Neural Computing and Applications (2020) 32:8047–8063
https://doi.org/10.1007/s00521-019-04307-5

ORIGINAL ARTICLE

Detecting outliers in industrial systems using a hybrid ensemble scheme

Biao Wang[1] · Zhizhong Mao[2]

Received: 19 November 2018 / Accepted: 17 June 2019 / Published online: 24 June 2019

Abstract

The massive growth of process data in industrial systems has promoted the development of data-driven techniques, while the presence of outliers in process data always deteriorates the effectiveness. This paper focuses on detecting outliers in industrial systems under the assumption that no labeled training data are available. Our method is on the basis of ensemble learning, and the base learners include both one-class classifiers and multi-class classifiers. The core idea is that one-class classifier ensemble model is used to address the problem of missing label, and the usage of multi-class classifier ensemble model is to further improve its performance when outlier examples are available. The essential motivation for this proposal is that results from a classifier trained using only positive data will not be as good as the results using both positive and negative data. We investigate the performance of the proposed scheme with a series of experiments. Ten benchmark data sets and two real-world industrial systems are used, and the results approve the performance of our detection scheme.

Keywords Outlier detection · Ensemble learning · Machine learning · Industrial system

1 Introduction

Modern process industry has embraced the dawn of a data-based epoch due to the difficulty in deriving physical model for complicated processes. Many data-driven techniques have been developed to facilitate industrial systems in the last years. However, the presence of outliers, to a great extent, prevents their applications in practical systems. This can be found from both process control systems and process monitoring systems. This motivates us to develop an outlier detection scheme dedicated to industrial systems.

1.1 Motivation and challenges

System identification is always a key step for process control. In adaptive control or predictive control, the result of system identification will influence the control performance heavily. Due to the difficulty in constructing mechanism models, many data-based models have been proposed to describe process systems, e.g., transfer function models, state-space models, polynomial models, and spectral models. Then, process data will be used to learn system parameters. Note that no matter which model is used, the data quality will always play an important role, and hence, a set of unrepresentative training samples may trigger biased identification result.

From the perspective of process monitoring, traditional models obtained from first principles are of much difficulty since modern industrial processes are usually large scale and complex. This is similar to the scenario in process control. In this situation, constructing efficient and reliable monitoring systems from process data becomes alternative. Of most studies, methods on the basis of multivariate statistics, such as PCA (principal component analysis) and PLS (partial least square), have drawn great attention. Actually, they have been applied in many industrial applications. We have noted that one important step in applying these data-based techniques is to obtain the portion of data representing the normal operating condition. But by disrupting the correlation structure of PCA or PLS

✉ Zhizhong Mao
maozhizhong@ise.neu.edu.cn

1 School of Automation, Shenyang Aerospace University, Shenyang 110136, China

2 School of Information Science and Engineering, Northeastern University, Shenyang 110819, China

model, the presence of outliers may trigger more false alarms or conceal true alarms.

As a consequence, outlier detection for industrial systems is a very practical work. Due to the particular characteristics of industrial data, however, many traditional detection methods can hardly be applied directly. Here, we summarize some challenges of detecting outliers for industrial systems.

A. Online data are lack of labels. It is very expensive, if not impossible, to label all measurements for online applications. This point may be the greatest challenge for many traditional outlier detection approaches, such as classification-based methods, in which training set must include sufficient instances sampled from both normal and abnormal classes.
B. The detection should be implemented online. This point can be a great challenge for nearest-neighbor-based or distance-based methods since the detection phase of these methods is usually time-consuming. This computation complexity is unaffordable for large-scale data.
C. Process data are usually noisy even faulty. In noisy environments, the robustness to noise can be more important than the performance results themselves. This point has also triggered great troubles for many outlier detection approaches, especially for those based on statistical models.

1.2 Related work

Outlier detection has always been in an extraordinary effervescence in data mining and machine learning. It has numerous applications, such as credit card fraud detection, discovery of criminal activities in electronic commerce, video surveillance, weather prediction, and pharmaceutical research. Many outlier detection methods have been proposed [1]. They can be categorized into five groups, i.e., classification-based techniques, nearest-neighbor-based techniques, cluster-based techniques, statistics-based techniques, and information-theory-based techniques [2]. Restricted by the data feature in industrial systems, these methods have rarely been applied here. For example, classification-based techniques usually require training sets containing both normal data and outliers; neighbor-based techniques usually have more computational complexity at the test phase. These problems are always difficult to solve for application here.

While, in the field of process industry, most proposed outlier detection methods are used for fault diagnosis, researches dedicated to detecting outliers in industrial systems are limited. Based on graph theory and spatiotemporal correlations of physical processes, [3] proposes a fully distributed general anomaly detection scheme for general large-scale networked industrial sensing systems. In [4], RBF network is used to construct the model of controlled object at first; then, auto-regression hidden Markov model is used to identify outliers based on fitting residuals. By taking flow information, application data, and the packet order into account, [5] proposes a self-learning anomaly detection approach for industrial control systems. Two types of anomalies are classified in [6] in the energy system of steel industry. A dynamic-time-warping-based method that combines adaptive fuzzy C means is proposed for the trend anomaly, and a k-nearest neighbor algorithm is designed for deviant anomalies. For improving fault detection performance, [7] proposes an online outlier identification and removal scheme based on neural network. In our early works [8, 9], we have proposed to use one-class classifier ensemble model as outlier detector in industrial systems. As information about outliers has not been developed, its performance can be further improved.

1.3 Contribution

In this paper, we extend our early work [10] with the aim of improving its detection performance. We propose to construct a binary classifier ensemble detection model when sufficient process measurements are available for training. We also find that the number of measurements from abnormal conditions is still much smaller than that under normal condition. To alleviate this problem hence, we propose to use an over-sampling algorithm to process the training set. Due to the usage of more system information, we expect this detection model to outperform our early work. In addition, we use the sliding window technique to update the training sets in order to make the detection model adaptive.

The remainder of this paper is structured as follows: some fundamentals used throughout this paper will be introduced in Sect. 2. Details regarding the proposed detection scheme will be presented in Sect. 3. Section 4 will show the experimental results and analysis. Finally, Sect. 5 concludes the paper and suggests some directions for future research.

2 Fundamentals

In this section, we introduce some preliminary knowledge regarding one-class classification, binary classification, and ensemble learning.

2.1 One-class classification

One-class classification (OCC) is among the most difficult, but very promising, areas of the contemporary machine learning [11, 12]. It works with the assumption that during the training phase it has only objects originating from a single class at our disposal. 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. This is common in many real-life applications, such as intrusion detection system, fault diagnosis, or object detection. As we have no access to any counterexamples during the training phase, constructing an efficient model and selecting optimal parameters for it become a very demanding task. Figure 1 illustrates several one-class classifiers trained with a banana-shaped data set. Note that only target points (denoted by "*") are available during the training phase.

2.2 Binary classification

In classification problem, the goal is to learn a mapping from inputs $\mathbf{x}$ to outputs y, where $y \in \{1, \ldots, C\}$, with C being the number of classes. If $C = 2$, this is called binary classification; if $C > 2$, this is called multi-class classification. One way to formalize the classification problem is as function approximation. We assume $y = f(x)$ for some unknown function f, and the goal of learning is to estimate the function f given a labeled training set and then to make predictions using $\hat{y} = \hat{f}(x)$. Our main goal is to make predictions on novel inputs, meaning ones that we have not seen before, since predicting the response on the training set is easy.

In this paper, we only focus on the binary classification problem ($C = 2$). Here we only introduce several common binary classifiers and briefly categorize them into the following three groups.

A. Rule-based classifiers: Rule learning for classification is an important branch of classification problem. Based on this mechanism, many machine learning algorithms have been proposed, among which fuzzy rule learning [13] and decision tree [14] may be the most representatives.
B. Neural network (NN)-based classifiers: Using NN as a classifier has never been a novel proposal. However, many modified methods based on NN were always developed due to its strong learning ability in last decades, such as incremental RBFNN, decremental RBFNN [15] and evolutionary RBFNN [16].

Support vector machine (SVM)-based classifiers: the presence of SVM is dedicated to classification problem. Methods like c-SVM [17] and ν-SVM [18] have been used extensively.

2.3 Ensemble learning

We have seen a significant development of algorithms known as ensemble classifiers or multiple classifier systems. The underlying idea of ensemble learning for classification problems is to build a number of base classifiers and then combine their outputs using a fusion rule. It has been observed that classifier ensembles usually outperform single classifiers for many classification problems [19]. The reason may be that a combination of multiple classifiers reduces risks associated with choosing an insufficient single classifier [20, 21]. Note that two key problems should be considered when developing ensemble modes. One is diversity enhancement, and the other is decision fusion.

For the first issue, two main categories, i.e., homogenous ensemble and heterogeneous ensemble, are two main techniques. In homogeneous ensemble models, all base learners are equipped with the same form. Enhancement of ensemble diversity could be achieved via the manipulations of input data and the base learner as follows: ① partition the training data set into several sample subsets. Bagging [22] and Random Forest [23] are two representative methods. They have also been developed for OCC ensemble models; ② partition the feature space of training data into several feature subsets. Random Subspace [24] should be the most famous representatives that can also be incorporated by ensembles of OCC models; ③ diversify the parameters of the single base classifier. Furthermore, by diversifying parameters of the single base learner, several "different learners" can be generated. While in heterogeneous ensemble models, different base learners are used and trained with the same training set, the most popular heterogeneous ensemble is *Stacking* [25].

Referring to the problem of decision fusion, the most popular solution should be majority voting scheme, which assigns the label of the class predicted by the highest number of classifiers from the pool to a new object. In addition, continuous output for each individual is also proposed in order to apply more sophisticated fusion algorithms. Simple fusions of support functions such as supremum and averaging do not involve learning process. When fuser weight selection began to be treated as a specific learning process, several algorithms such as linear function, mixture of Gaussian, perceptron and evolutionary algorithms have been developed as models of the fuser. Recently, several heuristic search algorithms like genetic algorithm and firefly algorithm have developed.

3 The proposed outlier detection scheme

In this section, we firstly outline the proposed detection scheme, followed by details regarding the two detection models.

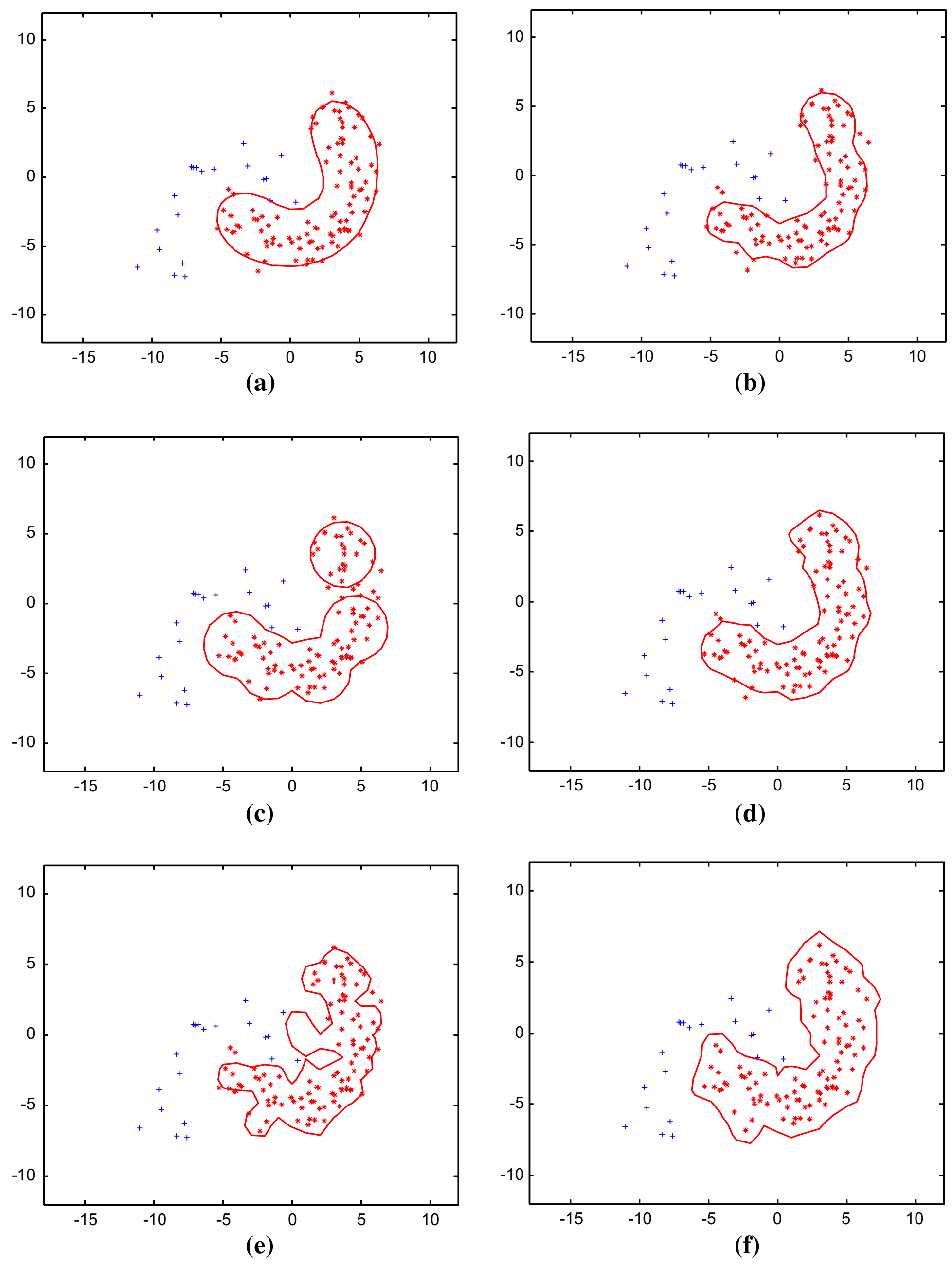

Fig. 1 **a** SVDD; **b** Parzen density data description; **c** *K*-means data description; **d** *K*-nearest neighbor data description ($k = 5$); **e** self-organizing map data description; **f** minimum spanning tree data description

3.1 Outline of detection scheme

We illustrate the general flow of the proposed detection scheme in Fig. 2, from which we can find three main phases. At the *initial training* phase, an OCC ensemble model is trained with process data collected online. Then, this model is used for detecting subsequent measurements at the *online detection* phase. Note that at this phase, we set two buffers to store normal samples (buffer 1) and outliers (buffer 2). When the collected samples in these two buffers meet certain criterion, the program will go to the *transition* phase. At this phase, we train a binary classifiers ensemble model, instead of the OCC ensemble model, with the measurements stored in both two buffers. It is noteworthy that this is the main difference from the method in our early work [10]. As stated in [26], results from a classifier trained using only positive data will not be as good as the results using positive and negative data. Thus, unless one has only training samples known to be from one class and no other information, one-class learners are likely not the best approach.

Note that we will encounter another problem called *data imbalance* when constructing binary classifiers at the transition phase. A classifier affected by the data imbalance

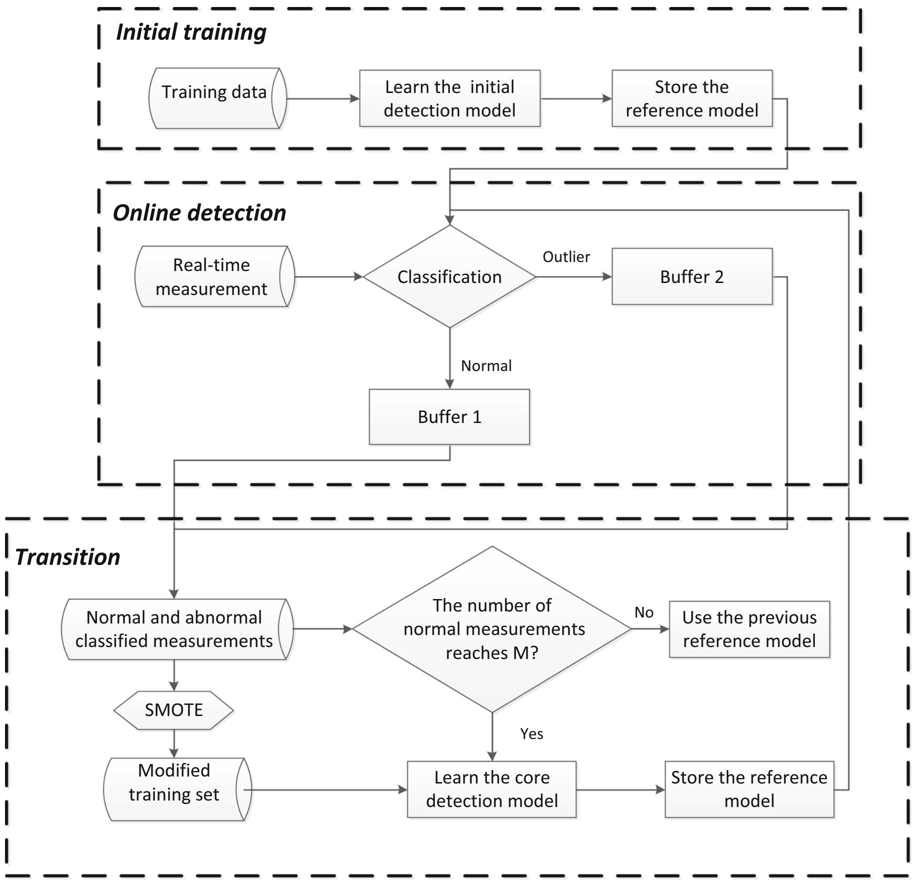
Initial training
Training data
Learn the initial detection model
Store the reference model
Online detection
Real-time measurement
Classification
Outlier
Buffer 2
Normal
Buffer 1
Transition
Normal and abnormal classified measurements
The number of normal measurements reaches M?
No
Use the previous reference model
SMOTE
Yes
Modified training set
Learn the core detection model
Store the reference model

Fig. 2 General structure of the whole outlier detection scheme

problem would see strong accuracy overall but very poor performance on the minority class. To cope with this problem, we use a resampling algorithm called SMOTE (synthetic minority over-sampling technique) [27] to preprocess the training set prior to the training of the binary classifier ensemble model. Then, a more balanced training set can be obtained, with which we can train a more accurate binary classifier ensemble model. In this paper, we call this ensemble model as *core detection model* and use it in the subsequent detection.

3.2 Initial detection model

In this paper, we assume that none history data is available for training the initial detection model. This assumption is reasonable in many practical applications. Training samples thus, for the initial model, can only come from samples collected in real time. Suppose we begin to construct the initial detection model at time t_0, then only samples collected before t_0 can be used. Thereby, the size of training set is heavily restricted by the requirement for early detection. Note that the determination of time t_0 is not an easy work. A too large value indicates a long time before the detection, while a too small value may imply an insufficient training set. This may need certain specific knowledge.

Due to the lack of labeled samples, one-class classifiers will be used as the detector as this phase. To improve the performance of single model, we construct a one-class classifier ensemble model as the initial detection model. The construction process can be demonstrated in Fig. 3. We construct a two-level heterogeneous ensemble model in order to obtain diverse base learners. At the first level, we partition the input space into several feature subspaces with algorithm RSM (random subspace method). Then for each feature subset, a heterogeneous ensemble model is constructed. The reason why we use heterogeneous models, rather than homogeneous models, is that the limited training samples in this phase can be explored sufficiently since all training samples will be used by each base learner.

Note that we use three one-class classifiers, i.e., Parzen density (PD), support vector data description (SVDD), and K-means (KM), to construct sub-ensemble models at each feature subset. The reason is that these three classifiers belong to different learning paradigms and perform better at the corresponding category. As categorized in [28], PD is a density-based classifier, SVDD belongs to the boundary-based paradigm and KM is a method based on reconstruction.

One crucial step in any ensemble combination is that of score normalization to account for the fact that the different

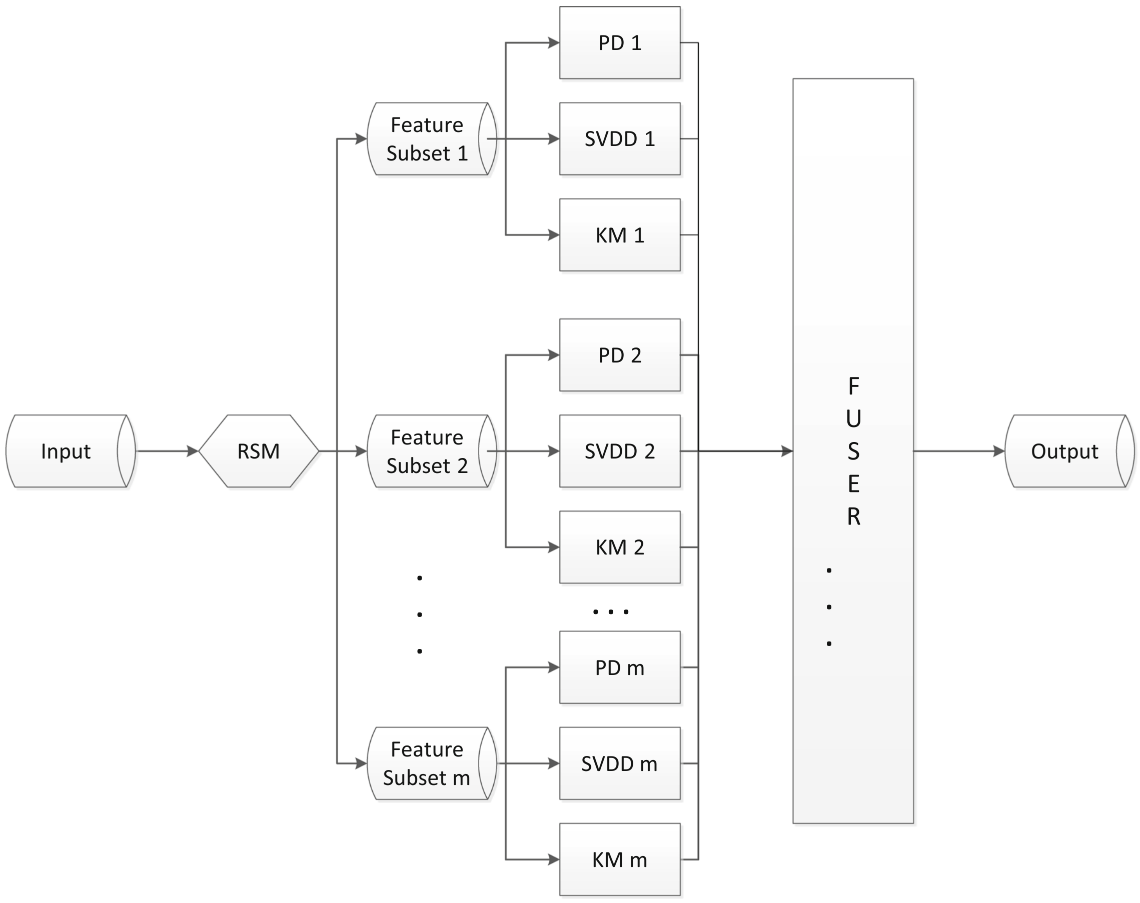

Fig. 3 General structure of initial detection model

algorithms may use scores on different scales, or they might even have a different ordering of the scores [29]. So we should normalize scores of individual classifiers in order to combine their outputs without inadvertently overweighting any one. Generally, the calibration methods for supervised classifiers fall under two categories, i.e., parametric and nonparametric. Parametric methods assume that the probabilities follow certain well-known distributions, and nonparametric methods usually employ smoothing, binning, and Bagging methods to infer probability. While these methods cannot be directly applied to the outlier detection models due to the absence of labeled samples, one solution is to treat the missing labels as hidden variables and apply the expectation–maximization (EM) algorithm to maximize the expected likelihood of the data [30]. Due to the extensive usage of logistic regression for transforming classification outputs into probability, here we employ this method to process the outputs of one-class classifiers. Meanwhile, EM algorithm is used to learn the parameters of logistic regression.

Let $X = \{x_1, x_2, \ldots, x_N\}$ denote a set of N d-dimensional observations, and $F = \{f_1, f_2, \ldots, f_N\}$ denote the corresponding outlier scores assigned to observations in X. Without loss of generality, it is assumed that the higher f_i is, the more likely x_i is an outlier. The objective is to estimate the probability that x_i is an outlier given its outlier score f_i, i.e., $p_i = P(O|f_i)$. Then, the probability that x_i is normal can be computed accordingly by $P(M|f_i) = 1 - p_i$. Here, O and M represent outlier class and normal class, respectively. According to the Bayesian theorem:

$$p_i = P(O|f_i) = \frac{P(f_i|O)P(O)}{P(f_i|O)P(O) + P(f_i|M)P(M)} \tag{1}$$

Then, we can describe it with a logistic function $p_i = 1/(1 + exp(-a_i))$, from which we can obtain:

$$a_i = \log \frac{P(f_i|O)P(O)}{P(f_i|M)P(M)} \tag{2}$$

Here, a_i can be deemed a discriminant function that classifies x_i into one of the two classes. It has been proved that for a Gaussian distribution with equal covariance matrices, a_i can be simplified to a linear function, i.e., $a_i = Af_i + B$. Then, we have:

$$p_i = \frac{1}{1 + \exp(-Af_i - B)} \tag{3}$$

The task hence has been converted to estimate parameters A and B from the training samples. Here, we let t_i be a binary variable whose value is 1 if x_i belongs to the outlier class and 0 if it is normal. Then, t_i can be described by a Bernoulli distribution:

$$p(t_i|f_i) = p_i^{t_i}(1 - p_i)^{1-t_i} \tag{4}$$

Here we assume that the observations are drawn independently; then, we can compute the likelihood of X with:

$$p(t_1, \ldots t_N|F) = \prod_{i=1}^{N} p_i^{t_i}(1 - p_i)^{1-t_i} \tag{5}$$

In contrast to supervised classification where labeled samples $\{x_i, t_i\}$ are available, we cannot learn the parameters directly by minimizing the function in Eq. 6 in the situation of outlier analysis. In this paper, EM algorithm [31] is used to estimate the missing labels and parameters simultaneously.

As we have mentioned previously, the aim of transforming outlier scores of base classifiers is to use more sophisticate fusion algorithms. In this paper, we employ an algorithm called exponential induced ordered weighted averaging (EIOWA) proposed in [32]. It has been proved that EIOWA outperforms many common fusion rules like *majority voting*, *mean*, *max*, and *product* for most used data sets. Note that in our early work [10], we also employ EIOWA as the fusion rule and prove its effectiveness.

Algorithm 1. Initial Detection Model

Input: Training set S_{Train}, the number of feature subsets m, threshold M, new measurement x_j.

```
1. RSM$(S_{Train}) \rightarrow S_{T1},\ldots,S_{Tm}$; // Use RSM to generate m feature subsets
2. for $i = 1$ to $m$
3.     $S_{Ti} \rightarrow (\mathrm{PD}_i, \mathrm{SVDD}_i, \mathrm{KM}_i)$; // Train m sub-models
4. end for;
5. Test point $x_j$
6. for $i = 1$ to $m$
7.     $\mathrm{PD}_i(x_j) = p^1{}_{ji}$;
8.     $\mathrm{SVDD}_i(x_j) = p^2{}_{ji}$;
9.     $\mathrm{KM}_i(x_j) = p^3{}_{ji}$;
10. end for
11. $\mathrm{E}(p^1{}_{j1},\ldots,p^1{}_{jm},\ldots,p^3{}_{j1},\ldots,p^3{}_{jm}) = P_j$; // Use EIOWA to fuse the outputs of sub-models
12. $\mathrm{E}(\xi_1,\ldots,\xi_m) = T_j$; // Calculate the threshold
13. if $P_j \geq T_j$
14.     Declare $x_j$ as normal and transfer it into buffer 1;
15. else
16.     Declare $x_j$ as outlier and transfer it into buffer 2;
17. end if
19. N1 = number of samples in buffer 1, N2= number of samples in buffer 2;
20. if N1 ≥ M
21.     Go to the transition phase and clear buffer 1;
22. else
23. Go to next test point;
```

Output: Label of x_j.

Thus, parameters can be optimized by maximizing the likelihood function. Generally, it is more computationally convenient to minimize the negative log-likelihood function:

$$\begin{aligned}\mathrm{NL}(t_1, \ldots t_N|F) &= -\sum_{i=1}^{N}[t_i \log p_i + (1 - t_i)\log(1 - p_i)] \\ &= \sum_{i=1}^{N}[(1 - t_i)(Af_i + B) \\ &\quad + \log(1 + \exp(-Af_i - B))]\end{aligned} \tag{6}$$

We conclude the initial model in the form of pseudo-code and show it in Algorithm 1. Note that in this algorithm, we need to determine two parameters in advance, i.e., the number of feature subsets m and the threshold of model transition M. We have to admit that these two parameters are determined in a heuristic manner. Parameter m is related to the dimension of data set, i.e., high-dimensional data set will deduce more feature subsets. While the parameter M is totally user-defined, we have no prior information about the data set. Specific values of these parameters will be introduced at the experimental parts.

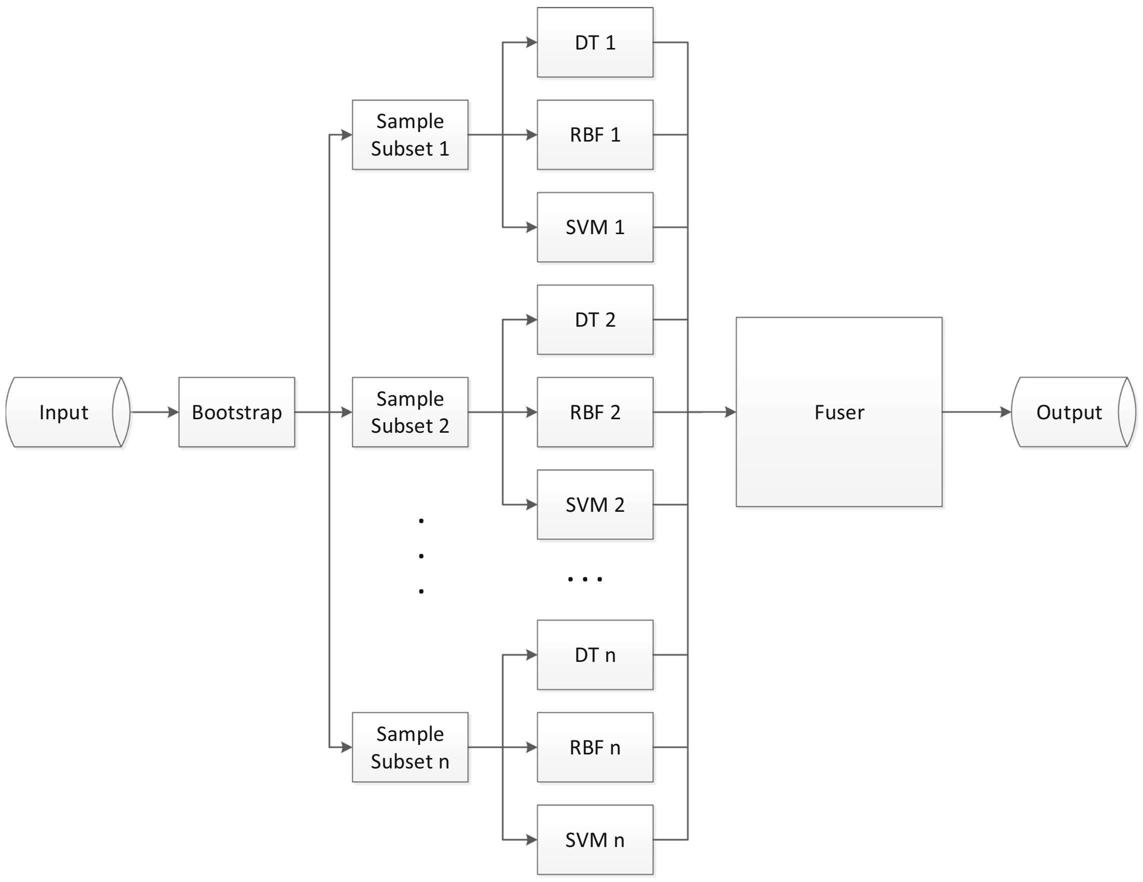

Fig. 4 General structure of core detection model

3.3 Core detection model

It has been observed that the results from a classifier trained using only positive data will not be as good as the results using positive and negative data. Thus, unless one has only training samples known to be from one class and no other information, one-class learners are likely not the best approach [26]. Inspired by this statement, we propose to construct a binary classifier ensemble model with both normal samples and outliers collected previously.

Although we can collect some outliers with the initial model, the quantity is too small compared with the normal samples. To mitigate this data imbalance problem, we employ a resampling method called SMOTE to process the samples that will be used to train the binary classifiers. For random over-sampling, several researchers agree that this method can increase the likelihood of occurring overfitting since it makes exact copies of existing instances. Therefore, we use a more sophisticated approach. Through this algorithm, abnormal class is over-sampled by taking each outlier and introducing synthetic examples along the line segments joining any or all nearest neighbors from abnormal class. With this modified training set, we train a heterogeneous ensemble of binary classifiers, in which bootstrap sampling is employed. We call this model as the *core detection model* and show its structure in Fig. 4.

Different from the initial model, bootstrap sampling method is also used to generate more subsets in addition to RSM. As concluded in [33], the *Bagging* techniques generally outperform *Boosting* in noisy data environments. Such an advantage of Bagging is of much significance for application we are interested here since practical process usually suffers from noise. As a consequence, the combination of bootstrap sampling will not only reduce the computational load, but also enhance the model robustness. Note that we use three base learners, i.e., decision tree, RBF neural network, and SVM, for each sample subset. Such a configuration could not only guarantee the diversity, but enhance the accuracy of the ensemble model. With regard to the used fusion method, majority voting is used here since outputs of base learners are predictive labels.

We also summarize the core detection model at transition phase in the form of pseudo-code and show it in Algorithm 2. Note that in this algorithm, the detector will update once samples in buffer 1 (normal samples) reach N, which is usually determined by the user. The objective of this setting is to make the detector adaptive toward new measurements by retraining it. The rationale behind this is that samples located nearer by time or distance may resemble each other more.

Algorithm 2. Core Detection Model.

Input: Training set $\boldsymbol{S_{Train}}$, the number of sample subsets $\boldsymbol{n}$, threshold $\boldsymbol{N}$, new measurement $\boldsymbol{x_j}$.

```
1. Bootstrap $(S_{Train}) \rightarrow S_{T1},...,S_{Tn}$; //Use Bootstrap to generate n sample subsets
2. for $i$=1 to n
3.     $S_{Ti} \rightarrow \mathrm{DT}_i, \mathrm{NN}_i, \mathrm{SVM}_i$; //Use subsets to train base learners
4. end for;
5. Test point $x_j$;
6. for $i$=1 to n
7.     $\mathrm{DT}_i(x_j) = L^D{}_{ji}$;
8.     $\mathrm{NN}_i(x_j) = L^N{}_{ji}$;
9.     $\mathrm{SVM}_i(x_j) = L^S{}_{ji}$;
10. end for;
11. $\mathrm{MV}(L^D{}_{j1},...,L^D{}_{jn},L^N{}_{j1},...,L^N{}_{jn},L^S{}_{j1},...,L^S{}_{jn}) \rightarrow L_j$; //Use majority voting to get the final label
12. if $L_j$=Normal
13.     Buffer 1 $\leftarrow x_j$;
14. else
15.     Buffer 2 $\leftarrow x_j$;
16. end if;
17. M1=number of samples in buffer 1, M2=number of samples in buffer 2;
18. if M1=N
19.     Retrain the detector and clear buffer 1;
20. else
21. Go to next test point.
```

Output: Label of $\boldsymbol{x_j}$.

4 Experiments and analysis

In this section, we carry out extensive experiments to investigate the performance of the proposed outlier detection scheme. Because our detection scheme integrates two different models with the consideration of varying situations in realistic applications, we firstly investigate the performance of these two models separately. For the initial detection model, we use 10 data sets with small size of training samples to represent the initial detection phase. For the core detection model, we use 10 data sets whose sizes are larger. Moreover, a fraction of outlier samples are also available in this experiment. Finally, we evaluate the whole detection scheme on two practical industrial systems.

4.1 Evaluation of the initial detection model

At this part, we investigate the performance of the initial detection model (IDM). All procedures are implemented in MATLAB with the help of toolboxes *PRTools*[1] and *dd_tools*.[2] To simulate the scenario of initial detection phase in practice, data sets, baselines, and metrics are all devised particularly.

4.1.1 Data Sets

Data sets used in this part are all taken from the *UCI Repository*.[3] A brief description is given in Table 1. Note that most of these data sets have multiple classes; we choose the most two prominent classes and filter out the remaining classes. This process is similar to that in most other researches. In addition, we use fivefold cross-validation to process data sets in order to reduce bias induced by randomness in splitting data sets. Then, averaging results will be listed.

4.1.2 Baselines

We compare IDM with the following four types of competitors:

1. A heterogeneous ensemble model (HeE). Three one-class classifiers (PD, SVDD, and KM) are directly used to construct a heterogeneous ensemble model. The difference from IDM is the lack of reconstruction of feature space by RSM. The used fusion rule is identical with that of IDM.
2. RSM series. This includes three random subspace methods whose base learners are PD, SVDD, and KM, respectively. The used fusion rules are all identical with that of IDM.
3. Bagging series. This includes three Bagging ensemble models whose base learners are PD, SVDD, and KM,

[1] http://prtools.org/.

[2] https://www.tudelft.nl/ewi/over-de-faculteit/afdelingen/intelligent-systems/pattern-recognition-bioinformatics/pattern-recognition-laboratory/data-and-software/dd-tools/.

[3] http://archive.ics.uci.edu/ml/index.php.

Table 1 Description of data sets used for evaluating IDM

No.	Data set	# Examples	# Normal examples	Features
1	Breast	286	201	9
2	C.H. disease	303	165	13
3	E-coli	220	77	7
4	Hepatitis	155	32	19
5	Glass	214	17	9
6	Imports	159	71	25
7	Ionosphere	351	225	34
8	Soybean	183	91	35
9	Vowel	180	90	13
10	Wine 2	178	71	13

respectively. The used fusion rules are all identical with that of IDM.

4. IDM with majority voting fusion algorithm (IDM_MV). The difference from the IDM is that outputs of base classifiers are directly used in the majority voting fusion rule.

4.1.3 Metrics

Traditionally for a basic two-class classification problem, the most frequently used metrics are accuracy and error rate. In this paper, we denote the outlier class as the positive class and the target class as the negative class. Following this convention along with the confusion matrix given in Table 2, accuracy and error rate are defined as:

$$\text{Accuracy} = \frac{\text{TN}+\text{TP}}{(\text{TN}+\text{FP})+(\text{TP}+\text{FN})}; \quad \text{Error Rate} = 1 - \text{accuracy} \tag{7}$$

But for certain situations where the ratio between sizes of two classes is very large, the accuracy metric can be deceiving [34]. Then, we use a famous evaluation criterion that takes into account both normal and outlier class, i.e., receiver operating characteristic (ROC). It allows the visualization of the trade-off between true-negative rate (TNR) and false-negative rate (FNR).

Table 2 Confusion matrix

	Actual label	
	Positive class	Negative class
Predicted label		
Positive class	True positive (TP)	False positive (FP)
Negative class	False negative (FN)	True negative (TN)

$$\text{TNR} = \frac{\text{TN}}{\text{TN}+\text{FP}}, \quad \text{FNR} = \frac{\text{FN}}{\text{FN}+\text{TP}} \tag{8}$$

Because the error on the majority class can be estimated relatively well, it is assumed that for all one-class classifiers a threshold can be set beforehand on the target error. Therefore, by varying this threshold and measuring the error on the outliers, an ROC curve can be obtained. Although the ROC curve can give a good summary of the performance of a one-class classifier, it is difficult to compare two ROC curves. The most common way is to summarize a ROC curve in a single number, i.e., the area under the ROC curve (AUC). This value integrates the FNR over varying TNR. A higher value indicates a better separation between target and outlier objects.

In addition to the AUC values, we also use two types of nonparametric statistic tests to provide a statistical comparison.

The first test is *Friedman ranking test*, which can check whether the assigned ranks are significantly different from assigning to each classifier an average rank via assessing the ranks of methods over all examined data sets. Under the null hypothesis that all the algorithms are equivalent, the Friedman statistic:

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{9}$$

This statistic is distributed according to χ_F^2 with (k - 1) degrees of freedom when N and k are big enough ($N > 10, k > 5$). N and k are the number of data sets and algorithms, respectively. R_j is the average rank of algorithms for all the data sets. Generally, Friedman's χ_F^2 is undesirably conservative, and a better statistic has been derived as:

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2} \tag{10}$$

This statistic is distributed according to the F-distribution with $(k-1, (k-1)(N-1))$ degrees of freedom.

If the result of Friedman test indicates statistical significance among the compared methods, we then use two post hoc tests, i.e., Nemenyi test and Bonferroni–Dunn test, to check whether the ranking difference between each pair of methods is significant or not. The performance of two algorithms is significantly different if the corresponding average ranks differ by at least the critical difference (CD):

$$\text{CD} = q_\alpha\sqrt{\frac{k(k+1)}{6N}} \tag{11}$$

The values of q_α for both Nemenyi test and Bonferroni–Dunn test can be found in [35].

Table 3 Comparison of AUC values of FHO and other four groups of methods

No.	HeE	RSM			Bagging			IDM_MV	IDM
		PD	SVDD	*K*-means	PD	SVDD	*K*-means		
1	66.9 (4)	52.6 (6)	68.2 (2)	59.9 (5)	46.9 (8)	48.1 (7)	45.4 (9)	67.8 (3)	**69.1 (1)**
2	72.1 (5)	73.0 (3)	72.8 (4)	72.0 (6)	65.0 (7)	64.8 (8)	64.1 (9)	73.2 (2)	**74.8 (1)**
3	94.3 (3)	90.5 (6)	94.8 (2)	92.9 (5)	80.3 (9)	87.8 (7)	86.3 (8)	93.9 (4)	**95.5 (1)**
4	72.6 (4)	72.8 (3)	**73.9 (1)**	72.0 (5)	63.6 (8)	66.0 (7)	62.0 (9)	71.4 (6)	73.5 (2)
5	91.2 (4)	88.4 (6)	93.3 (2)	92.1 (3)	81.2 (9)	87.1 (7)	86.2 (8)	90.3 (5)	**94.3 (1)**
6	87.4 (3)	84.7 (6)	88.2 (2)	86.4 (4)	78.9 (9)	81.3 (7)	79.9 (8)	85.9 (5)	**88.7 (1)**
7	94.6 (3)	**95.5 (1)**	91.7 (6)	92.7 (5)	86.6 (7)	84.9 (9)	84.6 (8)	93.5 (4)	95.3 (2)
8	94.4 (3)	91.3 (6)	93.9 (4)	92.5 (5)	78.5 (9)	81.7 (7)	79.4 (8)	96.1 (2)	**96.5 (1)**
9	88.4 (4)	88.0 (5)	90.3 (2)	87.9 (6)	79.4 (8)	82.5 (7)	77.8 (9)	89.6 (3)	**99.2 (1)**
10	95.2 (4)	94.2 (6)	95.8 (3)	94.2 (5)	86.1 (9)	89.3 (7)	86.4 (8)	97.4 (2)	**97.9 (1)**
Overall average	85.71 (3.7)	83.10 (4.8)	86.29 (2.8)	84.26 (4.9)	74.65 (8.3)	77.35 (7.3)	75.21 (8.4)	85.91 (3.6)	**88.48 (1.2)**

Best results are given in bold

Table 4 Result of Friedman test (significance level is 0.05)

F-statistic	Critical value	Significant difference
49.21	5.79	Yes

4.1.4 Results and analysis

Comparison of AUC values of all methods on all data sets is presented in Table 3. In addition, we also provide ranks of all methods on each data set and averaging values over all data sets. Then, statistical results with respect to Friedman test and two post hoc tests are listed in Tables 4, 5 and 6.

Based on the results in Table 3, we can easily observe that IDM outperforms other methods on most data sets (8 out of 10). Specifically, IDM outperforms HeE on all data sets. This result indicates that using RSM to reconstruct the feature space could improve the performance of heterogeneous ensemble models. Then comparing the result of IDM with those of RSM series, we can find that IDM outperforms RSM_PD and RSM_SVDD on nine data sets and outperforms RSM_KM on all data sets. This result implies that constructing sub-ensemble models on subsets generated by RSM could improve the performance of RSM-based models. Then from the results of Bagging series methods, we can find that their deviation from IDM is greater than that of other competitors. This result indicates that using Bagging technique on small-scale data sets may be an unwise decision. Finally, we can also see the improvement of the usage of EIOWA fusion method from the result of IDM_MV.

Then, we investigate the performance of all methods with statistic tests. From Table 4, we can find that the value of F-statistic (49.21) is much larger than the critical value (5.79). This indicates that the null hypothesis of Friedman is rejected and there is significant difference among these methods. Then from the results presented in Tables 5 and 6, we find that the differences between IDM and IDM_MV, RSM_SVDD, HeE are not significant. It is clear that the differences between FHO-ME and RS-PD, RS-KM and all Bagging series method are significant. This indicates that the Bagging technique is not suitable for small-scale data sets again. Furthermore, we can also find the significant differences between RSM series method and Bagging series methods. Thus, we can conclude that the usage of feature portioning technique has improved the classification performance significantly. In addition, usage of the proposed fusion method could also improve the classification performance with respect to AUC values. Nevertheless, results represented by statistical tests indicate that this improvement is not significant. Maybe with more data sets we could obtain more positive result for our method.

Table 5 Comparison result of Nemenyi test (0.1 significance level)

CD	Averaging rank								
	IDM	RSM SVDD	IDM MV	HeE	RSM PD	RSM KM	Bagging SVDD	Bagging PD	Bagging KM
3.5	1.2	2.8	3.6	3.7	4.8	4.9	7.3	8.3	8.4

Table 6 Comparison result of Bonferroni–Dunn test (0.1 significance level)

CD	Average rank								
	IDM	RSM SVDD	IDM MV	HeE	RSM PD	RSM KM	Bagging SVDD	Bagging PD	Bagging KM
3.1	1.2	2.8	3.6	3.7	4.8	4.9	7.3	8.3	8.4

Table 7 Description of data sets used for evaluating CDM

No.	Data set	# Example	IR	Features
1	Segmentation	2308	6.01	19
2	Yeast 1	1484	8.11	8
3	Yeast 2	1004	9.14	8
4	Vowel	988	10.10	13
5	Shuttle	1829	13.87	9
6	Solar flare	1066	23.79	11
7	Car	1728	24.04	6
8	Chess	2935	35.23	6
9	Abalone	2338	39.31	8
10	Wine	1482	58.28	11

4.2 Evaluation of the core detection model

At this part, we investigate the performance of the core detection model (CDM). Data sets used here include more samples than those in Sect. 4.1. Baseline methods here are also particularly selected, and all procedures are implemented on KEEL.[4]

4.2.1 Data sets

Data sets used in this experiment are all taken from *KEEL-data set Repository*. A simple description is shown in Table 7, from which we can find that all sizes of samples are larger than those in former experiments. Here we also use fivefold cross-validation to process data sets.

4.2.2 Baselines

We compare CDM with the following five competitors:

1. SVDD with negative examples (SVDD-N). When negative examples are taken into account, the decision boundary could become tighter than that of the original SVDD.
2. CDM_OC. This model is identical with the CDM except for the usage of one-class classifiers as base learners.
3. IDM. This is the initial detection model.
4. A heterogeneous ensemble of binary classifiers (HeB). DT, RBF neural network and SVM are directly used as base learners in this ensemble model.
5. CDM without SMOTE (CDM_woS). This model is identical with CDM except for the usage of algorithm SMOTE.

4.2.3 Metrics

It is noteworthy that reducing the number of false negatives is more significant than that of false positives in practice. Thus, we employ another metric called *F*-measure that can adjust the ratio of FPR to FNR, in order to evaluate the performance of different methods more reasonably.

$$F_m = \frac{(1+\beta^2)\mathrm{PPV}\times\mathrm{TNR}}{\beta^2\mathrm{PPV}+\mathrm{TNR}} \tag{12}$$

where

$$\mathrm{PPV} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FN}} \tag{13}$$

Here, we set $\beta = 0.5$, indicating that more penalty will be assigned on false positives. In addition, statistical tests are also used in this experiment.

4.2.4 Results and analysis

Configurations of base learners used in ensemble models are listed in Table 8. We should note that these parameters are not determined by any optimizing technique. They are only default values in KEEL. This is similar to that in the initial model, where all parameters of one-class classifiers are also not determined by any optimizing technique. It is observed that optimizing base learners is unnecessary for an ensemble model. Results with respect to F-measure are presented in Table 9. Statistical results are shown in Table 10.

From Table 9, firstly we can find that CDM has achieved the best averaging value of F-measure, and its averaging rank is also the best. This indicates that CDM performs better for most cases. Then, we compare the result of CDM with that of CDM-OC; we can see that CDM outperforms IDM on seven data sets. This comparison implies that binary classifier ensemble models usually perform better than one-class classifier ensemble models

[4] www.keel.es.

Table 8 Configurations of base learners used in ensemble models

Type	Algorithm	Parameter Setting
Decision tree	C 4.5	Instances per leaf = 2
Neural network	RBFN	Neurons = 50
Support vector machine	c-SVM	Kernel function = RBF; c = 100

Table 9 Comparison of F-measure values of all the methods

No.	CDM	CDM woS	HeB	CDM OC	IDM	SVDD-N
1	**0.9648 (1)**	0.9152 (2)	0.8849 (5)	0.8910 (4)	0.9092 (3)	0.7986 (6)
2	**0.9774 (1)**	0.9530 (3)	0.9330 (4)	0.9548 (2)	0.9101 (5)	0.9016 (6)
3	0.9175 (2)	0.9028 (3)	0.8870 (5)	**0.9213 (1)**	0.8992 (4)	0.8656 (6)
4	**0.8986 (1)**	0.8516 (3)	0.8113 (4)	0.7738 (5)	0.8627 (2)	0.7517 (6)
5	**0.9434 (1)**	0.8339 (5)	0.8891 (4)	0.9007 (3)	0.9112 (2)	0.8002 (6)
6	**0.8527 (1)**	0.8426 (3)	0.8397 (4)	0.8435 (2)	0.8076 (5)	0.7663 (6)
7	**0.9119 (1)**	0.8767 (3)	0.8490 (5)	0.9015 (2)	0.8701 (4)	0.8251 (6)
8	**0.9546 (1)**	0.8908 (3)	0.8809 (4)	0.9084 (2)	0.8778 (5)	0.8290 (6)
9	0.9226 (2)	0.9071 (4)	0.8883 (6)	**0.9312 (1)**	0.9135 (3)	0.8925 (5)
10	0.9194 (2)	0.8832 (3)	0.8796 (4)	**0.9232 (1)**	0.8661 (5)	0.8447 (6)
Averaging	**0.9263**	0.8857	0.8743	0.8949	0.8828	0.8275
Value	**1.3**	3.2	5.5	2.3	3.8	5.9

Best results are given in bold

Table 10 Result of Friedman tests for all the data sets (significance level is 0.05)

F-statistic	Critical value	Significant difference
11.79	3.90	Yes

Table 11 Comparison of all methods against each other with the Nemenyi test

CD	CDM	CDM OC	CDM woS	IDM	HeB	SVDD-N
2.1661	1.3	2.3	3.2	3.8	5.5	5.9

when outlier samples are available during the training phase. However, it is noteworthy that this statement is valid when these two types of ensemble models have identical structure, since we can find that CDM_OC outperforms two binary classifier ensemble models according to the results of CDM-woS and HeB. Then, we can also see the advantage of CDM when we compare its performance with that of SVDD-N. Meanwhile, from the results CDM and HeB we can find that using Bagging technique could further improve the performance of heterogeneous ensemble models. In addition, we can also find that Bagging technique outperforms RSM, which can be found from the results of CDM_OC and IDM. Finally, we can find the significance of algorithm SMOTE from the results of CDM and CDM_woS.

The statistical significance of the ranking differences has been verified by the Friedman test, followed by two post hoc tests for pairwise comparisons. In Table 10, the value of F-statistic (11.79) is much larger than critical value (3.90). Such a result indicates that the null hypothesis of Friedman test should be rejected. Then from the results in Tables 11 and 12, we can find that CDM outperforms IDM, HeB, and SVDD-N significantly. The advantage over CDM-OC and CDM-woS is not significant, even though we can still conclude a success of our method.

4.3 Experiment on electric arc furnace system

To evaluate the performance of the whole detection scheme, we first carry out an experiment on a real-world industrial system, i.e., electric arc furnace (EAF) system. EAFs are used extensively in industry to convert scrap metal into molten steel. The EAF system is a very complex and highly energy-intensive process. Approximately 60% of the energy consumed by the EAF represents electrical energy and the other 40% accounts for chemical energy, resulting from the burner materials and the chemical reactions occurring within the furnace. This high-energy

Table 12 Comparison of all methods against each other with the Bonferroni–Dunn test

CD	CDM	CDM OC	CDM woS	IDM	HeB	SVDD-N
1.9461	1.3	2.3	3.2	3.8	5.5	5.9

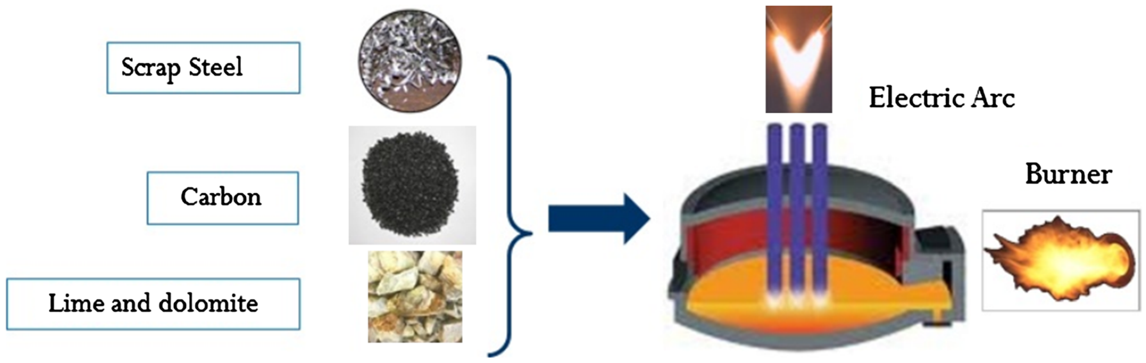

Fig. 5 A simple operational procedures of EAF

consumption of the EAF motivates the development of control and optimization strategies that would reduce production costs, while maintaining targeted steel quality and meeting environmental standards.

A simple schematic diagram of the EAF operation can be shown in Fig. 5. The scrap is loaded into the furnace and the roof is then closed, before the electrodes bore down the scrap to transfer electric energy. Natural gas and oxygen are injected into the furnace from the burners which get combusted releasing chemical energy that is also absorbed by the scrap. The scrap keeps melting through absorbing electrical, chemical and radiation energy. When sufficient amount of space is available within the furnace, another scrap charge is added and melting continues until a flat batch of molten steel is formed at the end of the batch. Through the evolution of carbon monoxide from molten metal a slag layer is formed, which contains most of the oxides from the reactions of the metals with oxygen. Slag chemistry is adjusted through oxygen and carbon lancing, besides some direct addition of carbon, lime and dolomite through the roof of the furnace. Cooling panels are used to cool down the roof and the walls of the furnace, in addition to the gas and molten metal zones.

We can find real-time measurements of several variables which have been used to design the controller [36]. In this paper, a 17-dimensional data set taken from eight variables (primary voltage, secondary voltage, primary current, secondary current, short net resistance, short net reactance, and arc impedance) is used. This data set comes from a realistic 30-ton three AC EAF system. We only choose 5000 examples from the original data set. In our data set, 500 examples are outliers, which distribute in the data set separately. Sizes of training sets for initial model and core model are 100 and 500, respectively. Here we use a robust data scaling method proposed in [37], since our data set contains outliers. In particular, we use the median of each feature values to represent its mean value, and the standard deviation is computed with half observation that is more close to the corresponding median.

Table 13 Comparison results on EAF data set

	HEOD	AR-HMM	Hy-EOCC	OCSVM	MOG
AUC	**0.9827**	0.9273	0.9717	0.9566	0.9190
F-score	**0.9649**	0.9407	0.9593	0.9480	0.9382

Best results are given in bold

We compare our detection scheme with the method proposed in [4] (AR-HMM) and our early work [10] (Hy-EOCC), as well as two traditional outlier detection methods, i.e., one-class SVM (OCSVM) [38] and mixture of Gaussian (MOG) [39]. Results are listed in Table 13. We refer to our whole detection scheme as hybrid ensemble outlier detection model (HEOD), since it includes two distinct detection models.

According to this comparison, we can easily find the higher performance of our detection scheme (HEPD) with respect to both two criteria. Note that AUC value is a criterion evaluating the general performance of outlier detection methods. We can hence say that our detection scheme has achieved the best general result over its competitors on EAF data set. We should claim that other detection models also adopt our updating rule for justified results. Via the comparison of HEOD and Hy-EOCC, we can confirm the effectiveness of the binary ensemble in the core detection model. In addition, we can also see that ensemble detection models (HEOD and Hy-EOCC) usually outperform single detection models. In this experiment, each training set has relatively less examples, to which MOG may subject. For the result with respect to F-score, only one threshold is fixed to investigate the false-positive rate and true-positive rate. Also, our detection scheme has got the best result. This indicates that HEOD has the lowest false-positive rate when the true-positive rate is fixed at a certain value. We can also explain this phenomenon as that HEOD can identify more outliers than other methods when accepting the same normal samples at the training phase.

4.4 Experiment on wind tunnel system

Then, we carry out an experiment on another industrial system, i.e., wind tunnel (WT) system. WT systems are used for testing scale models, mostly of airplanes, in the speed region of 0 to Mach 1.3. It is necessary to keep the Mach number constant at a predefined set point because most measured variables are a function of the Mach number, for given test conditions of stagnation pressure and temperature. This motivates the control of WT systems.

A schematic diagram of transonic wind tunnel is shown in Fig. 6. The air is injected into the wind tunnel through the main control hydraulic servo valve and the main injector. Then, it passes the third and the fourth corner in order to reach the stilling chamber, where air speed is low relatively. After the stilling chamber, the potential energy is translated to kinetic energy before it reaches the test section. The test section is closed in a large plenum chamber, and scale model is mounted at this section. Air exchange can be implemented through slots at the top and bottom walls. Then part of the air is injected back into the wind tunnel through the plenum exhaust valve and the plenum injector, and the remaining part is distributed uniformly through mesh screens. After the mesh screen, the air goes through a diffuser and deflects when passing the first corner. Finally, part of the air is ejected out through the main exhaust hydraulic servo valve, and the rest returns to the compressor intake.

As discussed in [40], there are totally five main impacting variables (displacement of the main control hydraulic servo valve, the displacement of the main exhaust hydraulic servo valve, the displacement of mesh screen hydraulic servo valve, the stagnation pressure, and the angle of attack) that have strong connection with the control of WT. Therefore, a 5-dimensional WT data set is used in this experiment. This data set also comes from a realistic wind tunnel system, and the working condition is 0.578 Mach number and 110 kPa total pressure. Data scaling is implemented on this data set. We select 5000 examples from the original data set, of which 500 are outliers. We also compare our method with AR-HMM, Hy-EOCC, OCSVM and MOG.

Comparison results are listed in Table 14. Based on this result, we can also find the advantage of HEOD over its competitors with respect to both criteria, while some little difference exists compared with the results on EAF data set. We should firstly note that results of all methods in terms of both criteria become worse than those on EAF data set. The reason should be that outliers in this

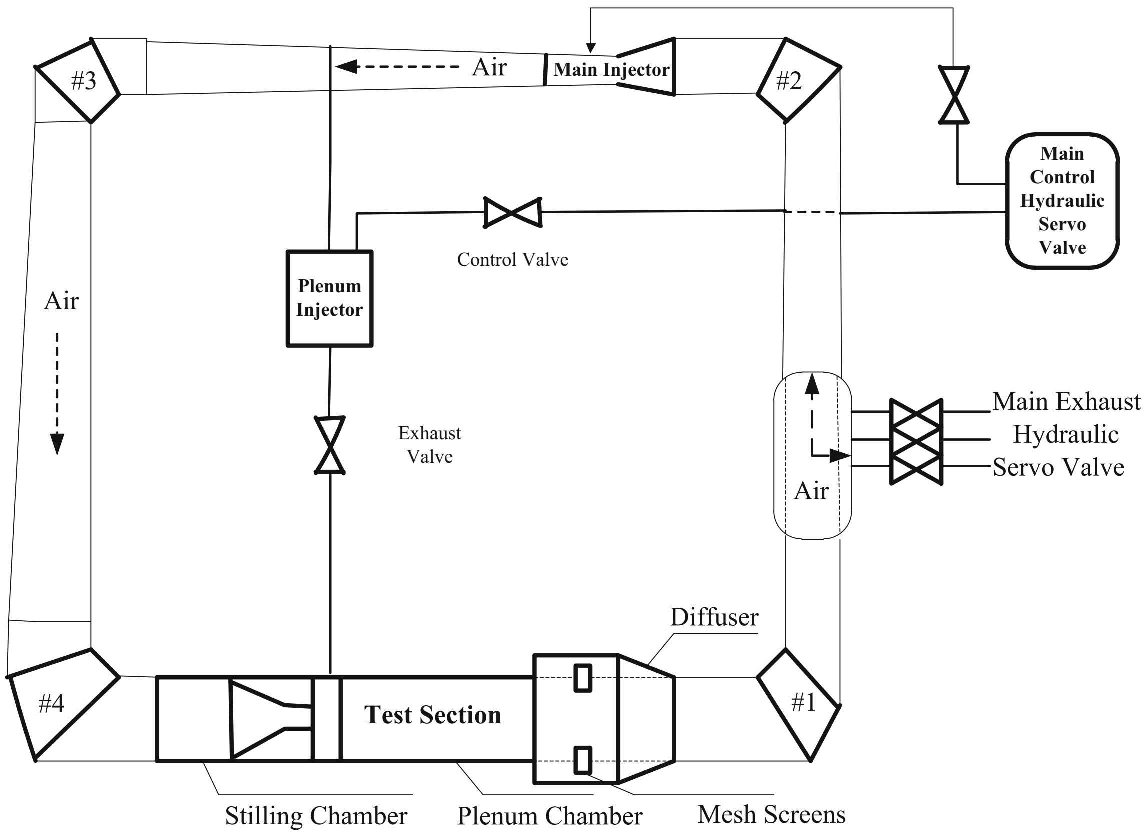

Fig. 6 Schematic diagram of transonic wind tunnel systems

Table 14 Comparison result on WT data set

	HEOD	AR-HMM	Hy-EOCC	OCSVM	MOG
AUC	**0.9544**	0.9181	0.9502	0.9318	0.9370
F-score	**0.9478**	0.9193	0.9453	0.9281	0.9302

Best results are given in bold

experiment are more close to the normal pattern so that detection methods can hardly identify certain ones. We then find that the difference between HEOD and Hy-EOCC also becomes smaller in this experiment. The reason may be that outlier examples in the training set are more different from those at the test phase. In addition, we have also found that MOG outperforms OCSVM, which is different from that on EAF data set. This indicates that training samples in this experiment are more representative. In summary, the advantage of our detection scheme is also prominent in this experiment.

Finally, we discuss the issue of computational complexity. For all detection methods used in our experiments on both EAF data set and WT data set, two types can be categorized, i.e., single model and ensemble model. Note that ensemble detection methods usually include more submodels due to their mechanism. This will usually induce more training and test time compared with single models. If we can apply the parallel computing technique, however, the computational complexity can be greatly reduced. Assuming that we use a multi-core computer or more computers to construct the ensemble model, the consuming time should approach to that of the single model.

5 Conclusions

An outlier detection scheme dedicated to industrial systems is proposed in this paper. A one-class classifier ensemble model is constructed as the initial detection model. Because we aim to implement the detection as early as possible, the training samples for this model are limited. Thus, random subspace is used to construct the ensemble model. Experimental results have shown that this initial detection model outperforms many competitors. Then a binary classifier ensemble model (core detection model) is constructed when more training samples are available. Experimental results approve this core detection model when we compare its performance with several competitors. Finally, we investigate the performance of the whole detection scheme with two real-world industrial systems. Its effectiveness has been verified by both systems.

However, several points are still open issues, such as the determination of number of subsets, when using Bagging and RSM, and a more sophisticate fusion method for binary classifiers. These problems may be the direction of future researches.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 51634002 and 61702070) and National Key R & D Program of China (Grant No. 2017YFB0304104).

Compliance with ethical standards

Conflict of interest No conflict of interest exits in the submission of this manuscript, and manuscript is approved by all authors for publication.

References

1. Wang Z et al (2015) Incremental multiple instance outlier detection. Neural Comput Appl 26(4):957–968
2. Chandola V, Banerjee A, Kumar V (2009) Anomaly detection: a survey. ACM Comput Surv 41(3):1–58
3. Chen PY, Yang S, Mccann JA (2014) Distributed real-time anomaly detection in networked industrial sensing systems. IEEE Trans Ind Electron 62(6):1–1
4. Liu F, Mao Z, Su W (2012) Outlier detection for process control data based on a non-linear auto-regression hidden Markov model method. Trans Inst Meas Control 34(5):527–538
5. Schuster F, Paul A, König H (2013) Towards learning normality for anomaly detection in industrial control networks. Springer, Berlin, pp 61–72
6. Zhao J et al (2014) Adaptive fuzzy clustering based anomaly data detection in energy system of steel industry. Inf Sci 259(3):335–345
7. Ferdowsi H, Jagannathan S, Zawodniok M (2014) An online outlier identification and removal scheme for improving fault detection performance. IEEE Trans Neural Netw Learn Syst 25(5):908–919
8. Wang B, Mao Z, Huang K (2017) Detecting outliers in complex nonlinear systems controlled by predictive control strategy. Chaos Solitons Fractals 103:588–595
9. Wang B, Mao Z (2018) Detecting outliers in electric arc furnace under the condition of unlabeled, imbalanced, non-stationary and noisy data. Meas Control 51(3–4):83–93
10. Wang B, Mao Z (2017) One-class classifiers ensemble based anomaly detection scheme for process control systems. Trans Inst Meas Control 40(12):3466–3476
11. Cabral GG, Oliveira ALI, Cahú CBG (2009) Combining nearest neighbor data description and structural risk minimization for one-class classification. Neural Comput Appl 18(2):175–183
12. Wang J et al (2017) Dynamic hypersphere SVDD without describing boundary for one-class classification. Neural Comput Appl 3:1–11
13. Cordón O, Jesus MJD, Herrera F (1999) A proposal on reasoning methods in fuzzy rule-based classification systems. Int J Approx Reason 20(1):21–45
14. Shlien S (1990) Multiple binary decision tree classifiers. Pattern Recognit 23(7):757–763
15. Broomhead DS, Lowe D (1988) Multivariable functional interpolation and adaptive networks. Complex Syst 2(3):321–355
16. Rivas VM et al (2004) Evolving RBF neural networks for time-series forecasting with EvRBF. Inf Sci 165(3):207–220
17. Vapnik V, Cortes C (1995) Support vector networks. Mach Learn 20(3):273–297

18. Scholkopf B et al (2000) New support vector algorithms. Neural Comput 12(5):1207–1245
19. Sesmero MP et al (2012) A new artificial neural network ensemble based on feature selection and class recoding. Neural Comput Appl 21(4):771–783
20. Tian J, Gu H, Liu W (2011) Imbalanced classification using support vector machine ensemble. Neural Comput Appl 20(2):203–209
21. Ge S et al (2016) Dynamic Clustering Forest: an ensemble framework to efficiently classify textual data stream with concept drift. Inf Sci 357:125–143
22. Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140
23. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
24. Ho TK (1998) The random subspace method for constructing decision forests. IEEE Trans Pattern Anal Mach Intell 20(8):832–844
25. Wolpert DH (1992) Stacked generalization. Neural Netw 5(2):241–259
26. Manevitz LM, Yousef M (2001) One-class SVMs for document classification. J Mach Learn Res 2(1):139–154
27. Chawla NV et al (2002) SMOTE: synthetic minority over-sampling technique. J Artif Intell Res 16(1):321–357
28. Tax DMJ (2001) One-class classification (concept-learning in the absence of counter-examples). Delft University of Technology, Delft
29. Haijun Z et al (2011) Textual and visual content-based anti-phishing: a Bayesian approach. IEEE Trans Neural Netw 22(10):1532–1546
30. Gao J, Tan PN (2006) Converting output scores from outlier detection algorithms into probability estimates. In: Sixth international conference on data mining
31. Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc 39(1):1–38
32. Parhizkar E, Abadi M (2015) BeeOWA: a novel approach based on ABC algorithm and induced OWA operators for constructing one-class classifier ensembles. Neurocomputing 166:367–381
33. Khoshgoftaar TM, Van Hulse J, Napolitano A (2011) Comparing boosting and bagging techniques with noisy and imbalanced data. IEEE Trans Syst Man Cybern A Syst Hum 41(3):552–568
34. He H, Garcia EA (2009) Learning from imbalanced data. IEEE Trans Knowl Data Eng 21(9):1263–1284
35. Demsar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7(1):1–30
36. Li L, Mao Z (2012) A direct adaptive controller for EAF electrode regulator system using neural networks. Neurocomputing 82(4):91–98
37. Chiang LH, Pell RJ, Seasholtz MB (2003) Exploring process data with the use of robust outlier detection algorithms. J Process Control 13(5):437–449
38. Schölkopf B et al (2014) Estimating the support of a high-dimensional distribution. Neural Comput 13(7):1443–1471
39. Bishop CM (1995) Neural networks for pattern recognition. Oxford University Press, Oxford
40. Wang X, Yuan P, Mao Z (2015) Ensemble fixed-size LS-SVMs applied for the Mach number prediction in transonic wind tunnel. IEEE Trans Aerosp Electron Syst 51(4):3167–3181