PLOS ONE

RESEARCH ARTICLE

An incremental anomaly detection model for virtual machines

Hancui Zhang[1]*, Shuyu Chen[1], Jun Liu[3], Zhen Zhou[4], Tianshu Wu[2]

1 College of Software Engineering, Chongqing University, Chongqing, China, **2** College of Computer Science, Chongqing University, Chongqing, China, **3** College of Software Engineering, Chongqing University of Posts and Telecommunications, Chongqing, China, **4** School of computer science and technology, Southwest Minzu University, Chengdu, China

* zhc_813@126.com

Citation: Zhang H, Chen S, Liu J, Zhou Z, Wu T (2017) An incremental anomaly detection model for virtual machines. PLoS ONE 12(11): e0187488. https://doi.org/10.1371/journal.pone.0187488

Editor: Yong Deng, Southwest University, CHINA

Received: October 17, 2016

Accepted: August 30, 2017

Published: November 8, 2017

Data Availability Statement: The KDD Cup 1999 data set is available at http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

Funding: The authors are grateful to the editor and anonymous reviewers for their valuable comments on this paper. The work of this paper is supported by National Natural Science Foundation of China (Grant No. 61272399 and No. 61572090), the Science and Technology Research Project of Chongqing Municipal Education Committee (Grant No. KJ1704081), and Research Fund for the Doctoral Program of Higher Education of China (Grant No. 20110191110038). The funders had no role in study design, data collection and analysis,

Abstract

Self-Organizing Map (SOM) algorithm as an unsupervised learning method has been applied in anomaly detection due to its capabilities of self-organizing and automatic anomaly prediction. However, because of the algorithm is initialized in random, it takes a long time to train a detection model. Besides, the Cloud platforms with large scale virtual machines are prone to performance anomalies due to their high dynamic and resource sharing characters, which makes the algorithm present a low accuracy and a low scalability. To address these problems, an Improved Incremental Self-Organizing Map (IISOM) model is proposed for anomaly detection of virtual machines. In this model, a heuristic-based initialization algorithm and a Weighted Euclidean Distance (WED) algorithm are introduced into SOM to speed up the training process and improve model quality. Meanwhile, a neighborhood-based searching algorithm is presented to accelerate the detection time by taking into account the large scale and high dynamic features of virtual machines on cloud platform. To demonstrate the effectiveness, experiments on a common benchmark KDD Cup dataset and a real dataset have been performed. Results suggest that IISOM has advantages in accuracy and convergence velocity of anomaly detection for virtual machines on cloud platform.

Introduction

As cloud computing continues to develop, cloud platform based on virtualization technology is becoming increasingly popular in the fields of medicine, biology, geology and scientific computing and so on. The scale of virtual machines in cloud platform is continuously growing, and the applications deployed on virtual machines are more and more complex. At the same time, competition for resources in the cloud platform, resource sharing, virtual machine overload are prone to cause abnormalities which will make a part of the virtual machines downtime and will affect the reliability and availability of the entire cloud platform seriously. Therefore, it is highly desirable to provide an effective anomaly detection algorithm for virtual machines in cloud platform [1–4].

decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

Nowadays, statistics, clustering, classification, and nearest-neighbor algorithms are commonly used methods of detecting abnormal [5]. Method under a certain probability model is based on statistical anomaly detection algorithm, nevertheless, a hypothetical probability model has to be predefined [6]. SVM-based [7] algorithms, Bayes networks [8] and neural networks [9] are the main several classification algorithms for anomaly detection used to learn a classify model (classifier) from the training data (labeled data instances) and then to predict a test instance using the classifier. Here, classification-based techniques rely on various labeled instances, which is hard to get and be labeled in real problems. Moreover, for multi-class classification, to get a variety of labeled normal classes is almost impossible. Clustering is an unsupervised learning method, which can find groupings in input instances without any prior knowledge [10–12]. Therefore, comparatively, clustering is a useful tool in dealing with a large, complex and unforeseen dataset with many variables and unknown structures.

Self-Organizing Map (SOM) algorithm is a clustering and data analysis algorithm proposed by Kohonen in 1982 [13, 14]. Since that time, it has been widely used and the improvement of the algorithm has been going on. In order to solve the initialization problem of the size of SOM network, Growing SOM algorithm [15]has been proposed. The main idea is to form a single n-dimensional space (e.g. triangle, tetrahedron) as the basic building blocks in the training process of generating dynamic SOM network. Meanwhile, for the purpose of achieving SOM algorithm rapidly, TS-SOM (Tree Structured Sel-Organizing Maps) [16–18] method was proposed to realize the SOM by using the recursive nature of the tree structure itself. Furthermore, in order to address the problem of SOM network size and parameter optimization, Polani [19], Kubota R [20], and Deep K [21]et al. applied Genetic Algorithm to evolve SOM network to figure out the fitness function which optimized SOM network, to get the optimal solution of the initial size of SOM network, and to train the training neighborhood of the SOM network by the optimized function. Due to the characteristics of non-supervisory, nonlinear, lower complexity and mapping high-dimensional data into low-dimensional space, SOM has been widely used in pattern recognition [22, 23], biology [24, 25], medical applications [26], climate [27, 28], neural networks [29, 30], image analysis [31] and other fields.

However, the application of SOM in anomaly detection for virtual machine is still in its infancy. Hoz E D L [32] et al. presented a Growing Hierarchical SOM (GHSOM) for network intrusion detection on the basis of NSGA-II algorithm for feature selection to reduce the complexity of SOM. A new SOM map arose or not in GHSOM need to be calculated with the quantization error of each unit. Although it reduced the computational complexity of dataset dimension as a result of feature selection, a lot of calculations and comparisons were still required and the training time increase. In addition, they proposed a Probabilistic SOM (PSOM) approach that hybridizes statistical technique and SOM for network intrusion detection where Principal Component Analysis (PCA) and Fisher Discriminant Ratio (FDR) were considered for feature selection and noise removal [33]. In PSOM, the SOM map was trained by a Gaussian Mixture Model (GMM) first, while a further tuning of the map was achieved by calculating the prior probabilities and posterior probability of the SOM units. Consequently, PSOM would spend a lot of training time and computing costs on the input samples to achieve a good classification result. When in case of the large scale and high dynamic of virtual machines under the cloud platform, they would appear a high false positive rate and a low accuracy rate. Jun L [34] et al. applied Self-Organizing Maps method for anomaly detection of cloud platform and presented a unified modeling method based on SOM with a detection region. However, they did not optimize the SOM algorithm and it significant affected the accuracy and speed of detection. Wang H [35] et al. proposed a Simulated Annealing-based SOM (SOMSA) method for intrusion detection. The Simulated Annealing (SA) algorithm was used to refine the weight of SOM to get the optimal point by a form of probability. Nevertheless, SA

is a random search algorithm. In each process of refining, the chose of best matched neuron and its neighborhoods is performed in random with probability. As a result, this method achieved the accuracy at the cost of training time due to the characteristic of random search. Based on the process units of anomaly detection, Song Y [36] et al. proposed a Statistic Pattern-based SOM (SP-based SOM) method, which had advantages on a small-scale system. Whereas, because of the feature of complex and dynamic of cloud platform and a large amounts of performance metrics, this method was not suitable for cloud platform with large-scale virtual machines. Moreover, Song Y et al. did not optimize the initialization and the training neighborhood of SOM, which significantly affects the speed of modeling SOM network.

To address these problems, this paper presents an Improved Incremental SOM (IISOM) anomaly detection algorithm for virtual machines on cloud platform, in which a heuristic SOM algorithm has been proposed to initialize SOM network and a Weighted Euclidean Distance (WED) method has been used to improve the training neighborhood. As mentioned above, Wang H [35] et al. utilized the Simulated Annealing-based SOM algorithm to obtain global optimal solution, which has advantages in improving the accuracy of anomaly detection for virtual machines on cloud platform. But due to the expense of training time, it is not functional for anomaly detection of virtual machines with high dynamic. Here, in order to obtain a global optimal solution and reduce the training time, IISOM introduces the heuristic-based initialization algorithm to consider the similarity of input data instance and the characteristics of SOM convergence to estimate the weight of SOM, which has the advantages in global optimal clustering the same as SOMSA. Moreover, IISOM is superior to SOMSA subjected to detection accuracy and convergence rate as well as the quality of model. The effectiveness of the proposed method can be substantiated by systematic analysis associated with the benchmark dataset KDD Cup and real dataset. It can be found that the IISOM has a much higher training speed and accuracy than the traditional SOM algorithm and SOMSA algorithm.

The rest of this paper is organized as follows. In Materials and Methods, the traditional SOM algorithm and the SOMSA algorithm are described. And according to the lacks of traditional SOM and SOMSA method applied for anomaly detection of virtual machines in cloud platform, the optimistic IISOM method is proposed. Then carry out the experiments to show the performance evaluation. Advantages and limitations of the IISOM are discussed in the discussion part and also we give some expectations for the future work. And finally, conclude this paper.

Materials and methods

Traditional SOM algorithm

Self-Organizing Map (SOM) is a popular neural network model that using unsupervised learning rules to analyze, cluster, and model various datasets. Commonly, it is used to map a high dimension input space into a low dimensional discrete map space. Meanwhile it preserves the topological properties of the original input space [37, 38].

Generally, SOM consists of two layers: the input layer, denoting the runtime measurement vectors, and the output layer, usually consisting of a two-dimensional lattice type of neurons, illustrated by Fig 1. The training of SOM network is usually done in two phases: ordering or self-organizing phase to get a rough training order and then convergence phase to fine-tune the map and to provide the ability for detection. Here we give a brief introduction.

Suppose the input vector of runtime measurements is $x(t) = [x_1, x_2, x_3, \ldots, x_d] \in R^D$, and there are N^*N neurons in the output space, a two-dimensional lattice network, in which each neuron n_{uv} is associated with a coordinate(u, v), $1 \leq u, v \leq N$ and a weight vector W_{uv}. Here,

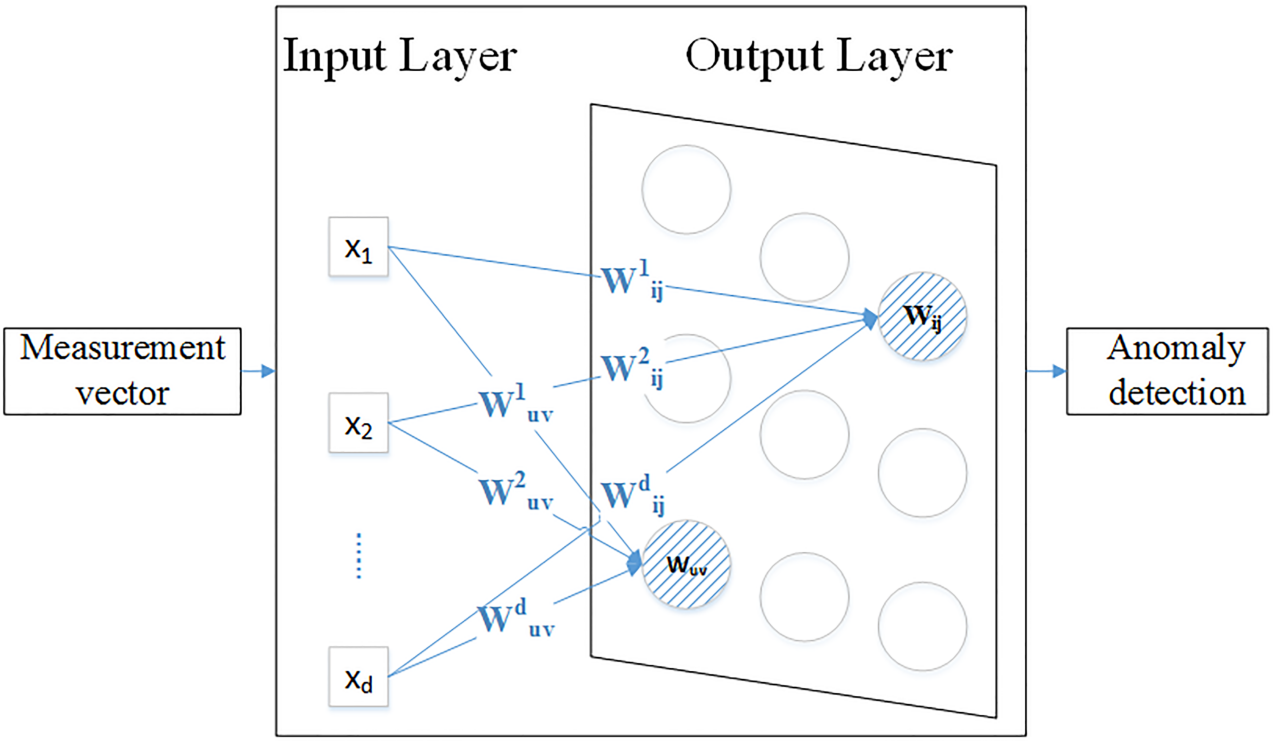

Fig 1. The general structure SOM network. SOM network consists of the Input Layer and Output Layer. Each input vector X is a D-dimensional vector, and each neurons in the output layer is associated with a coordinate(u, v) and a weight vector W_{uv}.

https://doi.org/10.1371/journal.pone.0187488.g001

x_h ($1 \leq h \leq D$)denotes one system-level performance metrics, such as CPU, memory, disk I/O, or network traffic and so on, and the weight vector W_{uv} should have the same length as the measurement vector x(t).

The SOM network training process is the neuron competition process, which works by computing the distances between the input measurement vector and each neuron's weight vector in the map and selecting the neuron with the smallest distance as the winning neuron or excited neuron. Several choices can be made for the definition of the distance function such as Euclidean distance, Manhattan distance, Cosine similarity etc. And then the winning neuron's weight vector as well as its neighborhood neurons are updated as shown in Fig 2. The basic formula for updating the weight vector of a given neuron (u, v) at time t is given in Eq (1).

$$W_{uv}(t) = W_{uv}(t-1) + N_C[x(t) - W_{uv}(t-1)] \quad (1)$$

Where W_{uv}(t-1) is the weight vector at time t-1, and x(t) is the input vector of system-lever performance metrics at time t. N_C is a neighborhood function of the winning neuron or excited neuron C, and a Gaussian function is usually used as the neighborhood function. The function is described in Eq (2).

$$N_C(t) = \gamma(t) \times exp(-\frac{\| I_C - (i,j) \|^2}{2\delta^2(t)}) \quad (2)$$

Where N_C is the neighborhood of the excited neuron C, γ(t) is the learning-rate factor at time t, which determines how much each weight vector changed at time t, I_C is the index of the excited neuron C, (i, j) is the coordinate of neuron n_{ij} in the map, δ(t) is the size of the neighborhood at time t. γ(t) and δ(t) is monotonically decreasing with time t.

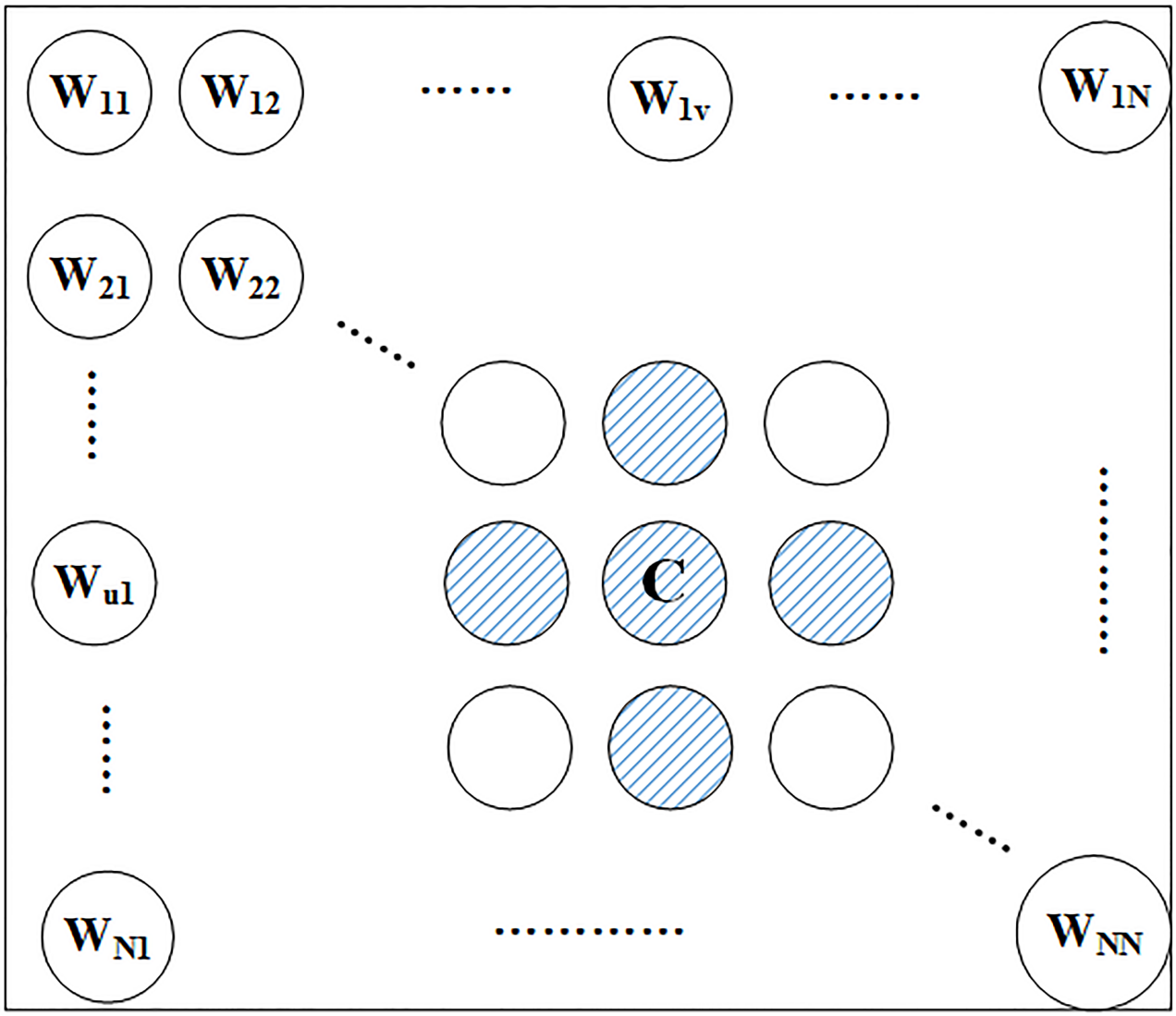

Fig 2. The nearest neighbors of excited neuron C. Each neuron is labeled with it's coordinate (u, v), that is W_{uv}. C: the excited neuron. The circles with blue line is the nearest neighbors of excited neuron C.

https://doi.org/10.1371/journal.pone.0187488.g002

The winning neuron or excited neuron is determined by the squared Euclidean distance between the input vector x(t) and the weight vector W_{uv}, for each input vector the excited neuron (u, v) is determined in Eq (3).

$$C = \begin{cases} \arg\min_{uv} \| x(t) - W_{uv}(0) \|, & t = 1 \\ \arg\min_{uv} \| x(t) - W_{uv}(t-1), & t = 2, 3, \ldots \end{cases} \tag{3}$$

To determine whether the training process of SOM network is convergence, the following inequality Formula Eq (4) is used.

$$\frac{1}{N \times N} \sum_{i,j=1}^{N} \| W_{ij}(t) - W_{ij}(t-1) \| \leq \varepsilon \tag{4}$$

Where ε is a sufficient small number we predefined, which denotes the average deviation of the SOM weight vector at time t and t-1, $\| \bullet \|$ is the Euclidean distance between two vectors.

Simulated annealing-based SOM algorithm

Simulated Annealing-based SOM (SOMSA) algorithm applies Simulated Annealing algorithm to SOM to optimize the training process, which could be divided into two steps. First of all, use traditional SOM algorithm to train input samples to get the Best Matching Neurons (BMN) or winner neurons and their neighborhoods. Secondly, use simulated annealing (SA) algorithm to adjust the weight of BMN and its neighborhoods to find the global optimization solution (clusters).

Simulated Annealing (SA) [39] is an optimization algorithm to find a global minimum of a problem among many local minimal, in which containing a solution space with a set of all possible solutions (clusters) and an objective function.

In SOMSA, the BMNs and their neighborhoods are regarded as solutions S, and the inter-class total distance as the objective function J. The objective function of SOMSA is described in Eq (5).

$$J_S = \sum_{i}^{m} \sum_{X \in S_i} \| X - C_{si} \| \tag{5}$$

Where X is the input vector, C_{si} is the center of cluster i in solution S, $\| \bullet \|$ is the Euclidean distance between the input sample X and its cluster center C_{si}, J_S is the sum distance of each input sample to its cluster center.

Keep the SA algorithm going on until the optimal performance is achieved. At each iteration it replaces the current cluster center by a random nearby center chosen with a probability p, depending on the difference between the new objective function and the current objective function. If $J_{S'}$-J_S is less than zero, then set S' as the current optimal cluster solution, otherwise accept the new solution by the probability p. The formula is as Eq (6).

$$p = exp(\frac{|J'_S - J_S|}{k \times J_S}) \tag{6}$$

Where k is a constant, J_S is the current objective function with a cluster center that the input sample belongs to, $J_{S'}$ is the new objective function when the input sample chosen a random nearby cluster center.

During the training process, random initial weight W is needed to be set, which is the same as traditional SOM algorithm. Although the SOMSA algorithm optimizes the accuracy and can effectively prevent the local optimum through the probability function, it still costs more time to train SOM network due to the random initial value and the random selection strategies, which cannot detect performance anomalies of large-scale virtual machines on cloud platform in real-time, leading to a low scalability and adaptability. At each iteration more calculations and comparisons are required to performed simultaneously. Moreover, an annealing speed α and the iteration number N have to be predefined, too.

In order to address the problem, the following study attempts to improve the scalability, adaptability, accuracy and high convergence rate of SOM model by modeling an incremental SOM model that combines a heuristic-based initial optimization algorithm and a Weighted Euclidean Distance algorithm into SOM.

An improved incremental SOM anomaly detection algorithm

Incremental model-An iterative regression process based on heuristic initialization algorithm

Commonly, when using the SOM method to execute the anomaly detection, at least 1000 samples were needed to make the SOM map into a roughly order. However, under cloud platform, the virtual machines are deployed randomly and dynamically. It is not possible to obtain the performance metrics of virtual machines before deployment. So an incremental model SOM method or an iterative regression SOM is more suitable for anomaly detection of virtual machines on cloud platform.

The incremental SOM starts from the initialization optimization of the SOM network. The common method of SOM network initialization is called the random initialization which selects N samples from the input space randomly. N is the number of neurons in SOM, and it constitutes the initial associated weight vector of each neuron. Relevant research [40–43] indicates that although the initial associated weight vector is determined without any prior knowledge, the SOM network is still able to reach the ordering state after several training iterations. However, it obtains the ordering state at the cost of longer training time of SOM model training. In this paper, we proposed a heuristic-based initialization method for SOM network to effectively initialize the SOM network and shorten the training time.

By analyzing the convergence phase of SOM, it can be shown that, during the training period of a sample x, the weight of a neuron in SOM network will be tuned as long as the neighborhood size N_C is reduced to contain only one neuron that is the cluster center itself. The set of those samples are recorded as $TS(W_{nv})$, and the equation can be described in Eq (7).

$$W_{uv}(t+1) = W_{uv}(t) + \gamma(t)[x - W_{uv}(t)] \qquad \forall x \subset TS(W_{uv}) \tag{7}$$

Assume that W_{uv} will eventually converges to $\bar{W}_{uv}$, then when there exists a sample x belongs to $TS(W_{uv})$, it can be derived as Eq (8).

$$\begin{aligned} \lim_{t\to\infty} W_{uv}(t+1) &= \lim_{t\to\infty} W_{uv}(t) + \lim_{t\to\infty} \gamma(t)[x - W_{uv}(t)] \\ \Rightarrow \bar{W}_{uv} &= \bar{W}_{uv} + \lim_{t\to\infty} \gamma(t)[x - W_{uv}(t)] \\ \because \mathrm{t} \to \infty, \lim_{\mathrm{t}\to\infty} \gamma(\mathrm{t}) &> 0 \\ \therefore x &= \bar{W}_{uv} \end{aligned} \tag{8}$$

Taking account of all the input samples that belong to $TS(W_{uv})$, it can get the relations in Eq (9).

$$\begin{aligned} E_{TS(W_{uv})}(x) &= \bar{W}_{uv} \\ \Rightarrow \frac{\int_{TS(W_{uv})} x \cdot p(x)}{\int_{TS(W_{uv})} p(x)} &= \bar{W}_{uv} \end{aligned} \tag{9}$$

where p(x) is the probability density function of the input space. It can be seen obviously that it's hard to get the distribution and the probability density of the input space to calculate the weight of neuron W_{uv}, but it can get the approximately estimate value of W_{uv} based on the input samples, which can be used as the idea initialization value of SOM network.

The heuristic-based initialization method is an iteration method by estimating the weight of neuron W_{uv} use the equation in Eq (10).

$$W_{uv} = \frac{\sum_{h=1}^{L} n(x_h) \cdot x_h}{|C_{uv}|} \tag{10}$$

Where $|C_{uv}|$ denotes the number of training samples in the cluster C_{uv}, L is the number of types of input vectors that in cluster C_{uv}, $n(x_h)$ is the times of one kind of sample x_h occur in C_{uv}, and $|C_{uv}| = \sum_{h=1}^{L} n(x_h)$.

The convergence of the training process of the SOM initialization can be checked using Eq (11).

$$\frac{1}{N \times N} \sum_{u=1}^{N} \sum_{v=1}^{N} \| W_{uv} - W_{uv}^{l} \| \leq \varepsilon \tag{11}$$

where $W_{uv}{}^{l}$ is the last iteration weight value of neuron n_{uv}, W_{uv} is the current weight value of n_{uv}, $\| \bullet \|$ is the Euclidean distance between them, and ε is a sufficiently small real number.

During the heuristic-based initialization training process, just a small-scale number of samples are collected from the sample dataset. Instead of directly assigning the selected sample vector values to neurons, it trains each neuron by taking account of the number of samples in each cluster and the times one sample occur in the cluster, which makes the trained SOM network close to the final convergence state and improves the training speed efficiently.

Weighted Euclidean Distance (WED)

Generally, in each iterative training process of SOM, it will compare the Euclidean distance of the input samples to each neurons' weight vector in the map to determine the training center neuron or winning neuron C_{uv}, then amend the weight vector of C_{uv} and its neighborhood neurons according to the neighborhood function. The Euclidean distance measures the distance between two vectors by the square deviations of each dimension of them, during which the variance value of each dimension is treated equally. Nevertheless, actually each dimension x_i of the input space vector $x(t) = [x_1, x_2, x_3, \ldots, x_d] \in R^D$ may subject to different distribution. And if the variance of one dimension x_i of the input vector is much larger than the others, it will cause the imbalance problem of the training center neuron to determine due to the single dimension plays a decisive role, adversely affect the quality of the SOM model.

To address the problem, the Weighted Euclidean Distance (WED) method is proposed to represent the contributions of each dimension in the competition process of SOM through calculating the weight of each dimension in the input vector in Eq (12).

$$weight(x_i) = \frac{\left(\frac{v(x_i)}{sum_v}\right)^{-1}}{\left(\frac{\sum_{i=1}^{d} v(x_i)}{sum_v}\right)^{-1}} \tag{12}$$

Where $v(x_i)$ is the variance of each dimension in input space, sum_v is the sum of variance of each dimension, that is $sum_v = \sum_{i=1}^{d} v(x_i)$.

According to the weight determined by Eq (12), we get the formula of WED as follow in Eq (13).

$$WED(X, W) = \sqrt{\sum_{i=1}^{d} weight(x_i) \cdot (x_i - W_i)^2} \tag{13}$$

Neighborhood-based training domain searching algorithm

As we know, at the convergence phase, each iteration has few impact on the associated weight vector of neurons in the SOM. Meanwhile, system-level performance metrics values can be regarded as substantially stable during a very short time because of the local properties of the virtual machine. Therefore, reducing the search space during the SOM network training process can effectively lessen the search times, decrease the computational complexity, and shorten the training time. Then we combine the WED with neighbor-based searching algorithm together to train the central point of training domain to reduce the complexity of the search operations in the SOM network training process for determining the central point of the training domain.

The training process is shown in Fig 3. To describe the system-lever performance metrics of all the virtual machines (vm_s) in the domain, let the token p_k as the pointer vector associated with virtual machine k (vm_k), $n_{matched}$ as the neuron that matches the current state, $C_{uv}(k^t)$ as the central point of the training neighborhood determined by the runtime measurement samples of virtual machine vm_k at time t. So at time t, $p_k(t)$ points to $C_{uv}(k^t)$. Assuming at time t + 1, $p_k(t + 1)$ still points to $C_{uv}(k^t)$, then get the neighborhood of $p_k(t + 1)$ at time t + 1, which could be described as Left($p_k(t + 1)$), Right($p_k(t + 1)$), Top($p_k(t + 1)$), Down($p_k(t + 1)$). After that, using WED to compare the input vector vm_k^{t+1} with $C_{uv}(k^t)$ and its neighborhoods to

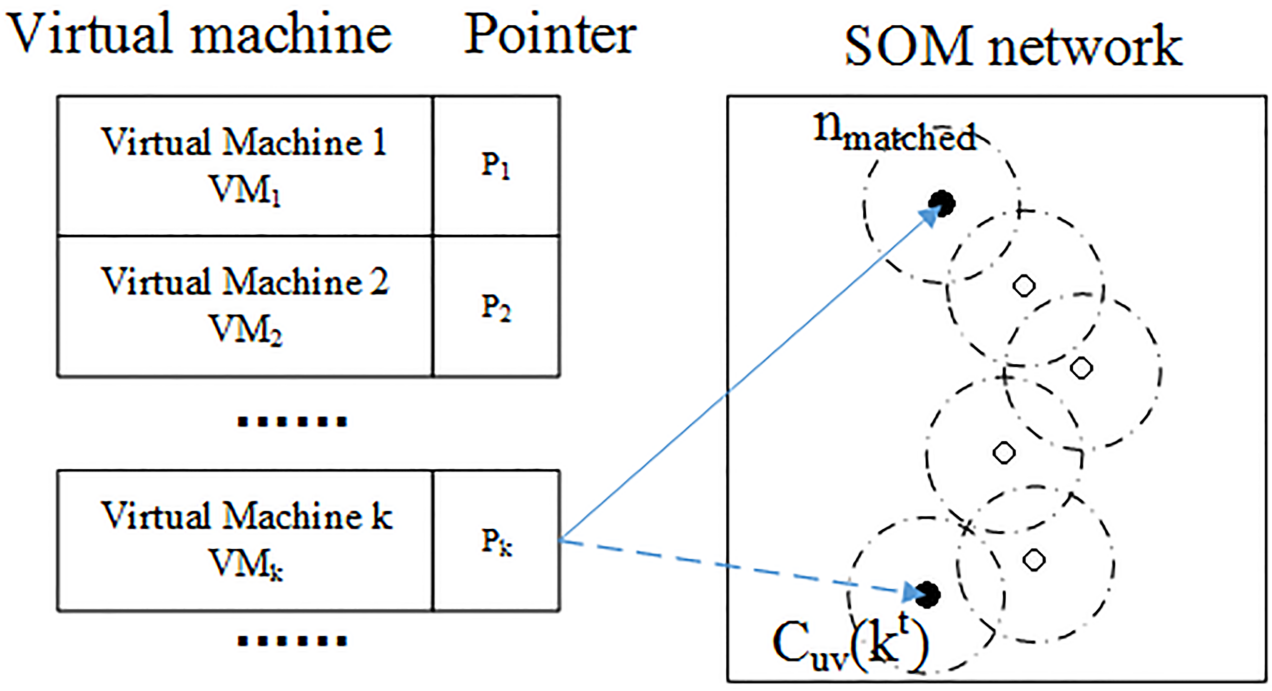

Fig 3. The process of neighborhood-based searching training domain algorithm. P_k is the point to virtual machine k, $n_{matched}$ is the best matching neuron at time t + 1, and $C_{uv}(k^t)$ is the excited neuron at time t.

https://doi.org/10.1371/journal.pone.0187488.g003

find the best matching neuron $n_{matched}$, which has the smallest Weighted Euclidian Distance. If $n_{matched} \neq C_{uv}(k^t)$, then change the pointer $p_k(t+1)$ to point to $n_{matched}$ and go to next iteration. If $n_{matched} = C_{uv}(k^t)$, replace $p_k(t)$ with $p_k(t+1)$ and end the searching process.

Compared with traversal search algorithm, this method only using a small part of the SOM network, which could efficiently shorten the training time to complete and decrease the computing complexity.

As stated above, IISOM is an effective anomaly detection algorithm for virtual machines on cloud platform. The IISOM detection model for virtual machines mainly consists of two parts (Fig 4). First is the iterative regress initialization phase. In this phase, the similarity of each data instance and the contributions of each dimension of data are considered and process is iterative until a predetermined minimum value ε is achieved. Due to the initialization process could approximately simulate the distributions of data, the roughly ordering SOM performs well in anomaly detection. Thus, when a new data instance is collected, the anomaly detection takes place in advance. If it is detected to be an anomaly, the alarm is raised. Otherwise, use it to train model until the model convergence. And the second part is the iterative convergence phase, where the neighborhoods of virtual machines are considered. The convergence process of SOM will not be retrained till the changing rate of running virtual machines exceed the

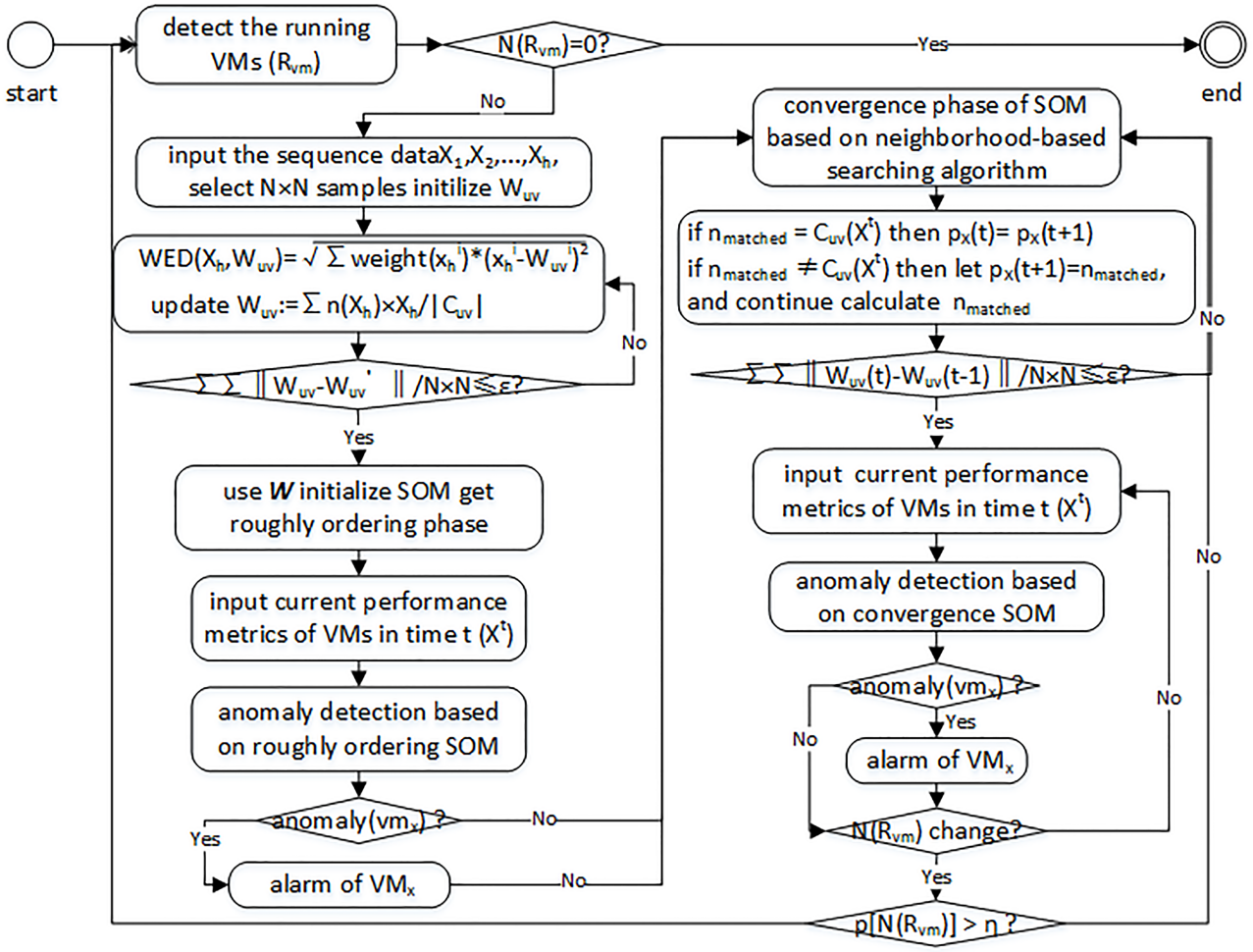

Fig 4. The flowchart of anomaly detection for virtual machines based on IISOM.

https://doi.org/10.1371/journal.pone.0187488.g004

threshold η, which is predefined to display the sensitivity to the change of virtual machines and we set it to be 0.4.

Experiments and discussions

Experimental environment

In this paper, the open source cloud platform OpenStack [44, 45] was used to build the experimental cloud platform with the physical servers for running virtual machines and the physical servers for running cloud management components. All the physical servers are installed the operation system CentOS6.5, while the former installs the hypervisor Xen3.2 [46], and the latter installs the cloud management components. And 100 virtual machines were deployed on this experimental cloud platform.

The runtime performance metrics set of virtual machines in this platform is collected by tools such as libxenstat and libvirt [47, 48]. And during the runtime, three types of fault injection method were used to simulate system failures, that is memory leak, CPU Hog and the network Hog [49–51]. A subset of key performance metrics collected is shown in Fig 5.

Experimental program and results

Experiment 1: Performance evaluate of anomaly detection for virtual machine. The performance of real-time and accuracy of IISOM was evaluated by comparing with the traditional SOM and SOMSA method.

Training: first of all, choose several virtual machines from 100 virtual machines deployed on the cloud platform, and randomly select one fault (memory leak, CPU Hog and network Hog) to inject into. Then collect 1000 runtime performance measurements from the 100 virtual machines during 10 rounds (one second per round) as the training data.

Anomaly detection: in order to estimate the performance of real-time and accuracy of the methods, one of the three faults was randomly injected into the 100 virtual machines per second and the duration time is 1 minute. The anomalies were then detected by the three trained models, traditional SOM (TSOM) model, SOMSA model and IISOM model. And the detection results were recorded.

The experimental results are shown in the following tables, Tables 1 and 2.

It can be seen from Table 1 that compare to TSOM model and SOMSA model, the IISOM takes the shortest response time to anomalies. For IISOM, it owns to the neighborhood-based domain searching method, which controls the searching domain into a small range for every input samples then speed up the detection time. And SOMSA model achieves a better detection time than TSOM model due to the SA optimization algorithm.

Table 2 shows the comparisons of detection accuracy among the TSOM model, SOMSA model and the IISOM model, all of them could obtain a good detection accuracy. Regardless of the detection time, the SOMSA model almost has the same accuracy rate with IISOM model. However, because of the dynamic characteristics of the virtual machines on the cloud platform, a shorten detection time is necessary. So, IISOM method can have a better performance in anomaly detection of virtual machines on the cloud platform than the others.

Experiment 2. The effect of clustering on KDD cup. The objective of this set of experiments was to evaluate the effect of clustering on the benchmark KDD Cup [52] dataset to cluster the kind of anomalies. In KDD Cup dataset, there are 5 types of samples, the normal samples and four kinds of abnormal samples, each sample contains 41 indicators.

We select 1000 samples from the KDD Cup dataset as the training dataset, which were labeled as normal, Probe attacked, DOS attacked, U2R attacked and R2L attacked, each 200. Meanwhile, 2000 samples from KDD Cup dataset are selected as testing dataset. Two steps are

Category	Metric	Description
cpu	Cpu_idel	CPU idle rate
	Cpu_user	CPU rate of user process
	Cpu_system	CPU rate of system process
	Cpu_iowait	wait time for IO operation
	Cpu_running	Number of tasks to be run
	VCpu_run	Run time of virtual CPU
	Vcpu_run_rate	Usage rate of virtual CPU
memory	Mem_swapd	Number of swap space used
	Mem_free	Available space
	Mem_si	The num. of virtual page paged in
	Mem_so	The num. of virtual page paged out
	Mem_total	Total size of memory
	VMem_cur	Virtual machine memory size
	VMem_rate	Usage rate of virtual machine
	VMem_max	Maximum memory occupied
disk	Disk_await	Average wait time for I/O
	Disk_util	Disk utilization
	Disk_r	I/O Read times per seconds
	Disk_w	I/O write times per seconds
	Disk_queue	I/O queue length
	Disk_rq	Average I/O data size
	Disk_sv	Average I/O service time
net	Net_rbyte	Received bytes per second
	Net_sbyte	Send bytes per second
	Net_rpacket	Received packets per second
	Net_spacket	Send packets per second
	Net_rloss	Loss of received packets
	Net_sloss	Loss of send packets
	Net_load	Load of network
	Vnet_rrate	Rate of received data
	Vnet_srate	Rate of send data

Fig 5. A subset of performance metrics. Four categories of virtual machines' performance are listed.

https://doi.org/10.1371/journal.pone.0187488.g005

Table 1. The comparisons of TSOM, SOMSA and IISOM on the detection time.

	Injecttime(s)	Responsetime(s)	Detectiontime(s)
TSOM	1	2.33	77.88
SOMSA	1	2.21	73.76
IISOM	1	1.72	64.54

The fault is injected in every per second, and response time is the time the model detect the fault. The detection time is the total time during the fault injection.

https://doi.org/10.1371/journal.pone.0187488.t001

Table 2. The comparisons of TSOM, SOMSA and IISOM on detection accuracy.

	AccuracyRate(%)
TSOM	≈ 97.6
SOMSA	≈ 98.9
IISOM	≈ 99.0

The Accuracy Rate is the rate of correctly detecting the anomalies during the fault injection.

https://doi.org/10.1371/journal.pone.0187488.t002

carried out to evaluate these algorithms: training and detection. 1000 testing samples are used to train the detection models of IISOM, SOMSA and traditional SOM, while 2000 testing samples are used to evaluate the detection performance among these models. The results are shown in Tables 3 and 4.

According to them, we get the results of metrics in Table 5.

Table 3. The experimental results of IISOM and traditional SOM on KDD cup test dataset.

Result	IISOM					TSOM				
Real label	**Normal**	**Probe**	**Dos**	**U2R**	**R2L**	**Normal**	**Probe**	**Dos**	**U2R**	**R2L**
Normal	376	1	8	0	15	361	4	9	2	26
Probe	8	379	7	0	6	17	371	9	0	3
Dos	16	8	546	0	2	24	5	537	2	4
U2R	11	0	9	192	16	13	1	7	190	17
R2L	13	4	2	0	381	14	7	3	0	376

Normal, Probe, Dos, U2R and R2L are five types of KDD Cup dataset.

https://doi.org/10.1371/journal.pone.0187488.t003

Table 4. The experimental results of IISOM and SOMSA on KDD cup test dataset.

Result	IISOM					SOMSA				
Real label	**Normal**	**Probe**	**Dos**	**U2R**	**R2L**	**Normal**	**Probe**	**Dos**	**U2R**	**R2L**
Normal	376	1	8	0	15	370	2	7	2	19
Probe	8	379	7	0	6	15	375	6	1	3
Dos	16	8	546	0	2	24	5	537	2	4
U2R	11	0	9	192	16	13	1	7	192	15
R2L	13	4	2	0	381	10	7	3	1	379

Normal, Probe, Dos, U2R and R2L are five types of KDD Cup dataset.

https://doi.org/10.1371/journal.pone.0187488.t004

Table 5. The metrics results of IISOM, traditional SOM and SOMSA on KDD cup dataset.

Algorithm	TP	FP	TN	FN	R_{TP}	R_{FP}	R_p	$R_{accurate}$
IISOM	1498	24	376	102	0.936	0.060	0.984	0.937
TSOM	1474	41	361	126	0.921	0.102	0.973	0.917
SOMSA	1490	30	370	110	0.931	0.075	0.980	0.930

TP is the number of True Positive samples, FP is the number of False Positive samples, TN is the number of True Negative samples, FN is the number of False Negative samples. R_{TP} is the rate of TP, R_{FP} is the rate of FP, R_p is the rate of precision and $R_{accurate}$ is the accuracy of the Algorithm.

https://doi.org/10.1371/journal.pone.0187488.t005

Table 6. The experimental results of IISOM and traditional SOM on our own test dataset.

Result	IISOM				TSOM			
Real label	**Normal**	**CPU Anomaly**	**Mem Anomaly**	**Net Anomaly**	**Normal**	**CPU Anomaly**	**Mem Anomaly**	**Net Anomaly**
Normal	489	5	3	3	477	7	10	6
CPUAnomaly	2	475	11	12	6	475	8	11
MemAnomaly	1	14	483	2	3	9	481	7
NetAnomaly	9	12	8	471	10	13	11	466

CPUAnomaly, MemAnomaly and NetAnomaly are three types of abnormal performance samples collected by fault injection.

https://doi.org/10.1371/journal.pone.0187488.t006

We estimate the detection accuracy trough four parts. One is the true positive rate R_{TP}, which describes the sensitivity of the detection algorithm to detect anomalies when an anomaly occurs. R_{FP} is opposite. R_p is the rate of precision, which shows the ratio of correctly identified anomalies account for all detected anomalies. $R_{accurate}$ presents the proportion of true positives and false positives account for all testing samples.

Experiment 3. The effect of clustering on real dataset. Due to the runtime performance of virtual machine has high dynamic, experiments on the real dataset are conducted, too. 1000 running data from our cloud platform were selected as the training sample, including normal samples, CPU anomaly samples, memory anomaly samples and network anomaly samples, each 250. Similarly, experiments based on these 2000 testing samples are carried out and the results are shown in Tables 6 and 7.

Summarized from Tables 6 and 7, we get the metric results in Table 8, that is the true positive rate R_{TP} and the false positive rate R_{FP}, the precision R_p and the accuracy rate $R_{accurate}$.

It can be found from results (Tables 5 and 8) that IISOM has the highest accuracy followed with SOMSA and traditional SOM. IISOM leverages heuristic-based initialization algorithm rather than random initialization used in traditional SOM and SOMSA algorithms, which

Table 7. The experimental results of IISOM and SOMSA on our own test dataset.

Result	IISOM				SOMSA			
Real label	**Normal**	**CPU Anomaly**	**Mem Anomaly**	**Net Anomaly**	**Normal**	**CPU Anomaly**	**Mem Anomaly**	**Net Anomaly**
Normal	489	5	3	3	481	8	7	4
CPUAnomaly	2	475	11	12	7	477	6	10
MemAnomaly	1	14	483	2	4	11	479	6
NetAnomaly	9	12	8	471	7	13	10	470

CPUAnomaly, MemAnomaly and NetAnomaly are three types of abnormal performance samples collected by fault injection.

https://doi.org/10.1371/journal.pone.0187488.t007

Table 8. The metrics results of IISOM, traditional SOM and SOMSA on Real dataset.

Algorithm	TP	FP	TN	FN	R_{TP}	R_{FP}	R_p	$R_{accurate}$
IISOM	1918	11	489	71	0.964	0.022	0.994	0.967
TSOM	1899	23	477	78	0.961	0.046	0.988	0.959
SOMSA	1907	19	481	74	0.963	0.038	0.990	0.962

TP is the number of True Positive samples, FP is the number of False Positive samples, TN is the number of True Negative samples, FN is the number of False Negative samples. R_{TP} is the rate of TP, R_{FP} is the rate of FP, R_p is the rate of precision and $R_{accurate}$ is the accuracy of the Algorithm.

https://doi.org/10.1371/journal.pone.0187488.t008

makes it have a fast learning ability for a runtime performance and detect in time. Though SOMSA uses simulated annealing algorithm to optimize the excited neuron's weight and its neighborhoods to improve the accuracy, it takes a longer time to train the self-organizing map network by randomly selecting adjacent center neurons to replace the current center neurons. So SOMSA does not work well on anomaly detections for virtual machines with high dynamic. Besides, IISOM algorithm takes into account that the contributions of each dimension to self-organizing map network is different. If each dimension in the input space is treated equally, just like traditional SOM algorithm and SOMSA algorithm, it may lead to the imbalance of self-organizing map network, in which a great change in some dimensions cannot cause the attention of the SOM and a subtle change in some dimensions may result in a serious fluctuation. Weighted Euclidean Distance algorithm uses different weight values to calculate the similarity (Euclidean Distance) between the input space (input samples) and the output space (neurons) to balance the contribution and improve the quality of the trained SOM model. These two experiments (based on KDD Cup dataset and the real dataset) approved the result that IISOM has a higher accuracy than traditional SOM and SOMSA algorithm, improving the true positive rate and lowing the false positive rate.

Experiment 4. Estimate the performance of models. Although IISOM performs better in the known attacks or anomalies in the same testing samples than the other algorithms, there may be of inaccuracy for unknown attacks or anomalies. So, we conduct an experiment on scalability to verify whether IISOM could detect unknown attacks or anomalies better than traditional SOM and SOMSA or not. The results of three SOMs are shown in Table 9.

Two types of attacked samples are selected, in which DOS has the maximum number in the whole KDD Cup testing dataset while U2R (User-to-Root) is the minimum. Using the SOM model we trained in Experiment 2 to test these two kinds of attacked samples. From Table 9, it can be interpreted that IISOM algorithm has a higher true positive rate than traditional SOM and SOMSA. In addition, we compare the accuracy rate of known anomalies detecting with unknown anomalies detecting to evaluate the quality of models. After training and testing, it can be seen from Table 9 that although the false positive rate of IISOM is relatively higher than

Table 9. The experimental results of IISOM and traditional SOM on our own test dataset.

Performance Algorithm	Dos		U2R		Known Anomalies	Unknown Anomalies
	R_{TP}	R_{FP}	R_{TP}	R_{FP}		
IISOM	0.932	0.015	0.887	0.019	0.953	0.941
TSOM	0.806	0.031	0.695	0.018	0.912	0.7783
SOMSA	0.854	0.011	0.793	0.023	0.932	0.866

Dos and U2R are two attacked samples. Table shows the accuracy of detecting Known Anomalies and unknown anomalies, the rate of false positive and rate of true positive.

https://doi.org/10.1371/journal.pone.0187488.t009

SOMSA on DOS attacked samples, it has an obvious advantage in unknown anomalies detection than SOMSA. Furthermore, it can be found that whether the intrusion is large or not, IISOM algorithm performance as well as usual.

In order to embody the advantage of the IISOM algorithm, two figures are used to reflect the true positive rate R_{TP} and the false positive rate R_{FP}. Fig 6 shows the true positive rate of the IISOM, traditional SOM and SOMSA with different iterations. Fig 7 shows the false positive rate of the IISOM, traditional SOM and SOMSA with different iterations.

From these two figures, it can be seen that for all the three SOM algorithms, accuracy rate keeps rising with the increase of iteration, while the false rate reduces. For IISOM algorithm, it has a relatively higher true positive rate and a lower false positive rate than the other two algorithms at the same number of iterations. For SOMSA, when the iterations is lower than 400, both true positive rate and false positive rate are relatively smooth, while as the number of iterations continues to increase, the accuracy rate is improved significantly than the traditional SOM.

Experiment 5. Estimate the parameters. The main advantage of the IISOM method is that it could balance the contribution of each dimension in the input space, rapidly and automatically search the Best Matching Neurons (BMN) by competition, avoiding giving any prior knowledge. However, several parameters should be tuned in order to obtain the fitness and smoothness SOM network. Here, we study the performance of IISOM by varying one parameter at a time.

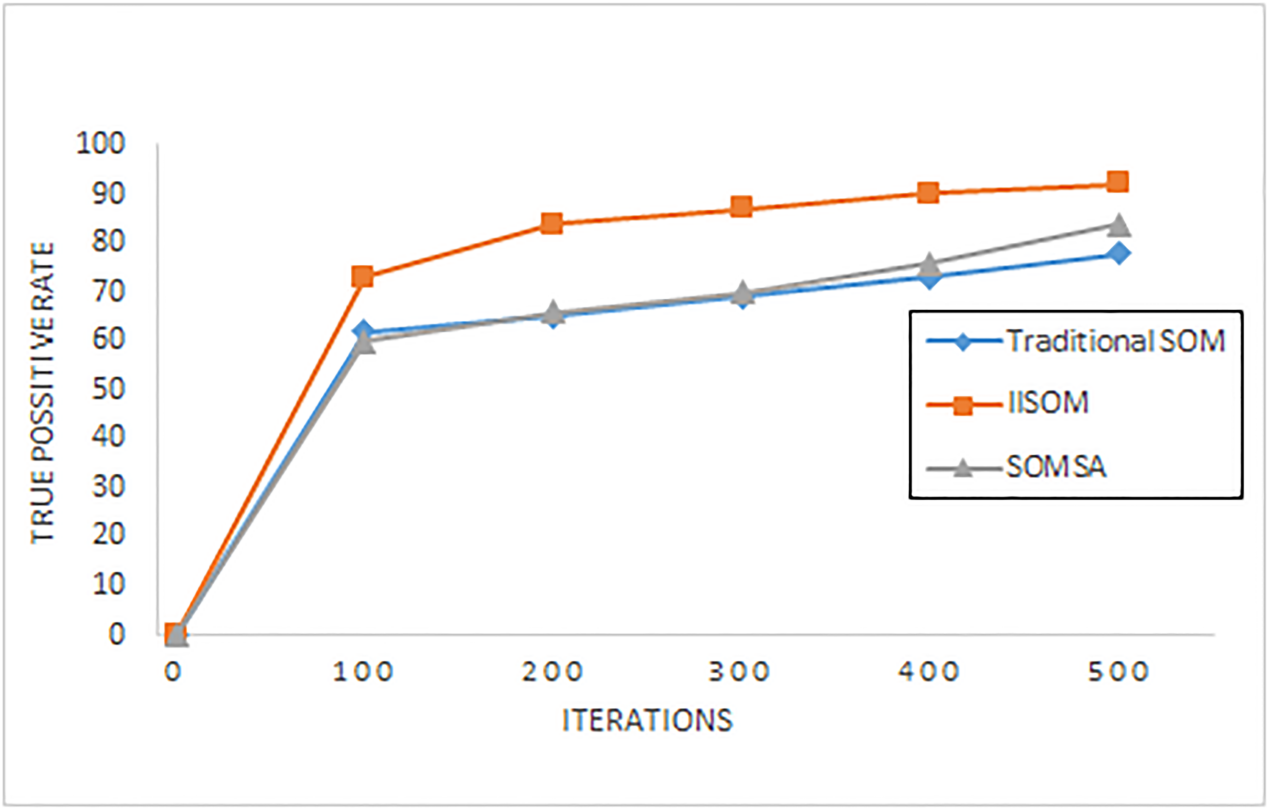

Fig 6. The true positive rate for IISOM, traditional SOM and SOMSA with different iterations.

https://doi.org/10.1371/journal.pone.0187488.g006

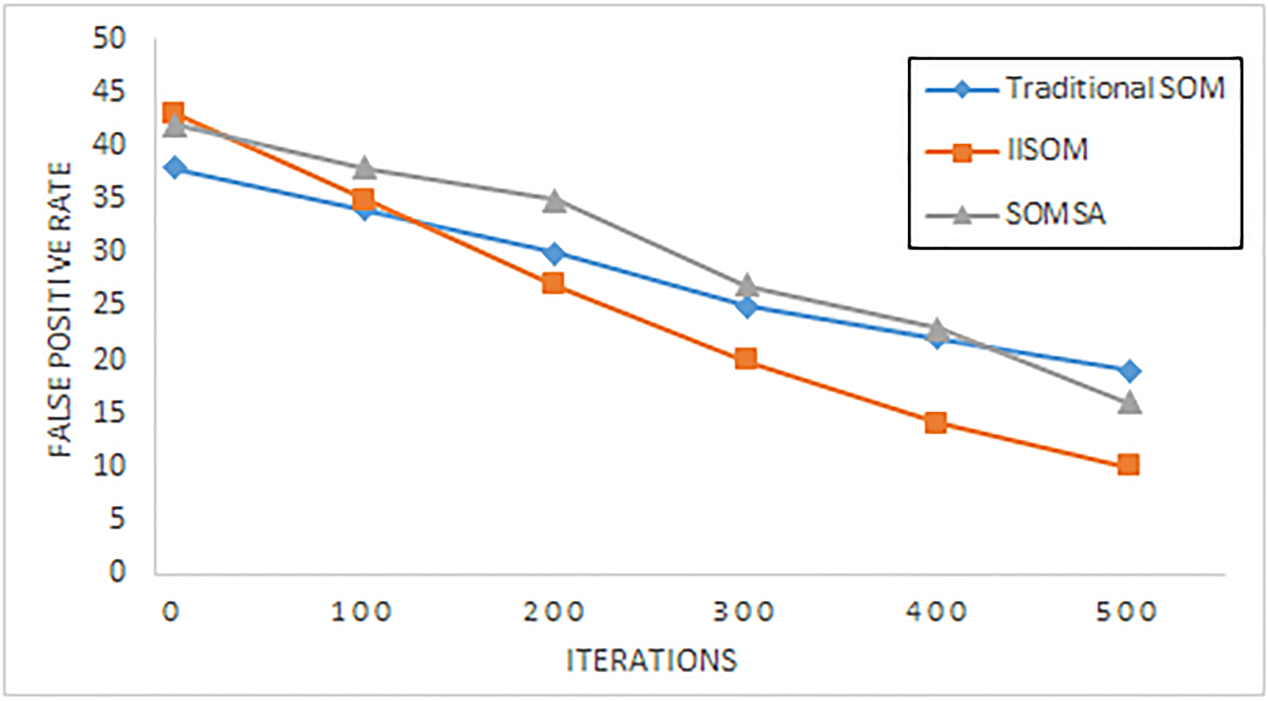

Fig 7. The false positive rate for IISOM, traditional SOM and SOMSA with different iterations.

https://doi.org/10.1371/journal.pone.0187488.g007

The parameters we discuss about are the map size of SOM k, the neighborhood size δ and the learning rate γ. In order to illustrate their influence on the detection, the Table 10 and following Figs (Figs 8 and 9) display the impact on the quality and accuracy of IISOM.

As we can see from Table 10, two types of quantities are used to evaluate the quality of the SOM that is quantization error (QE) and topographic error (TE) [53]. QE is the average distance between each input vector and its cluster center, the smaller the distance, the higher the detection accuracy. TE is the proportion of all input vectors for which first and second cluster centers are not adjacent, a smaller value comes to a smooth model. IISOM automatically balances the influence of each dimension in the input space by weighted Euclidean distance algorithm, which could maintain the topology preservation and reduce the distance between each input samples with its cluster center. Combined with the experimental results, it can be found that the map size of SOM has a few influence on the quality of the IISOM when the initial size of training neighborhood (radius) and learning rate are set to be 3.5 and 1, respectively.

Table 10. The impact of map size of SOM on the quality of IISOM.

Map size of SOM	QE	TE
5 × 5	0.12	0.01
10 × 10	0.11	0.02
15 × 15	0.11	0.01
18 × 18	0.10	0.02
21 × 21	0.09	0.02

QE is the quantzation error, TE is the topographic error.

https://doi.org/10.1371/journal.pone.0187488.t010

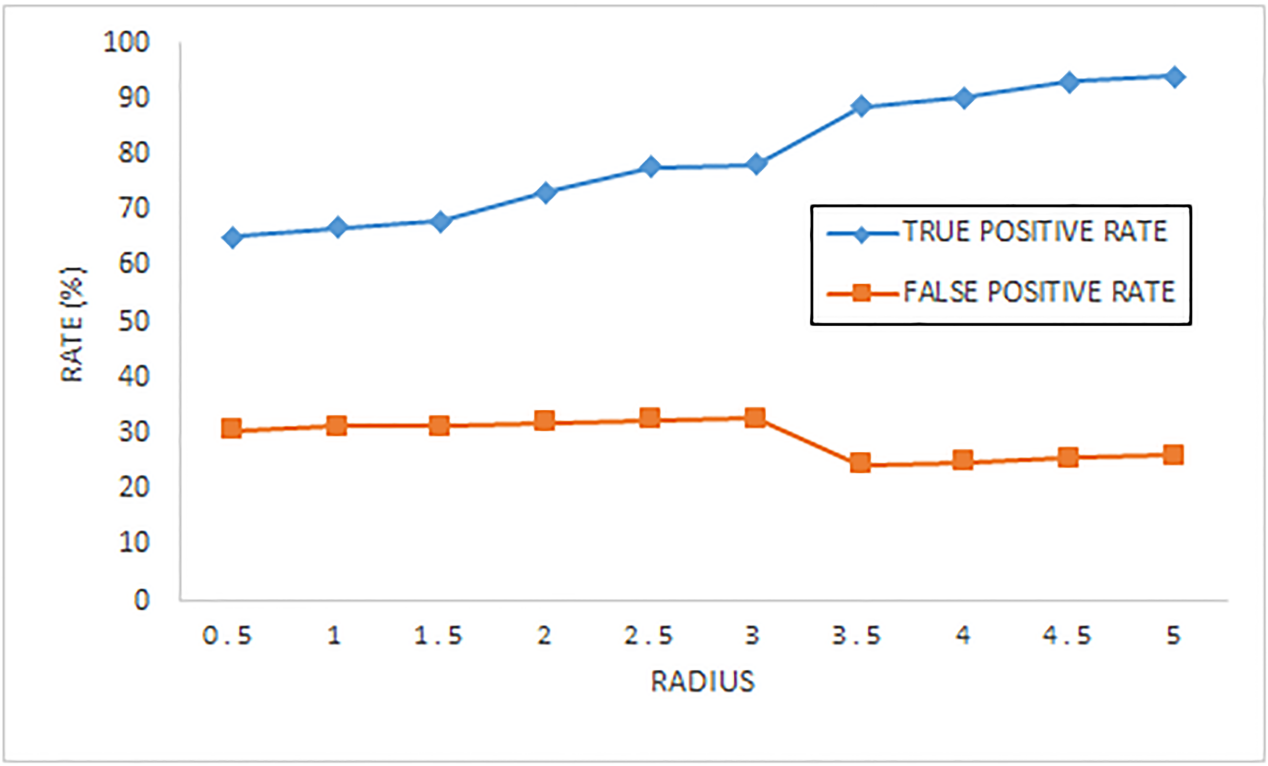

Fig 8. The performance for IISOM by varying the initial neighborhood size(radius).

https://doi.org/10.1371/journal.pone.0187488.g008

To evaluate the neighborhood size (radius) and learning rate well, the map size of SOM is set to be 15 × 15. Fig 8 displays the true positive rate and false positive rate of IISOM with different initial neighborhood size (radius), in which the initial learning rate is set to be 1. Both true positive rate and false positive rate achieve a better value around the radius 3.5, greater than or less than the value the accuracy rate is low. Fig 9 illustrates the performance of IISOM in terms of the number of iterations and the accuracy rate with different initial learning rate. From this figure we can see that the initial learning rate is related to the necessary iterations for convergence, a smaller or larger value will lead to more iterations. In the training process of IISOM, the learning rate is monotone decreasing during the iteration, therefore, it is efficient to choose a good initial learning rate. A small initial learning rate makes a small contribution of each sample to train the SOM network, which causes under fitting the status of testing data to decline the accuracy rate. Vice versa, a large one leads to over fitting, decreasing the convergence velocity. As shown in Fig 9, when the initial learning is 1, the accuracy rate achieves the highest value and the iterations is lowest, in which the initial neighborhood size is set to be 3.5.

Discussion

Cloud platforms with characteristics of resource sharing, allocation in demand and virtualization appear more and more users to lease resources in pay-as-you-go fashion and deploy their own systems on it to improve the utilization of hardware and software resources and reduce the cost. However, due to the ever-growing complexity and dynamic of cloud computing systems, it is susceptible to resource contentions, software bugs, hardware failures or administrators' mistakes, which can significantly affect the system performance. In order to avoid the

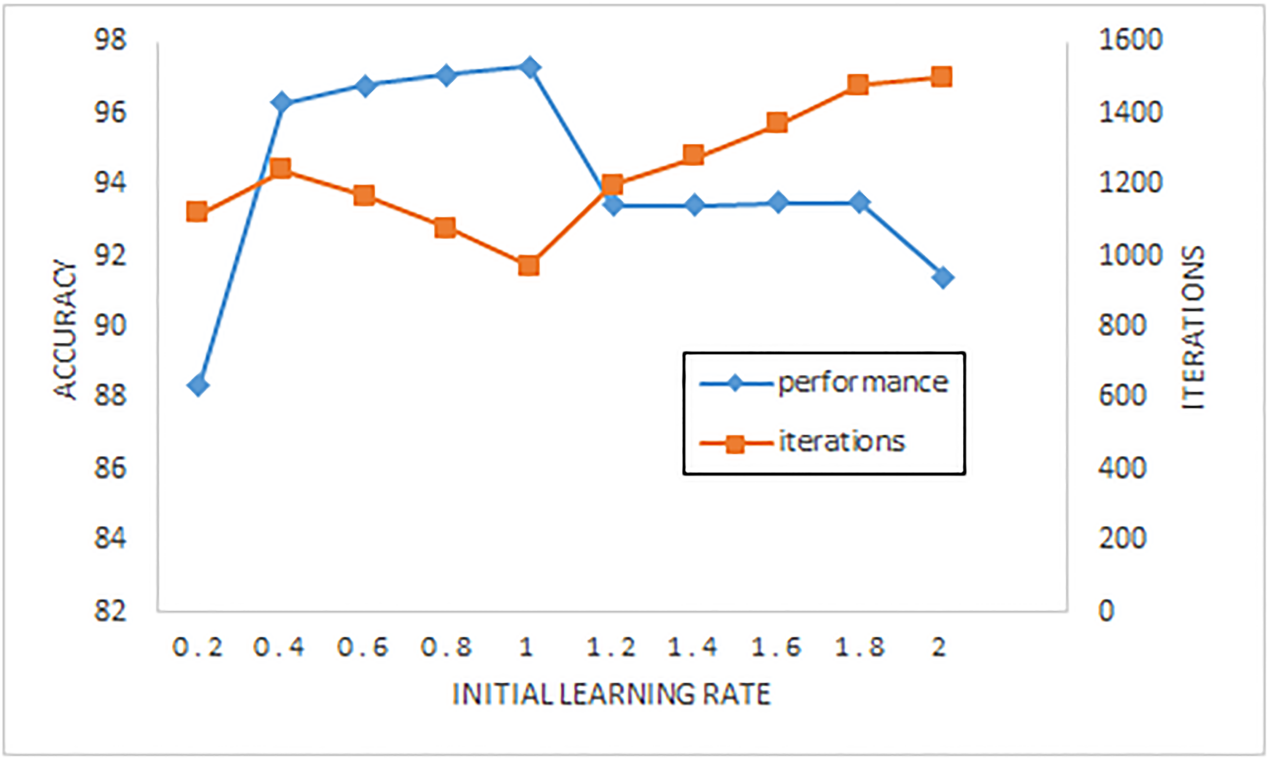

Fig 9. The performance for IISOM by varying the initial learning rate.

https://doi.org/10.1371/journal.pone.0187488.g009

performance anomaly, a number of anomaly detection techniques [54–57] were proposed to proactively and reactively detect anomalies, but due to their detection mechanisms, they mostly lack scalability and often require prior knowledge, thus making them unsuitable for virtual machines detection on cloud platforms. Thus, we propose the Improved Incremental SOM (IISOM) algorithm to accelerate detecting process and improve the quality of detection model by considering the virtual machines' high dynamic and complexity.

Strengths and limitations. From the experimental results, it can be seen that IISOM is superior to the traditional SOM and SOMSA algorithm for anomaly detection of virtual machines. With the help of heuristic-based initialization method and neighborhood-based searching method, IISOM greatly enhances the detection accuracy and the detection rate. Besides, by considering the contribution of each dimension of the performance metrics, the IISOM algorithm outperforms the traditional SOM and SOMSA algorithm, achieving a highest quality model than the others.

There are nonetheless several limitations of the current IISOM method. The parameters of map size of SOM, learning rate and neighborhood size still have to be predefined by empirical, and the current heuristic-based initialization method takes on the opinion that the initial weight value of neurons would approximately reflect the data distribution. But the data distribution is usually unknown in advance, several iterative regression computations have to be done during the initialization.

Future work. Anomaly detection for virtual machines on cloud platforms has a high demand of algorithm for detection accuracy, self adaption and real-time capability. As the lack of adaptive map size, a data-driven density-based detection method will be introduced into the IISOM to improve the detection model. In addition, with the growing scale of cloud platform,

it will be increasingly subjected to novel attacks and other anomalies. Thus, incorporating some novelty detection method into this IISOM method is a promising future direction. What's more, to enhance the technology and keep up with the trends, we plan to apply this IISOM to some new fields, such as mobile Augmented Reality [58], Mobile Landmark Recognition [59], Mobile Visual Location Recognition [60], Image processing [61, 62] and so on, to deal with the high dimensional data and meet the demands of real-time capability.

Conclusion

Considering with the feature of complexity and high dynamicity of cloud platform, the Improved Incremental SOM (IISOM) algorithm is proposed to identify and predict anomalies to keep the platform dependable. Weighted Euclidean Distance (WED) method and a heuristic-based initial method are incorporated into the Self-organizing Map (SOM) algorithm to reduce computational overhead, shorten the training time and self-adaptation in dynamic environment. To evaluate the proposed algorithm, five experiments are performed. Experiment 1 is carried out to evaluate the real-time property and detection accuracy of the IISOM compared to SOMSA and TSOM. The different detection models based on KDD Cup dataset and real dataset are trained through experiments 2 and 3 to evaluate the performance. Experiment 4 is set to estimate the quality or performance of the models when encounters unknown anomalies. Compared with traditional SOM and SOMSA on the four experiments, it shows that IISOM has more advantages in detecting accuracy, convergence velocity, and performance even though the anomalies are large and unknown. For further analysis of IISOM algorithm, several important parameters are considered in Experiment 5. From which, it can be summarized that the map size of SOM k has few influence in the model training process due to the dimensional balance based on WED algorithm, while the initial neighborhood size δ and the initial learning rate γ are sensitive to the model training. It can be found that the model performs well when δ is 3.5 and γ is 1.

Acknowledgments

The work of this paper is supported by National Natural Science Foundation of China (Grant No.61272399 and No.61572090), the Science and Technology Research Project of Chongqing Municipal Education Committee (Grant No. KJ1704081) and Research Fund for the Doctoral Program of Higher Education of China (Grant No.20110191110038).

Author Contributions

Conceptualization: Hancui Zhang, Shuyu Chen.

Data curation: Tianshu Wu.

Formal analysis: Hancui Zhang, Zhen Zhou.

Investigation: Jun Liu.

Methodology: Hancui Zhang, Zhen Zhou.

Project administration: Hancui Zhang, Shuyu Chen.

Resources: Hancui Zhang, Tianshu Wu.

Software: Zhen Zhou, Tianshu Wu.

Supervision: Shuyu Chen, Jun Liu.

Validation: Hancui Zhang, Tianshu Wu.

Visualization: Zhen Zhou, Tianshu Wu.

Writing – original draft: Hancui Zhang.

Writing – review & editing: Hancui Zhang, Jun Liu.

References

1. Gunarathne T, Wu T-L, Choi JY, Bae S-H, Qiu J. Cloud computing paradigms for pleasingly parallel biomedical applications. Concurrency and Computation-Practice & Experience. 2011; 23(17):2338–54. https://doi.org/10.1002/cpe.1780
2. Zhao J, Hu L, Ding Y, Xu G, Hu M. A Heuristic Placement Selection of Live Virtual Machine Migration for Energy-Saving in Cloud Computing Environment. Plos One. 2014; 9(9). https://doi.org/10.1371/journal.pone.0108275
3. Shanahan HP, Owen AM, Harrison AP. Bioinformatics on the Cloud Computing Platform Azure. Plos One. 2014; 9(7). https://doi.org/10.1371/journal.pone.0102642
4. Yazar S, Gooden GEC, Mackey DA, Hewitt AW. Benchmarking Undedicated Cloud Computing Providers for Analysis of Genomic Datasets. Plos One. 2014; 9(9). https://doi.org/10.1371/journal.pone.0108490
5. Chandola V, Banerjee A, Kumar V. Anomaly Detection: A Survey. Acm Computing Surveys. 2009; 41(3). https://doi.org/10.1145/1541880.1541882
6. Paschalidis IC, Chen Y. Statistical. Anomaly Detection with Sensor Networks. Acm Transactions on Sensor Networks. 2010; 7(2). https://doi.org/10.1145/1824766.1824773
7. Wang G, Chen S, Liu J. Anomaly-based Intrusion Detection using Multiclass-SVM with Parameters Optimized by PSO. International Journal of Security and Its Applications. 2015; 9(6):227–42. https://doi.org/10.14257/ijsia.2015.9.6.22
8. Tylman W. Seismic facies analysis from pre-stack data using self-organizing maps. Proceedings of International Conference on Dependability of Computer Systems. 2008: 211–218.
9. Sani Y, Mohamedou A, Ali K, Farjamfar A, Azman M, Shamsuddin S. An Overview of Neural Networks Use in Anomaly Intrusion Detection Systems. 2009 Ieee Student Conference on Research and Development: Scored 2009, Proceedings. 2009:89–92.
10. Zhou Z, Chen S, Lin M, Wang G, Yang Q. Minimizing Average Startup Latency of VMs by an Optimized VM Templates Caching Mechanism Based on K-Medoids Clustering in an IaaS System with Multi-cluster of Servers. International Journal of Grid and Distributed Computing. 2015; 8(3):275–97. https://doi.org/10.14257/ijgdc.2015.8.3.27
11. Russo T, Scardi M, Cataudella S. Applications of Self-Organizing Maps for Ecomorphological Investigations through Early Ontogeny of Fish. Plos One. 2014; 9(1). https://doi.org/10.1371/journal.pone.0086646
12. Kourki M, Riahi MA. Seismic facies analysis from pre-stack data using self-organizing maps. Journal of Geophysics and Engineering. 2014; 11(6). https://doi.org/10.1088/1742-2132/11/6/065005
13. Kohonen T. SELF-ORGANIZED FORMATION OF TOPOLOGICALLY CORRECT FEATURE MAPS. Biological Cybernetics. 1982; 43(1):59–69. https://doi.org/10.1007/BF00337288
14. Kohonen T. Self-Organizing Maps. 2nd edition. Berlin; Springer. 1997.
15. Fritzke B. Growing self-organizing networks—history, status quo, and perspectives. High-Level Workshop on the Theory of Methodology and Applications of the SOM. 1999:131–141.
16. Koikkalainen P. Progress with the Tree-Structured Self-Organizing Map. Proc Ecai. 1994; 78(9):211–5.
17. Koikkalainen P, Oja E. Self-organizing hierarchical feature maps. Proceedings of 1990 International Joint Conference on Neural Networks. 1990. https://doi.org/10.1109/IJCNN.1990.137727
18. Laaksonen J, Koskela M, Oja E. Application of tree structured self-organizing maps in content-based image retrieval. Proc Icann Edinburgh Uk. 1999; 1:174–9 vol.1.
19. Polani D. On the Optimization of Self-Organizing Maps by Genetic Algorithms. Proceedings of the Workshop on Self-Organizing Maps. 1999:157–169.
20. Kubota R, Horio K, Yamakawa T. Binary Self-Organizing Map with Modified Updating Rule and Its Application to Reproduction of Genetic Algorithm. IEICE—Transactions on Information and Systems. 2007; E90-D(1):382–3. https://doi.org/10.1093/ietisy/e90-1.1.382
21. Deep K, Dipti. A new hybrid Self Organizing Migrating Genetic Algorithm for function optimization. Evolutionary Computation cec ieee Congress on. 2007:2796–803.

22. Chalasani R, Principe JC. Self-organizing maps with information theoretic learning. Neurocomputing. 2015; 147:3–14. https://doi.org/10.1016/j.neucom.2013.12.059
23. Júnior AHS, Barreto GA, Corona F. Regional models: A new approach for nonlinear system identification via clustering of the self-organizing map. Neurocomputing. 2014; 147:31–46.
24. Rzhetsky A, Zheng T, Weinreb C. Self-correcting maps of molecular pathways. Plos One. 2006; 1(1): e61–e61. https://doi.org/10.1371/journal.pone.0000061 PMID: 17183692
25. Soledad D, Federico M, Antonio M, Juan Julián M, Carlos B. A novel representation of genomic sequences for taxonomic clustering and visualization by means of self-organizing maps. Bioinformatics. 2015; 31(5):736–44. https://doi.org/10.1093/bioinformatics/btu708
26. Skupin A, Biberstine JR, Boerner K. Visualizing the Topical Structure of the Medical Sciences: A Self-Organizing Map Approach. Plos One. 2013; 8(3). https://doi.org/10.1371/journal.pone.0058779 PMID: 23554924
27. Lennard C, Hegerl G. Relating changes in synoptic circulation to the surface rainfall response using self-organising maps. Climate Dynamics. 2015; 44(3–4):861–79. https://doi.org/10.1007/s00382-014-2169-6
28. Chávez-Arroyo R, Lozano-Galiana S, Sanz-Rodrigo J, Probst O. Statistical-dynamical downscaling of wind fields using self-organizing maps. Applied Thermal Engineering. 2015; 75:1201–9. https://doi.org/10.1016/j.applthermaleng.2014.03.002
29. Huang DW, Gentili RJ, Reggia JA. Self-organizing maps based on limit cycle attractors. Neural Networks the Official Journal of the International Neural Network Society. 2015; 63:208–22. https://doi.org/10.1016/j.neunet.2014.12.003 PMID: 25562568
30. Meschino GJ, Comas DS, Ballarin VL, Scandurra AG, Passoni LI. Automatic design of interpretable fuzzy predicate systems for clustering using self-organizing maps. Neurocomputing. 2015; 147:47–59. https://doi.org/10.1016/j.neucom.2014.02.059
31. Abdelsamea MM, Gnecco G, Gaber MM. An efficient Self-Organizing Active Contour model for image segmentation. Neurocomputing. 2015; 149(PB):820–35. https://doi.org/10.1016/j.neucom.2014.07.052
32. Hoz E D L, Ortiz A, Ortega J, Martínez-Álvarez A. Feature selection by multi-objective optimisation: Application to network anomaly detection by hierarchical self-organising maps. Knowledge-Based Systems, 2014, 71:322–338. https://doi.org/10.1016/j.knosys.2014.08.013
33. Hoz E D L, Ortiz A, Ortega J, Prieto B. A PCA filtering and probabilistic SOM for network intrusion detection. Neurocomputing, 2015, 164:71–81. https://doi.org/10.1016/j.neucom.2014.09.083
34. Liu J, Chen S, Zhou Z and Wu T. An Anomaly Detection Algorithm of Cloud Platform Based on Self-Organizing Maps. Mathematical Problems in Engineering. 2016; 1:1–9.
35. Wang H, Xu Z, Wang C, Yuan Z. A New Algorithm Combining Self Organizing Map with Simulated Annealing Used in Intrusion Detection. Biomedical Engineering and Informatics. 2009.
36. Song Y, Jiang Q, Yan X. Fault diagnosis and process monitoring using a statistical pattern framework based on a self-organizing map. Journal of Central South University. 2015; 22(2):601–9. https://doi.org/10.1007/s11771-015-2561-3
37. Vesanto J. SOM-based data visualization methods. Intelligent Data Analysis, 1999, 3(2):111–126. https://doi.org/10.1016/S1088-467X(99)00013-X
38. Vesanto J, Alhoniemi E. Clustering of the self-organizing map. IEEE Transactions on Neural Networks, 2000, 11(3):586–600. https://doi.org/10.1109/72.846731 PMID: 18249787
39. Wu S, Chow T W S. Self-organizing and self-evolving neurons: a new neural network for optimization. IEEE Transactions on Neural Networks, 2007, 18(2):385–96. https://doi.org/10.1109/TNN.2006.887556 PMID: 17385627
40. Juan A, Vidal E. Comparison of Four Initialization Techniques for the K -Medians Clustering Algorithm. Springer Berlin Heidelberg. 2000:842–52.
41. He J, Lan M, Tan CL, Sung SY, Low HB. Initialization of cluster refinement algorithms: A review and comparative study. Proceedings of IEEE International Joint Conference on Neural Networks. 2004.
42. Lin C-H, Liu J-C, Ho C-H. Anomaly detection using LibSVM training tools. International conference on Information Security and Assurance. 2008:166–171.
43. Kiziloren T, Germen E. Anomaly Detection with Self-Organizing Maps and Effects of Principal Component Analysis on Feature Vectors. Revista Brasileira De Cirurgia Cardiovascular. 2009; 14(4):355–7.
44. Aguado J G. MonPaaS. An Adaptive Monitoring Platform as a Service for Cloud Computing Infrastructures and Services. Services Computing IEEE Transactions on, 2015, 8(1):65–78. https://doi.org/10.1109/TSC.2014.2302810
45. Milojičić D, Llorente I M, Montero R S. OpenNebula: A Cloud Management Tool. Internet Computing IEEE, 2011, 15(2):11–14. https://doi.org/10.1109/MIC.2011.44

46. Jin H, Qin H, Wu S, et al. CCAP: A Cache Contention-Aware Virtual Machine Placement Approach for HPC Cloud. International Journal of Parallel Programming, 2013, 43(3):403–420. https://doi.org/10.1007/s10766-013-0286-1
47. Cho Y, Choi J, Choi J, Lee M. Towards an integrated management system based on abstraction of heterogeneous virtual resources. Cluster Computing. 2014; 17(4):1215–23. https://doi.org/10.1007/s10586-014-0391-y
48. Li J, Jia Y, Liu L, Wo T. CyberLiveApp: A secure sharing and migration approach for live virtual desktop applications in a cloud environment. Future Generation Computer Systems. 2013; 29(1):330–40. https://doi.org/10.1016/j.future.2011.08.001
49. Zhenbo X U, Zhang J, Zhongxing X U. Melton: a practical and precise memory leak detection tool for C programs. Frontiers of Computer Science, 2015, 9(1):34–54. https://doi.org/10.1007/s11704-014-3460-8
50. Wojek C, Dollar P, Schiele B, et al. Pedestrian Detection: An Evaluation of the State of the Art. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2012, 34(4):743–761. https://doi.org/10.1109/TPAMI.2011.155
51. Chin Y J, Berger T. A software-only videocodec using pixelwise conditional differential replenishment and perceptual enhancements. IEEE Transactions on Circuits & Systems for Video Technology, 1999, 9(3):438–450. https://doi.org/10.1109/76.754773
52. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.
53. Kiviluoto K. Topology preservation in self-organizing maps. IEEE International Conference on Neural Networks. 1996.
54. Wang C, Viswanathan K, Choudur L, et al. Statistical techniques for online anomaly detection in data centers. Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on, 2011:385–392.
55. Guan Q, Fu S. Adaptive Anomaly Identification by Exploring Metric Subspace in Cloud Computing Infrastructures. Proceedings of the IEEE Symposium on Reliable Distributed Systems, 2013:205–214.
56. Tan Y, Nguyen H, Shen Z, et al. PREPARE: Predictive Performance Anomaly Prevention for Virtualized Cloud Systems. International Conference on Distributed Computing Systems. IEEE Computer Society, 2012:285–294.
57. Peter B, Griffith R, Sutton C, Fox A, Jordan M, Patterson D. Statistical machine learning makes automatic control practical for internet datacenters. Usenix Wrkshp Hot Topics Cloud Computing, 2009.
58. Wei B, Guan T, Duan L, et al. Wide area localization and tracking on camera phones for mobile augmented reality systems. Multimedia Systems, 2015, 21(4):381–399. https://doi.org/10.1007/s00530-014-0364-2
59. Guan T, Wang Y, Duan L, et al. On-Device Mobile Landmark Recognition Using Binarized Descriptor with Multifeature Fusion. Acm Transactions on Intelligent Systems & Technology, 2015, 7(1):1–29. https://doi.org/10.1145/2795234
60. Zhang Y, Guan T, Duan L, et al. Inertial sensors supported visual descriptors encoding and geometric verification for mobile visual location recognition applications. Signal Processing, 2015, 112:17–26. https://doi.org/10.1016/j.sigpro.2014.08.029
61. Ji R, Gao Y, Hong R, et al. Spectral-Spatial Constraint Hyperspectral Image Classification. IEEE Transactions on Geoscience & Remote Sensing, 2014, 52(3):1811–1824. https://doi.org/10.1109/TGRS.2013.2255297
62. Ji R, Cao L, Wang Y. Joint Depth and Semantic Inference from a Single Image via Elastic Conditional Random Field. Pattern Recognition, 2016. https://doi.org/10.1016/j.patcog.2016.03.016