Multimedia Tools and Applications (2019) 78:30923–30942
https://doi.org/10.1007/s11042-019-7579-3

# Anomaly detecting and ranking of the cloud computing platform by multi-view learning

**Jing Zhang**[1]

Received: 17 January 2019 / Revised: 24 February 2019 / Accepted: 3 April 2019 /
Published online: 17 April 2019


**Abstract**
Anomaly detecting as an important technical in cloud computing is applied to support smooth running of the cloud platform. Traditional detecting methods based on statistic, analysis, etc. lead to the high false-alarm rate due to non-adaptive and sensitive parameters setting. We presented an online model for anomaly detecting with the machine learning theory. However, most existing methods based on machine learning linked all features from difference sub-systems into a long feature vector directly, which is difficult to both exploit the complement information among sub-systems and ignore multi-view features enhancing the classification performance. Aiming to above problems, the proposed method automatic fuses multi-view features and optimizes the discriminative model to enhance the accuracy. This model takes advantage of extreme learning machine (ELM) to improve detection efficiency. ELM is the single hidden layer neural network, which is transforming iterative solution of the output weights to solution of linear equations and avoiding the local optimal solution. Moreover, we rank anomies according to the relationship between samples and the classification boundary, and then assigning weights for ranked anomalies, retraining the classification model finally. Our method exploits the complement information among subsystems sufficiently, and avoids the influence from the imbalance distribution, therefore, deal with various challenges from the cloud computing platform. We deploy the privately cloud platform by Openstack, verifying the proposed model and comparing results to the state-of-the-art methods with better efficiency and simplicity.

**Keywords** Anomaly detection · Cloud computing · Extreme learning machine · Multi-view fusing

## 1 Introduction

Cloud computing makes possible auto-scaling and using resources at all time, moreover, avoid waste or out of expectation when developers deploy applications [13, 39]. However,

✉ Jing Zhang
zhangjing_0412@163.com

[1] School of Computer and Information Technology, Liaoning Normal University, Liushu South Street NO.1 Ganjingzi District, Dalian 116018, China

multiple anomalies in the cloud platform become both the major bottleneck for high available and the primary cause to impede development, it is important to how to build efficient anomaly detecting models. Previous studies in anomaly detecting focus on statistical analysis of running curves, which is computing and comparing curves and setting the threshold values to find anomalies by operators. The kind of methods produces both the low accuracy and high false-alarm ratio due to adjust parameters by manually in unknown data distribution. Therefore, the anomaly detecting problem in cloud computing is often formulated as a data association problem based on machine learning. Machine learning models that contains data drive, non-linear fitting and incremental provide preconditions for the problem of detection defining and solving.

In detecting-by-learning techniques, the performance depends on the detection accuracy and the challenge from the high-dimension vector that consists of multiple sub-systems [15]. Information redundancy and noisy from different features will result in failures of detecting. Therefore, feature extraction and dimensionality reduction as efficient means enhance the detecting performance in cloud computing based on virtualization. Sub-space learning extracts principal features by building the optimum projection space such as principal component analysis (PCA) [18], locality preserving projections (LPP) [24], linear discriminant analysis (LDA) [31]. In order to improve expression of features, independent component analysis (ICA) [3] is proposed. Robust dimensionality reduction methods by sparse representation and low-rank learning including robust PCA model [30], sub-space recover model [10], etc. Above methods are applied in information selection in cloud computing due to both high-integrity date description and high-effective computing. Guan, etc. exploit PCA to obtain the most relevant components, and then adaptive kalman filter (AKF) regard as classifier to enhance the detecting performance [8]. Fu, etc., utilize PAC to extract features of the cloud platform, which is different to the research from Guan, and it is selecting features in indicator vectors by mutual information method, and then extracting the most important features on selected features based on PCA [5]. Lan, etc., analyze in such a way as to distinguish between PCA and ICA in cloud computing detecting, and accept that ICA achieves higher and faster performance in experiments [9]. However, above sub-space methods only contain the global structure and ignore the local structure in original datasets. Local projection remaining model achieves more accurate describe of data [12, 29]. In order to solve the problem that linear method induces high computational complexity and limited expression original information, Fujimaki, etc., rebuild the state space by inducing kernel model [7]. Faschi, etc., achieve the feature extraction model based on regression metric analysis, which is used to find anomaly in cloud platform [4]. However, many of these methods are limited due to all of features from sub-systems are linked to a long feature vector ignoring the complementary information from difference sub-systems. Moreover, above methods do not consider to optimize the discriminative model to enhance the performance. Finding effective and robust learning models relies on appropriate discriminative model building processing of the anomaly detecting. Deng, etc. utilize SMO network to predict anomaly in Iaas cloud platform obtaining accurate results [2]. Sauvanaud, etc. exploit the dynamic index set both to update anomaly class and to compute the classification of the mean value adapting the changing of data distribution [16]. Fu, etc. first, describe the normal of the system based on Bayesian model; second, confirm the anomaly and obtain labeled samples; finally, semi-supervised model based on decision trees is used to predict anomaly in future [6]. Lan, etc. proposed the automatic recognition method for large scale distributed systems, which is using unsupervised model to detect abnormal nodes increasing the performance [9]. In order to reduce complexity of the algorithm, Wang, etc. find K-nearest nodes based on R-tree index method [20]. Unsupervised models overcome the difficult that samples tag labels

by manual and achieve better performance in various of application fields [21–23, 25–28, 32–37], but these models high dependent on the distribution of samples. Moreover, it is an important influence how to design the metric method. Therefore, anomaly detecting methods by supervised have gained widespread concern. Beak, etc. tag samples and then utilize classification model to find anomaly [1]. Wang, etc. utilize the entropy model transforming samples into time series to enhance the accuracy [19]. Tan, etc. build on anomaly early-warning system by mixing between the Markov Model and Enhanced Bayesian Networks [17]. Yao, etc. proposed the accompanied detection model based on C4.5 classification model, which is defining both log-primary from all of samples and log-accompanied from abnormal samples. In this method, log-primary are used to train the classification model, and log-accompanied are used to recognize the type of anomalies [38]. Liu, etc. exploit SMO network to achieve automatic recognition and detection [11]. However, above methods may not be sufficiently for the definition of detecting-by-learning. For this reason, first, ignore the complementary information from multiple sub-systems. Moreover, divide into two single steps including extraction feature and classification, which is resulting from the failure supervised information in extraction feature processing. Second, the data distribution is imbalance from the cloud platform, which is reducing the detecting accuracy. Third, it is different from the traditional classification problem of anomaly detecting that is the preorder step of anomaly handing, and the most suitable handing way is used to approach anomaly ranked. Therefore, the target that the anomaly detecting problem is defined as data associate based on detecting-by-learning is obtain the set of anomalies and anomalies ranking by learning models.

Overview of our algorithm is illustrated in Fig. 1. We pose anomaly detecting as a data learning problem, which is solved by multi-view learning model. Our method achieves both automatic fuses multi-view features from multiple sub-systems of the cloud platform and obtains optimized discriminative model by improving extreme learning machine. In order to handle anomaly with distinguished methods and consider the imbalance problem, we proposed the novel method to rank the set of anomaly, and then using ranking results to

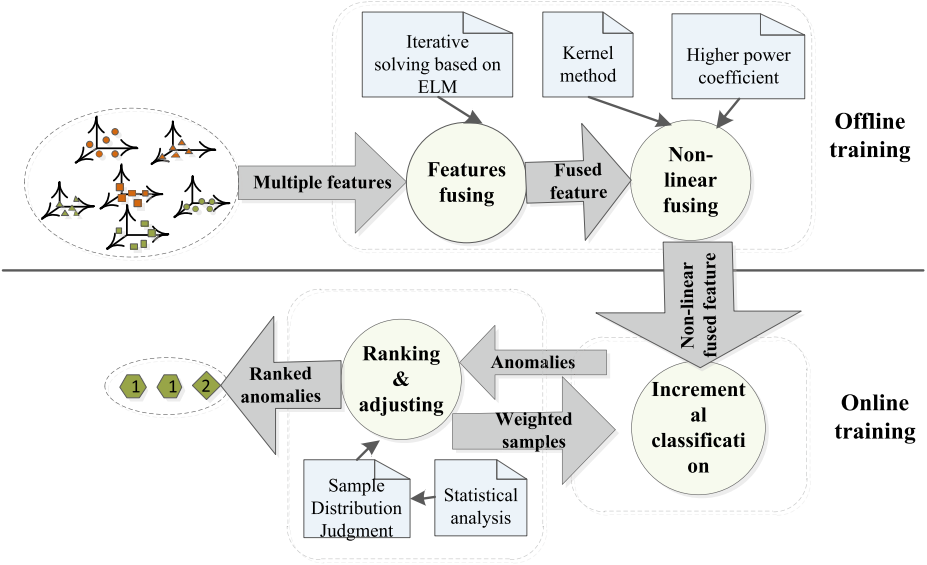


**Fig. 1** The workflow of the proposed method

optimize the classification model to enhance robust under imbalance distribution. The proposed model based on ELM has the following characteristics:

1. We provide an online method to detect anomaly by multi-view learning based on ELM without manual intervention.
2. The proposed model achieves that multi-view features automatic fuse from multiple sub-systems according to supervised information by iterating to minimize the training error, which is exploiting the complementary information substantially and obtain the optimal solution space under currently features.
3. Ranking anomalies by proposing the novel model for post-processing, and then generate weight to retrain the classification model to enhance the robustness for imbalance distribution.
4. Through the proposed model by learning, we manage various challenges from high-speed data streams, high-dimension index sets, imbalance distribution of anomalies and so on.

For the rest of this paper, we introduce ELM in Section 2. In Section 3, we proposed multi-view model to obtain the set of anomaly, ranking anomaly and optimal the classification model by means of adapting weights from ranking results. In Section 4, we utilize collected data from the private cloud platform to evaluate the proposed method and comparing it with existing detecting techniques.

## 2 Preliminaries: ELM and OSELM

### 2.1 ELM

In order to facilitate the understanding of our method, this section briefly reviews the related concepts and theories of ELM and developed OSELM.

Extreme learning machine is improved by single hidden layer neural network (SLFNs): assume given $N$ samples $(X, T)$, where $X = [x_1, x_2, ..., x_N]^T \in \mathbb{R}^{d\times N}$, $T = [t_1, t_2, ..., t_N]^T \in \mathbb{R}^{\tilde{N}\times N}$, and $t_i = [t_{i1}, t_{i2}, ..., t_{im}]^T \in \mathbb{R}^m$. The method is used to solve multi-classification problems, and thereby the number of network output nodes is $m(m \geq 2)$. There are $\widetilde{N}$ hidden layer nodes in networks, and activation function $h(\cdot)$ can be Sigmoid or RBF: $\sum_{i=1}^{\widetilde{N}} \beta_i h(a_i x_j + b_j) = o_j$ where $j = 1, \cdots, \tilde{N}$, $a_j = [a_{j1}, a_{j2}, \cdots, a_{jd}]^T$ is the input weight vector, and $\beta_j = [\beta_{j1}, \beta_{j2}, ..., \beta_{jm}]^T$ is the output weight vector. Moreover, $a_j$, $b_j$ can be generated randomly, which is known by. Written in matrix form: $H\beta = T$, where $H_i = [h_1(a_1 x_1 + b_1), \cdots, h_N(a_{\widetilde{N}} x_1 + b_{\widetilde{N}})]$. Moreover, the solution form of $H\beta = T$ can be written as: $\hat{\beta} = H^{\dagger} T$, where $H^{\dagger}$ is the generalized inverse matrix of $H$. ELM minimize both the training errors and the output weights. The expression can be formulated based on optimization of ELM:

$$\begin{aligned} \text{Minimize}: & \tfrac{1}{2}\|\beta\|_2^2 + C\tfrac{1}{2}\textstyle\sum_{i=1}^{N}\|\xi_i\|_2^2 \\ \text{Subject to}: & t_i\beta \cdot h(x_i) \geq 1 - \xi_i, i = 1, ..., N \\ & \xi_i \geq 0, i = 1, ..., N \end{aligned} \tag{1}$$

where $\xi_i = \left(\xi_{i,1} \cdots \xi_{i,m}\right)$ is the vector of the training errors. We can solve the above equation based on KKT theory by Lagrange multiplier, and can obtain the analytical expression of the output weight: $\hat{\beta} = H^T \left(\frac{I}{C} + HH^T\right)^{-1} T$. The output function of ELM is: $f(x) = h(x)\hat{\beta} = h(x)H^T \left(\frac{I}{C} + HH^T\right)^{-1} T$.

### 2.2 OSELM

The above model is used to solve classification problem for static batch data. Aiming to this problem Rong et al. proposed an increment classification model OSELM [14]. It is an online solving algorithm based on ELM. The model trains the $\Delta N(\Delta N \geq 1)$ chunk of new samples to obtain new model, then uses matrix calculation with the original model. Through the above calculation, the new output weight matrix $\hat{\beta}_{N+\Delta N}$ is obtained. When the new $\Delta N$ chunk arrives, the hidden output weight matrix is updated. The expression is listed as follows:

$$H_{N+\Delta N} = \begin{bmatrix} h(x_1; a_1, b_1) & \cdots & h(x_1; a_{\tilde{N}}, b_{\tilde{N}}) \\ \vdots & & \vdots \\ h(x_N; a_1, b_1) & \cdots & h(x_N; a_{\tilde{N}}, b_{\tilde{N}}) \\ h(x_{N+1}; a_1, b_1) & \cdots & h(x_{N+1}; a_{\tilde{N}}, b_{\tilde{N}}) \\ \vdots & & \vdots \\ h(x_{N+\Delta N}; a_1, b_1) & \cdots & h(x_{N+\Delta N}; a_{\tilde{N}}, b_{\tilde{N}}) \end{bmatrix} = \begin{bmatrix} h_1 \\ \vdots \\ h_N \\ h_{N+1} \\ \vdots \\ h_{N+\Delta N} \end{bmatrix} = \begin{bmatrix} H_N \\ H_{\Delta N} \end{bmatrix}$$

where $h_{N+k} = \left[h(x_{N+k}; a_1, b_1) \cdots h(x_{N+k}; a_{\tilde{N}}, b_{\tilde{N}})\right]^T (k = 1, \ldots, \Delta N)$ is the $k$th new sample corresponding the vector. Therefore, the output vector is $T_{\Delta N} = \left[t_{N+1} \cdots t_{N+\Delta N}\right]^T$. Therefore, the incremental expression of the output weight is obtained:

$$\hat{\beta}_{N+\Delta N} = (H_N^T H_N + H_{\Delta N}^T H_{\Delta N})^{-1}(H_N^T T_N + H_{\Delta N}^T T_{\Delta N}) \tag{2}$$

Let $G_0 = (H_N^T H_N)^{-1}$, and the incremental expression $G_1$ can be written as:

$$G_1{}^{-1} = G_0{}^{-1} + H_{\Delta N}^T H_{\Delta N} \tag{3}$$

According to the (4) and (5), the new output weight matrix $\hat{\beta}_{N+\Delta N}$ becomes:

$$\hat{\beta}_{N+\Delta N} = G_1(H_N T_N + H_{\Delta N}^T T_{\Delta N}) = \beta_N + G_1 H_{\Delta N}^T (T_{\Delta N} - H_{\Delta N}\beta_N) \tag{4}$$

where $G_1 = (G_0^{-1} + H_{\Delta N}^T H_{\Delta N})^{-1}$.

According to the above equation, we formulate the further expression:

$$G_1 = G_0 - G_0 H_{\Delta N}^T (I_{\Delta N} + H_{\Delta N} G_0 H_{\Delta N}^T)^{-1} H_{\Delta N} G_0 \tag{5}$$

From the above learning process, OSELM trains new model by adjusting the original model according to (7) when the new dynamic samples are arriving.

## 3 The proposed anomaly detecting and ranking model

In order to obtain anomalies in real-time from data stream of the cloud computing platform, we proposed an incremental detecting model based on multi-view features, ranking abnormal samples that is prerequisite of anomaly handling generate weights to feedback adjustment the classification model at current to enhance the robustness for imbalance samples of anomaly. The workflow of proposed model is show in Fig. 1, and it divides into two parts: local training and online training. In local training processing, the proposed multi-view features model is used to automatic fuse difference features and achieve optimized discriminative model. In online training processing, first, fuse multiple features according to the local learning structure, which reduce time consuming from retraining all of samples including local and online samples. Second, rank anomalies detected to handle differently.

Finally, set self-adapting weights for anomalies that are used to adjust the classification model to avoid the influence of imbalance distribution.

### 3.1 Multi-view features fusion and discriminative optimization

The number of state indicators from difference subsystems in the cloud computing platform belongs to the range from dozens to hundreds, which is composed the high-dimensional feature space. However, the information and noisy between sub-systems will influence the detected accuracy in subsequent calculations and reduce the performance of detecting. Most traditional models link multiple features into the long vector, and then extract principal components from this vector to avoid information redundancy. It is difficult to mining potential and complementary information from multiple features. Aiming to above problems, we automatic fuse multiple features and optimize classification model by iterative solving. The proposed method based on ELM that is used to classify samples. However, when the sample contains various features, the ELM model is difficult applied to solve multiple features of the same sample. In this paper, the the proposed fusion method based on ELM is describe as follow:

Given $N$ the different samples, which contain $V$ features of each sample collected in multiple ways. The feature $v$ corresponds to the samples are: $\left(x_i^{(v)}, t_i^{(v)}\right)$, where $X^{(v)} = \left[x_1^{(v)}, \ldots, x_N^{(v)}\right]^T \in \mathbb{R}^{D\times N}$, $t_i^{(v)} = \left[t_{i1}^{(v)}, \ldots, t_{im}^{(v)}\right]^T \in \mathbb{R}^m$. Meanwhile, the same sample corresponds to the same class, then there is: $t_i^{(1)} = t_i^{(2)} = t_i^{(V)}$. The output weight $\beta$ is solved by using the samples that contain combined multi-view features. The optimization equation is as follows:

$$\begin{aligned} &Minimize: \tfrac{1}{2}\|\beta\|_2^2 + \sum_{v=1}^{V} k^{(v)} \left(\sum_{j=1}^{N} \left\|\varepsilon_j^{(v)}\right\|_2^2\right) \\ &Subject\ to: \left(k \cdot H^{(v)}\right)\cdot\beta = t_i^T - \left(\varepsilon_i^{(v)}\right)^T \\ &\sum_{v=1}^{V} k^{(v)} = 1k > 0 \end{aligned} \tag{6}$$

where $k_j$ is the combined parameter that corresponds to the single-features. $\xi_i^{(v)} = \left(\xi_{i1}^{(v)}, \ldots, \xi_{im}^{(v)}\right)$ is training error vector that corresponds to feature $v$. $\beta$ is the output weight vector for different feature space in the (2). According to the (2), the target is to obtain the minimum value of the training error that combine the different feature space. However, according to the equation, the solution of $[k_1, k_2]$ may be (0,1)/(1,0). In this situation, it will degenerates to the single feature model, and other features are failed. Therefore, we introduction the high power factor $r$, and define $r \geq 2$. The equation is improved as follows:

$$\begin{aligned} &Minimize: \tfrac{1}{2}\|\beta\|_2^2 + \sum_{v=1}^{V} k^{r(v)} \left(\sum_{j=1}^{N} \left\|\varepsilon_j^{(v)}\right\|_2^2\right) \\ &Subject\ to: \left(k^{r(v)} \cdot H^{(v)}\right)\cdot\beta = t_i^T - \left(\varepsilon_i^{(v)}\right)^T \\ &\sum_{v=1}^{V} k^{r(v)} = 1k > 0 \end{aligned} \tag{7}$$

Iterative computing is used to solve both fusion coefficient and output weights due to generate interaction between feature fusion and optimization of hidden layer output weights. First, solve initial output weights by uniform fusing multi-view features. Second, testing

samples of single feature with the help of solved weights. Finally, adjust the fusion coefficient according to errors of testing, moreover, repeating the firstly step. Therefore, we exploit Lagrange multiplier method, the equation is transformed into:

$$L = \frac{1}{2}\|\beta\|_2^2 + \sum_{v=1}^{V} k^{r(v)} \left( \sum_{j=1}^{N} \left\| \varepsilon_j^{(v)} \right\|_2^2 \right) - a\left( \sum_{v=1}^{V} \left( k^{r(v)} \cdot H^{(v)} \right) \cdot \beta - t_i^T + \sum_{v=1}^{V} \left( \varepsilon_i^{(v)} \right)^T \right) - b\left( \sum_{v=1}^{V} k^{r(v)} - 1 \right) \tag{8}$$

And then, ferivation both variables $\beta$, $k$ and Lagrange multipliers $a$, $b$, and obtain output weights display expression as follow:

$$\beta = \left[ \left(k_1 H^{(1)}\right)^T + \cdots + \left(k_v H^{(2)}\right)^T \right] \left[ I/C + k_1 H^{(1)} \left(k_1 H^{(1)}\right)^T + \cdots + k_v H^{(v)} \left(k_v H^{(v)}\right)^T \right]^{-1} T, \tag{9}$$

and the fusion coefficient:

$$k_v = \frac{1/\sum_{i=1}^{N} \beta^T h_i}{\sum_{v=1}^{V} \left( 1/\sum_{i=1}^{N} \beta^T h_i \right)} \tag{10}$$

In this paper, we improve above model to incremental detecting method. According to data generation from the cloud computing platform, the training process is divided into offline and online:

Offline training: sampling multi-view features from the cloud platform, training fusion model from accumulating data, and solve both $\beta$ and $k$ according to (9) and (10).

Online: the solved offline coefficient is used to fuse new dataset $x_{\Delta N}$, and obtain the updated output matrix. According to the (9), combine with the matrix $H_{\Delta N}$ to solve the incremental output weight matrix as follow:

$$\beta_{N+\Delta N} = S_1 (H_N T_N + H_{\Delta N}^T T_{\Delta N}) = \beta_N + S_1 H_{\Delta N}^T (T_{\Delta N} - H_{\Delta N} \beta_N) \tag{11}$$

where $S_1^{-1} = S_0^{-1} + H_{\Delta N}^T H_{\Delta N}$. Therefore, our model achieves anomaly detecting in the cloud computing platform at real-time.

### 3.2 Anomaly ranking and model adjusting

Anomaly detecting is the precondition of anomalies handling to ensure high efficient running of the cloud computing platform. However, anomalies can be divided into difference ranks according to difference threats. Aiming to this problem, we proposed the novel method to ranking anomalies and the result of ranking is used to adjust the classification model to solve the imbalance problem. The workflow of the proposed model is shown in Fig. 2, including 4 steps: 1) training the classification model by the original training dataset, moreover, obtaining the output hidden weight and the output matrix; 2) according to statistical analysis, we find that it is closely related between the output matrix and the location of sample (verify it in the next paragraph). The classification matrix $T_{n\times c}$ can be obtained from the equation $H_{n\times m}\beta_{m\times c} = T_{n\times c}$, where the category number is the dimension of $T$, and the maximum value of any row of the matrix $f_{max}(T_{i\times c})$ is the class of the sample. We exploit the matrix $T$ to describe the location of each sample, achieving the sequence $l$ by ranking

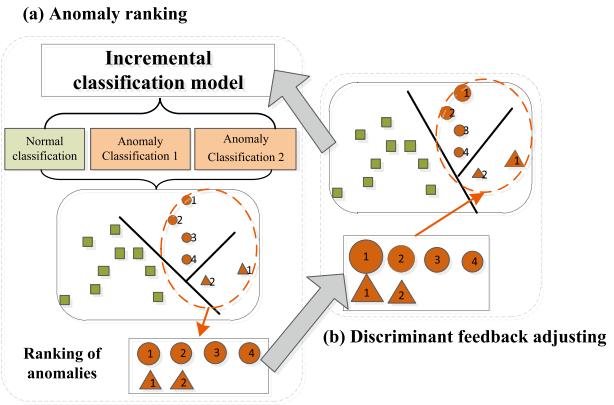


**Fig. 2** Abnormal samples ranking and discriminant model feedback adjusting

$f_{max}(T_{i\times c})$. The sequence $f_{pos}(f_{max}(T_{i\times c}))$ correspond the location sequence, which is smaller value corresponding the closer distance between the samples and the classification boundary; 3) weight samples ranked by $f_{max}(T_{i\times c})/\sum_{i=1}^{N} f_{max}(T_{i\times c})$, due to the small number for abnormal samples, weighted samples can adjusting the adaptability of classification to imbalance distribution; 4) weighted samples are used to retrain the classification model to enhance the robustness.

In order to verify step 2, we obtain the statistics result of all of samples from the testing dataset. First, rank the vector $f_{max}(T_{i\times c})$ ; and then, divide the vector $f_{max}(T_{i\times c})$ into ten ranges according to values, and count the number for each range. The more intuitive is shown in Fig. 3, the right of Fig. 3 is the histogram that is the samples number from then evenly-distributed ranges, and the left of Fig. 3 is the scatter-plots that is corresponding the histogram. From the Fig. 3, we can know locations from three kinds of samples, in 1th type, the number of 20% samples in front of the vector $l_{head}$ is rather less and samples close to the classification boundary. In 2th type, the number of 20% samples in rear of the vector $l_{rear}$ is rather less and samples far from the classification boundary. In 3th type, remaining samples that account for the largest proportions and these samples locate in the

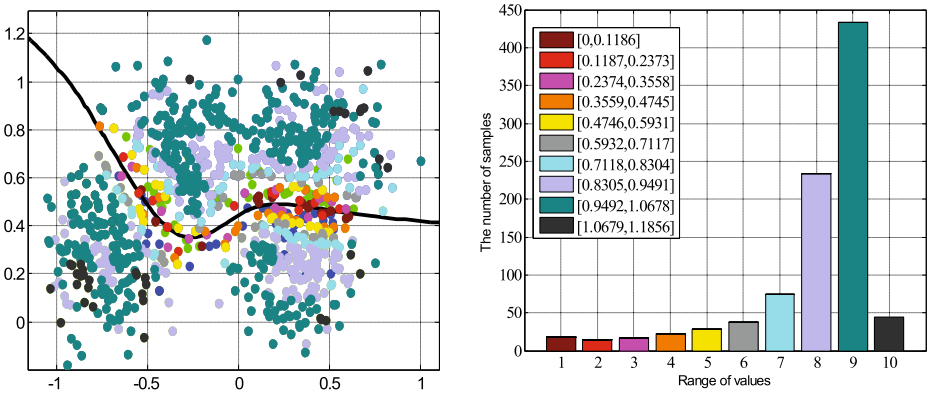


**Fig. 3** Statistical analysis of samples distribution

medium position of the classification region. Therefore, above conclusion that is the vector $f_{max}(T_{i\times c})$ representing the location of the sample is verified by analysis.

The proposed method is described as follows:

**Algorithm 1** Anomaly detecting and ranking based on multiple features fusing.

**Input:**the 2th frame data from multiple sub-systems,
and parameters: the number of hidden nodes and randomly set input weights and bias values.
**Execution:** from the 2th frame to the Nthe frame
1. Obtain multiple features from difference sub-systems;
2. Using (6) and (7) fuse multiple features, solving the optimal hidden output weight $\beta$ by iterating, and achieve the anomaly set from negative samples;
3. Ranking objects from the anomaly set and provide weights for anomalies according to ranking results;
4. Retrain all of samples containing weighted anomalies to enhance the classification accuracy.

**Output:** anomalies ranked at current frame.

## 4 Experiments

### 4.1 Experiments setting

In order to measure the performance of the proposed anomaly detecting method, we deploy the experiment environment based on OpenStack, the framework is shown in Fig. 4. that

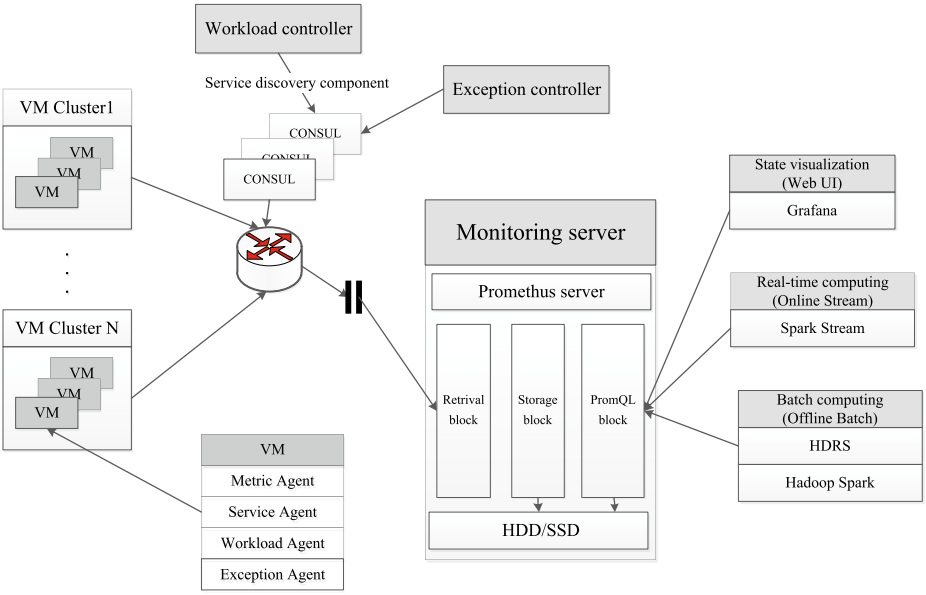


**Fig. 4** The experiments environment deployment

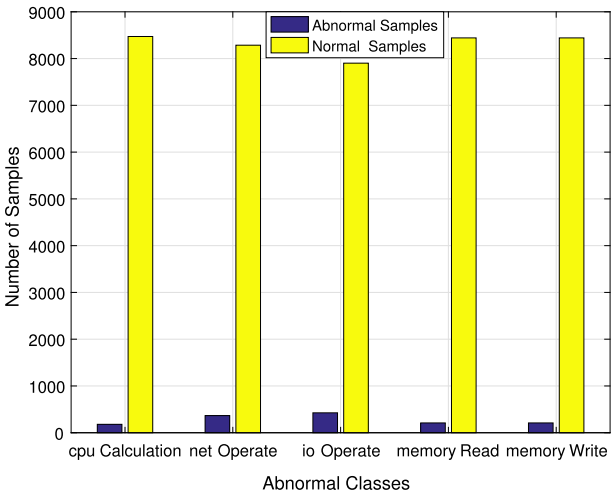


**Fig. 5** Please write your figure caption here

divided into three parts mainly. Monitoring server is built to collect and send state data of virtual machines from clusters by Promethus application. Workload sever is used to controller the workload in all of virtual machines. Each virtual machine (VM) is installed separately difference agent including Matrix, Service, Workload, Exception to detect state information of VM. Moreover, the specific setting is as the following: (1) The operating

**Table 1** The attributes description (CPU and Memory) of sub-systems

| CPU attributes set | Description | Memory attributes set | Description |
|---|---|---|---|
| node_cpu_idle | Free percentage | node_memory | Available memory |
| node_cpu_iowait | Wating I/O time | node_memory_Buffers | Block device cache |
| node_cpu_softing | Response software interruption time | node_memory_Cached | Character device cache |
| node_cpu_system | Proportion of Kernel Operations | node_memory_Swapd | Number of Use Spaces |
| node_cpu_user | User Process Propo | node_memory_MemTotal | Total physical memory |
| node_cpu_nice | Change process | node_memory_Memfree | Idle number |
| node_cpu_irq | Response to hardware interrupt time | node_memory_Slab | Kernel uses memory |
| node_cpu_cs | Process switching time | node_memory_Sheme | Process shared memory |
| node_cpu_running | Number of Runnable Tasks | node_memory_VmallocTotal | Virtual Machine Memory Volume |
| node_vcpu_run | Virtual Machine Runtime | node_memory_VmallocRate | Virtual Machine Memory Utilization Rate |
| node_cpu_runrate | Virtual Machine Utilization Rate | node_memory_VmallocMax | AMaximum occupancy of virtual machines |

**Table 2** The attributes (I/O and Network) description of sub-systems

| I/O attributes set | Description | Network attributes set | Description |
|---|---|---|---|
| node_disk_await | I/O waiting time | node_network | The amount of data received per second |
| node_disk_svc_time | I/O service time | node_network_transmit_bytes | The amount of data sended per second |
| node_disk_read_time_ms | Number of readings per second | node_network_receive_packets | Packages received per second |
| node_disk_write_time_ms | Number of writing per second | node_network_transmit_packets | The amount of data sended per second |
| node_disk_sectors_written | Reading sector count | node_network_trLoss_packets | Number of Packets Lost on Acceptance |
| node_disk_sectore_written | Writing sector count | node_network_trLoss_packets | Number of Packets Lost When Sending |
| node_disk_io_time_weighted | Percentage of operating time | node_netstat_TcpExt_TCPOFOQueue | TCP sequence |
| node_disk_bytes_written | Reading bit number | node_netstat_TcpExt_TCPOrigDataSent | TCP traffic |
| node_disk_bytes_written | Writing bit number | node_netstat_TcpExt_TCPLos | TCP untraffic |
| node_disk_Vread_time | Number of virtual block reads per second | node_Vnetwork_receive_bytes | Virtual Network Accepts Data Volume |
| node_disk_Vwrite_time | Write times per second for virtual blocks | node_Vnetwork_transmit_bytes | The amount of data sent by virtual network |

system is Ubuntu16.4.01 in the virtual services of the master and services, and we install the virtualization hypervisor KVM+Qemu. (2) The manage component is OpenStack in the cloud computing platform. (3) Performance indicator vectors of virtual services are collected from Prometheus. (4) In order to simulate the working state of each virtual service, Sysbench and Webbench are used benchmarking frameworks. We choose 5 computers as work nodes, and deploy 3 virtual services in each node.

### 4.2 Dataset collecting and descripting

In this paper, we grab the state data of the virtual service from Metric Agent service to Retrieval in real-time. In order to avoid network congestion and to ensure transmission

efficiency of computing data, we collect samples under 10s frequency, and sustained collection 3 hours. Therefore, we obtain the total items is 16200 where offline samples 20% of all samples (3240 items). Online detecting sustains 36 minutes where the number of training samples is 1620 and the number of testing samples is 1620. We collect 4 kinds of state data from virtual services including CPU, memory, disk I/O, network-service. Each item is described as follows ¡time-stamp, host, attribute-set (CPU), attribute-set (memory), attribute-set (disk-I/O), attribute-set (network-service). The samples contain both normal samples and anomalies is shown in Fig. 5, and the specific description of attributes is shown in Tables 1 and 2.

CPU anomaly: run computation programs in virtual services to achieve the very high CPU utilization rate. In this paper, the CPU anomalies are repressed as cpu_Calculation.

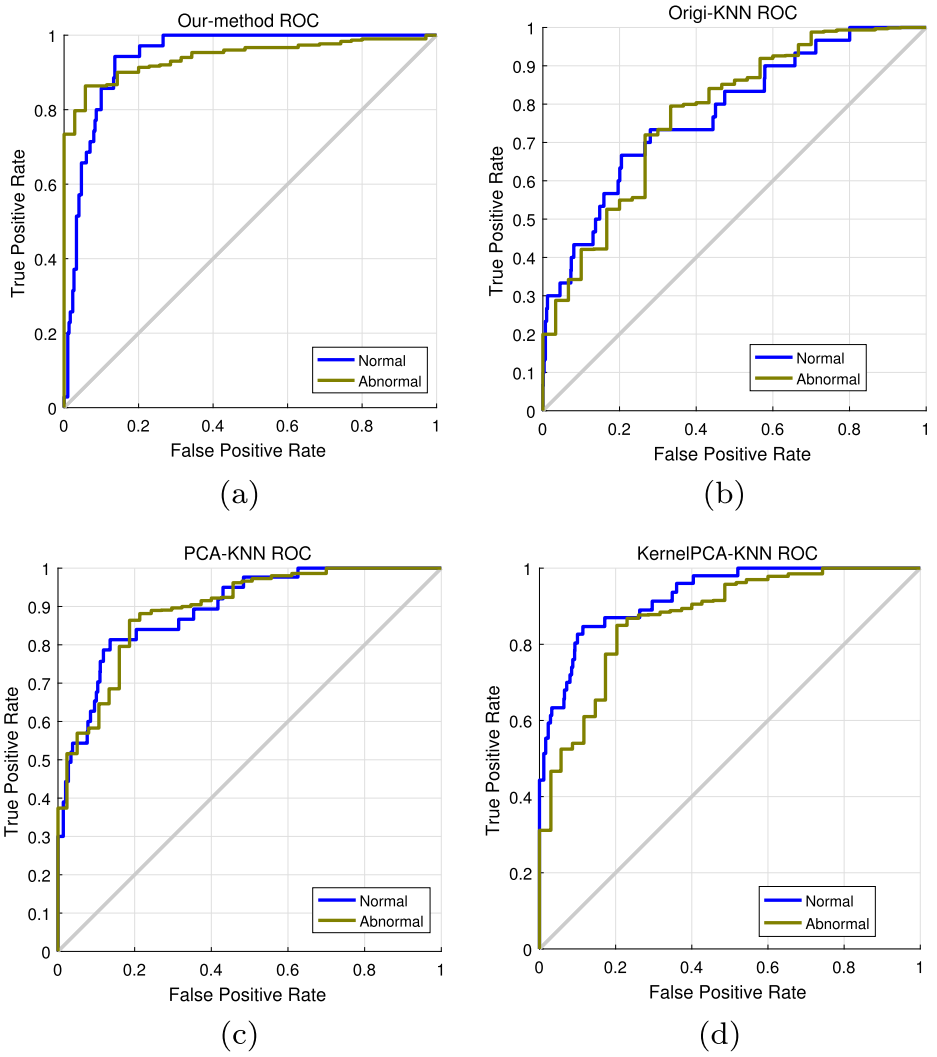


**Fig. 6** Comparison of ROC curves of cpu_Calculation anomalies

I/O anomaly: creating, writing, reading a large number of files in virtual services achieve system I/O anomalies. In this paper, the I/O anomaly is repressed as io_Operate. Network anomaly: send a large number of requiring to achieve network anomalies injection and induce the high occupancy rate of network resources of virtual services. In this paper, the Network anomaly is repressed as net_Operate. Memory anomaly: reading/writing the fixed-size memory block to increase the memory load. In this paper, memory anomalies include two classes: memory_Read and memory_Write.

### 4.3 Experiment results and analysis

Figure 5 is the histogram of sample size, and compares the positive and negative of dataset due to datasets contained anomalies are imbalance distribution.

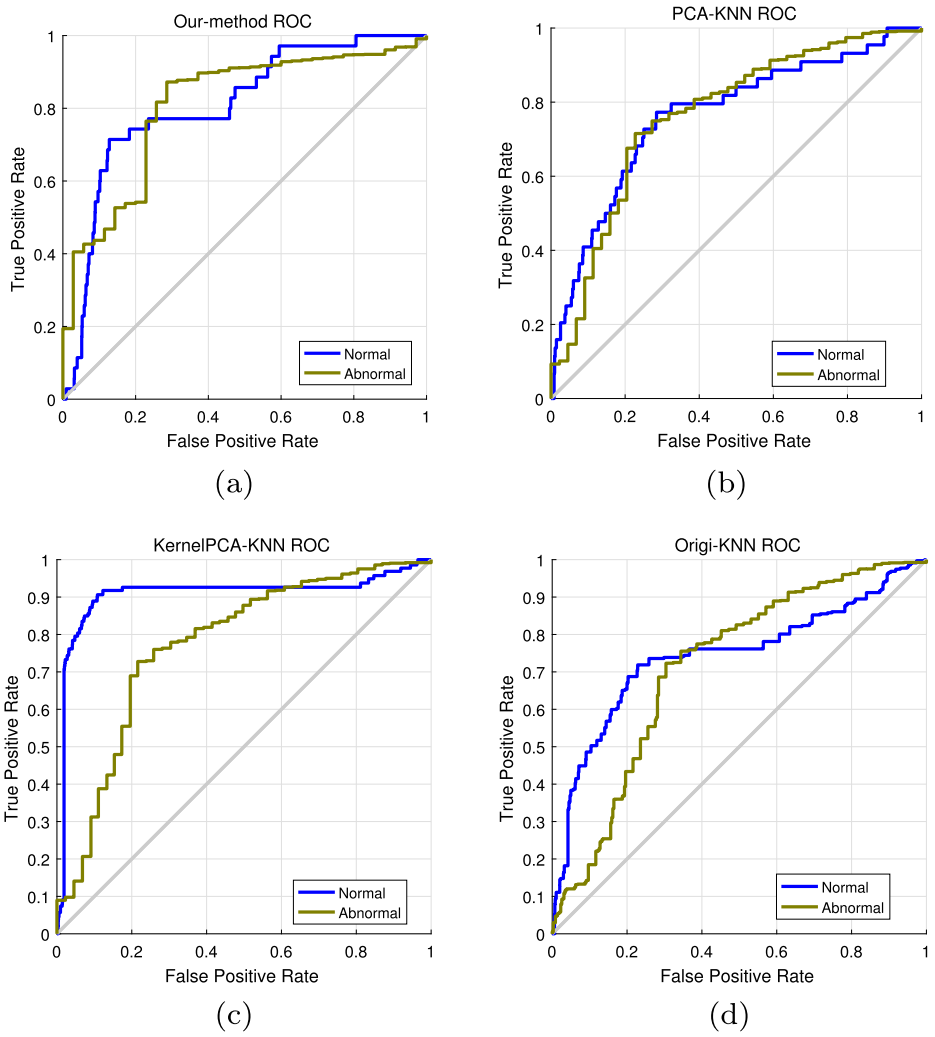


**Fig. 7** Comparison of ROC curves of io_Operate anomalies

In order to verify the performance of anomaly detecting in cloud computing platform, we utilize ROC curve to visual represent experiment results, therefore, we obtain both (false positive rate, RateFP) and (true positive rate, RateTP) of each sample, the defined as follows:

$$Rate_{TP} = E_{TP}/E_P;\ Rate_{FP} = E_{FP}/E_N \tag{12}$$

where the number of positive is $E_P$, and the number of negative is $E_N$. $E_{TP}$ and $E_{TN}$ that are the number of classification results are used to compose the coordinate of ROC curve. ROC curve describes the performance by plotting points and linking these points as a curve. When the location of the point is closer to 1, we can know that the classification achieves more performance. In this paper, we choose 3 models to compare and analysis the effectiveness of the proposed model. (1) OrigiF-KNN model, link all features from all subsystems from the cloud computing platform directly, and then KNN model is used to

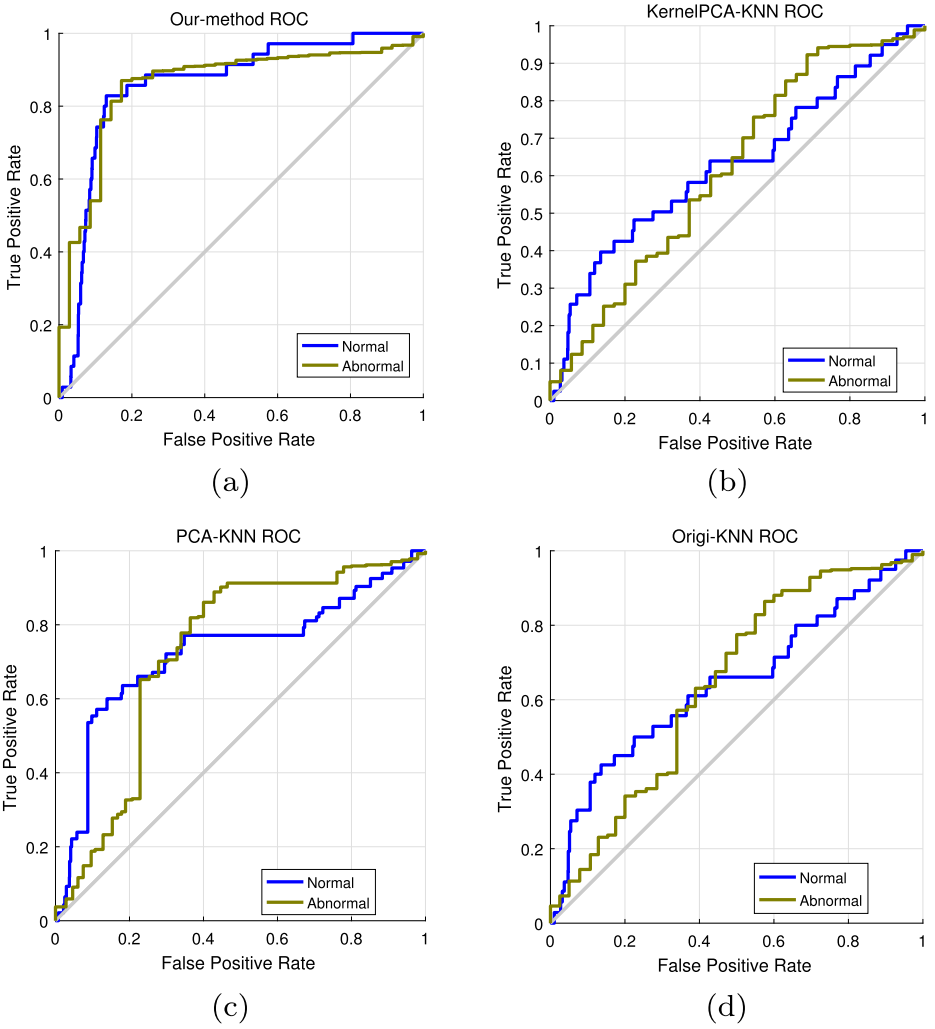


**Fig. 8** Comparison of ROC curves of net_Operate anomalies

detect anomalies. (2) PCA-KNN model, utilize PCA to reduce the dimension of the linked feature, and then KNN model is used as the classification model. (3) KernelPCA-KNN model, utilize kernel PCA to reduce the dimension of the linked feature, and then KNN model is used as the classification model. Figure 6 shows experiment results including our model and above 3 models, where blue curves are detection results from normal samples, and yellow curves are detection results from anomalies. From Figs. 6, 7, 8, 9 and 10, the proposed method enhance the detecting accuracy due to fully utilizes the complementary information from difference subsystems. Figure 6 is comparison between difference models when cpu_calculation is injected in the cloud computing platform. We can learn that Origi-KNN obtains the unsatisfied result due to the linked feature contains a lot of noisy and redundancy information to distribute the detecting result. KernelPAC-KNN by non-linear

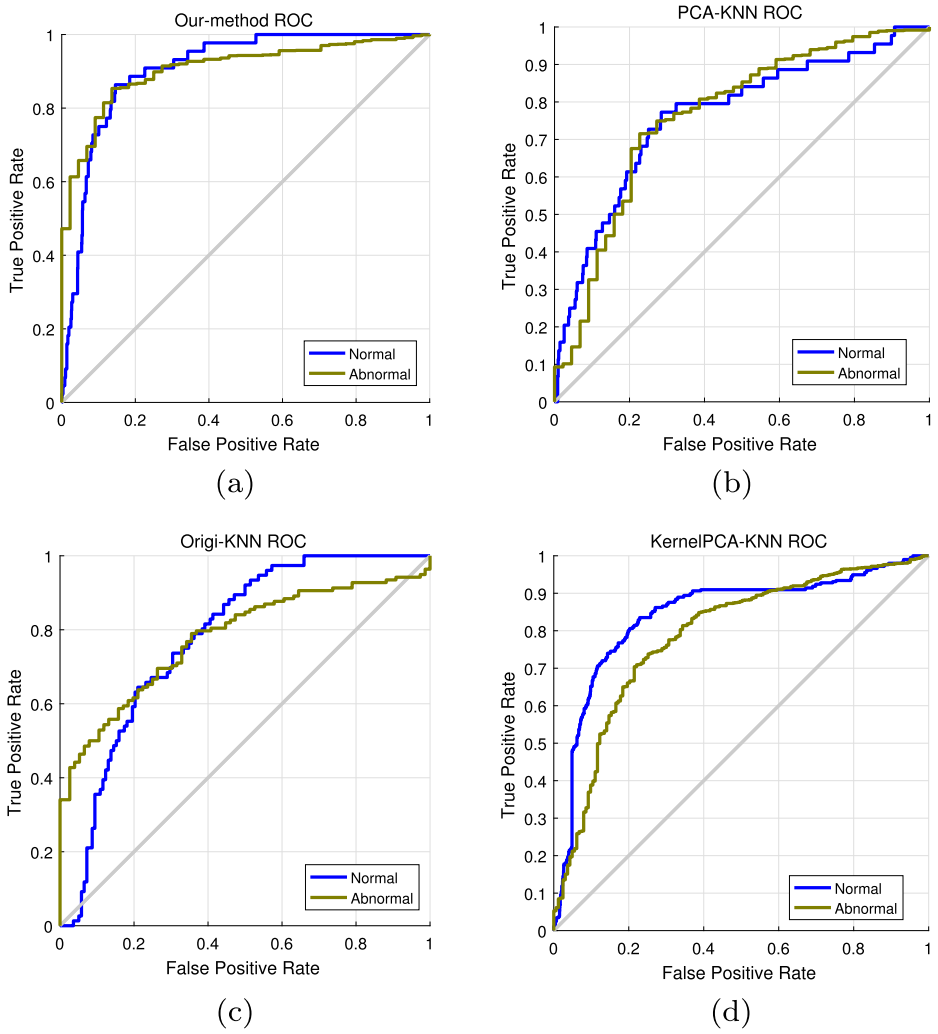


**Fig. 9** Comparison of ROC curves of memory_Read anomalies

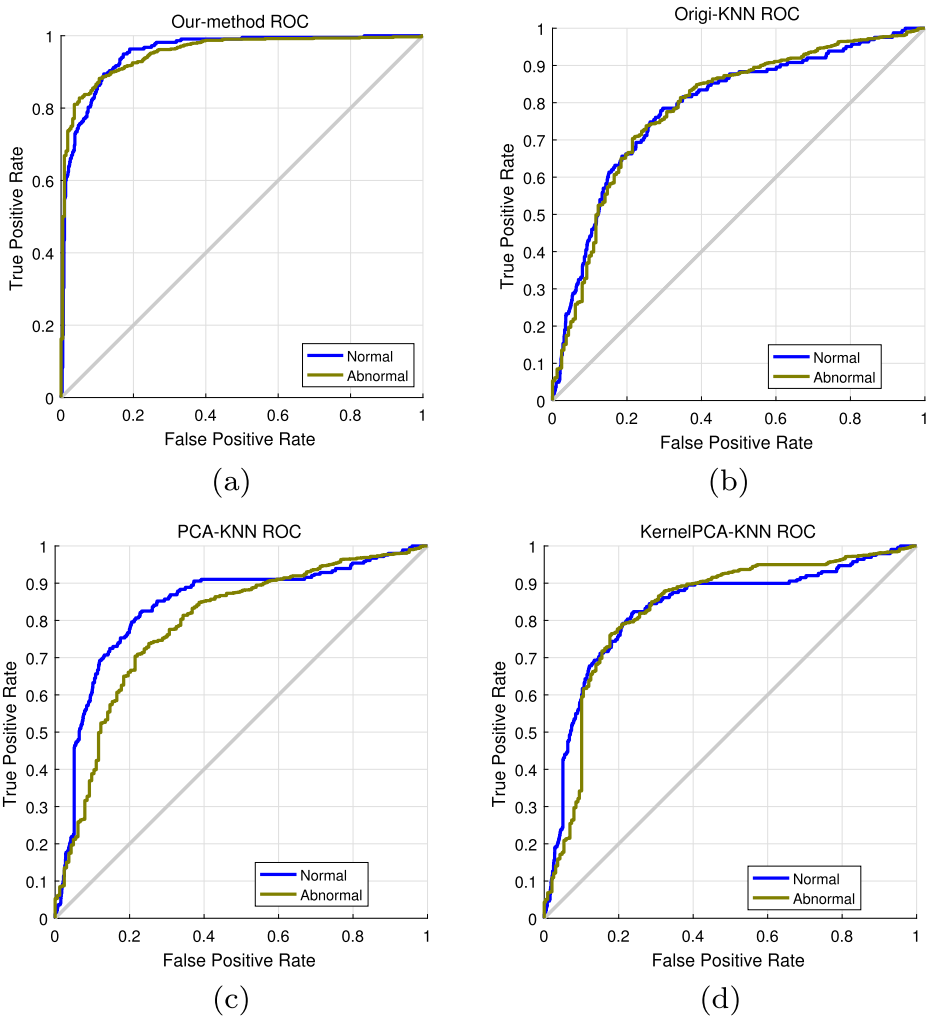


**Fig. 10** Comparison of ROC curves of memory_Write anomalies

mapping obtains higher performance comparing with PCA-KNN. Figure 7 shows difference detecting results under io_Operate anomalies, our model combines with the supervised information to achieve multi-view features automatic fusion, and optimize the solution space of the classification model to enhance the performance. When net_Operate anomalies are injected in the cloud computing platform, curves of Origi-KNN and KernelPCA-KNN are closer to the diagonal position of ROC, then detecting accuracies of both models are random from Fig. 8. The proposed model obtain the satisfied performance. In Figs. 9 and 10, we inject both memory_Read and memory_Write in the cloud computing platform. PCA-KNN, KernelPCA-KNN and our model achieve better results under the kind of anomaly.

In order to guarantee real-time performances due to dynamic generating data from the cloud computing platform, we proposed the incremental learning model that divided into two parts including offline and online learning. In offline learning processing, we achieve

multiple features fusing from difference sub-systems by iterating. In online learning processing, only learns new data at the current time and directly utilizes fusing parameters from offline learning. Therefore, more time is consumed in location learning to obtain favorable fusing from multiple features and optimized output weights, and linear time is used to calculate the incremental solution according to the (11). Moreover, we statistics each AUC value that is the area under ROC curve and the closer the AUC value is to 1, the better the classification performance in Table 3. Table 3 shows comparing results with three

**Table 3** Comparison of AUC values in online anomaly detection

| Anomaly types | Algorithms | Incremental batches | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Cpu_ calculation | Our_ method | 0.65 | 0.68 | 0.70 | 0.70 | 0.78 | 0.80 | 0.88 | 0.90 | 0.90 | 0.90 |
| | Origi_ KNN | 0.42 | 0.48 | 0.48 | 0.49 | 0.59 | 0.60 | 0.65 | 0.69 | 0.68 | .071 |
| | PCA_ KNN | 0.55 | 0.59 | 0.62 | 0.65 | 0.64 | 0.70 | 0.72 | 0.78 | 0.77 | 0.77 |
| | KernelPCA_ KNN | 0.55 | 0.56 | 0.60 | 0.71 | 0.70 | 0.75 | 0.74 | 0.78 | 0.78 | 0.79 |
| Net_ Operate | Our_ method | 0.64 | 0.65 | 0.70 | 0.70 | 0.78 | 0.78 | 0.78 | 0.80 | 0.80 | 0.81 |
| | Origi_ KNN | 0.45 | 0.46 | 0.44 | 0.55 | 0.59 | 0.65 | 0.68 | 0.70 | 0.72 | 0.72 |
| | PCA_ KNN | 0.53 | 0.55 | 0.61 | 0.70 | 0.70 | 0.70 | 0.79 | 0.79 | 0.80 | 0.80 |
| | KernelPCA_ KNN | 0.33 | 0.38 | 0.38 | 0.49 | 0.50 | 0.62 | 0.65 | 0.65 | 0.65 | 0.65 |
| Io_ Operate | Our_ method | 0.58 | 0.58 | 0.65 | 0.66 | 0.70 | 0.70 | 0.71 | 0.73 | 0.75 | 0.75 |
| | Origi_ KNN | 0.39 | 0.40 | 0.53 | 0.59 | 0.61 | 0.62 | 0.62 | 0.63 | 0.63 | 0.63 |
| | PCA_ KNN | 0.55 | 0.55 | 0.57 | 0.67 | 0.67 | 0.68 | 0.70 | 0.72 | 0.72 | 0.73 |
| | KernelPCA_ KNN | 0.41 | 0.41 | 0.51 | 0.50 | 0.54 | 0.57 | 0.59 | 0.62 | 0.64 | 0.64 |
| Memory_ Read | Our_ method | 0.65 | 0.66 | 0.70 | 0.79 | 0.79 | 0.80 | 0.88 | 0.91 | 0.91 | 0.91 |
| | Origi_ KNN | 0.55 | 0.55 | 0.60 | 0.61 | 0.65 | 0.68 | 0.73 | 0.75 | 0.76 | 0.76 |
| | PCA_ KNN | 0.59 | 0.60 | 0.67 | 0.72 | 0.75 | 0.75 | 0.75 | 0.75 | 0.77 | 0.77 |
| | KernelPCA_ KNN | 0.60 | 0.67 | 0.71 | 0.72 | 0.76 | 0.79 | 0.82 | 0.83 | 0.83 | 0.83 |
| Memory_ Write | Our_ method | 0.70 | 0.70 | 0.70 | 0.76 | 0.80 | 0.80 | 0.82 | 0.82 | 0.82 | 0.85 |
| | Origi_ KNN | 0.70 | 0.71 | 0.72 | 0.77 | 0.76 | 0.77 | 0.78 | 0.79 | 0.79 | 0.79 |
| | PCA_ KNN | 0.64 | 0.68 | 0.68 | 0.69 | 0.75 | 0.76 | 0.82 | 0.81 | 0.84 | 0.84 |
| | KernelPCA_ KNN | 0.65 | 0.67 | 0.71 | 0.72 | 0.76 | 0.79 | 0.82 | 0.83 | 0.83 | 0.83 |

kinds of detecting methods at every 6 minutes. All detecting methods obtain enhance of performances with time goes on, however, our algorithm achieves more satisfactory detecting results due to exploits both complementary information from different sub-system and supervisory information from classification labels.

## 5 Conclusion

We presented an effective method for anomaly detecting of the cloud computing platform by building the incremental machine learning model. Our problem is formulated as the binary classification problem in real-time, whose solution is learned through a improved multiple features ELM model. The proposed model automatic fuses multiple features from difference sub-systems and obtains the optimized classification solution by minimizing the training error sum; ranked anomalies are determined by the relation between samples and the classification boundary, and weighting samples ranked retrain the classification model. We can deal with various challenges in anomaly detecting, such as imbalance distribution, high dimensional features and others, effectively through multi-view learning and feed adjusting. Our model is fast and generalize well to many other sequences from the cloud computing platform.

**Acknowledgments** This work was supported by the National Natural Science Foundation of China under grants 61373127, 61772252, the Young Scientists Fund of the National Natural Science Foundation of China under grants 61702242 and the Doctoral Scientific Research Foundation of Liaoning Province under grants 20170520207.

The authors would like to thank the anonymous reviewers for the valuable suggestions they provided.

## References

1. Baek S, Kwon D, Kim J et al (2017) Unsupervised labeling for supervised anomaly detection in enterprise and cloud networks. In: 2017 IEEE 4th international conference on cyber security and cloud computing (CSC loud). IEEE, pp 205–210
2. Dean DJ, Nguyen H, Gu X (2012) Unsupervised behavior learning for predicting performance anomalies in virtualized cloud systems. In: Proceedings of the 9th ACM international conference on autonomic computing, pp 191–200
3. Du KL, Swamy MNS (2014) Independent component analysis. Springer, London
4. Farshchi M, Schneider JG, Weber I et al (2017) Metric selection and anomaly detection for cloud operations using log and metric correlation analysis. J Syst Softw 137(5):531–549
5. Fu S (2011) Performance metric selection for autonomic anomaly detection on cloud computing systems. In: Proceedings of the global telecommunications conference. IEEE, Washington, pp 1–5
6. Fu S (2011) Performance metric selection for autonomic anomaly detection on cloud computing systems. In: 2011 IEEE global telecommunications conference, pp 1–5
7. Fujimaki R, Yairi T, Machida K (2005) An approach to spacecraft anomaly detection problem using kernel feature space. In: Proceedings of the eleventh ACM SIGKDD international conference on knowledge discovery in data mining. ACM, New York, pp 401–410
8. Guan Q, Fu S (2013) Adaptive Anomaly identification by exploring metric subspace in cloud computing infrastructures. In: Proceedings of the 2013 IEEE 32nd international symposium on reliable distributed systems. IEEE, Washington, pp 205–214
9. Lan ZL, Zheng ZM, Li YW (2010) Toward automated anomaly identification in large-scale systems. IEEE Trans Parallel Distrib Syst 21(2):174–187
10. Liu GC, Yan SC (2011) Latent low-rank representation for subspace segmentation and feature extraction. In: Proceedings of the IEEE international conference on computer vision. IEEE, Washington, pp 1615–1622
11. Liu J, Chen S, Zhou Z et al (2016) An anomaly detection algorithm of cloud platform based on self-organizing maps. Mathematical Problems in Engineering
12. Liu C, Jaja J, Pessoa L (2018) LEICA Laplacian Eigenmaps for group ICA decomposition of fMRI data. NeuroImage 169:363–373

13. Rittinghouse JW, Ransome JF (2016) Cloud computing: implementation, management, and security. CRC press, Boca Raton
14. Rong HJ, Huang GB, Sundararajan N et al (2009) Online sequential fuzzy extreme learning machine for function approximation and classification problems[J]. IEEE Trans Syst Man Cybern Part B Cybern 39(4):1067–1072
15. Saleem M, Rajouri JK (2017) Cloud computing virtualization. International Journal of Computer Applications Technology and Research 6(7):290–292
16. Sauvanaud C, Silvestre G, Kaaniche M et al (2015) Data stream clustering for online anomaly detection in cloud applications. In: 2015 eleventh European dependable computing conference (EDCC). IEEE, pp 120–131
17. Tan Y, Nguyen H, Shen Z, Gu X, Venkatramani C, Rajan D. (2012) PREPARE predictive performance anomaly prevention for virtualized cloud systems. In: Proceedings of 32nd IEEE international conference on distributed computing systems, pp 285–294
18. Vidal R, Ma Y, Sastry SS (2016) Principal component analysis. Springer, New York
19. Wang C, Talwar V, Schwan K, Ranganathan P (2010) Online detection of utility cloud anomalies using metric distributions. In: Proceedings of the 2010 IEEE/IFIP network operations and management symposium, pp 96–103
20. Wang DY (2012) Research and implementation of anomaly detection technology for cloud Computin. Shanghai Jiaotong University
21. Wang Y, Lin X, Wu L, Zhang W (2015) Effective multi-query expansions: robust landmark retrieval ACM multimedia
22. Wang Y, Lin X, Wu L, Zhang W, Zhang Q, Huang X (2015) Robust subspace clustering for multi-view data by exploiting correlation consensus. IEEE Trans Image Process 24(11):3939–3949
23. Wang Y, Zhang W, Wu L et al (2016) Iterative views agreement: An iterative low-rank based structured optimization method to multi-view spectral clustering[C]. In: IJCAI international joint conference on artificial intelligence, pp 2153–2159
24. Wang R, Nie F, Hong R et al (2017) Fast and orthogonal locality preserving projections for dimensionality reduction. IEEE Trans Image Process 26(10):5019–5030
25. Wang Y, Lin X, Wu L, Zhang W (2017) Effective multi-query expansions: collaborative deep networks for robust landmark retrieval. IEEE Trans Image Processing 26(3):1393–1404
26. Wang Y, Zhang W, Wu L, Lin X, Zhao X (2017) Unsupervised metric fusion over multiview data by graph random walk-based cross-view diffusion. IEEE Transactions on Neural Networks and Learning Systems 28(1):57–70
27. Wang Y, Wu L (2018) Beyond low-rank representations: orthogonal clustering basis reconstruction with optimized graph structure for multi-view spectral clustering. Neural Netw 103:1–8
28. Wang Y, Wu L, Lin X, Gao J (2018) Multiview spectral clustering via structured low-rank matrix factorization. IEEE Transactions on Neural Networks and Learning Systems 29(10):4833–4843
29. Ward JL, Lumsden SL (2016) Locally linear embedding: dimension reduction of massive protostellar spectra. Mon Not R Astron Soc 461(2):2250–2256
30. Wright J, Ganesh A, Rao S et al (2009) Robust principal component analysis: exact recovery of corrupted low-rank matrices. Advances in Neural Information Processing Systems 87(4):2080–2088
31. Wu L, Shen C, Hengel A (2017) Deep linear discriminant analysis on fisher networks: a hybrid architecture for person re-identification. Pattern Recogn 65:238–250
32. Wu L, Wang Y, Gao J, Li X (2018) Where-and-when to look: deep siamese attention networks for video-based person re-identification. IEEE Trans. Multimedia:1–1
33. Wu L, Wang Y, Li X, Gao J (2018) Deep attention-based spatially recursive networks for fine-grained visual recognition. IEEE Transactions on Cybernetics 2018(99):1–12
34. Wu L, Wang Y, Gao J, Li X (2018) Deep adaptive feature embedding with local sample distributions for person re-identification. Pattern Recogn 73:275–288
35. Wu L, Wang Y, Li X, Gao J (2018) What-and-where to match: deep spatially multiplicative integration networks for person re-identification. Pattern Recogn 76:727–738
36. Wu L, Wang Y, Shao L (2019) Cycle-consistent Deep generative hashing for cross-modal retrieval. IEEE Trans Image Processing 28(4):1602–1612
37. Wu L, Wang Y, Shao L, Wang M (2019) 3D PersonVLAD: learning deep global representations for video-based person re-identification. IEEE Trans. Neural Networks and Learning Systems
38. Yao X, Wand HM et al (2012) Fault detection mechanism based on adjoint state tracking in cloud computing system. J Comput Sci 35(5):856–870
39. Zhen Z Research on anomaly detection strategy and algorithms aware of running environment for virtual machines in the cloud platform. Chongqing: Chongqing University. Department of Computer

**Jing Zhang** was born in 1984. She received her M.S. degree in Computer and Application Technology from Northeast Petroleum University in 2011, and Ph.D. degree in in Computer and Application Technology from Dalian University of Technology in 2016. Her research interests include pattern recognition and machine learning.