)

$$\hat{p}\left(correct_j\right) = \frac{N_j}{K} \ j = 1, \ldots, L \tag{4}$$

where N_j is the number of neighbor patterns that are correctly classified by classifier C_j. As our base classifiers can output probability estimate

Algorithm 1 AKNN.

Input: Training set $S_{tr} = \{x_{tr1}, \ldots, x_{trN}\} \in \mathcal{R}^n$; test point $x_t \in \mathcal{R}^n$; k;
Output: $\boldsymbol{X}_t^{kN}$, k nearest neighbor patterns of x_t.
1. $SVDD(S_{tr}) \rightarrow g(x) = 0\%$ Train SVDD model on S_{tr} to derive the decision boundary.
2. $x_d = \arg\min_{x_i} x_t - x_i,\ s.t.\ g(x_i) = 0\%$ Find the closest point to x_t.
3. $\boldsymbol{n}_d = \nabla_d g = (n_d^1, \ldots, n_d^n)^T\%$ Compute the gradient vector at x_d.
4. **for** $i = 1 : N$
5. $\quad d_i^{cl} = |x_{tri} - x_{tri}^{cl}|$
6. **end**
7. $D = \frac{1}{N}\sum_{i=1}^{N} d_i^{cl},\ B_t = |x_t - x_t^{cl}|,\ A = D - B_t$
8. **for** $j = 1 : n$
9. $\quad R_j(x_t) = |\boldsymbol{u}_j^T \cdot \boldsymbol{n}_d|\%$ Compute the feature relevance.
10. $\quad \omega_j(x_t) = \frac{exp(AR_j(x_t))}{\sum_{i=1}^{n} exp(AR_i(x_t))}\%$ Compute the features weight.
11. **end**
12. **for** $i = 1 : N$
13. $\quad D_t^i = D(x_t - x_{tri}) = \sqrt{\sum_{j=1}^{n} \omega_j(x_t)(x_t^j - x_{tri}^j)^2}\%$ Compute the weighted distance.
14. **end**
15. $[D, index] = sort(D_t^1, \ldots, D_t^N)\%$Arrange all values of D_t^i in ascending order.
16. $\boldsymbol{X}_t^{kN} = \{x_{trindex[1]}, \ldots, x_{trindex[k]}\}\%$Output the k nearest neighbors of x_t

for each input vector, we can use this information to reformulate Eq. (5). Assume that a neighbor pattern $X_t^i \in \omega_l,\ l \in \{O, M\}$ (O, M denotes the outlier and normal class, respectively), then $P^j(\omega_l|X_t^i)$ provided by classifier C_j can be deemed its measure of competence for X_t^i. As a result, the competence of classifier C_j on pattern X_t can be derived by averaging competences on all neighbor patterns.

$$\hat{p}(correct_j) = \frac{1}{K}\sum_{i=1}^{K} P^j(\omega_l^i|X_t^i) \tag{5}$$

where ω_l^i is the label of X_t^i. Then a weight can be assigned to each neighbor pattern in order to reduce the uncertainty in the definition of the neighbor size.

$$\hat{p}(correct_j) = \frac{\sum_{i=1}^{K} P^j(\omega_l|X_t^i)W_i}{\sum_{i=1}^{K} W_i} \tag{6}$$

where $W_i = 1/d_i$, and d_i is the distance from pattern X_t^i to X_t (defined in Section 3.2).

As outputs of base classifiers have been transformed into posterior probability estimates, we can exploit this information to measure classifier competence.

$$\hat{p}(correct_j|C_j(X_t) = \omega_l) = \hat{p}(X_t \in \omega_l|C_j(X_t) = \omega_l) = \frac{N_{ll}}{\sum_{j\in\{M,O\}} N_{jl}} \tag{7}$$

where N_{ll} is the number of neighbor patterns that have been correctly classified by C_j to class ω_l, and $\sum_{j\in\{M,O\}} N_{jl}$ is the total number of neighbor patterns that have been classified to class ω_l by classifiers C_j. Then we exploit the posterior probabilities of neighbor patterns to reformulate Eq. (8) according to the Bayesian theory.

$$\hat{p}(X_t \in \omega_l|C_j(X_t) = \omega_l) = \frac{\hat{p}(C_j(X_t) = \omega_l|X_t \in \omega_l)\hat{p}(\omega_l)}{\sum_{i\in\{M,O\}} \hat{p}(C_j(X_t) = \omega_l|X_t \in \omega_i)\hat{p}(\omega_i)} \tag{8}$$

The term $\hat{p}(C_j(X_t) = \omega_l|X_t \in \omega_l)$ indicates the probability that patterns belonging to class ω_l are correctly classified. It can be estimated as

$$\hat{p}(C_j(X_t) = \omega_l|X_t \in \omega_l) = \frac{1}{\sum_{j\in\{M,O\}} N_{jl}} \sum_{X_t^i \in \omega_l} P^j(\omega_l|X_t^i) \tag{9}$$

The term $\hat{p}(\omega_l)$ denotes the prior probability of class ω_l, which can be estimated as

$$\hat{p}(\omega_l) = \frac{\sum_{j=1}^{L} N_{lj}}{K} \tag{10}$$

Then a weight is also assigned to each neighbor pattern to reduce the uncertainty triggered by the neighbor size. Finally, competence of classifier C_j on test pattern X_t can be derived by substituting Eqs. (10) and (11) into (9).

$$\hat{p}(correct_j|C_j(X_t) = \omega_l) = \frac{\sum_{X_t^i \in \omega_l} P^j(\omega_l|X_t^i)W_i}{\sum_{i=1}^{K} P^j(\omega_l|X_t^i)W_i} \tag{11}$$

4.4. Decision making

In DCS, the most competent classifier will be selected to classify the corresponding test pattern. However, a critical point in DCS is that the choice of one individual classifier over the rest will depend on how much we trust the estimate of the generalization of this classifier. In contrast to DCS, DES can distribute the risk of the over-generalization by choosing a group of classifiers instead of one individual classifier for a test pattern. In order to achieve a trade-off between DCS and DES, a switching scheme between these two selection mechanisms seems to be necessary. In [40], a threshold is set on classifier competence. If differences between one classifier and all the other classifiers are larger than this threshold, then this classifier will be used to classify this test pattern. Otherwise, a classifier will be randomly selected to classify this test pattern. In [41], a threshold is also used to determine whether the best classifier exists or not. The difference is that a majority-voting rule is used instead of random selection if the best classifier does not exist. Note that the determination of the value of threshold is very subjective and has great influence on the final result. In [42], statistical inference is used to construct the switching mechanism. In this mechanism, a parametric statistical test, F-test is used to investigate whether the best classifier exists or not.

It is observed that non-parametric tests are safer than parametric tests, since they do not assume normal distributions or homogeneity of variance. Meanwhile, they can be applied to extensive measures, such as classification accuracies and error ratios, even model sizes and computation times [43]. Here we use two non-parametric statistical tests to construct the switching scheme. The first one is Friedman test that is used to check if the competence measures of all classifiers are equivalent. Then a post-hoc test, Nemenyi test is used to check if competence measures of a pair of classifiers are equivalent. More details about these two tests can be found in [22]. It is noteworthy that each data point in the validation set is regarded as a data set when calculating the statistic. Our switching scheme can be concluded as followings.

Step 1. Use Friedman test on all classifiers. If the null hypothesis is not rejected, go to step 2; otherwise, go to step 3.
Step 2. Use the averaging value of all classifier outputs as the final result.

Step 3. Use Nemenyi test on the best classifier and the second best classifier. If the null hypothesis is rejected, go to step 4; otherwise, go to step 5.

Step 4. Use this best classifier to classify this test pattern.

Step 5. Remove the best classifier from the original classifier pool and put it in a new pool. Implement this process until the null hypothesis could be rejected. Then use the averaging output of classifiers in the new pool as the final result.

At step 1, if the null hypothesis of Friedman test is not rejected, then none classifier performs significantly better than other classifiers. In this case, we average outputs of all classifiers and use the result as the final output (step 2). Otherwise, we need to discover classifiers of higher performance statistically. At step 3, we use Nemenyi test on the best two classifiers. If the null hypothesis is rejected, then we use this best classifier to classify this test pattern (step 4). Otherwise, we remove this best classifier to a new pool, and use Nemenyi test on the rest classifiers. When the implementation of Nemenyi test is over, we use the averaging output of classifiers in the new pool as the final result (step 5).

4.5. Theoretic proof of dynamic outlier detection

Suppose that we have trained L one-class classifiers $(C_1, \dots, C_L)$ in Section 3.1 as base learners of our ensemble model. These L classifiers have divided the feature space $\mathfrak{R}^n$ into K small regions $(R_1, \dots, R_K)$. These regions are not associated with specific classes, nor do they need to be of a certain shape or size. For any test pattern, we should decide which classifier from the pool will take charge of it. This is equal to decide which classifier we should nominate for each small region R_j. Note that the number of classifiers L is not necessarily equal to the number of small regions K, because some classifier might never be nominated or some classifiers might be nominated for more than one region. Let C^B be the best classifier (often referred to as single best classifier) over the whole feature space from the pool. Let $p(C_i|R_j)$ denote the probability of correct classification by C_i in region R_j. According to the Bayes theory, the objective of an outlier detection algorithm or one-class classifier is to maximize the probability of correct classification that can be computed by

$$p(correct) = \sum_{j=1}^{K} p(R_j)p(C_{i(j)}|R_j) \tag{12}$$

where $C_{i(j)}$ denotes the classifier responsible for region R_j, and $p(R_j)$ is the prior probability of region R_j, which can be represented by the fraction of patterns falling into the region R_j.

Then we assume that we have selected the most competent classifier, i.e.

$$p(C_{i(j)}|R_j) \geq p(C_i|R_j),\ \forall i = 1, \dots, L \tag{13}$$

From Eqs. (13) to (14), we have the following conclusion.

$$p(correct) \geq \sum_{j=1}^{K} p(R_j)p(C^B|R_j) \tag{14}$$

This equation has proved that one DCS method should outperform the single best model, under the condition that the most competent classifier can be selected for each small region. It is noteworthy that this conclusion holds regardless of the way of feature space partition.

Then we prove that under some hypotheses, we can achieve the optimal Bayesian result by the dynamic selection of non-optimal classifiers. Similar proof have been provided in DCS for multi-classification problem. Here we extend if to outlier detection. In outlier detection or one-class classification, an optimal Bayesian result can be obtained by maximizing the following probability

$$\sum_{i\in\{M,O\}} \int p(X|\omega_i)p(\omega_i)dX \tag{15}$$

where M and O represent the normal and outlier class, respectively. R_i denotes the decision region of class i. The classifier that maximize the above probability is referred to as optimal Bayes classifier. Let the corresponding decision region and decision boundary be $R_i^B,\ i \in \{M, O\}$ and $f^B(X) = 0$, respectively.

Then we also assume that we have trained L one-class classifiers $(C_1, \dots C_j \dots, C_L)$ as base learners of our ensemble model. Let $R_i^j,\ i \in \{M, O\},\ j = 1, \dots, L$ denote the decision region of class i of classifier C_j. Let $f^j(X) = 0$ denote the decision boundary of classifier C_j. Here we define two decision regions associated with classifier C_j: ① $R_{i+}^j = R_i^j \cap R_i^B,\ i \in \{M, O\}$ represents the optimal decision region for class i; ② $R_{i-}^j = R_i^j - R_{i+}^j$ represents the non-optimal decision region for class i. We can thus have $R_i^j = R_{i+}^j \cup R_{i-}^j$. Then we also assume that L classifiers have divided the feature space $\mathfrak{R}^n$ into K small parts $(P_1, \dots P_k \dots, P_K)$. These small parts are not associated with specific classes, nor do they need to be of a certain shape or size. Let $f_k(X) = 0$ denote the decision boundary of part P_k, then

$$\sum_j \sum_i \beta_i^j(X) f^j(X) = 0,\ \forall X :\ f_k(X) = 0 \tag{16}$$

$$f^j(X) \neq 0,\ \forall X :\ f_k(X) > 0,\ \forall i, j \tag{17}$$

where represent binary functions like indicated function, $\beta_i^j(X) = \{0, 1\}$, $\sum_j \sum_i \beta_i^j(X) = 1$. This implies that the boundary of part P_k is regarded as a sum of pieces of decision boundaries of all base classifiers.

Assume that the following two conditions are satisfied.

Hypothesis 1: $R_i^B = \cup_{j=1}^{L} R_{i+}^j,\ i \in \{M, O\}$

Hypothesis 2: $\forall X :\ f^B(X) = 0 \rightarrow \sum_{j=1}^{L} \alpha_j(X) f^j(X) = 0$

Firstly, we can know that

$$\forall i, j\ R_i^j = \bigcup_{k\in A_i^j} P_k\ A_i^j \subset \{1, \dots, K\} \tag{18}$$

where A_i^j is a subset of $\{1, \dots, K\}$, it contains subscripts of small parts P_k, that constitute decision region R_i^j. According to hypothesis 2, we know that the optimal decision boundaries coincide piecewise with base classifier boundaries. We can hence extend Eq. (19) to the optimal decision region by:

$$\forall i, j\ R_{i+}^j = \bigcup_{k\in A_{i+}^j} P_k\ A_{i+}^j \subset \{1, \dots, K\} \tag{19}$$

This conclusion indicates that small parts $P_k,\ k \in A_{i+}^j$ have constituted the optimal decision region R_{i+}^j. Therefore, we can achieve optimal Bayesian results for any pattern in these small parts. Then, we need to prove that all small parts $P_k,\ k \in \{1, \dots, K\}$ will be contained in optimal decision regions. According to hypothesis 1 and Eq. (20), we have

$$\forall i\ R_i^B = \bigcup_{j=1}^{k} R_{i+}^j = \bigcup_{k\in A_{i+}} P_k,\ i \in \{M, O\} \tag{20}$$

where $A_{i+} = \bigcup_{j=1}^{L} A_{i+}^j$. Because the decision regions of the optimal Bayes classifier can cover the entire feature space, we can obtain the following conclusion

$$A_{M+} \cup A_{O+} = \{1, \dots, k, \dots K\} \tag{21}$$

Therefore

$$\bigcup_{i\in\{M,O\}} R_i^B = \bigcup_{k=1}^{K} P_k = \mathfrak{R}^n \tag{22}$$

This conclusion is equivalent to the following one

$$\forall k\ \exists i, j :\ P_k \subseteq R_{i+}^j \tag{23}$$

For any test pattern, there exists a non-optimal classifier that can achieve the optimal Bayesian result. However, the two proposed hypotheses must hold. In addition, we must have approach to selecting this non-optimal classifier from the pool.

Table 1
Description of data sets used in experiments.

No.	Name	Examples	Features	Classes
1	Balance	526	4	3
2	Bank marketing	45,211	17	2
3	Breast-cancer	286	9	2
4	Breast-Wisconsin	699	9	2
5	Car evaluation	1728	6	4
6	Chess	3196	36	2
7	Diabetes	768	8	2
8	Glass identification	214	10	6
9	Heart disease	303	14	4
10	Iris	150	4	3
11	Ionosphere	351	34	2
12	Liver	345	6	2
13	Nursery	12,960	8	5
14	Page blocks	5473	10	5
15	Pima	768	8	2
16	Sonar	208	60	2
17	Voting records	435	16	2
18	Wine quality	4898	12	11
19	Yeast	1484	8	10
20	Zoo	101	16	7

Table 2
Confusion matrix.

		Actual Label	
		Target Class	*Outlier Class*
Predicted Label	*Target Class*	True Negative (TN)	False Negative (FN)
	Outlier Class	False Positive (FP)	True Positive (TP)

Table 3
Ten one-class classifiers used in the second series of experiments.

Label	One-class Classifier	dd_tools
1	Mixture of Gaussians data description	mog_dd
2	Parzen density estimator	parzen_dd
3	Auto-encoder neural network	autoenc_dd
4	K-means data description	kmeans_dd
5	K-center data description	kcenter_dd
6	Principal component analysis data description	pca_dd
7	Self-organizing map data description	som_dd
8	Minimum spanning tree data description	mst_dd
9	K-nearest neighbor data description	knndd
10	Support vector data description	svdd

5. Experimental investigations

5.1. Data sets

20 data sets from UCI Repository are used in this experiment. A simple description is shown in **Table 1**. These data sets are originally used for multi-class classification. In order to apply them to validate outlier detection models, some adaptations are necessary. We regard the majority class as the target class, and sample 70% target points to constitute the training set. Then the remaining target points, as well as data points from all the other classes are used to constitute the test set. In order to reduce the uncertainty triggered by random sampling, the above process is repeated five times, and the averaging results will be listed.

5.2. Competitors

In order to put the experimental results into context, we compare our method with the following competitors.

- *SB*: single best. We take the one-class classifier from the pool that performs best. Note that this classifier can only be known after obtaining results of all base classifiers.
- *AVR*: averaging outputs of all base classifiers.
- *PRO*: product of outputs of all base classifiers.
- *OCClustE*: a clustering based ensemble model proposed in [14].
- *DCS-EM*: a dynamic outlier detection method proposed in [21].
- *DCS-P*: a dynamic outlier detection method proposed in [22].
- *DEOD-K*: this is another version of our method. The difference from our method is that DEOD-K use traditional KNN rule to determine the validation set.
- *Oracle*: The oracle works as follows: assign the correct class label to the test point if and only if at least one individual OC classifier produces the correct class label.

The comparison with SB is to investigate if the dynamic ensemble model can outperform single models. The aim of comparing our method with AVR, PRO, and OCClustE is to check whether accuracy improvement can be achieved over static ensemble models. The comparisons with DCS-EM and DCS-P are used to verify the effectiveness of the procedures of validation set determination and competence estimation. The objective of the comparison with DEOD-K is to investigate whether the proposed adaptive KNN rule can contribute more than traditional KNN to our dynamic outlier ensemble. Finally, the result of oracle can show the gap of our method from the ideal state.

5.3. Metrics

With regard to the criterion that evaluates detection results, traditional metrics used in multi-class classification like accuracy and error rate are not used here. Instead, we use a criterion called ROC (receiver operating characteristic) curve. The ROC curve represents the trade-off between the true negative rate (TPR) and the false negative rate (FPR). Here patterns from outlier class and target class are denoted as positive and negative, respectively. Following this convention along with the confusion matrix given in **Table 2**, index TNR and FNR can be formulated as

$$TNR = \frac{TN}{TN + FP}, \; FNR = \frac{FN}{FN + TP} \tag{24}$$

Note that these two indexes are two mutually exclusive quantities, ROC curve can hence evaluate the general performance of classifiers, rather than performance at one working point. In general, the area under the curve (AUC) is used to measure the performance of outlier detection algorithms. The AUC of specific algorithm is defined as the surface area under its ROC curve. It can be easily found that for an outlier detection task, the AUC of a perfect algorithm is one, implying that all outliers have been identified along with none misclassified normal data.

In order to present a detailed comparison among a group of machine learning algorithms, one must use statistical tests to prove that the reported differences among classifiers are significant [44]. Here we use two statistical tests mentioned in Section 4.4 to provide statistical comparisons. Specifically, the Friedman ranking test is used to check if the assigned ranks are significantly different from assigning to each classifier an average rank. While the Nemenyi ranking test is used to check if two ranks are different significantly, once the null hypothesis of Friedman has been rejected. The significance level is fixed at 0.05 for both tests.

5.4. Result and analysis

Two series of experiments are carried out in this paper. In the first series, we use homogeneous ensemble models. Algorithm SVDD is used as the base classifier. The error of target is set 0.1. In the second series, we use heterogeneous ensemble models and 10 one-class classifiers are used as base learners. A simple description of these classifiers is shown in **Table 3**. Then we show results of the first and second series of experiment with respect to AUC value in **Tables 4** and **5**, respectively. The results of Friedman test and Nemenyi test are listed in **Tables 6** and **7**, respectively.

Table 4
Results of homogeneous ensemble models. Subscripts indicate corresponding ranks (except Oracle). Best results are in bold for each data set.

Dataset	SB	ARV	PRO	OCClustE	DCS-EM	DCS-P	DEOD-K	Our	Ora
1	85.73_7	85.28_8	85.92_6	86.74_4	86.32_5	87.35_2	87.06_3	$\mathbf{87.71_1}$	99.35
2	93.04_7	94.42_4	93.73_6	93.01_8	95.16_2	94.27_5	94.72_3	$\mathbf{95.37_1}$	100
3	64.10_8	65.24_6	64.78_7	66.02_5	67.04_2	66.92_3	66.68_4	$\mathbf{67.31_1}$	92.46
4	89.05_7	89.74_5	88.43_8	89.15_6	91.02_2	$\mathbf{91.46_1}$	90.77_3	90.77_3	100
5	74.61_7	75.52_5	73.82_8	76.19_2	$\mathbf{76.55_1}$	76.19_2	75.09_6	75.87_4	94.07
6	93.14_6	92.96_8	93.10_7	93.84_4	94.02_3	94.53_2	93.21_5	$\mathbf{94.68_1}$	100
7	63.25_5	61.96_8	62.04_7	62.43_6	64.29_3	64.87_2	64.00_4	$\mathbf{65.17_1}$	91.97
8	80.17_8	80.93_6	80.56_7	81.67_5	83.02_2	$\mathbf{84.95_1}$	82.41_4	83.00_3	98.22
9	86.26_7	87.10_6	87.73_5	85.59_8	87.81_4	88.26_3	89.10_2	$\mathbf{89.47_1}$	99.89
10	89.94_8	90.43_6	90.22_7	91.09_5	91.89_2	$\mathbf{92.29_1}$	91.65_3	91.65_3	97.52
11	83.57_7	83.59_6	83.50_8	84.04_5	83.41_4	84.26_2	84.02_3	$\mathbf{84.53_1}$	94.79
12	67.81_5	$\mathbf{69.78_1}$	67.25_8	69.06_2	68.03_4	68.42_3	67.33_6	67.33_6	97.83
13	87.17_8	88.01_6	88.22_5	87.45_7	89.16_4	89.91_3	90.90_2	$\mathbf{91.46_1}$	100
14	93.29_6	92.16_7	92.00_8	93.77_5	95.02_2	$\mathbf{95.43_1}$	94.21_3	94.21_3	100
15	71.02_7	71.09_6	70.28_8	71.47_5	71.62_3	71.14_4	$\mathbf{72.36_1}$	$\mathbf{72.36_1}$	99.14
16	90.59_6	90.38_7	90.24_8	91.27_4	91.32_3	91.78_2	90.84_5	$\mathbf{92.66_1}$	99.92
17	89.93_5	88.94_7	89.09_6	88.24_8	91.17_2	$\mathbf{91.29_1}$	90.50_4	90.76_3	100
18	79.13_6	77.29_8	79.50_5	77.99_7	$\mathbf{81.79_1}$	$\mathbf{81.79_1}$	79.78_4	81.04_3	98.27
19	78.29_8	80.01_5	79.13_7	79.40_6	$\mathbf{81.98_1}$	80.92_3	80.54_4	81.27_2	97.76
20	82.77_8	83.82_5	83.82_5	82.94_7	84.28_3	84.11_4	84.90_2	$\mathbf{85.81_1}$	99.91

Table 5
Results of heterogeneous ensemble models. Subscripts indicate corresponding ranks (except Oracle). Best results are in bold for each data set.

Dataset	SB	ARV	PRO	OCClustE	DCS-EM	DCS-P	DEOD-K	Our	Ora
1	85.90_6	84.22_8	85.68_7	86.14_5	86.75_4	87.82_2	87.46_3	$\mathbf{88.63_1}$	99.35
2	94.08_5	92.69_8	93.51_6	93.12_7	95.33_2	94.36_4	94.85_3	$\mathbf{95.57_1}$	100
3	68.14_4	67.02_6	66.32_7	65.69_8	68.54_3	$\mathbf{69.65_1}$	67.62_5	68.97_2	92.46
4	91.14_3	88.72_7	88.49_8	89.19_6	$\mathbf{92.38_1}$	91.97_2	90.68_4	90.68_4	100
5	$\mathbf{76.23_1}$	75.52_5	73.82_8	76.19_2	$\mathbf{76.55_1}$	76.19_2	75.09_6	75.87_4	94.07
6	94.14_6	93.16_8	93.60_7	94.84_4	95.02_3	95.73_2	94.21_5	$\mathbf{96.17_1}$	100
7	63.25_5	61.96_8	62.04_7	62.43_6	64.29_3	64.87_2	64.00_4	$\mathbf{65.17_1}$	91.97
8	83.47_5	81.93_8	82.56_7	83.17_6	85.02_2	$\mathbf{85.75_1}$	83.91_4	84.34_3	98.22
9	90.73_3	87.99_8	88.73_7	89.59_6	89.89_5	90.22_4	91.10_2	$\mathbf{91.26_1}$	99.89
10	90.94_5	90.13_6	89.22_8	89.49_7	91.80_3	92.17_2	91.35_4	$\mathbf{92.69_1}$	97.52
11	84.27_6	83.79_7	83.54_8	85.04_4	84.71_5	85.26_3	85.36_2	$\mathbf{85.93_1}$	94.79
12	$\mathbf{69.14_1}$	68.85_2	67.21_3	66.06_7	65.46_8	67.42_4	66.78_6	67.03_5	97.83
13	88.17_7	88.01_8	89.22_5	88.45_6	90.16_4	91.28_2	90.84_3	$\mathbf{91.90_1}$	100
14	93.24_6	92.86_7	92.05_8	93.77_5	94.02_4	$\mathbf{94.86_1}$	94.35_2	94.35_2	100
15	71.41_5	70.09_8	70.78_7	71.17_6	72.62_3	71.65_4	72.88_2	$\mathbf{73.07_1}$	99.14
16	90.32_5	90.08_7	89.94_8	90.57_4	91.02_3	91.28_2	90.32_5	$\mathbf{92.06_1}$	99.92
17	91.86_3	88.16_8	89.49_6	88.90_7	90.67_5	91.31_4	92.04_2	$\mathbf{92.76_1}$	100
18	79.23_2	77.07_7	77.85_5	$\mathbf{79.51_1}$	76.79_8	77.29_6	78.48_4	78.90_3	98.27
19	81.29_5	79.41_8	79.83_7	80.40_6	81.48_4	$\mathbf{82.62_1}$	81.97_2	81.97_2	97.76
20	83.79_6	83.42_7	83.00_8	83.94_5	84.69_4	84.91_3	85.81_2	$\mathbf{86.66_1}$	99.91

Table 6
Results of Friedman tests (8 comparative methods on 20 data sets).

Experiment	Critical Value	F_F **Value**	Null Hypothesis
Series 1	2.08	27.28	Reject
Series 2	2.08	18.29	Reject

5.4.1. Homogeneous ensemble

Our method outperforms all the other competitors on 11 out of 20 data sets. But its gap from the oracle is still very large. As oracle is an abstraction method rather than a real model, we exclude it from the comparison. In addition, we can find that dynamic ensemble methods perform better than single model and static ensemble models generally. The advantage of static ensemble models over single model cannot be found from this experiment.

Now we compare our method with single best model. For all 20 data sets, our dynamic ensemble model can outperform SB. Statistical results also show that a significant difference exists between these two methods. This result indicates that, in homogeneous ensemble paradigm, none single model has strong performance on the whole feature space. For small local regions, our dynamic selection scheme can accurately esti-

Table 7
Results of Nemenyi tests (8 comparative methods on 20 data sets).

Experiment	CDValue	Averaging Rank							
		SB	AVR	PRO	OCClutE	DCS-EM	DCS-P	DEOD-K	Our
Series 1	2.35	6.80	6.00	6.80	5.40	2.65	2.30	3.55	2.05
Series 2	2.35	4.45	7.01	6.85	5.40	3.75	2.80	3.50	1.85

mate the classifier competence, and select more appropriate one(s) for corresponding test patterns. But the gap from the oracle can, at the same time, show that selecting the optimal Bayes classifier is still very challenging, since the two hypotheses (Section 4.5) can hardly be satisfied in practice. Then we compare our method with three static ensemble models. For most data sets, our method can outperform these three models, and the statistical result also indicates the advantage of our method. In static ensemble models, outputs of all base classifiers are aggregated by corresponding functions. Although static ensemble can alleviate the model selection problem, and avoid weak results. Its superiority over single best models is not absolute, which can also be found in other researches. By dynamic selection for each test pattern, weak classifiers would be excluded, and only the most competent one(s) can be retained. But we should always note that the success of a dynamic model must rely on an accurate estimation of classifier competence, which also related much to the corresponding validation set. When comparing our method with other two dynamic models, we can still find the superiority of our method. But this is not supported by the statistical results. This result indicates that each procedure of a dynamic selection model will have great influence on the final performance. This can also be found when we compare our method with DEOD-K, where the traditional KNN rule is used. Our proposed adaptive KNN rule can found nearest neighbor patterns that have identical class posterior probabilities with the test pattern. Then the classifier competence estimation should be more accurate.

5.4.2. Heterogeneous ensemble

In this experiment, 10 one-class classifiers are used as base learners. As shown in Section 3.1, different classifiers may have corresponding competent regions. This is also an inspiration for dynamic classifier selection. From Table 5 we can find that our method has outperformed its competitors on 12 out of 20 data sets. And being similar with the result in homogeneous ensembles, dynamic outlier ensembles usually perform better than single model and static outlier ensembles. Meanwhile, we have found an interesting point in this experiment that model SB gets much improvement. This can be shown by averaging ranks of all methods (Table 7). For data set 5 and 12, SB model has obtained the best performance. This result indicates that, for some data sets, base classifiers may behave strong on the whole feature space. In such cases, static ensembles, even dynamic ensembles can hardly achieve much improvement. Another reason for this situation may be the mismatch between validation set and test set. If the distribution of test patterns has significant difference from the distribution of validation patterns, then competence measures estimated on validation set can hardly be generalized to the test set. This can also be regarded as the overfitting of dynamic models. Therefore, in this case, dynamic ensemble selection may be effective.

The statistical comparisons with two tests are shown in Tables 6 and 7. We should note that the objective of Friedman test is to investigate whether one method can be singled out from its competitors. The null hypothesis is thus that the performance of all competitors are equivalent. From Table 6 we can clearly see that this null-hypothesis has been rejected, indicating the existence of statistically stronger method(s). Then we can proceed with the post-hoc test, i.e. Nemenyi test to statistically compare pairwise methods. The performance of two methods is significantly different if the corresponding averaging ranks differ by at least the critical difference (CD) [43]. As shown in Table 7, the CD values for both experiments are 2.35. As a result, our method has statistical superiority over SB, AVE, PRO, and OCClutE for both homogeneous and heterogeneous ensembles. We have to admit that the superiority over other dynamic competitors have not been presented by this result.

5.5. Further discussions

In previous experiments, we have compared our dynamic outlier ensemble with several competitors, including single best model, static outlier ensembles, and other dynamic outlier ensembles. For most data sets, our detector have obtained better performance than these competitors. Then statistical comparisons are implemented with two statistical tests in order to investigate whether the superiority is significant. Results have shown that most dynamic outlier ensembles could outperform single model and static outlier ensembles significantly. This experimental conclusion seems very clear and has been proved by other dynamic outlier ensembles. Then we investigate *what will happen if we remove traditional static ensembles (SB, AVR, PRO) from the comparison,* since the inclusion of these detectors in statistical comparison may lead to biased result.

According to the results listed in Tables 4 and 5, the comparative results between OCClutE, DCS-EM, DCS-P, DEOD-K and our method with respect to numeric criterion have not changed. The result of Friedman test (95% significance level) is shown in **Table 8**. We can see that the null hypothesis of equal performance has been rejected by the results with respect to averaging ranks. This implies the significant difference among these comparative methods. Then Nemenyi test (95% significance level) is implemented and the result is listed in **Table 9**. The critical difference values in both series of experiments are 1.36, which indicates that significant difference exists only if the absolute deviation of averaging ranks between two methods exceeds 1.36. For homogeneous ensembles (series 1), the averaging rank of our method is 1.95. Therefore, any method whose averaging rank exceeds 3.41 would be outperformed by our method significantly. We have noted that only the averaging rank of OCClutE exceeds 3.41. In addition, the averaging rank of DEOD-K is 3.40, which is much closed to 3.41. This result has subtle difference from previous experiment in which traditional static ensembles are included. Specifically, the efficiency of our method has been verified more clearly in this experiment.

For heterogeneous ensembles (series 2), the averaging rank of our method is 1.60. Consequently, any method with averaging rank exceeding 2.96 will be significantly outperformed by our method. According to the results in Table 9, we can easily see that only DCS-P has the averaging rank being smaller than 2.96. Compared with the result on homogeneous ensembles, the statistical superiority of our method is more explicit. Similarly, when compared with the result in previous experiment, the performance of our method can be highlighted again. As a result, we can conclude that the statistical efficiency of our method becomes more clearly when we remove some traditional static outlier ensembles.

Finally, from the result in this experiment, we can find that our method has outperformed DEOD-K significantly on both homogeneous and heterogeneous ensembles. Recall that method DEOD-E is a modified version of our method, in which traditional KNN is used to determine validation sets. As a result, we can claim that the proposed adaptive KNN rule for discovering more reasonable validation sets has resulted in significant improvement.

Table 8
Results of Friedman tests (5 comparative methods on 20 data sets).

Experiment	Critical Value	F_F Value	Null Hypothesis
Series 1	2.49	5.40	Reject
Series 2	2.49	10.55	Reject

Table 9
Results of Nemenyi tests (5 comparative methods on 20 data sets).

Experiment	CD Value	Averaging Rank				
		OCClutE	DCS-EM	DCS-P	DEOD-K	Our
Series 1	1.36	4.50	2.60	2.20	3.40	1.95
Series 2	1.36	4.45	3.35	2.25	3.15	1.60

6. Conclusions

In this paper, we present a dynamic outlier detection method on the basis of one-class classification. In order to determine an effective validation set for each test pattern, we propose an adaptive KNN rule based on algorithm SVDD. By exploiting the posterior probabilities of classifiers, we estimate their competences and make the final decision through a switching scheme. In order to validate the effectiveness of our method, we carry out experiments on 20 benchmark data sets and compare with several competitors. Results have shown that in most cases, a dynamic selection scheme will perform better than single model and static ensemble models.

By comparing our method with oracle, however, significant difference still exists. Therefore, in future, we should study more regarding validation set determination and competence estimation, in order to reduce the gap from oracle.

Declaration of Competing Interest

None.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.inffus.2020.05.001.

References

[1] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection:a survey, ACM Comput. Surv. 41 (3) (2009) 1–58.
[2] G. Giacinto, R. Perdisci, M.D. Rio, F. Roli, Intrusion detection in computer networks by a modular ensemble of one-class classifiers, Inf. Fusion 9 (1) (2008) 69–82.
[3] R.M. Cruz, R. Sabourin, G.D. Cavalcanti, Dynamic classifier selection: recent advances and perspectives, Inf. Fusion 41 (2018) 195–216.
[4] T. Woloszynski, M. Kurzynski, P. Podsiadlo, G.W. Stachowiak, A measure of competence based on random classification for dynamic ensemble selection, Inf. Fusion 13 (3) (2012) 207–213.
[5] B. Krawczyk, L.L. Minku, J. Gama, J. Stefanowski, M. Wozniak, Ensemble learning for data stream analysis: a survey, Inf. Fusion 37 (C) (2017) 132–156.
[6] R.M.O. Cruz, R. Sabourin, G.D.C. Cavalcanti, META-DES.Oracle: Meta-learning and feature selection for dynamic ensemble selection, Inf. Fusion 38 (2017) 84–103.
[7] T. Woloszynski, M. Kurzynski, A probabilistic model of classifier competence for dynamic ensemble selection, Pattern Recognit. 44 (10–11) (2011) 2656–2668.
[8] K. Woods, W.P.K. Jr, K. Bowyer, Combination of multiple classifiers using local accuracy estimates, IEEE Trans. Pattern Anal. Mach. Intell. 19 (4) (1997) 405–410.
[9] P.C. Smits, Multiple classifier systems for supervised remote sensing image classification based on dynamic classifier selection, IEEE Trans. Geosci. Remote Sens. 40 (4) (2002) 801–813.
[10] D.M.J. Tax, M.V. Breukelen, R.P.W. Duin, J. Kittler, Combining multiple classifiers by averaging or by multiplying? Pattern Recognit. 33 (9) (2000) 1475–1485.
[11] P. Juszczak, R.P.W. Duin, Combining one-class classifiers to classify missing data,", in: Proceedings of the International Workshop on Multiple Classifier Systems, Cagliari, Italy, 2004, pp. 92–101.
[12] B. Wang, Z. Mao, One-class classifiers ensemble based anomaly detection scheme for process control systems, Trans. Inst. Meas. Control 40 (12) (2017) 3466–3476.
[13] C. Lai, D.M.J. Tax, R.P.W. Duin, E. Pękalska, P. Paclík, On combining one-class classifiers for image database retrieval,", in: Proceedings of the *International Workshop on Multiple Classifier Systems*, 2002, pp. 212–221.
[14] B. Krawczyk, M. Woźniak, B. Cyganek, Clustering-based ensembles for one-class classification, Inf. Sci. 264 (6) (2014) 182–195.
[15] B. Wang, Z. Mao, K. Huang, Detecting outliers in complex nonlinear systems controlled by predictive control strategy, Chaos Solitons Fractals 103 (2017) 588–595.
[16] L.I. Kuncheva, C.J. Whitaker, Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy, Mach. Learn. 51 (2) (2003) 181–207.
[17] B. Krawczyk, One-class classifier ensemble pruning and weighting with firefly algorithm, Neurocomputing 150 (150) (2015) 490–500.
[18] E. Parhizkar, M. Abadi, BeeOWA: A novel approach based on ABC algorithm and induced OWA operators for constructing one-class classifier ensembles, Neurocomputing 166 (2015) 367–381.
[19] H. Ykhlef, D. Bouchaffra, An efficient ensemble pruning approach based on simple coalitional games, Inf. Fusion 34 (2017) 28–42.
[20] D.V.R. Oliveira, G.D.C. Cavalcanti, R. Sabourin, Online pruning of base classifiers for dynamic ensemble selection, Pattern Recognit. 72 (2017) 44–58.
[21] B. Krawczyk, Dynamic classifier selection for one-class classification, Knowl. Based Syst. 107 (2016) 43–53.
[22] B. Wang, Z. Mao, Outlier detection based on a dynamic ensemble model: applied to process monitoring, Inf. Fusion 51 (2019) 244–258.
[23] R.P.W. Duin, On the choice of smoothing parameters for parzen estimators of probability density functions, IEEE Trans. Comput. C-25 (11) (1976) 1175–1179.
[24] D.M.J. Tax, R.P.W. Duin, Support vector data description, Mach. Learn. 54 (1) (2004) 45–66.
[25] B. Schölkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, R.C. Williamson, Estimating the support of a high-dimensional distribution, Neural Comput. 13 (7) (2001) 1443–1471.
[26] L. Manevitz, M. Yousef, One-class document classification via neural networks, Neurocomputing 70 (7-9) (2007) 1466–1481.
[27] O. Taylor, J. MacIntyre, Adaptive local fusion systems for novelty detection and diagnostics in condition monitoring,", in: Sensor Fusion: Architectures, Algorithms, and Applications II, 3376, 1998, pp. 210–218. International Society for Optics and Photonics.
[28] C.C. Aggarwal, S. Sathe, Outlier Ensembles, Springer, 2017.
[29] C.C. Aggarwal, S. Sathe, Theoretical Foundations and Algorithms for Outlier Ensembles, ACM SIGKDD Explor. Newsl. 17 (1) (2015) 24–47.
[30] A. Lazarevic, V. Kumar, Feature bagging for outlier detection, in: Presented at the Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, 2005.
[31] G. Rätsch, S. Mika, B. Schölkopf, K.-R. Müller, Constructing boosting algorithms from SVMs: An application to one-class classification, IEEE Trans. Pattern Anal. Mach. Intell. 9 (2002) 1184–1199.
[32] S. Rayana, W. Zhong, L. Akoglu, Sequential ensemble learning for outlier detection: a bias-variance perspective, in: *Proceedings of the* IEEE 16th International Conference on Data Mining (ICDM), IEEE, 2016, pp. 1167–1172.
[33] M. Salehi, X. Zhang, J.C. Bezdek, C. Leckie, "Smart sampling: a novel unsupervised boosting approach for outlier detection,", in: Proceedings of the Australasian Joint Conference on Artificial Intelligence, Springer, 2016, pp. 469–481.
[34] L. Breiman, Bagging Predictors, Mach. Learn. 24 (2) (1996) 123–140.
[35] T.K. Ho, The random subspace method for constructing decision forests, in: Proceedings of the IEEE Transactions on Pattern Analysis & Machine Intelligence, 20, 1998, pp. 832–844.
[36] L. Swersky, H.O. Marques, J. Sander, R.J.G.B. Campello, A. Zimek, On the evaluation of outlier detection and one-class classification methods, in: Proceedings of the IEEE International Conference on Data Science and Advanced Analytics, 2016, pp. 1–10.
[37] D.M. Tax, R.P. Duin, Support vector data description, Mach. Learn. 54 (1) (2004) 45–66.
[38] D.M.J. Tax, One-Class Classification (Concept-Learning in the Absence of Counter-Examples) (2001).
[39] J. Gao, P.N. Tan, Converting output scores from outlier detection algorithms into probability estimates, in: Proceedings of the Sixth International Conference on Data Mining, 2006, pp. 212–221.
[40] G. Giacinto, F. Roli, Dynamic classifier selection based on multiple classifier behaviour, Pattern Recognit. 34 (9) (2001) 1879–1881.
[41] G. Giacinto, F. Roli, Methods for dynamic classifier selection, in: Proceedings of the Tenth International Conference on Image Analysis and Processing, 1999, pp. 659–664.
[42] L.I. Kuncheva, Switching between selection and fusion in combining classifiers: an experiment, IEEE Trans. Syst. Man Cybern. Part B Cybern. 32 (2) (2002) 146–156.
[43] J. Demsar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (1) (2006) 1–30.
[44] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Inf. Sci. 180 (10) (2010) 2044–2064.