International Journal of Security and Its Applications
Vol. 10, No. 12 (2016), pp.173-186
http://dx.doi.org/10.14257/ijsia.2016.10.12.14

An Anomaly Detection Algorithm based on Online Learning Lagrangian SVM for Cloud Computing Environment

GuiPing Wang[1], Ren Li[1], and XiaoYi Yuan[2]

[1]*College of Information Science and Engineering, Chongqing Jiaotong University, Chongqing, China*
[2]*Chongqing Real Estate College, Chongqing, China*
wgp@cqjtu.edu.cn

Abstract

Anomaly detection under Cloud computing environment plays an important role in detecting anomalous virtual machines (VMs) before real failures occur. In order to accurately characterize the new trend of VMs' performance, new samples are collected, detected, and selectively added into the training sample set. The newly added samples are used for updating the detection model, so as to improve detection accuracy. However, increasing number of training samples causes both much storage space and CPU time. To overcome this challenge, this article proposes an anomaly detection algorithm based on online learning Lagrangian SVM (termed OLLSVM) for detecting anomalous VMs. Online learning includes incremental learning and decremental learning. Single-sample and batch incremental learning algorithms are designed to update the detection model when adding a single sample or a set of samples. Similarly, single-sample and batch decremental learning algorithms are designed for deleting a single sample or a set of samples. The strategies for selecting sample(s) to be added or deleted are also designed. This article conducts experiments on Cloud datasets and KDD Cup datasets. The experimental results show that, compared with traditional Lagrangian SVM (LSVM) which retrains the detection model each time when adding or deleting sample(s), OLLSVM achieves almost similar high detection accuracy but much higher time efficiency.

Keywords: *Cloud Computing; Virtual Machine (VM); Anomaly Detection; Lagrangian Support Vector Machine (LSVM); Online Learning*

1. Introduction

In order to improve the dependability ([1][2]) of Cloud platforms, anomalous virtual machines (VMs) are needed to be detected before real failures occur. Therefore, anomaly detection under Cloud computing environment (e.g., [3][4]) plays an important role in detecting VMs' anomalous performance or state represented by a set of performance metrics. For a distributed system, an *anomaly* refers to the state of the monitored distributed system (or a physical / virtual node in the system) that deviates from the expected normal states, or deviates from the states of most of the time (or most of other nodes in the system).

In a production Cloud computing platform, along with the operation of the platform and the anomaly detection system, new samples of monitoring data (e.g., performance metrics) are collected and detected in real-time. The detected samples are verified by administrators. These verified samples can be added into the training sample set, so as to reflect the new trend of VMs' performance or state. These added samples are used for updating the detection model, so as to improve detection accuracy. However, these added samples also cause a larger and larger training sample set. The training sample set requires more and more storage space. Moreover, training the detection model on this

ISSN: 1738-9976 IJSIA

training sample set also requires more and more CPU time.

To overcome the above challenge, this article introduces the ideas of online learning into designing an anomaly detection algorithm for Cloud computing environment. To be specific, this article adopts Lagrangian Support Vector Machine (LSVM) ([5][6]) to train the anomaly detection model. Moreover, it introduces and tailors the online learning algorithms of LSVM first proposed by Duan et al. ([7][8]) to cope with the problem of increasing number of training samples. Online learning includes *incremental learning* and *decremental learning*. Single-sample and batch incremental learning algorithms are designed to update the detection model when adding a single sample or a set of samples into the training sample set. Similarly, single-sample and batch decremental learning algorithms are designed for deleting a single sample or a set of samples from the training sample set. In addition, this article also designs strategies for selection sample(s) to be added or deleted. Ultimately, this article designs a complete algorithm for anomaly detection based on online learning Lagrangian SVM (termed OLLSVM).

To evaluate the effectiveness of OLLSVM, this article conducts experiments on Cloud datasets and KDD Cup datasets. The experimental results show that, compared with traditional Lagrangian SVM (LSVM) which retrains the detection model each time when adding or deleting sample(s), OLLSVM achieves almost similar high detection accuracy. Most importantly, OLLSVM achieves much higher time efficiency than LSVM.

The remainder of this article is organized as follows. Section 2 summarizes related work. Section 3 gives preliminaries of Lagrangian SVM. Section 4 presents the proposed OLLSVM in detail, including incremental learning and decremental learning. Section 5 conducts experiments. Section 6 concludes this article and looks into future work.

2. Related work

Delgado et al. ([9]) review 179 classifiers divided into 17 families including discriminant analysis, Bayesian, neural networks, support vector machines (SVM), boosting, bagging, stacking, and random forests (RF). They evaluate these classifiers on 121 UCI datasets. The evaluation results show that SVM ([10]) with Gaussian kernel implemented in C language using LibSVM ([11]) is one of the best classifiers. Classification based techniques are widely used for anomaly detection. In literature there are plentiful researches (e.g., [12]-[15]) which train SVM-based classifiers as the anomaly detection model.

For the training sample sets only including normal samples, Yin et al. ([12]) present a new fault detection algorithm based on a proposed robust one class SVM (1-class-SVM) which is a special variant of standard SVM. Since traditional 1-class-SVM is sensitive to the outliers in the training sample set, Yin et al. introduce designed penalty factors to depress the influences of outliers. There still exists the challenge of increasing number of training samples for the training sample sets only including normal samples. However, Yin et al. do not concentrate on this challenge in the proposed robust 1-class-SVM.

In order to improve the dependability of Cloud platform, Fu et al. ([14]) propose a hybrid self-evolving anomaly detection framework, which combines one-class and two-class SVMs. The standing point of equipping with two SVMs is to solve highly imbalanced datasets rather than the challenge of increasing number of training samples.

Parsa and Naree ([15]) propose a new online anomaly detection approach for software systems to detect fault-suspicious execution paths at runtime. They design a new semantic kernel function for a SVM classifier. The designed kernel function uses a new sequence matching algorithm to measure similarities among program execution paths. Therefore, they concentrate on designing an application-specific kernel function for SVM-based classifiers in anomaly detection.

Despite the above researches, when designing a detection model for a production Cloud platform, increasing number of training samples is still a challenge which are not

well solved in literature. This article introduces the ideas of online learning into SVM-based classifiers to solve this challenge. Online learning includes two aspects: *incremental learning* and *decremental learning*.

Incremental learning refers to updating the detection model only based on the newly added training samples, thus avoiding retraining the detection model on the entire training sample set. Incremental learning may greatly improve real-time performance of the detection model. If a single training sample is added each time, such as a sample of performance metrics of a single VM in a point-in-time, the learning is called *single-sample incremental learning*. If a set of training samples is added each time, such as the samples of all VMs in a monitoring domain in a point-in-time, the learning is called *batch incremental learning*. Decremental learning refers to updating the detection model after selectively deleting a single sample or a set of samples form the training sample set, thus avoiding the sample set taking up too much storage space. Similarly, decremental learning is called *single-sample* or *batch decremental learning* when a single sample or a set of samples is deleted each time.

For standard SVMs, several research efforts (e.g., [16]-[18]) in literature have been conducted to introduce online learning into SVMs. Lagrangian SVM (LSVM) ([5][6]) simplifies the quadratic programming problem of standard SVMs, thus obtaining a fast technique for training SVMs. LSVM can solve classification problems with millions of samples ([6]). Duan et al. ([7][8]) introduce online learning into LSVM. This article tailors the online learning algorithms first proposed in [7][8] to design a complete anomaly detection algorithm based on online learning LSVM. This article also designs strategies for selecting sample(s) to be added or deleted.

3. Lagrangian SVM (LSVM)

The primal optimization problem of standard SVM is:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{l}\xi_i , \tag{1}$$

s.t. $y_i((\boldsymbol{w}\cdot\boldsymbol{x}_i) + b)\geq 1 - \xi_i,\ i = 1, 2, \ldots, l,$ (2)

$\xi_i\geq 0,\ i = 1, 2, ..., l.$ (3)

where $(\boldsymbol{x}_i, y_i)$, $i = 1, 2, \ldots, l$ is the training sample set; l is the number of training samples; $\boldsymbol{w}$ and b are the normal vector and bias of the classification hyperplane respectively; C is the penalty parameter; ξ_i are slack variables.

LSVM improves and simplifies the above optimization problem as ([5][7]):

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}(\|\boldsymbol{w}\|^2 + b^2) + \frac{C}{2}\boldsymbol{\xi}^T\boldsymbol{\xi} \\ \text{s.t.} \quad & y_i((\boldsymbol{w}^T\boldsymbol{x}_i) + b) + \xi_i \geq 1,\ i = 1, 2, \ldots, l. \end{aligned} \tag{4}$$

The Lagrangian function of LSVM is:

$$L = \frac{1}{2}(\|\boldsymbol{w}\|^2 + b^2) + \frac{C}{2}\boldsymbol{\xi}^T\boldsymbol{\xi} - \sum_{i=1}^{l}\alpha_i(y_i((\boldsymbol{w}^T\boldsymbol{x}_i) + b) + \xi_i - 1), \tag{5}$$

where $\boldsymbol{\alpha} = [\alpha_1\ \alpha_2\ ...\ \alpha_l]^T$ is the Lagrangian multiplier vector, $\alpha_i\geq 0$.

The dual optimization problem of LSVM is:

$$\min_{0\leq\boldsymbol{\alpha}\in R^n} \quad \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{Q}\boldsymbol{\alpha} - \boldsymbol{e}^T\boldsymbol{\alpha}, \tag{6}$$

where $\boldsymbol{Q} = \frac{\boldsymbol{I}}{C} + \boldsymbol{H}\boldsymbol{H}^T$, $\boldsymbol{H} = \boldsymbol{D}[\boldsymbol{A} \quad -\boldsymbol{e}]$, $\boldsymbol{D} = diag(y_1, y_2, ..., y_l)$, $\boldsymbol{A} = [\boldsymbol{x}_1\ \boldsymbol{x}_2\ ...\ \boldsymbol{x}_l]^T$, $\boldsymbol{I}$ is the l-by-l identity matrix, $\boldsymbol{e}$ represents an l-dimensional column vector where all the elements are 1.

The iterative formula of LSVM is:

$$\boldsymbol{\alpha}^{i+1} = \boldsymbol{Q}^{-1}(((\boldsymbol{Q}\boldsymbol{a}^i - \boldsymbol{e}) - \lambda \boldsymbol{a}^i)_+ + \boldsymbol{e}),\ i = 0, 1, 2, \ldots,\ \lambda > 0, \tag{7}$$

where the symbol $(\boldsymbol{x})_+$ means setting the negative elements in the vector $\boldsymbol{x}$ as 0. If $0<\lambda<2/C$, given an arbitrary initial value $\boldsymbol{\alpha}^0$, Eq. 7 is always of global linear convergence.

4. The proposed anomaly detection algorithm - OLLSVM

The online learning algorithms in OLSVM include incremental learning and decremental learning, as well as the strategies for selecting sample(s) to be added or deleted, which are detailed as follows.

4.1. Incremental learning

Incremental learning includes single-sample and batch incremental learning designed for the situation of adding a single sample and a set of samples respectively.

(1) Single-sample incremental learning

Let $T = \{ (\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_l, y_l) \}$ be the initial training sample set; l is the number of samples. Assume that the newly added sample is $(\boldsymbol{x}_{l+1}, y_{l+1})$. When this new sample is added into T, α_{l+1} is first initialized as 0. This new sample is judged by the following KKT condition:

$$\begin{cases} \text{if } \alpha_i = 0, \quad y_i f(\boldsymbol{x}_i) \geq 1 \\ \text{if } \alpha_i > 0, \quad y_i f(\boldsymbol{x}_i) < 1 \end{cases}. \tag{8}$$

If this new sample satisfies $y_{l+1}f(\boldsymbol{x}_{l+1})\geq 1$, then $\alpha_{l+1} = 0$ matches the above KKT condition, thus $\boldsymbol{x}_{l+1}$ is a non-support vector. $\boldsymbol{x}_{l+1}$ has no effect on the currently constructed classifier. Therefore, $\boldsymbol{x}_{l+1}$ can be abandoned. If this new sample satisfies $y_{l+1}f(\boldsymbol{x}_{l+1})<1$, then $\alpha_{l+1} = 0$ does not match the above KKT condition, thus the classifier should be updated.

According to the iterative formula of LSVM (Eq. 7), after adding the new sample $(\boldsymbol{x}_{l+1}, y_{l+1})$ the classifier represented by the Lagrange multiplier vector $\boldsymbol{\alpha}_{new}$ can be computed as ([7]):

$$\boldsymbol{\alpha}_{new}^{i+1} = \boldsymbol{Q}_{new}^{-1}(((\boldsymbol{Q}_{new}\boldsymbol{a}_{new}^i - \boldsymbol{e}) - \lambda \boldsymbol{a}_{new}^i)_+ + \boldsymbol{e}),\ i = 0, 1, 2, \ldots,\ \lambda > 0, \tag{9}$$

where the initial value $\boldsymbol{a}_{new}^0 = \begin{bmatrix} \boldsymbol{a} \\ 0 \end{bmatrix}$, $\boldsymbol{\alpha}$ is the solution before adding $(\boldsymbol{x}_{l+1}, y_{l+1})$. Therefore, the key step for solving $\boldsymbol{\alpha}_{new}$ is to solve $\boldsymbol{Q}_{new}^{-1}$.

If T is linearly separable, $\boldsymbol{Q}_{new}^{-1}$ can be computed as ([7]):

$$\boldsymbol{Q}_{new}^{-1} = C\left(\boldsymbol{I} - \begin{bmatrix} \boldsymbol{H} \\ \boldsymbol{h} \end{bmatrix} \left(\boldsymbol{B} - \frac{\boldsymbol{B}\boldsymbol{h}^T\boldsymbol{h}\boldsymbol{B}}{1 + \boldsymbol{h}\boldsymbol{B}\boldsymbol{h}^T} \right) (\boldsymbol{H}^T \boldsymbol{h}^T) \right), \tag{10}$$

where $\boldsymbol{B}$ is the value of $\boldsymbol{Q}^{-1}$ obtained in the last computation; $\boldsymbol{h} = y_{l+1}[\boldsymbol{x}_{l+1}^T \quad -1]$. Therefore, by using the iterative formula Eq. 9, the new solution $\boldsymbol{\alpha}_{new}$ can be obtained according to the original solution $\boldsymbol{\alpha}$.

If T is nonlinearly separable, $\boldsymbol{Q}_{new}^{-1}$ can be computed as ([7]):

$$\boldsymbol{Q}_{new}^{-1} = \begin{bmatrix} \boldsymbol{Q}^{-1} + (\boldsymbol{Q}^{-1}\boldsymbol{b}\boldsymbol{b}^T\boldsymbol{Q}^{-1})/(d - \boldsymbol{b}^T\boldsymbol{Q}^{-1}\boldsymbol{b}) & -\boldsymbol{Q}^{-1}\boldsymbol{b}/(d - \boldsymbol{b}^T\boldsymbol{Q}^{-1}\boldsymbol{b}) \\ -\boldsymbol{b}^T\boldsymbol{Q}^{-1}/(d - \boldsymbol{b}^T\boldsymbol{Q}^{-1}\boldsymbol{b}) & 1/(d - \boldsymbol{b}^T\boldsymbol{Q}^{-1}\boldsymbol{b}) \end{bmatrix}, \tag{11}$$

where $d = \frac{1}{C} + K(\boldsymbol{g}, \boldsymbol{g}^T)$, $\boldsymbol{g} = [\boldsymbol{x}_{l+1}^T \quad -1]$, $\boldsymbol{b} = \boldsymbol{D}K(\boldsymbol{G}, \boldsymbol{g}^T)y_{l+1}$, $\boldsymbol{G} = [\boldsymbol{A} \quad -\boldsymbol{e}]$, K is the kernel function $K(\boldsymbol{x}, \boldsymbol{y}) = (\Phi(\boldsymbol{x})\cdot\Phi(\boldsymbol{y}))$.

The single-sample incremental learning algorithm first proposed in [7] is tailored as follows:

Algorithm 1: single-sample incremental learning algorithm in OLLSVM

Input: the initial training sample set $T = \{ (\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_l, y_l) \}$, where $\boldsymbol{x}_i \in R^n$, $y_i \in \Psi = \{1, -1\}$, $i = 1, 2, ..., l$; the newly added sample $(\boldsymbol{x}_{l+1}, y_{l+1})$.

Output: the classifier $LSVM_{l+1}$ represented by the new Lagrange multiplier vector $\boldsymbol{\alpha}_{new} = [\alpha_1\ \alpha_2\ ...\ \alpha_l\ \alpha_{l+1}]^T$.

Step 1: Given the parameter C, choose the parameter λ which satisfies $0<\lambda<2/C$; the precision $\varepsilon>0$; Train the detection model on T and obtain the initial classifier denoted as $LSVM_l$.

Step 2: Read the newly added sample $(\boldsymbol{x}_{l+1}, y_{l+1})$; Let $i = 0$, $\alpha_{l+1}^i = 0$.

Step 3: Judge $(\boldsymbol{x}_{l+1}, y_{l+1})$ by the KKT condition (Eq. 8). If this new sample matches KKT, then abandon it; the classifier $LSVM_l$ does not change. Otherwise, add $(\boldsymbol{x}_{l+1}, y_{l+1})$ into T and obtain a new training sample set; Compute $\boldsymbol{Q}_{new}^{-1}$ according to Eq. 10 (T is linearly separable) or Eq. 11 (T is nonlinearly separable).

Step 4: Compute $\boldsymbol{\alpha}_{new}^{i+1}$ according to the iterative formula (Eq. 9).

Step 5: If $\| \boldsymbol{\alpha}_{new}^{i+1} - \boldsymbol{\alpha}_{new}^{i} \| \le \varepsilon$, then $\boldsymbol{\alpha} = \boldsymbol{\alpha}_{new}^{i+1}$, this algorithm ends and the new classifier $LSVM_{l+1}$ is obtained. Otherwise, let $i = i+1$, turn to Step 4.

(2) Batch incremental learning

Assume T is the original training sample set, and the number of samples in T is l; the newly added training sample set is $\overline{T}$, and the number of samples in $\overline{T}$ is $\bar{l}$. According to the iterative formula of LSVM (Eq. 7), after adding the new training sample set $\overline{T}$ the classifier represented by the Lagrange multiplier vector $\boldsymbol{\alpha}_{new}$ can be computed as ([7]):

$$\boldsymbol{\alpha}_{new}^{i+1} = \boldsymbol{Q}_{new}^{-1}(((\boldsymbol{Q}_{new}\boldsymbol{a}_{new}^{i} - \boldsymbol{e}) - \lambda \boldsymbol{a}_{new}^{i})_{+} + \boldsymbol{e}),\ i = 0, 1, 2, \ldots,\ \lambda > 0, \tag{12}$$

where the initial value $\boldsymbol{a}_{new}^{0} = \begin{bmatrix} \boldsymbol{a} \\ \overline{\boldsymbol{\alpha}}^{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{0}_{\bar{l}\times 1} \end{bmatrix}$; $\boldsymbol{0}_{\bar{l}\times 1}$ represents a column vector where all the elements are 0; namely, in the initial value $\overline{\boldsymbol{\alpha}}^{0}$ the elements associated with the samples in $\overline{T}$ are set as 0; $\boldsymbol{\alpha}$ is the solution before adding $\overline{T}$. Therefore, the key step for solving $\boldsymbol{\alpha}_{new}$ is still to solve $\boldsymbol{Q}_{new}^{-1}$.

If T is linearly separable, and $\bar{l} \ge n+1$, then $\boldsymbol{Q}_{new}^{-1}$ is computed as ([7]):

$$\boldsymbol{Q}_{new}^{-1} = C\left(\boldsymbol{I} - \boldsymbol{H}_{new}\left(\frac{\boldsymbol{I}}{C} + \boldsymbol{H}^T\boldsymbol{H} + \overline{\boldsymbol{H}}^T\overline{\boldsymbol{H}} \right)^{-1} \boldsymbol{H}_{new}^T \right), \tag{13}$$

where $\overline{\boldsymbol{H}} = \overline{\boldsymbol{D}}[\overline{\boldsymbol{A}} \quad -\boldsymbol{e}]$, $\boldsymbol{H}_{new} = \begin{bmatrix} \boldsymbol{H} \\ \overline{\boldsymbol{H}} \end{bmatrix}$; $\overline{\boldsymbol{D}}$ and $\overline{\boldsymbol{A}}$ are the matrices $\boldsymbol{D}$ and $\boldsymbol{A}$ associated with $\overline{T}$. If $\bar{l} < n+1$, let $\boldsymbol{B}$ be the value of $\boldsymbol{Q}^{-1}$ obtained in the previous computation, then $\boldsymbol{Q}_{new}^{-1}$ can be computed as ([7]):

$$\boldsymbol{Q}_{new}^{-1} = C(\boldsymbol{I} - \boldsymbol{H}_{new}(\boldsymbol{B} - \boldsymbol{B}\overline{\boldsymbol{H}}^T(\boldsymbol{I} + \overline{\boldsymbol{H}}\boldsymbol{B}\overline{\boldsymbol{H}}^T)^{-1}\overline{\boldsymbol{H}}\boldsymbol{B})\boldsymbol{H}_{new}^T). \tag{14}$$

If T is nonlinearly separable, and $\bar{l} \ge l$, then $\boldsymbol{Q}_{new}^{-1}$ is computed as ([7]):

$$\boldsymbol{Q}_{new}^{-1}=\begin{bmatrix}(\boldsymbol{Q}-\boldsymbol{B}\boldsymbol{E}^{-1}\boldsymbol{B}^{T})^{-1} & -(\boldsymbol{Q}-\boldsymbol{B}\boldsymbol{E}^{-1}\boldsymbol{B}^{T})^{-1}\boldsymbol{B}\boldsymbol{E}^{-1}\\ -\boldsymbol{E}^{-1}\boldsymbol{B}^{T}(\boldsymbol{Q}-\boldsymbol{B}\boldsymbol{E}^{-1}\boldsymbol{B}^{T})^{-1} & \boldsymbol{E}^{-1}\boldsymbol{B}^{T}(\boldsymbol{Q}-\boldsymbol{B}\boldsymbol{E}^{-1}\boldsymbol{B}^{T})^{-1}\boldsymbol{B}\boldsymbol{E}^{-1}+\boldsymbol{E}^{-1}\end{bmatrix},$$

(15)

where $\boldsymbol{B}=\boldsymbol{D}K(\boldsymbol{G},\overline{\boldsymbol{G}}^{T})\overline{\boldsymbol{D}}$, $\boldsymbol{E}=\frac{\boldsymbol{I}}{C}+\overline{\boldsymbol{D}}K(\overline{\boldsymbol{G}},\overline{\boldsymbol{G}}^{T})\overline{\boldsymbol{D}}$. If $\bar{l}$ <*l*, then $\boldsymbol{Q}_{new}^{-1}$ is computed as ([7]):

$$\boldsymbol{Q}_{new}^{-1}=\begin{bmatrix}\boldsymbol{S} & -\boldsymbol{S}\boldsymbol{B}\boldsymbol{E}^{-1}\\ -\boldsymbol{E}^{-1}\boldsymbol{B}^{T}\boldsymbol{S} & \boldsymbol{E}^{-1}\boldsymbol{B}^{T}\boldsymbol{S}\boldsymbol{B}\boldsymbol{E}^{-1}+\boldsymbol{E}^{-1}\end{bmatrix}, \quad (16)$$

where $\boldsymbol{S}=(\boldsymbol{Q}-\boldsymbol{U}\boldsymbol{U}^{T})^{-1}$, $\boldsymbol{U}=\boldsymbol{B}\boldsymbol{E}_1^{T}$, $\boldsymbol{E}_1$ is obtained after Choleskey decomposing $\boldsymbol{E}^{-1}$, i.e., $\boldsymbol{E}^{-1}=\boldsymbol{E}_1^{T}\boldsymbol{E}_1$.

The batch incremental learning algorithm first proposed in [7] is tailored as follows:

Algorithm 2: batch incremental learning algorithm in OLLSVM

Input: the initial training sample set T = { ($\boldsymbol{x}_1$, y_1), ($\boldsymbol{x}_2$, y_2), …, ($\boldsymbol{x}_l$, y_l) }, where $\boldsymbol{x}_i \in R^n$, $y_i \in \Psi = \{1, -1\}$, i = 1, 2, ..., l. Assume T is divided into several mutually exclusive subsets $T_1 \cup T_2 \cup ... \cup T_N$; the scale of T_i ($1 \le i \le N$) can be determined according to different cases. The newly added sample set is $\overline{T}$, the number of samples in $\overline{T}$ is $\bar{l}$.

Output: the classifier $LSVM_{l+\bar{l}}$ represented by the new Lagrange multiplier vector $\boldsymbol{\alpha}_{new} = [\alpha_1\ \alpha_2\ ...\ \alpha_l\ \alpha_{l+1}\ ...\ \alpha_{l+\bar{l}}]^T$.

Step 1: Given the parameter C, choose the parameter λ which satisfies $0<\lambda<2/C$; the precision $\varepsilon>0$; Let k=1, train the detection model on T_k and obtain the initial classifier denoted as $LSVM_k$.

Step 2: Read $\overline{T}$; let each element of $\boldsymbol{\alpha}$ associated with each sample in $\overline{T}$ be zero; i=0; Judge whether each sample in $\overline{T}$ satisfies the KKT condition (Eq. 8) using the classifier $LSVM_k$.

2.1: If a new sample satisfies the KKT condition, the sample is abandoned.

2.2: Reserve all the new samples that do not satisfy the KKT condition, and include these samples into a new sample set $\overline{Tr}_{k+1}$. The number of samples in $\overline{Tr}_{k+1}$ is denoted as t_{k+1}, while the number of samples before increment is denoted as m_k. A new training sample set is obtained by adding $\overline{Tr}_{k+1}$ into the previous sample set before increment. The inverse of the new matrix $\boldsymbol{Q}_{k+1}$ is computed as follows:

2.2.1: If T is linearly separable, turn to Step 3;

2.2.2: If T is nonlinearly separable, turn to Step 4.

Step 3: If $t_{k+1} \ge n+1$, compute the inverse of $\boldsymbol{Q}_{k+1}$ by Eq. 13; otherwise compute the inverse of $\boldsymbol{Q}_{k+1}$ by Eq. 14.

Step 4: If $t_{k+1} < m_k$, compute the inverse of $\boldsymbol{Q}_{k+1}$ by Eq. 16; otherwise compute the inverse of $\boldsymbol{Q}_{k+1}$ by Eq. 15.

Step 5: Compute $\boldsymbol{\alpha}_{new}^{i+1}$ according to the iterative formula (Eq. 12) of LSVM, the initial value $\boldsymbol{\alpha}_{new}^{0}$ in each iteration is given by: the values of the elements associated with the previous samples are set as the optimal values obtained in the previous computation; while the values of the elements associated with the new incremental samples are set as zero.

Step 6: If $\| \boldsymbol{\alpha}_{new}^{i+1} - \boldsymbol{\alpha}_{new}^{i} \| \le \varepsilon$, then the new solution $LSVM_{k+1}$ is obtained; let $k = k+1$, and turn to Step 2 until all the subsets have been trained. Otherwise, let $i = i+1$, and turn to Step 5.

(3) The strategy for selecting sample(s) to be added

In large scale and high dynamic Cloud environment, the state of VMs is also high dynamic. In order to improve detection accuracy of the anomaly detection model, new training samples should be added timely into the training sample set, so as to update the detection model by the single-sample or batch incremental learning algorithm. The principles for selecting sample(s) to be added includes: real-time, authoritativeness, and representativeness.

1) Real-time: in order to capture the trend of VMs' performance, it is necessary to periodically select part of samples as new training samples, so as to train the detection model which can accurately characterize the performance or state of VMs.

2) Authoritativeness: part of detected samples can be submitted to the administrator for verification; the verified samples are authoritative; these samples should be added as training samples in priority.

3) Representativeness: in SVM-based anomaly detection algorithms, decision function of the detection model only depends on support vectors; therefore, the samples determined by the KKT condition as non-support vectors should be abandoned in incremental learning algorithms.

Therefore, this article designs the following strategies for selecting sample(s) to be added (the sampling interval is denoted as t_{int}):

1) For each VM, periodically select a fixed sample in $L \times t_{int}$ (e.g., L=10) time span, or randomly select a sample in $L \times t_{int}$ time span as a candidate sample;

2) For each VM, if in $L \times t_{int}$ time span, a certain sample is verified by the administrator, then replace the sample selected in 1) with this verified sample as a candidate sample;

3) If the selected sample is determined by the KKT condition as a non-support vector, then abandon it; otherwise, add it into the training sample set.

4.2. Decremental learning

Decremental learning also includes single-sample and batch decremental learning designed for the situation of deleting a single sample and a set of samples respectively.

(1) Single-sample decremental learning

Assume that the sample $(\boldsymbol{x}_k, y_k)$ will be deleted from the original training sample set T. The key step of updating the detection model is to compute the matrix $\boldsymbol{Q}$ denoted as $\boldsymbol{Q}_{l\text{-}1}$ after deleting $(\boldsymbol{x}_k, y_k)$. The matrix $\boldsymbol{Q}$ associated with T is denoted as $\boldsymbol{Q}_l$. $\boldsymbol{Q}_l$ is first transformed by elementary row and column transformation, so as to make the original k-th row and k-th column are exchanged to the first row and first column. The obtained matrix is denoted as $\boldsymbol{K}$. Let $\boldsymbol{U} = \boldsymbol{K}^{-1}$. $\boldsymbol{K}$ and $\boldsymbol{U}$ are partitioned as:

$$\boldsymbol{K} = \begin{bmatrix} k_{11} & \boldsymbol{k}^T \\ \boldsymbol{k} & \boldsymbol{Q}_{l-1} \end{bmatrix}, \ \boldsymbol{U} = \begin{bmatrix} u_{11} & \boldsymbol{u}^T \\ \boldsymbol{u} & \boldsymbol{U}_{l-1} \end{bmatrix}. \tag{17}$$

Then $\boldsymbol{Q}_{l\text{-}1}^{-1}$ can be computed as ([8]):

$$\boldsymbol{Q}_{l-1}^{-1} = \boldsymbol{U}_{m-1} - \frac{\boldsymbol{u}\boldsymbol{u}^T}{u_{11}}. \tag{18}$$

According to the iterative formula of LSVM (Eq. 7), after deleting $(\boldsymbol{x}_k, y_k)$ the classifier represented by the Lagrange multiplier vector $\boldsymbol{\alpha}_{new}$ can be computed as ([8]):

$$\boldsymbol{\alpha}_{new}^{i+1} = \boldsymbol{Q}_{l-1}^{-1}((\boldsymbol{Q}_{l-1}\boldsymbol{a}_{new}^{i} - \boldsymbol{e}) - \lambda \boldsymbol{a}_{new}^{i})_{+} + \boldsymbol{e}), \ i = 0, 1, 2, \ldots, \ \lambda > 0. \tag{19}$$

Accordingly, the single-sample decremental learning algorithm ([8]) is listed as follows.

Algorithm 3: single-sample decremental learning algorithm in OLLSVM

Input: the initial training sample set $T = \{ (\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_l, y_l) \}$, where $\boldsymbol{x}_i \in R^n$, $y_i \in \Psi = \{1, -1\}$, $i = 1, 2, ..., l$; the sample to be deleted, $(\boldsymbol{x}_k, y_k)$.

Output: the classifier $LSVM_{l\text{-}1}$ represented by the new Lagrange multiplier vector $\boldsymbol{\alpha}_{new} = [\alpha_1 \ \alpha_2 \ ... \ \alpha_{l\text{-}1}]^T$.

Step 1: Given the parameter C, choose the parameter λ which satisfies $0<\lambda<2/C$; the precision $\varepsilon>0$; Train the detection model on T and obtain the initial classifier denoted as $LSVM_l$. Read the sample to be deleted, $(\boldsymbol{x}_k, y_k)$.

Step 2: Let $\boldsymbol{\alpha}_{new}^0 = \boldsymbol{\alpha} \setminus \{k\}$, which means deleting the k-th element from $\boldsymbol{\alpha}$. $\boldsymbol{\alpha}$ is the optimal solution before deleting $(\boldsymbol{x}_k, y_k)$; let $i = 0$.

Step 3: Compute the inverse matrix of $\boldsymbol{Q}_{l\text{-}1}$ according to Eq. 18.

Step 4: Compute $\boldsymbol{\alpha}_{new}^{i+1}$ according to Eq. 19.

Step 5: If $\| \boldsymbol{\alpha}_{new}^{i+1} - \boldsymbol{\alpha}_{new}^{i} \| \le \varepsilon$, then the new solution is obtained; otherwise, let $i = i+1$, and turn to Step 4.

(2) Batch decremental learning

Assume that the subset of training samples $\{ (\boldsymbol{x}_{\iota(1)}, y_{\iota(1)}), (\boldsymbol{x}_{\iota(2)}, y_{\iota(2)}), \ldots, (\boldsymbol{x}_{\iota(k)}, y_{\iota(k)}) \}$ will be deleted from T. $\boldsymbol{Q}_l$ is first transformed by elementary row and column transformation, so as to make the original k rows and k columns associated with k samples to be deleted are exchanged to the former k rows and k columns. The obtained matrix is denoted as $\hat{\boldsymbol{K}}$. Let $\boldsymbol{U} = \hat{\boldsymbol{K}}^{-1}$. $\hat{\boldsymbol{K}}$ and $\boldsymbol{U}$ are partitioned as:

$$\hat{\boldsymbol{K}} = \begin{bmatrix} \hat{\boldsymbol{Q}}_{11} & \hat{\boldsymbol{Q}}_{12} \\ \hat{\boldsymbol{Q}}_{12}^T & \boldsymbol{Q}_{l-k} \end{bmatrix}, \ \boldsymbol{U} = \begin{bmatrix} \boldsymbol{U}_{11} & \boldsymbol{U}_{12} \\ \boldsymbol{U}_{12}^T & \boldsymbol{U}_{l-k} \end{bmatrix}. \tag{20}$$

Then $\boldsymbol{Q}_{l\text{-}k}^{-1}$ can be computed as ([8]):

$$\boldsymbol{Q}_{l-k}^{-1} = \boldsymbol{U}_{l-k} - \boldsymbol{U}_{12}^T \boldsymbol{U}_{11}^{-1} \boldsymbol{U}_{12}. \tag{21}$$

According to the iterative formula of LSVM (Eq. 7), after deleting $\{ (\boldsymbol{x}_{\iota(1)}, y_{\iota(1)}), (\boldsymbol{x}_{\iota(2)}, y_{\iota(2)}), \ldots, (\boldsymbol{x}_{\iota(k)}, y_{\iota(k)}) \}$ the classifier represented by the Lagrange multiplier vector $\boldsymbol{\alpha}_{new}$ can be computed as ([8]):

$$\boldsymbol{\alpha}_{new}^{i+1} = \boldsymbol{Q}_{l-k}^{-1} (((\boldsymbol{Q}_{l-k} \boldsymbol{a}_{new}^{i} - \boldsymbol{e}) - \lambda \boldsymbol{a}_{new}^{i})_+ + \boldsymbol{e}), \ i = 0, 1, 2, \ldots, \ \lambda > 0. \tag{22}$$

where $\boldsymbol{a}_{new}^0$ represents the vector obtained through deleting k elements (from $\boldsymbol{\alpha}$) associated with the k samples to be deleted. $\boldsymbol{\alpha}$ is the optimal solution before deleting these samples.

Accordingly, the batch decremental learning algorithm ([8]) is listed as follows.

Algorithm 4: batch decremental learning algorithm in OLLSVM

Input: the initial training sample set $T = \{ (\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_l, y_l) \}$, where $\boldsymbol{x}_i \in R^n$, $y_i \in \Psi = \{1, -1\}$, $i = 1, 2, ..., l$; the subset of samples to be deleted, $\{ (\boldsymbol{x}_{\iota(1)}, y_{\iota(1)}), (\boldsymbol{x}_{\iota(2)}, y_{\iota(2)}), \ldots, (\boldsymbol{x}_{\iota(k)}, y_{\iota(k)}) \}$.

Output: the classifier $LSVM_{l\text{-}k}$ represented by the new Lagrange multiplier vector $\boldsymbol{\alpha}_{new} = [\alpha_1 \ \alpha_2 \ ... \ \alpha_{l\text{-}k}]^T$.

Step 1: Given the parameter C, choose the parameter λ which satisfies $0<\lambda<2/C$; the precision $\varepsilon>0$; Train the detection model on T and obtain the initial classifier denoted as $LSVM_l$. Read the subset of samples to be deleted.

Step 2: Let $\boldsymbol{\alpha}_{new}^0 = \boldsymbol{\alpha} \setminus \{\iota(1), \cdots, \iota(k)\}$, which means deleting k elements (from $\boldsymbol{\alpha}$) whose subscripts are $\iota(1)$, $\iota(1)$, ..., $\iota(k)$; let $i = 0$.

Step 3: Compute the inverse matrix of $\boldsymbol{Q}_{l\text{-}k}$ according to Eq. 21.

Step 4: Compute $\boldsymbol{\alpha}_{new}^{i+1}$ according to Eq. 22.

Step 5: If $\| \boldsymbol{\alpha}_{new}^{i+1} - \boldsymbol{\alpha}_{new}^{i} \| \le \varepsilon$, then the new solution is obtained; otherwise, let $i = i+1$, and turn to Step 4.

(3) The strategy for selecting sample(s) to be deleted

The above decremental learning algorithms only direct how to update the detection model when deleting sample(s) from the training sample set. As for the strategy for

selecting sample(s) to be deleted, some researches simply delete non-support vectors. However, non-support vectors may become support vectors after updating the detection model ([7]). These samples may provide important information in later. Deleting these samples may reduce the accuracy of anomaly detection.

Support vectors locate on the boundaries $g(\boldsymbol{x}) = \pm 1$, or in the side of misclassification. The support vectors are considered as equally important. Non-support vectors locate on the boundaries $g(\boldsymbol{x}) = \pm 1$, or in the side of correct classification. The farther the distance between non-support vectors and the classification hyperplane $g(\boldsymbol{x}) = 0$, the lower the importance of these samples in the current detection model. After training or updating the detection model every time, the distance between the boundary and the classification hyperplane is denoted as r; the importance of non-support vectors $\boldsymbol{x}_i$ can be measured according to the distance between $\boldsymbol{x}_i$ and the classification hyperplane. Specifically, this article defines the following importance factor $f_{imp}(\boldsymbol{x}_i)$ for each sample $\boldsymbol{x}_i$ in the training sample set. The $f_{imp}(\boldsymbol{x}_i)$ value has been normalized to [0, 1]. The smaller the $f_{imp}(\boldsymbol{x}_i)$ value, the lower the importance of $\boldsymbol{x}_i$.

$$f_{imp}(\boldsymbol{x}_i) = \begin{cases} 1, & \boldsymbol{x}_i \text{ is a support vector} \\ r / r_i, & \boldsymbol{x}_i \text{ is a non-support vector} \end{cases}. \tag{23}$$

In addition, since the support vectors are dynamically changed along with the update of the detection model, this article computes the average importance factor value $\bar{f}_{imp}(\boldsymbol{x}_i)$ for each sample $\boldsymbol{x}_i$.

This article proposes the following strategies for selecting sample(s) to be deleted: record the average importance factor value $\bar{f}_{imp}(\boldsymbol{x}_i)$ for each sample in the training sample set and take it as the criterion to delete sample(s); when it is necessary to delete sample(s), select the sample $\boldsymbol{x}_k$ with the smallest $\bar{f}_{imp}(\boldsymbol{x}_i)$ value or a subset $\{ (\boldsymbol{x}_{I(1)}, y_{I(1)}), (\boldsymbol{x}_{I(2)}, y_{I(2)}), \ldots, (\boldsymbol{x}_{I(k)}, y_{I(k)}) \}$ with the smallest $\bar{f}_{imp}(\boldsymbol{x}_i)$ values.

4.3. The proposed OLLSVM algorithm

Combined with the above incremental / decremental learning algorithms and strategies, a complete anomaly detection algorithm based on OLLSVM is formed. The concrete steps of OLLSVM are listed as follows.

Algorithm 5: the anomaly detection algorithm based on OLLSVM

Input: the initial training sample set $T = \{ (\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_l, y_l) \}$, where $\boldsymbol{x}_i \in R^n$, $y_i \in \Psi = \{1, -1\}$, $i = 1, 2, ..., l$; the newly collected sample, $\boldsymbol{x}_{new}$; the capacity limit of the training sample set, V_T.

Output: the classifier represented by the Lagrange multiplier vector, $\boldsymbol{\alpha}_{new}$; the label y_{new} corresponding to the new sample.

Step 1: Train the detection model on the initial training sample set T, obtain the initial classifier denoted as $LSVM_l$.

Step 2: Compute $y_{new} = f(\boldsymbol{x}_{new})$, where $f(\boldsymbol{x})$ is the decision function associated with $LSVM_l$; if y_{new} equals to +1, the new sample $\boldsymbol{x}_{new}$ is an abnormal state; otherwise, it is a normal state.

Step 3: According the designed strategies for selecting sample(s) to be added, determine whether or not add the sample $(\boldsymbol{x}_{new}, y_{new})$ into T, or add a newly accumulated sample subset $\overline{T}$ into T. The new training sample set is also denoted as T.

Step 4: If a single sample is added, update the detection model using the single-sample incremental learning algorithm; else update the detection model using batch incremental learning algorithm. The Lagrange multiplier vector of the updated detection model is denoted as $\boldsymbol{\alpha}_{new}$.

Step 5: Let the number of samples in the new training sample set T be l'. Compute or

update the $f_{imp}(\boldsymbol{x})$ and $\bar{f}_{imp}(\boldsymbol{x}_i)$ values of each sample in T according to Eq. 23.

Step 6: If $l' > V_T$, select a single or a set of samples to be deleted according to the proposed strategies.

Step 7: Update the detection model according to the single-sample or batch decremental learning algorithm, the Lagrange multiplier vector of the updated detection model is also denoted as $\boldsymbol{\alpha}_{new}$.

5. Experiments and analyses

5.1. Datasets

This article conducts experiments on Cloud datasets and KDD Cup datasets to evaluate the effectiveness of the proposed OLLSVM.

(1) Cloud datasets

In Cloud platforms, the performance or state of VMs can be characterized by a set of performance metrics including CPU utilization rate, available memory size, network traffic. These metrics indicate the heath state of a VM. The anomaly detection model determines the state of a VM as normal or abnormal based on a classifier learnt from a training sample set. This article first constructs an institute-wide Cloud platform and collects samples of performance metric data of VMs. The performance metric set contains 53 performance metrics, which can be classified into five categories: computation, storage, disk I/O, process, and network.

In addition, in order to construct the real datasets including abnormal samples and normal samples, four types of anomalies are injected to randomly selected VMs in the Cloud Platform, which are listed as follows.

1) Anomalous consumption of computation resource: A computing-intensive program runs in a VM, and persistently over-consumes computation resources.

2) Anomalous consumption of memory resource: A running program in a VM continues to apply for dynamic memory through malloc() function without releasing the assigned memory, which causes memory leak and over-consumption of the VM's memory resources.

3) Anomalous disk I/O operation: A running program in a VM persistently reads large files on disk, generating large amounts of disk I/O operations to simulate anomalous disk I/O.

4) Anomalous network access: Several users run LoadRunner from their own physical personal computer to simultaneously access a Web application server deployed on a VM, generating a large number of HTTP connections to simulate anomalous network behavior.

Based on the above methods, the required real datasets are constructed for experiments.

(2) KDD Cup datasets

KDD Cup 1999 dataset ([19]) is a benchmark dataset for testing and evaluating algorithms in anomaly detection, intrusion detection, data mining, etc. KDD Cup dataset is released by the fifth International Conference on Knowledge Discovery and Data Mining (KDD). KDD Cup dataset contains a wide variety of intrusions. It is simulated and collected in a military network environment. Each sample in this dataset represents a connection record from a source host to a destination host during a complete session. Each sample contains 41 metrics including duration time of the session, protocol type, bytes from the source host to the destination host, bytes from the destination host to the source host.

KDD Cup dataset contains a training dataset, which is collected from 7 weeks of network traffic. There are totally 4,898,431 samples in this training dataset. KDD Cup dataset also contains a test dataset, which is collected from 2 weeks of network traffic. There are totally 2,984,154 samples in this test dataset. Each connection (i.e., a sample) is labeled as either normal, or as an attack (Probe, DOS, U2R, or R2L).

5.2. Experimental results and analyses

This article tests the performance of OLLSVM compared with traditional SVM and LSVM in terms of both detection accuracy and time efficiency.

This article adopts four performance measures: *Sensitivity*, *Specificity*, *Precision*, and *F1-Measure*, which are defined as follows (F_P, False Positive; F_N, False Negative; T_P, True Positive; T_N, True Negative).

1) Sensitivity: the proportion of True Positive to the number of actual abnormal states.

$$Sensitivity = T_P /(T_P + F_N) \tag{24}$$

2) Specificity: the proportion of True Negative to the number of actual normal states.

$$Specificity = T_N /(F_P + T_N) \tag{25}$$

3) Precision: the proportion of True Positive to the number of detected positives.

$$Precision = T_P /(F_P + T_P) \tag{26}$$

4) F1-Measure:

$$F_1 = 2PR /(P + R), \tag{27}$$

where P is Precision, R is Recall which is defined as $T_P /(F_N + T_P)$ (i.e., Recall is equivalent with Sensitivity). Namely, F1-Measure is the harmonic mean of precision and recall.

For Cloud datasets, an initial training sample set including 5000 samples is constructed. The capacity (i.e., V_T) of the training sample set is 5200 (samples). The detection models based on SVM, LSVM, and OLLSVM are trained on this training sample set. A sample from every 10 test samples is randomly selected as the newly added training sample to simulate the situation of increasing number of training samples. The strategies for selecting sample(s) to be added are slightly adjusted. Each of the selected samples is added into the training sample set, regardless of whether or not it is a support vector. When the number of training samples reaches the limit of capacity, a training sample is deleted each time from the training sample set during the decremental learning process.

At the same time, a test sample set including 5000 samples is constructed. Table 1 lists performance measure results of these three algorithms on this test sample set.

The experimental results show that, the detection accuracy of SVM is the smallest among these three algorithms. Specially, LSVM achieves the highest detection accuracy, since LSVM retrains the detection model each time when adding or deleting sample(s). Although OLLSVM only updates the detection model each time according to the added or deleted sample(s), OLLSVM still achieves almost the same high accuracy as that of LSVM.

Table 1. Performance measure results of 3 algorithms on Cloud datasets

Algorithm	T_P	F_P	F_N	T_N	Sensitivity	Specificity	Precision	F1-Measure
SVM	2409	134	91	2366	0.964	0.946	0.947	0.955
LSVM	2433	106	67	2394	0.973	0.958	0.958	0.966
OLLSVM	2427	111	73	2389	0.971	0.956	0.956	0.963

This article also conducts experiments on KDD Cup datasets. The settings of the training sample set, the test sample set (also including 5000 samples), and the adopted strategies are the same as that in experiments on Cloud datasets. Table 2 lists performance measure results of these three algorithms on this test sample set. The experimental results also show that, LSVM achieves the highest detection accuracy, while OLLSVM achieves almost the same high accuracy as that of LSVM.

Table 2. Performance measure results of 3 algorithms on KDD Cup datasets

Algorithm	T_P	F_P	F_N	T_N	Sensitivity	Specificity	Precision	F1-Measure

SVM	2404	139	96	2361	0.962	0.944	0.945	0.953
LSVM	2428	113	72	2387	0.971	0.955	0.956	0.963
OLLSVM	2421	116	79	2384	0.968	0.954	0.954	0.961

Table 3 and Table 4 list time efficiency results in terms of the average time for retraining or updating the detection model when adding or deleting a training sample on Cloud datasets and KDD Cup datasets respectively. Averagely, it takes LSVM about 150 ms to retrain the detection model, while it takes OLLSVM only about 13 ms to update the detection model. Thus, OLLSVM shows great advantage in time efficiency compared with SVM and LSVM.

Table 3. Time efficiency results of 3 algorithms on Cloud datasets

Algorithm	Adding a training sample	deleting a training sample
SVM	356.3 ms	358.2 ms
LSVM	157.1 ms	158.6 ms
OLLSVM	13.5 ms	13.7 ms

Table 4. Time efficiency results of 3 algorithms on KDD Cup datasets

Algorithm	Adding a training sample	deleting a training sample
SVM	339.6 ms	341.7 ms
LSVM	144.7 ms	146.3 ms
OLLSVM	12.9 ms	13.1 ms

From the above experimental results, it is concluded that original LSVM achieves high performance measures since the detection model is retrained each time when adding a new sample or deleting a selected sample; yet the time efficiency of LSVM is relatively low. Although the detection accuracy of OLLSVM is slightly lower than that of LSVM (but also acceptable), the time efficiency of OLLSVM apparently outperforms that of LSVM. Therefore, OLLSVM can better meet the real-time requirements of anomaly detection under production Cloud platforms.

6. Conclusion and future work

In order to improve the dependability of Cloud platforms and cope with the challenge of increasing number of training samples at the same time, this article proposes an anomaly detection algorithm based on online learning Lagrangian SVM (termed OLLSVM). This article introduces and tailors the existing online learning algorithms of LSVM, as well as designs the strategies of selecting sample(s) to be added or deleted, so as to form a complete algorithm for anomaly detection. The experimental results verify that OLLSVM achieves evident improvement in time efficiency compared with LSVM.

The future work of this article will focus on introducing the proposed algorithms and strategies into other SVM-based anomaly detection algorithms, thus time efficiency is expected to be improved.

Acknowledgments

The authors are grateful to the editor and anonymous reviewers for their valuable comments on this article.

The research of this article is supported by National Natural Science Foundation of China (Grant No. 61272399 and No. 61572090), Chongqing Research Program of Basic Research and Frontier Technology (Grant No. cstc2015jcyjBX0014 and No. cstc2016jcyjA0304), and the Scientific and Technological Research Program of

Chongqing Municipal Education Commission (Grant No. KJ1500538 and No. KJ1600521).

References

[1] A. Avizienis, J. C. Laprie, B. Randell, and C. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing", IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 1, (**2004**), pp. 11-33.
[2] K. Joshi, G. Bunker, F. Jahanian, A. Moorsel, and Joseph Weinman, "Dependability in the Cloud: Challenges and Opportunities", Proceedings of the 39th IEEE/IFIP International Conference on Dependable Systems & Networks (DSN), (**2009**), pp. 103-104.
[3] H. S. Pannu, J. G. Liu, and S. Fu, "A Self-evolving Anomaly Detection Framework for Developing Highly Dependable Utility Clouds", Proceedings of IEEE Global Communications Conference, (**2012**), pp. 1605-1610.
[4] A. B. Sharma, H. F. Chen, M. Ding, K. J. Yoshihira, and G. F. Jiang, "Fault Detection and Localization in Distributed Systems Using Invariant Relationships", Proceedings of 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks, (**2013**), pp. 1-8.
[5] O. L. Mangasaian and D. R. Musicant, "Lagrangian Support Vector Machines", Journal of Machine Learning Research, vol. 1, no. 3, (**2001**), pp. 161-177.
[6] O. L. Mangasaian and D. R. Musicant, "Lagrangian Support Vector Machine", http://research.cs.wisc.edu/dmi/lsvm/.
[7] H. Duan, X. J. Shao, W. Z. Hou, G. P. He, and Q. T. Zeng, "An Incremental Learning Algorithm for Lagrangian Support Vector Machines", Pattern Recognition Letters, vol. 30, no. 15, (**2009**), pp. 1384-1391.
[8] H. Duan, H. Li, G. P. He, and Q. T. Zeng, "Decremental Learning Algorithms for Nonlinear Lagrangian and Least Squares Support Vector Machines", Proceedings of the First International Symposium on Optimization and Systems Biology, (**2007**), pp. 358-366.
[9] M. F. Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do We Need Hundreds of Classifiers to Solve Real World Classification Problems? ", Journal of Machine Learning Research, vol. 15, (**2014**), pp. 3133-3181.
[10] C. Cortes and V. N. Vapnik, "Support-vector Networks", Machine Learning, vol. 20, no. 3, (**1995**), pp. 273-279.
[11] C. C. Chang and C. J. Lin, "LIBSVM: A Library for Support Vector Machines", ACM Transactions on Intelligent Systems and Technology, vol. 2, no. 3, (**2011**), Article 27.
[12] S. Yin, X. P. Zhu, and C. Jing, "Fault Detection Based on a Robust One Class Support Vector Machine", Neurocomputing, vol. 145, (**2014**), pp. 263-268.
[13] A. K. Marnerides, S. Malinowski, R. Morla, and H.S. Kim, "Fault Diagnosis in DSL Networks Using Support Vector Machines", Computer Communications, vol. 62, (**2015**), pp. 72-84.
[14] S. Fu, J. G. Liu, and H. Pannu, "A Hybrid Anomaly Detection Framework in Cloud Computing Using One-Class and Two-Class Support Vector Machines", Proceedings of 8th International Conference on Advanced Data Mining and Applications (ADMA), (**2012**), pp. 726-738.
[15] S. Parsa and S. A. Naree, "A New Semantic Kernel Function for Online Anomaly Detection of Software", ETRI Journal, vol. 34, no. 2, (**2012**), pp. 288-291.
[16] N. Syed, H. Liu and K. Sung, "Incremental Learning with Support Vector Machines", Proceedings of International Joint Conference on Artificial Intelligence (IJCAI), Workshop on Support Vector Machines, (**1999**).
[17] G. Cauwenberghs and T. Poggio, "Incremental and Decremental Support Vector Machine Learning", Proceedings of Advances in Neural Information Processing Systems (NIPS), (**2000**).
[18] J. Zheng, F. R. Shen, H. J. Fan, and J. X. Zhao, "An Online Incremental Learning Support Vector Machine for Large-Scale Data", Neural Computing & Applications, vol. 22, (**2013**), pp. 1023-1035.
[19] S. Hettich and S. D. Bay, "The UCI KDD Archive", (**1999**). http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

