applied sciences

MDPI

Article

A New Method of Fuzzy Support Vector Machine Algorithm for Intrusion Detection

Wei Liu *, LinLin Ci and LiPing Liu

Computer Department, Beijing Institute of Technology, Beijing 100081, China; cilinlin_bit@126.com (L.L.C.); liuliping_bit@163.com (L.P.L.)

* Correspondence: liuwei_bit@126.com; Tel.: +86-159-0101-5684

Received: 27 December 2019; Accepted: 29 January 2020; Published: 5 February 2020

Abstract: Since SVM is sensitive to noises and outliers of system call sequence data. A new fuzzy support vector machine algorithm based on SVDD is presented in this paper. In our algorithm, the noises and outliers are identified by a hypersphere with minimum volume while containing the maximum of the samples. The definition of fuzzy membership is considered by not only the relation between a sample and hyperplane, but also relation between samples. For each sample inside the hypersphere, the fuzzy membership function is a linear function of the distance between the sample and the hyperplane. The greater the distance, the greater the weight coefficient. For each sample outside the hypersphere, the membership function is an exponential function of the distance between the sample and the hyperplane. The greater the distance, the smaller the weight coefficient. Compared with the traditional fuzzy membership definition based on the relation between a sample and its cluster center, our method effectively distinguishes the noises or outlies from support vectors and assigns them appropriate weight coefficients even though they are distributed on the boundary between the positive and the negative classes. The experiments show that the fuzzy support vector proposed in this paper is more robust than the support vector machine and fuzzy support vector machines based on the distance of a sample and its cluster center.

Keywords: support vector machine; SVDD; system call sequence; intrusion detection

1. Introduction

Intrusion detection systems (IDS) are essential to information security. IDS can be divided into signature-based IDS and anomaly-based IDS [1]. Both are based on pattern detection. Signature-based IDS matches the system behavior against the known attack and lacks the ability to detect zero-day attack. Anomaly-based methods construct normal behavior based on prior knowledge and judge the deviation between the current behavior and the normal behavior [2]. The advantage of the anomaly-based method is the ability to detect new attacks.

A system call requested by an application is a function built into the operation system kernel. A system call sequence is a detailed account of the system calls occurring on a host. The behavior of an application can be described in terms of the sequence of system calls. It is easy to get the system call sequence in real-time. Therefore, the data of a system call sequence is often used as audit data for analysis and classification of malicious processes.

The current methods of anomaly detection are based on traditional statistics, which is the study of the asymptotic theory. That is, the limit property can be reached when the number of samples approaches infinity. In intrusion detection systems, the observation samples are limited or even a small number. This cannot satisfy the preconditions of a detection method based on traditional statistics. As a result, the false alarm rate and missing rate are high.

The algorithms of a system call in anomaly-based IDS need a lot of data. This is because these algorithms are based on traditional statistics, which is the study of the asymptotic theory. That is, the limit property can be reached when the number of samples approaches infinity. Unfortunately, the anomaly data in the intrusion detection system is very limited. Therefore, we classify system calls using an SVM-based algorithm, which is based on statistical learning theory. Statistical learning theory makes the SVM-based classifier only depend on a small part of the support vectors (SVs). This is very helpful for the training of classifiers with insufficient data. SVM-based algorithm, like the other algorithm, also has an inherent shortcoming, that is, it is sensitive to noises and outliers. In order to distinguish noises near the boundary from SV, fuzzy support vector machine is proposed to solve the problem. Because there are no uniform guiding principles, the existing FSVM-based algorithms are inconsistent with reality when classifying system call sequences. Therefore, the purpose of our paper is to construct a fuzzy support vector machine that is suitable for classifying system call sequences.

Support vector machine [3–6], as a machine learning method based on statistical learning theory, derives from the idea of the dual form to solve the large dimensional problems, makes the classifier only depend on a small part of the support vectors, implements the structural risk minimization principle in statistical learning theory, and solves the problems of nonlinearity and local minima. A system call sequence can be converted into a vector in a high dimensional space by the frequency of short system call sequences of a certain length. Therefore, abnormal detection can be carried out based on SVM.

There are always noises and outliers in solving practical engineering applications due to statistical methods, human error and other factors. These noises and outliers cannot satisfy the precondition that all samples are independent and identically distributed. The noises and outliers near the boundary play the same role as SVs in constructing the optimal classification hyperplane. To solve this problem, researchers proposed fuzzy support vector machine (FSVM), that is, different weights are assigned to different samples, so that different samples contribute differently to the optimal classification hyperplane. In order to eliminate the influence of noises and outliers, the small weights are given to these samples. The design of the membership function is the key of the whole fuzzy algorithm and is no uniform criterion to be followed. At present, there are many ways to construct a membership function. Most of these methods are based on the distance between a sample and its cluster center. The closer the sample is to the cluster center, the greater the weight coefficient is. The noises and SVs distributed on the boundary are far from the cluster center. They are all given less weight coefficients. This is quite different from the objective situation. SVs should be given greater weight coefficients.

To solve the above problem, our paper proposed a new fuzzy support machine method based on support vector data description (SVDD). The contributions of our work are as follows:

- Our paper proposed a new fuzzy membership function which can effectively distinguish the noises and SVs distributed on the boundary based on SVDD. SVs are given larger weight coefficients while noises are given smaller coefficients. In this way, our method avoids imperfection of FSVM based on the distance between a sample and its cluster center. Such a fuzzy membership function structure method is more in line with reality.
- The method proposed in our paper uses the hyperplane, which passes through the cluster center and takes the line of two cluster centers as the normal vector to replace each cluster center. This is in accordance with the geometric principle of SVM. In other words, two hyperplanes with maximum space are used to separate the training samples. Therefore, using the hyperplane in class to replace the cluster center can better approximate the actual situation.
- Our method is more efficient, especially for anomaly-based IDS with high real-time requirements. In our method, the noises and outliers are identified by a sphere with minimum volume while containing the maximum of the samples. Some uncontributed vectors are eliminated by pre-extracting the boundary vector set containing the support vectors. This reduces the number of training samples and speeds up the training. It is important that IDS speed up detection as much as possible by reducing computation and storage.

The remainder of the paper is organized as follows. In Section 2, the previous work based on system calls are reviewed, while Section 3 describes the proposed method. In Section 4, experiments and evaluations are presented. Section 5 gives some conclusions.

2. Previous Work

Anomaly detection can be studied by system call sequence from different angles. These methods focus primarily on data processing, data representation and other feature selections derived from system call sequences. In this section, some methods of anomaly-based IDSs based on system call will be discussed.

As the original data of system call traces is large, preprocessing and feature selection methods contribute to obtaining typical features and avoiding the influence of irrelevant and redundant features on detection rate and processing cost [7,8]. Methods commonly used for natural language processing are used to preprocess system call traces. The n-gram method is used to construct the system call databases of normal behavior by a sliding window with a single length or multiple lengths [9,10]. Aron Laszka et al. [11,12] investigated and claimed that the optimal n-gram is 6-gram in UNM dataset and 7-gram in ADFA-LD dataset. Suaad et al. [13] continued to prove that 6-gram and 10-gram have the advantage of time efficiency and detection rate respectively in a dataset collected from a virtual machine. Feature selection methods reduce redundancy and irrelevance by selecting interesting features. These methods facilitate the reduction of computation time and storage requirements, understanding data out noise and avoiding the over-fitting problem, increasing the accuracy rate. Feature selection can be divided into th wrapper approach [14], filter approach [15] and hybrid approach [16] according to the correlation of algorithms. This depends on the feedback represented by the accuracy rate; the wrapper approach implements the selection of best features. The filter approach evaluates the attributes of a learning algorithm by using the statistical learning data. The wrapper approach gets better classification performance than filter approach at the expense of expensive computation. The filter approach is better suited to handle high dimensional data than the wrapper approach. A hybrid approach was produced by combining the advantages and disadvantages of the filter approach and wrapper approach.

The enumerating sequences-based [17–19] methods are simple and efficient to implement by removing system call parameters. During database construction stage, normal behaviors are represented by short system sequences. During the monitoring stage, the short sequences of testing data are obtained and tested. The enumerating sequences-based methods need constructing, updating and maintainance of the normal database for each individual program [20,21]. The Murmurhash [22–24] is utilized with the Bloom-filter-based method to ensure that it has the advantage over STIDE in terms of memory occupation, searching speed and privacy preservation. Although the Bloom-filter-based method shows simplicity and effectiveness, it has the limitation of false positives.

The system call sequence can be represented as a vector. Qing et al. extracted a minimized set of rules to define a normal behavior and detected anomaly behavior based on a rough set [25,26]. Pandit predefined workflow and added a knowledge base of workflow [27–29]. A search engine is then applied to discover the hidden knowledge [30]. The drawback of a rule-based approach [31–33], since these rules are derived from small-scale datasets, is that the rules are constantly updated. Matej et al. [34] presented a new tool data collection system for Windows PC. Different from previously distributed data collection systems, this system uses less resources based on host and client structures. The system shows good performance in the real test environment. However, it is just a preliminary stage and is going to be a lot of work. IDS plays an important role in the network. Qiuhua et al. [35] proved a classification algorithm based on data clustering and data reduction by mini batch K-Means algorithm in the training stage and sorting cluster in the detection stage. Experiments indicated that the computational complexity was reduced significantly and the accuracy maintained high. However, the implementation of this classification method is complex.

In recent years, neural networks have made remarkable achievements in computer vision [36–38] and natural language processing [39–41]. Researchers have also tried to use neural networks to process system call sequences [42–46]. AnRAD [47] performs probabilistic inference by self-structuring confabulation network. Their network continuously refines their knowledge base and is capable of fast incremental learning. Sheraz et al. [48] implemented intrusion detection in a real environment based on a convolutional neural network. In order to accelerate the training process, multiple GPUs must be deployed on a physical host. The challenge of solutions based on a neural network is pricey and space consuming due to the increasing amount of data.

Ambusaidi et al. [49] proved that their method contributes more critical features for the least square support vector machine to achieve a better detection rate and lower computation cost. Gideon [50] applied a semantic structure to system calls. This approach facilitates the representation of software behavior and obtains excellent results in UNM dataset and KD98 dataset. Wael et al. [51] presented a heterogeneous detector which consisted of sequence time-delay embedding, hidden Markov model [52–54] and a one-class support machine. In addition to satisfactory results, the heterogeneous detector also exhibits the reliability. Michael et al. [55] detected features of the system in the hypervisor. The experiments demonstrated that their detection accuracy achieves 90% whilst the method has the detecting ability of DoS attacks. The algorithms based on frequency can be realized at a lower computation cost by reducing the dimension of the frequency vectors [56]. SVM is sensitive to noises and outliers [57–60]. FSVM is based on fuzzy theory to reduce the influence of noises or outliers on the classification hyperplane [61–66]. Lin et al. [67] proposed a method based on the relation between samples and their cluster center. Zhang et al. [68] proposed a new FSVM method after considering the imperfection of a distance-based algorithm. However, the above methods also reduce the effect of SVs on the hyperplane when reducing the influence of noises or outliers on the hyperplane.

3. Methodology Based on SVDD

Due to human error, random error and other factors in the data collection process, the sample set contains a small number of noise samples. Noise samples have a great influence on the construction of the optimal classification hyperplane, which makes it deviate from the optimal position, reduces the normalization ability of the classifier, and affects the classification effect. FSVM assigns different weight coefficients to different samples. The purpose is to make each sample have a different effect on the optimal classification hyperplane.

Figure 1 shows the overview of our algorithm. The algorithm consists of three steps. The first step is to obtain the positive and negative minimum hyperspheres based on the training data and SVDD. By finding the minimum volume hypersphere containing a sample set, the target samples are included in the hypersphere as much as possible, and the non-target samples are excluded from the hypersphere. The second step is to calculate the distance between the samples and the hyperplanes. The sample position is determined by distance and radius difference. In the third step, different samples inside and outside the hyperspheres are represented by different functions. Our algorithm is described in detail below.

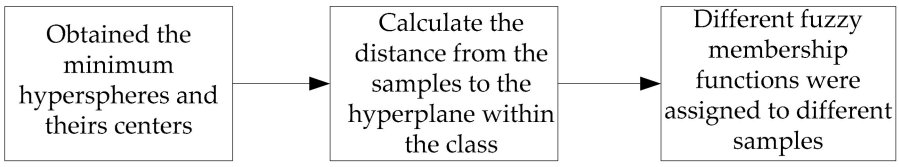

Figure 1. The overview of the proposed algorithm.

3.1. Analysis

The definition of fuzzy membership function is the key of FSVM algorithm. There are many definitions of fuzzy membership functions, but there are no general guidelines. Traditional fuzzy membership functions based on the distance between the sample and its cluster center are not effective

to distinguish noises or outliers from SVs. Sometimes the distance is not the only criterion for judging whether it is normal. As shown in Figure 2, point A on the left, in Figure 2a, has a high probability of being a valid sample. Point A on the right, in Figure 2b, has a high probability of being a noise or outlier. It is not enough to just rely on linear functions of distance. The relative position relation between samples should be considered. That is, fuzzy membership function must consider the affinity between samples.

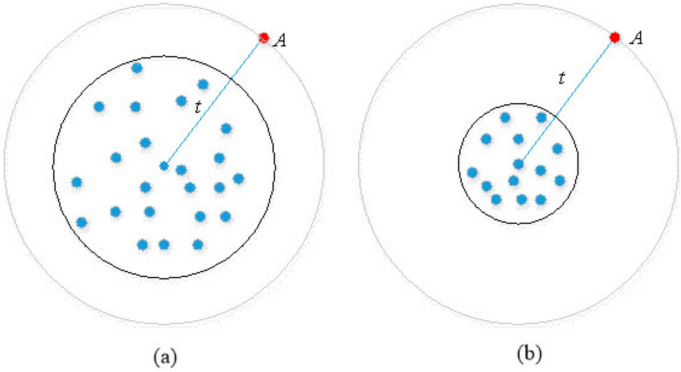

Figure 2. The difference of the affinity among samples at two different classes.

As shown in Figure 3, the distance between B and the cluster center is not the same as the distance between C and the cluster center. Since these two points have the same distance to the classification hyperplane, they contribute the same to the hyperplane. Compared with point B and point C, point D is further away from the cluster center, but closer to the classification hyperplane. Point D is the point that contributes the most to the hyperplane. So we have to take that into account when we design membership functions.

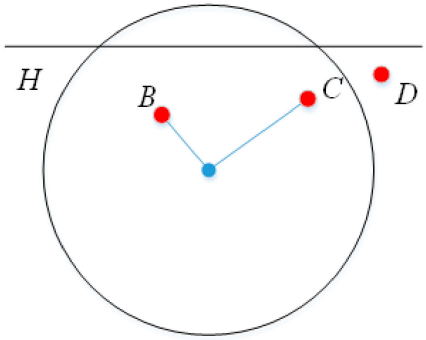

Figure 3. The membership function design based on the position of samples and the hypersphere.

The optimal classification hyperplane and its nearby support vectors are far away from the cluster center. The closer the sample is to the cluster center, the greater the weight coefficient. The noises and SVs distributed on the boundary are far from the cluster center. They are all given less weight coefficients. This is quite different from the objective situation. SVs should be given greater weight coefficients.

The optimal hyperplane of standard SVM is determined by SVs. The geometric principle of SVM is to use two hyperplanes with maximum spacing to separate the training samples as far as possible in the original space or feature space. Therefore, using the hyperplane in class to replace the center can better approximate the actual situation.

The solution of the optimal classification hyperplane of SVM is usually converted to solving quadratic programming problem. However, the solving complexity of the quadratic programming problem will increase significantly with the sample increase. When the sample size is large, the traditional fuzzy support vector machine needs large memory to store and calculate the kernel function matrix. Therefore, by selecting the boundary vector containing SVs in advance, the number

of training samples and the number of quadratic programming solutions can be reduced. This has practical significance for improving training speed and accuracy.

3.2. Support Vector Data Description

The task of one-class classification is to distinguish the target sample from non-target samples. The boundary has to be constructed by a hypersphere around the target samples. By finding the minimum volume hypersphere containing the sample set, the target samples are included in the hypersphere as much as possible, and the non-target samples are excluded from the hypersphere.

For ease of description, φ: $R^n \rightarrow H$ represents the mapping of the input space to the high-dimensional space. Assume $T = \{\boldsymbol{x}_i, i=1, 2, \cdots, l\}$ contains l data objects, and the hypersphere is described by center $\boldsymbol{a}$ and radius R. The minimum hypersphere can be obtained by solving the following quadratic programming (1).

$$\begin{gathered} \min_{R,\ \alpha,\ \xi} R^2 + C \sum_{i=1}^{l} \xi_i \\ s.t.\ \|\varphi(\boldsymbol{x}_i) - \boldsymbol{a}\|^2 \leq R^2 + \xi_i \\ \xi_i \geq 0,\ i = 1, 2, \ldots, l \end{gathered} \tag{1}$$

where $\|\cdot\|$ is the Euclidean distance, ξ_i is slack variables, and C is the trade-off between the volume of the hypersphere and the errors. Lagrangian multipliers α_i and β_i are introduced to construct a Lagrangian function. The Lagrangian function is constructed as shown in Equation (2).

$$L(R, \boldsymbol{a},\ \xi,\ \boldsymbol{\alpha},\ \boldsymbol{\beta}) = \mathrm{R}^2 + C \sum_{i=1}^{l} \xi_i - \sum_{i=1}^{l} \alpha_i \left[\mathrm{R}^2 + \xi_i - \|\varphi(\boldsymbol{x}_i) - \boldsymbol{a}\|^2\right] - \sum_{i=1}^{l} \beta_i \xi_i \tag{2}$$

The center $\boldsymbol{a}$ and radius R can be obtained from the solution to the dual problem and the solution to KKT conditions. $\boldsymbol{a}$ and R are calculated according to Formulas (3) and (4) respectively.

$$\boldsymbol{a} = \sum_{i=1}^{l} \alpha_i^* \varphi(x_i) \tag{3}$$

$$R^2 = \|\varphi(\boldsymbol{x}_j) - \boldsymbol{a}\|^2 = K(\boldsymbol{x}_j, \boldsymbol{x}_j) - 2\Sigma_{i=1}^{l} \alpha_i^* K(\boldsymbol{x}_i, \boldsymbol{x}_j) + \Sigma_{i=1}^{l} \Sigma_{j=1}^{l} \alpha_i^* \alpha_j^* K(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{4}$$

According to (5), if the Euclidean distance between the point $\varphi(x)$ and center $\boldsymbol{a}$ is less than radius R then it is normal. Conversely, it is the noise point when the Euclidean distance is greater than radius R.

$$\|\varphi(\boldsymbol{x}) - \boldsymbol{a}\|^2 = K(\boldsymbol{x}, \boldsymbol{x}) - 2\Sigma_{i=1}^{l} \alpha_i^* K(\boldsymbol{x}_i, \boldsymbol{x}) + \Sigma_{i=1}^{l} \Sigma_{j=1}^{l} \alpha_i^* \alpha_j^* K(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq R \tag{5}$$

3.3. Design Fuzzy Membership Function

According to the basic principle of the support vector machine, the optimal classification hyperplane is determined by the support vectors. If each class of the two classification problems is considered as a convex set, then these support vectors lie on the relative boundary of the two convex sets far away from the center of the two classes.

Given $T = \{(\boldsymbol{x}_i, y_i), i = 1, 2, \cdots, l\}$ contains l data objects. If l^+ and l^- respectively represent the number of positive samples $\boldsymbol{x}_i^+$, $i = 1, 2, \cdots, l^+$ and the number of negative samples $\boldsymbol{x}_i^-$, $i = 1, 2, \cdots, l^-$, then $l^+ + l^- = l$.

If $\boldsymbol{a}^+$ and a^- respectively represent the minimum hypersphere center of positive samples and the minimum hypersphere center of negative samples, R^+ and R^- respectively represent the minimum hypersphere radius of positive samples and minimum hypersphere radius of negative samples, then $\boldsymbol{a}^+$ and a^- are always located in the geometric center, if the normal vector of the hyperplane is established

by the maximum sum of the distance [69] from two centers to the hyperplane. As shown in Figure 4, in order to maximize the sum of the distances, it should satisfy $d=\|\boldsymbol{a}^+-A\|+\|a^--B\|\leq\|\boldsymbol{a}^+-O\|+\|a^--O\|$ $=\|\boldsymbol{a}^+-a^-\|$. That is, the distance is maximized $d=\|\boldsymbol{a}^+-a^-\|$ when hyperplane and vector $\boldsymbol{a}^+-a^-$ are perpendicular to each other.

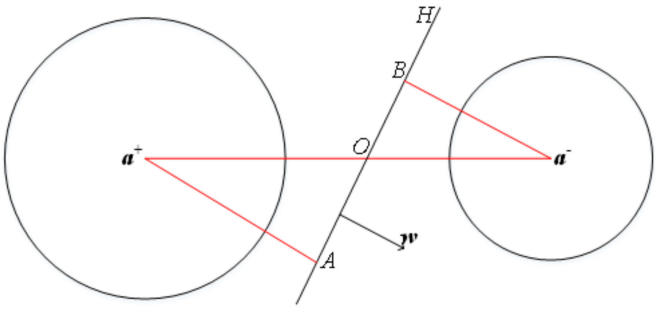

Figure 4. Maximize the sum of the distances.

For the given data, the relative positions of the two hyperspheres can be intersected, tangent and separated. The two separate cases and tangent case can be classified as one case. Although in the case of intersection, the radial basis kernel function can always choose parameters to make the two hyperspheres separated, this will cause an overfitting phenomenon. Therefore, our paper summarizes the above three cases as two cases of separation and intersection.

As shown in the Figures 5 and 6, the normal vector to the hyperplane is $\boldsymbol{w}=\boldsymbol{a}^+-a^-$ according to the principle of maximum distance. These two hyperplanes that go through $\boldsymbol{a}^+$ and a^- with a normal vector of a normal vector of $\boldsymbol{w}$ are H^+: $w^T(\boldsymbol{x}-\boldsymbol{a}^+)=0$ and H^-: $w^T(\boldsymbol{x}-a^-)=0$.

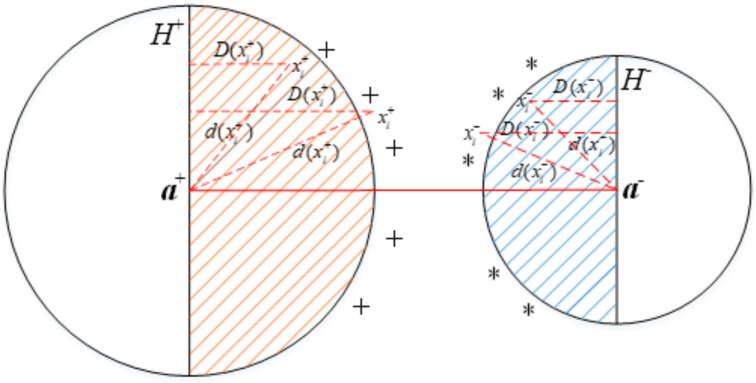

Figure 5. The case of separation.

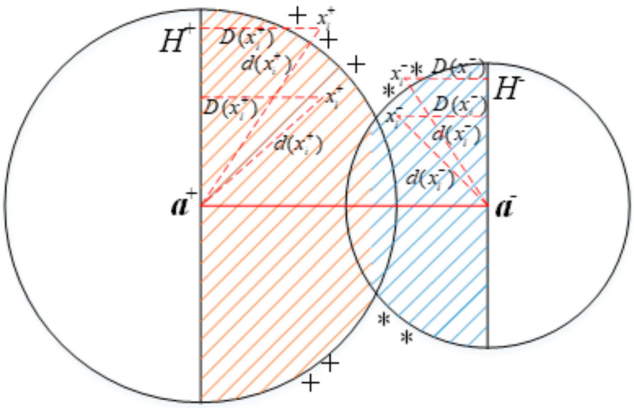

Figure 6. The case of intersection.

The optimal classification hyperplane is only determined by the SVs. Therefore, samples can be screened in advance, and those samples that may become support vectors can be selected to be trained as new training samples, which will simplify the computation of quadratic programming and improve the training speed. The optimal classification hyperplane lies between positive and negative hypersphere centers. Then, positive and negative samples between H^+ and H^- can be selected as the new training sample set. As shown in the above two figures, the shaded parts inside the hyperspheres are the normal sample, in which the positive class is represented by orange slashes, while the negative class is represented by blue slashes. The samples with "+" and "*" in the new training sample set located outside the hypersphere represent noises or outliers respectively.

If l^+_{new} and l^-_{new} are the number of positive and negative samples in the new sample set, then the distance between the samples and the hyperplane in each class is calculated according to (6).

$$\begin{cases} D\left(\boldsymbol{x}_i^+\right) = \frac{|>\boldsymbol{w}^T(\boldsymbol{x}-\boldsymbol{a}^+)|}{\|\boldsymbol{w}\|}, & i = 1, 2, \cdots, l^+_{new} \\ D\left(\boldsymbol{x}_i^-\right) = \frac{|\boldsymbol{w}^T(\boldsymbol{x}-\boldsymbol{a}^-)|}{\|\boldsymbol{w}\|}, & i = 1, 2, \cdots, l^-_{new} \end{cases} \tag{6}$$

where $\boldsymbol{w}^T(\boldsymbol{x}-\boldsymbol{a}^+) = 0$ and H^-: $\boldsymbol{w}^T(\boldsymbol{x}-a^-) = 0$ are the two hyperplanes that go through $\boldsymbol{a}^+$ and a^-; $\boldsymbol{w} = \boldsymbol{a}^+ - a^-$ is the normal vector to the hyperplane.

The distance between the samples and the center of the hypersphere is calculated according to (7).

$$\begin{cases} d\left(\boldsymbol{x}_i^+\right) = \|\boldsymbol{x}_i^+ - \boldsymbol{a}^+\|, & i = 1, 2, \cdots, l^+_{new} \\ d\left(\boldsymbol{x}_i^-\right) = \|\boldsymbol{x}_i^- - \boldsymbol{a}^-\|, & i = 1, 2, \cdots, l^-_{new} \end{cases} \tag{7}$$

Therefore, membership functions of both positive and negative sample points are constructed according to (8) and (9).

$$s_i^+ = \begin{cases} 0.6 * \frac{D\left(\boldsymbol{x}_i^+\right)}{d\left(\boldsymbol{x}_i^+\right)} * \frac{d\left(\boldsymbol{x}_i^+\right)}{R^+} + 0.4, & d\left(\boldsymbol{x}_i^+\right) \le R^+, \quad i = 1, 2, \cdots, l^+_{new} \\ 0.4 * \left[\frac{1}{\frac{d\left(\boldsymbol{x}_i^+\right)}{R^+} + \frac{D\left(\boldsymbol{x}_i^+\right)}{d\left(\boldsymbol{x}_i^+\right)}}\right]^P, & d\left(\boldsymbol{x}_i^-\right) > R^-, \quad i = 1, 2, \cdots, l^+_{new} \end{cases} \tag{8}$$

$$s_i^- = \begin{cases} 0.6 * \frac{D\left(\boldsymbol{x}_i^-\right)}{d\left(\boldsymbol{x}_i^-\right)} * \frac{d\left(\boldsymbol{x}_i^-\right)}{R^-} + 0.4, & d\left(\boldsymbol{x}_i^-\right) \le R^-, \quad i = 1, 2, \cdots, l^-_{new} \\ 0.4 * \left[\frac{1}{\frac{d\left(\boldsymbol{x}_i^-\right)}{R^-} + \frac{D\left(\boldsymbol{x}_i^-\right)}{d\left(\boldsymbol{x}_i^-\right)}}\right]^P, & d\left(\boldsymbol{x}_i^-\right) > R^-, \quad i = 1, 2, \cdots, l^-_{new} \end{cases} \tag{9}$$

The minimum value of the membership function inside the hypersphere is 0.4, and the maximum value of the membership function outside the hypersphere is 0.4. The membership value of the sample increases with the value of the distance between the sample and hyperplane. Given $p \ge 2$, the bigger p, the faster s_i^+ and s_i^- decay. Similarly, for nonlinear cases, mapping function $\varphi(x)$ is introduced by kernel function $K(x_i, x_j)$ to map data to a high-dimensional space. The normal vector is $\boldsymbol{w} = \boldsymbol{a}^+ - a^-$ by a rule for the maximum sum of distance from two hyperspheres' centers to the separation hyperplane. The hyperplanes of the two classes with $\boldsymbol{w}$ as the normal vector and going through $\boldsymbol{a}^+$ and a^- respectively are H^+: $\boldsymbol{w}^T(\varphi(x) - \boldsymbol{a}^+) = 0$ and H^-: $\boldsymbol{w}^T(\varphi(x) - a^-) = 0$. If l^+_{new} and l^-_{new} are the number of positive and negative samples in the new sample set, then the distance between the samples and the hyperplane in each class is calculated according to (10).

$$\begin{cases} D\left(\varphi\left(\boldsymbol{x}_i^+\right)\right) = \frac{|\boldsymbol{w}^T(\varphi(\boldsymbol{x})-\boldsymbol{a}^+)|}{\|\boldsymbol{w}\|}, \ \varphi(\boldsymbol{x}) = \varphi\left(\boldsymbol{x}_i^+\right), & i = 1, 2, \cdots, l^+_{new} \\ D\left(\varphi\left(\boldsymbol{x}_i^-\right)\right) = \frac{|\boldsymbol{w}^T(\varphi(\boldsymbol{x})-\boldsymbol{a}^-)|}{\|\boldsymbol{w}\|}, \ \varphi(\boldsymbol{x}) = \varphi\left(\boldsymbol{x}_i^-\right), & i = 1, 2, \cdots, l^-_{new} \end{cases} \tag{10}$$

The distance between the sample and its center of the hypersphere in each class is calculated according to (11).

$$\begin{cases} d\left(\varphi\left(\boldsymbol{x}_i^+\right)\right) = \left\|\varphi\left(\boldsymbol{x}_i^+\right) - \boldsymbol{a}^+\right\|, & i = 1, 2, \cdots, l_{new}^+ \\ d\left(\varphi\left(\boldsymbol{x}_i^-\right)\right) = \left\|\varphi\left(\boldsymbol{x}_i^-\right) - \boldsymbol{a}^-\right\|, & i = 1, 2, \cdots, l_{new}^- \end{cases} \tag{11}$$

Therefore, membership functions of both positive and negative sample points are constructed according to (12) and (13).

$$s_i^+ = \begin{cases} 0.6 * \frac{D\left(\varphi\left(\boldsymbol{x}_i^+\right)\right)}{d\left(\varphi\left(\boldsymbol{x}_i^+\right)\right)} * \frac{d\left(\varphi\left(\boldsymbol{x}_i^+\right)\right)}{R^+} + 0.4, & d\left(\varphi\left(\boldsymbol{x}_i^+\right)\right) \le R^+,\ i = 1, 2, \cdots, l_{new}^+ \\ 0.4 * \left[\frac{1}{\frac{d\left(\varphi\left(\boldsymbol{x}_i^+\right)\right)}{R^+} + \frac{D\left(\varphi\left(\boldsymbol{x}_i^+\right)\right)}{d\left(\varphi\left(\boldsymbol{x}_i^+\right)\right)}}\right]^P, & d\left(\varphi\left(\boldsymbol{x}_i^+\right)\right) > R^-,\ i = 1, 2, \cdots, l_{new}^+ \end{cases} \tag{12}$$

$$s_i^- = \begin{cases} 0.6 * \frac{D\left(\varphi\left(\boldsymbol{x}_i^-\right)\right)}{d\left(\varphi\left(\boldsymbol{x}_i^-\right)\right)} * \frac{d\left(\varphi\left(\boldsymbol{x}_i^-\right)\right)}{R^-} + 0.4, & d\left(\varphi\left(\boldsymbol{x}_i^-\right)\right) \le R^-,\ i = 1, 2, \cdots, l_{new}^- \\ 0.4 * \left[\frac{1}{\frac{d\left(\varphi\left(\boldsymbol{x}_i^-\right)\right)}{R^-} + \frac{D\left(\varphi\left(\boldsymbol{x}_i^-\right)\right)}{d\left(\varphi\left(\boldsymbol{x}_i^-\right)\right)}}\right]^P, & d\left(\varphi\left(\boldsymbol{x}_i^-\right)\right) > R^-,\ i = 1, 2, \cdots, l_{new}^- \end{cases} \tag{13}$$

4. Experimental Evaluation

This section presents the evaluation of our method in terms of detection performance, overhead and impaction of parameters. In order to test the performance of our algorithm, in addition to comparing the SVM-based algorithms SVM, FSVM1 [66], FSVM2 [65], and FSVM3 [68], the proposed algorithm is also compared with other algorithms in this section.

4.1. The Experimental Data

In order to facilitate an experimental performance comparison with similar studies, the system call datasets UNM_sendmail and UNM_live_lpr published by the New Mexico university are used in this section [70]. The UNM_sendmail data set consists of 346 normal traces and 25 abnormal traces. The abnormal data contains sunsendmailcp (sccp) intrusions and decode intrusions, in which the sccp intrusions enable the local user to obtain the root access by using special command options to make sendmail attach an e-mail message to a file, and the decode intrusions enable remote users to make changes to certain files on the local system. Data for UNM_live_lpr data set includes 15 months of activity and consists of 4298 normal tracks and 1003 abnormal tracks. The abnormal data contains lprcp symbolic link intrusions that take advantage of the vulnerability of the lpr program to control the files on the host computer and tamper with the contents of the files.

The trace consists of a sequence of system calls in chronological order. The meaning of trace varies from program to program. Each trace file lists pairs of numbers, the first number represents the process identity (PID) of the execution process, and the second number represents the specific system call (SC). The child processes forked by the parent process are traced individually. We take UNM_sendmail as an example to show the specific format of experimental data as follows:

8840 4, 8840 2, 8840 5, 8840 66, 8840 5, $\cdots$, 8840 6
8843 115, 8843 15, 8843 99, 8843 120, 8843 17, $\cdots$, 8843 12,
6545 2, 6545 5, $\cdots$, 6545 2, 6545 2, 6545 2

The data consists of different data units. Each data unit consists of PID and SC. In the experiment, data is segmented according to the PID number and the SCs after the same PID are arranged together in chronological order. In the above data, the numbers 8840, 8843 and 6545 are PIDs and the number 4,

2, 5, 66, … , 2 are SCs. The lists of system calls issued by 8840, 8843 and 6545 are denoted as R_{8840} = (4, 2, 5, 66, 5, … , 6), R_{8843} = (115, 15, 99, 120, … , 17) and R_{6545} = (2, 55, … , 2) respectively.

The vector form of a system call sequence composed of the frequency of the short system call sequences. Given n is the number of traces, m is the total number of short system call sequences. Then the ith element in the vector is the frequency at which the short system sequence numbered i occurs in the system call sequence of the process. The vector corresponding to the jth trace, represented by the frequency of short system call sequence, is denoted by $(f_{1j},\cdots,f_{mj})$. That is to say, samples are represented as (X_j, y_j) $j = 1,\cdots, n$, where $X_j = (f_{1j},\cdots,f_{mj})$, $y_j \in (+1, -1)$.

4.2. The Experimental Performance

The experimental data of UNM_live_lpr process were composed of 4298 normal tracks and 1003 abnormal tracks. The 1003 anomaly traces contain LPRCP attacks that control and tamper with host files. The experimental data for the UNM_sendmail process consisted of 346 normal traces and 25 abnormal traces, including 20 SCCPS that used E-mail to obtain root directory information and 5 decode attacks that remotely modified local files.

False alarm is to judge the normal behavior of the program as abnormal behavior. If L_N normal traces participate in the evaluation, and N_{FAR} normal traces are misjudged as abnormal traces, then the false alarm rate is equal to N_{FAR} $/L_N$. Detection rate refers to the proportion of detected abnormal traces in the total number of abnormal traces. If L_{AN} abnormal traces participate in the test and N_{HR} were detected, then the detection rate is N_{HR} $/L_{AN}$. The missing rate indicates the proportion of unrecognized abnormal traces in the total number of abnormal traces. If L_{AN} abnormal traces participate in the evaluation and N_M cannot be detected, then the missing rate is N_M $/L_{AN}$. $DR = \rho_{HR}*(1-\rho_{FAR})$ is used as the comprehensive detection formula, in which ρ_{HR} and ρ_{FAR} stands for detection rate and missing rate respectively.

Gauss kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\| \boldsymbol{x}-\boldsymbol{y} \|^2/2\sigma^2)$ is used in all algorithms during training. σ_1 and σ_2 can be selected according to the maximum error rate allowed in the target set. C_1 and C_2 can be adjusted according to the equality of upper bound of positive and negative error rate. That is, $1/l_1C_1 = 1/l_2C_2$, $1/l_1 \leq C_1 \leq 1$, $1/l_2 \leq C_2 \leq 1$. Grid search method is adopted to select the optimal parameters. The search range of parameter C is $\{2^{-24}, 2^{-23},\cdots, 2^{23}, 2^{24}\}$. The parameter σ and β both have a search range of $\{2^{-24}, 2^{-23},\cdots, 2^{23}, 2^{24}\}$. The step length and short sequence length are set to 1 and 6 respectively.

Table 1 summarizes the comparison results of detection performance between our method and the other eight methods on UNM_live_lp. The detection rate of SVM, FSVM1, FSVM2, FSVM3, and our algorithm are 61.53%,76.92%, 76.92%, 83.21%, and 84.61% respectively. The missing rates of SVM, FSVM1, FSVM2, FSVM3, and our algorithm are 38.47%, 23.08%, 23.08%, 16.23%, and 15.39% respectively. The false alarm rates of SVM, FSVM1, FSVM2, FSVM3, and our algorithm are 10.14%, 9.57%, 8.17%, 6.92%, and 4.51% respectively. In all the algorithms, our algorithm has the highest detection rate and the lowest false alarm rate and missing rate.

Table 1. Performance comparison on data set UNM_live_lp.

Algorithm	Detection Rate (%)	Missing Rate (%)	False Alarm Rate (%)
SVM	79.88%	20.12%	8.64%
FSVM1	83.71%	16.29%	7.30%
FSVM2	85.56%	14.44%	8.17%
FSVM3	86.20%	13.80%	6.92%
Our algorithm	92.63%	7.37%	6.16%

As shown in Table 1, SVM has the lowest detection rate, 79.88%, among the five algorithms. Due to the construction of a fuzzy membership function, the other four algorithms treat the different contributions of different samples in the construction of the objective function differently, which makes their average detection rate reach 87.03%. FSVM1 algorithm designs a fuzzy membership function

based on the linear function of the distance between the sample and its cluster center. The larger the distance is, the smaller the coefficients is. However, this membership design method is sometimes unable to distinguish abnormal points effectively. The FSVM2 algorithm not only considers the distance in the FSVM1 algorithm, but also considers the position relation between samples. FSVM2 algorithm determines the membership function according to the position of the sample and hypersphere. When the samples are located inside the hypersphere, the membership function is defined by the linear function of the distance. The coefficient of the sample decreases with the increase of the distance. When the samples are located outside the hypersphere, these samples are regarded as abnormal samples. The membership function is represented by different functions. The FSVM3 algorithm, like the FSVM2 algorithm, takes account of the distance between the sample and its cluster center and the affinity between samples. The difference between the FSVM3 and FSVM2 is that the FSVM3 algorithm is based on the SVDD algorithm by introducing two different parameters to control the affinity between positive and negative samples.

The method treats all samples equally during the training process, which makes the contribution of samples to constructing the optimal classification hyperplane equal. As a result, when training samples contain abnormal samples, the classification hyperplane obtained is not the optimal hyperplane. Our algorithm abides by the maximum sum of distance from the hypersphere to hyperplane, replaces the cluster center with the hyperplane inside the class, and designs the membership function according to the distance from the sample to the hyperplane inside the hypersphere. Therefore, it can be seen from the table that with the continuous improvement of the membership function, the detection rate of the algorithm increases successively. The detection rate changed from 83.71% of FSVM1 algorithm to 85.56% of FSVM2 algorithm, then to 86.20% of WCS-FSVM algorithm, and finally reached 92.63% of our algorithm.

The formula of comprehensive detection rate is closer to reality. The comprehensive detection rates of SVM, FSVM1, FSVM2, and FSVM3 are 72.97%, 77.59%, 78.56%, and 80.23% respectively. As show in Figure 7, our algorithm has the highest comprehensive detection rate, 86.92%, among five algorithms.

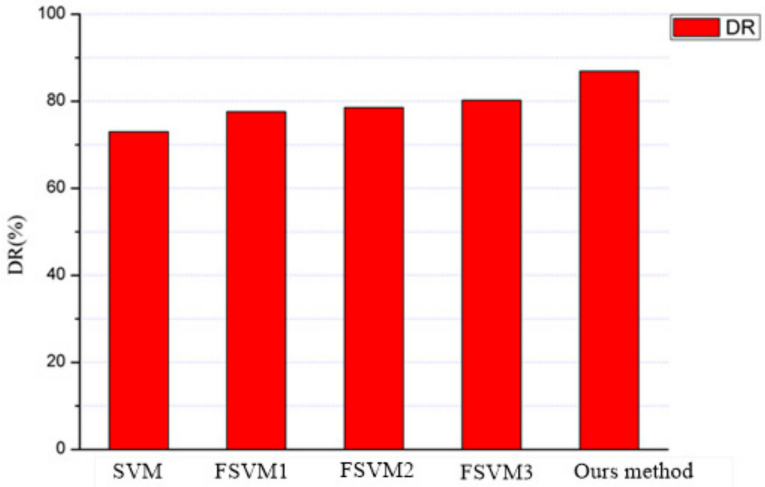

Figure 7. Performance comparisons on UNM_live_lpr data set.

At the same time, it is also evident from the data in Table 2 that the detection rate, false alarm rate and missing rate of UNM_sendmail data are all lower than that of UNM_live_lpr data. The reason for this is the amount of data. The UNM_sendmail process data consists of 346 normal tracks and 25 abnormal tracks, which is less experimental data than the 4298 normal tracks and 1003 abnormal tracks of the UNM_live_lpr process. Therefore, there is sufficient data for UNM_live_lpr process training, which is conducive to pre-extracting relative boundary vectors containing enough support vectors and ensuring sufficient data parameters. In the experiments of UNM_live_lpr process and UNM_sendmail process, SVM uses the same error penalty factor for all samples. It will have a negative

impact on the classification results when the SVM algorithm classifies unbalanced samples. This is the reason that the SVM always has the lowest detection rate among all algorithms. FSVM1 algorithm, FSVM2 algorithm, FSVM3 algorithm, and our algorithm are equivalent to using a penalty factor for each sample to be treated differently. When these algorithms are used to classify the quantity imbalance samples, good classification results can be obtained. In other words, membership function can be used to describe error penalty on samples. This is consistent with the conclusion that fuzzy support vector machine is equivalent to a multi-penalty support vector machine.

Table 2. Performance comparison on data set UNM_sendmail.

Algorithm	Detection Rate (%)	Missing Rate (%)	False Alarm Rate (%)
SVM	61.53%	38.47%	10.14%
FSVM1	76.92%	23.08%	9.57%
FSVM2	76.92%	23.08%	8.17%
FSVM3	83.21%	16.23%	6.92%
Our algorithm	84.61%	15.39%	4.51%

The comprehensive detection rates of SVM algorithm, FSVM1 algorithm, FSVM2 algorithm, FSVM3 algorithm, and our algorithm on UNM sendmail process data were 55.29%, 69.58%, 70.63%, 77.45%, and 80.79%, respectively. As shown in Figure 8, among the five detection algorithms, SVM algorithm has the lowest comprehensive detection rate and the weakest detection performance. The algorithm proposed in this paper has the highest comprehensive detection rate and the strongest detection performance. The comprehensive detection performance of FSVM1 algorithm, FSVM2 algorithm and FSVM3 algorithm also varies with the fuzzy membership function, but the overall trend is consistent with the test results obtained from UNM_live_lpr process data, that is, the detection performance is improved with the improvement of fuzzy membership function.

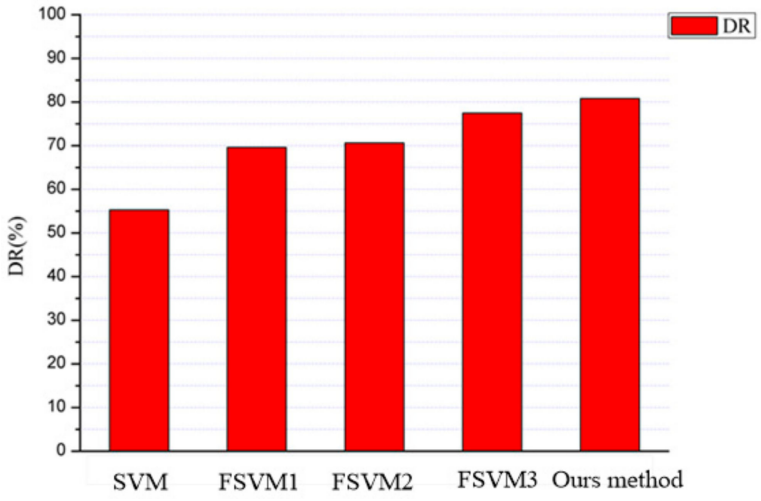

Figure 8. Performance comparisons on UNM_sendmail data set.

As shown in Tables 3 and 4, the length of the system call short sequence k has a direct impact on the detection performance. When k increases from 3 to 5, the detection rate, missing alarm rate and false alarm rate are increasing. When k is 6, our algorithm obtains the highest detection rate, the lowest false alarm rate and the lowest missing rate. When k is greater than 6, the detection rate of our algorithm starts to decrease, and the missing rate and false alarm rate start to increase. On UNM_sendmail data, the detection rate, false alarm rate, and missing alarm rate of PVFSVM algorithm have the same trend as on UNM_live_lpr data.

Table 3. Detection performance of different k on UNM_live_lpr data.

Our Method	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$	$k = 8$
Detection rate	78.30%	83.25%	86.31%	92.63%	88.26%	83.27%
Missing rate	21.70%	16.75%	13.69%	7.37%	11.74%	16.73%
False alarm rate	11.76%	11.32%	10.60%	6.16%	6.30%	7.56%

Table 4. Detection performance of different k on UNM_sendmail data.

Our Method	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$	$k = 8$
Detection rate	77.22%	82.45%	85.37%	84.61%	86.46%	81.37%
Missing rate	22.78%	17.55%	14.63%	15.39%	13.54%	18.63%
False alarm rate	14.67%	12.12%	11.68%	4.51%	6.85%	9.53%

As shown in Figures 9 and 10, if $k = 6$, the comprehensive detection rates of our algorithm on UNM_live_lpr and UNM_sendmail are 86.92% and 80.79% respectively, which are higher than the comprehensive detection rate corresponding to other k values on UNM_live_lpr and UNM_sendmail. When k increases from 3 to 6, the comprehensive detection rate also increases gradually. When k increases from 6 to 8, the comprehensive detection rate decreases gradually.

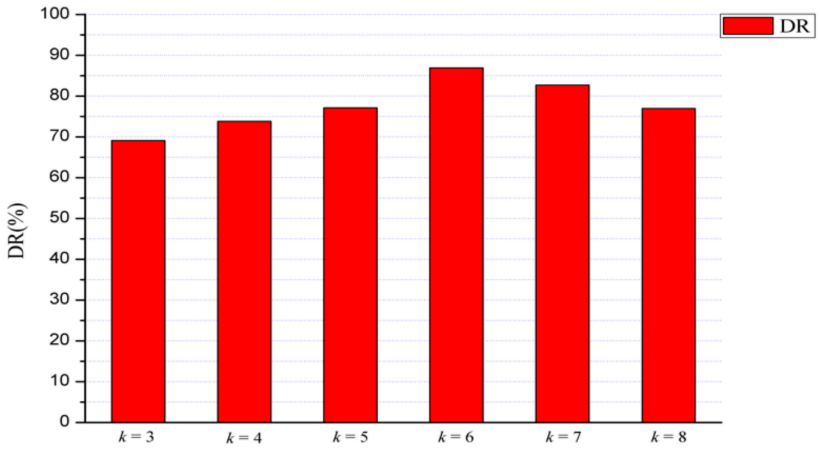

Figure 9. The comprehensive detection rate of different k on UNM_live_lpr data.

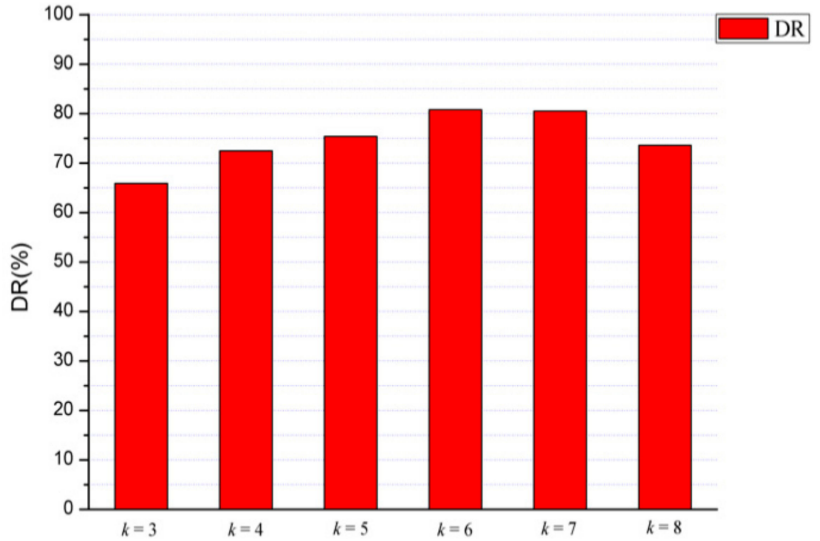

Figure 10. The comprehensive detection rate of different k on UNM_sendmail data.

This is consistent with professor Forrest's conclusion on the selection of short sequence length, and professor Wenke Lee's conclusion from the perspective of information theory that the best system call short sequence length is 6 or 7. Compared with the value of k of 6, when the value of short sequence is too small, the timing relation between short sequence patterns is lost. When the short sequence becomes longer, the short sequence pattern loses local information, no matter what kind of information loss will make the comprehensive detection performance worse. Therefore, the short sequence length can only be chosen as 6, which makes the detection performance of our algorithm reach the optimal level.

Table 5 summarizes the comparison results of detection performance between our method and the other eight methods on UNM_live_lp. The detection rate of N.bayes, Uniqueness, Hybrid Markov, Bayes1-step Markov, IPMA, Sequence matching, Compression, Closeness, and our algorithm are 65.11%, 39.1%, 45.6%, 62.6%, 45.05%, 36.7%, 33.9%, 76.5%, and 92.63% respectively. The missing rates of N.bayes, Uniqueness, Hybrid Markov, Bayes1-step Markov, IPMA, Sequence matching, Compression, Closeness, and our algorithm are 30.12%, 55.3%, 40.22%, 33.3%, 47.37%, 58.2%, 50.1%, 19.2%, and 7.37% respectively. The false alarm rates of N.bayes, Uniqueness, Hybrid Markov, Bayes1-step Markov, IPMA, Sequence matching, Compression, Closeness, and our algorithm are 4.77%, 2.30%, 14.26%, 4.1%, 7.58%, 5.91%, 16%, 4.3%, and 6.16% respectively.

Table 5. Comparison between different algorithms on UNM_live_lp.

Algorithm	Detection Rate (%)	Missing Rate (%)	False Alarm Rate (%)
N.bayes	65.11%	30.12%	4.77%
Uniqueness	39.1%	55.3%	2.30%
Hybrid Markov	45.6%	40.22%	14.26%
Bayes1-step Markov	62.6%	33.3%	4.1%
IPMA	45.05%	47.37%	7.58%
Sequence matching	36.7%	58.2%	5.91%
Compression	33.9%	50.1%	16%
Closeness	76.5%	19.2%	4.3%
Our algorithm	92.63%	7.37%	6.16%

These two methods, Compression method and Sequence Matching method, respectively consider the detection problem from the aspect of data reversibility and similarity matching degree, and fail to consider the sequence characteristics between system calls. Therefore, they are the two algorithms with the worst comprehensive detection performance. The IPMA method and Hybrid Markov method are complicated due to investigating the transition characteristics of system calls one by one. Although the ρ_{FAR} of IPMA method and Hybrid Markov method are 7.58% and 14.26%, the ρ_{MR} is 47.37% and 40.22%. The ρ_{MR} of Bayes 1-step Markov method is 33.3% and the ρ_{FAR} is 4.1%. The comprehensive test performance evaluation formula has a high value, so the test performance is good. Since this method looks the frequency of rare system calls, the ρ_{MR} of Uniqueness method is 55.3% and the ρ_{FAR} is 2.3%. However, this requires statistics for all the system calls that are used. N.bayes belongs to Naive Bayesian method in essence, and it has good noise tolerance and fast calculation, but the false alarm rate is too high. Therefore, the comprehensive detection performance is good. The closeness method extracts user behavior patterns from the perspective of combination. It shows good detection performance under different closeness thresholds, but it ignores the characteristics of attack intensity. That is, the attacker will complete the attack task in the shortest possible time and try to behave normally the rest of the time. As show in Figure 11, our algorithm has the highest comprehensive detection rate 86.92% among five algorithms. In all the algorithms, our algorithm has the highest detection rate, and the lowest false alarm rate and missing rate. The formula of the comprehensive detection rate is closer to reality. The comprehensive detection rate of N.bayes, Uniqueness, Hybrid Markov, Bayes1-step Markov, IPMA, Sequence matching, Compression, Closeness, and our algorithm are 62%, 36.91%, 39.13%, 60.03%, 41.63%, 34.82%, 28.47%, 73.22%, and 86.92% respectively.

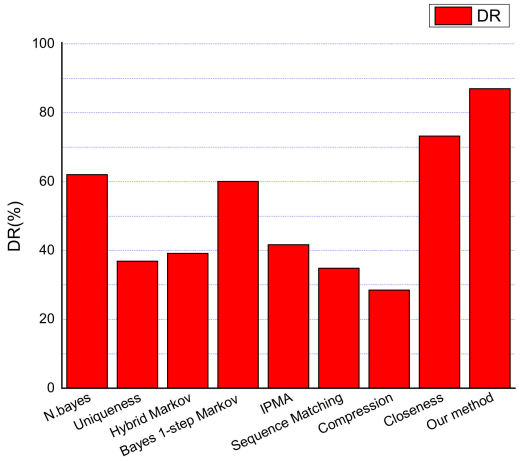

Figure 11. The DR comparation of different algorithms on UNM_live_lp.

Table 6 summarizes the comparison results of detection performance between our method and the other eight methods on UNM_sendmail. The detection rates of N.bayes, Uniqueness, Hybrid Markov, Bayes1-step Markov, IPMA, Sequence matching, Compression, Closeness, and our algorithm are 63.88%, 37.4%, 45.3%, 67.3%, 40.63%, 35.9%, 29.7%, 75.2%, and 84.61% respectively. The missing rates of N.bayes, Uniqueness, Hybrid Markov, Bayes1-step Markov, IPMA, Sequence matching, Compression, Closeness, and our algorithm are 32.12%, 60.3%, 40.44%, 30.8%, 42.37%, 60.2%, 50.5%, 20.1%, and 15.39% respectively. The false alarm rates of N.bayes, Uniqueness, Hybrid Markov, Bayes1-step Markov, IPMA, Sequence matching, Compression, Closeness, and our algorithm are 4%, 2.30%, 14.26%, 1.9%, 17%, 3.9%, 19.8%, 4.7%, and 4.51% respectively.

Table 6. Comparison between different algorithms on UNM_ sendmail.

Algorithm	Detection Rate (%)	Missing Rate (%)	False Alarm Rate (%)
N.bayes	63.88%	32.12%	4.00%
Uniqueness	37.4%	60.3%	2.30%
Hybrid Markov	45.3%	40.44%	14.26%
Bayes1-step Markov	67.3%	30.8%	1.9%
IPMA	40.63%	42.37%	17%
Sequence matching	35.9%	60.2%	3.9%
Compression	29.7%	50.5%	19.8%
Closeness	75.2%	20.1%	4.7%
Our algorithm	84.61%	15.39%	4.51%

Similarly, the Compression method and Sequence Matching method pay attention to reversibility and similarity matching degree, respectively, and fail to consider the sequence characteristic between system calls. The IPMA method and Hybrid Markov method are complicated due to investigating transition characteristics of system calls one by one. Although the ρ_{FAR} of IPMA method and Hybrid Markov method is 17% and 14.26%, the ρ_{MR} is 42.37% and 40.44%. The ρ_{MR} of Bayes 1-step Markov method is 30.8%, and the ρ_{FAR} is 1.9%. The comprehensive test performance evaluation formula has a high value, so the test performance is good. Since this method looks at the frequency of rare system calls, the ρ_{MR} of Uniqueness method is 60.3%, and the ρ_{FAR} is 2.3%. However, this requires statistics for all the system calls that are used. N.bayes belongs to Naive Bayesian method in essence, and it has good noise tolerance and fast calculation, but the false alarm rate is too high. Therefore, the comprehensive detection performance is good. The closeness method extracts user behavior patterns from the perspective of combination. It shows good detection performance under specific closeness thresholds, but it ignores the characteristics of attack intensity. That is, the attacker will complete the attack task in the shortest possible time and try to behave normally the rest of the time.

As shown in Figure 12, our algorithm has the highest comprehensive detection rate 86.92% among five algorithms. In all the algorithms, our algorithm has the highest detection rate, and the lowest false alarm rate and missing rate. The formula of comprehensive detection rate is closer to reality. The comprehensive detection rates of N.bayes, Uniqueness, Hybrid Markov, Bayes1-step Markov, IPMA, Sequence matching, Compression, Closeness, and our algorithm are 61.32%, 36.54%, 38.84%, 60.02%, 33.72%, 34.49%, 23.81%, 71.67%, and 80.79% respectively.

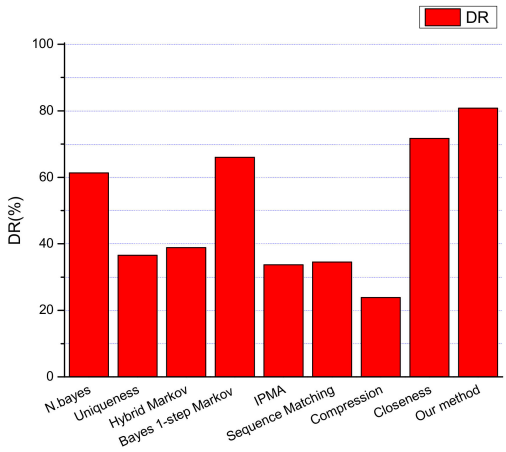

Figure 12. The DR comparation of different algorithms on UNM_sendmail.

5. Conclusions

In order to solve the defects of FSVM. This paper presents a new FSVM algorithm based on SVDD. In our method, the noises and outliers are identified by a hypersphere with minimum volume while containing the maximum of the samples. The definition of fuzzy membership considered not only the position of samples inside the hypersphere, but also the distance between the hyperplane and samples. For the samples inside the hypersphere, the fuzzy membership function is a linear function of the distance. The greater the distance is, the greater the coefficient is. For the samples outside the hypersphere, the fuzzy membership function is an exponential function of the distance. The greater the distance is, the smaller the coefficient is. Compared with the FSVM based on the relation between the sample and its cluster center, our algorithm effectively distinguishes the noises or outliers from samples. The experiments show that our FSVM is more robust than the SVM and FSVM based on the distance between the sample and its cluster center. Our algorithm is suitable for the data of system call sequence and can contribute to accurate prediction.

Author Contributions: Conceptualization and methodology, W.L. and L.C.; software and validation, W.L. and L.L.; formal analysis, W.L.; Writing–original draft preparation, W.L.; writing—review and editing, L.C.; supervision, L.C.; project administration, L.C.; funding acquisition, W.L. and L.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Natural Science Foundation of China under grant No. 61370134, the National High Technology Research and Development Program of China (863 Program) under grant No. 2013AA013901.

Acknowledgments: System call data from New Mexico university and the Massachusetts institute of technology are used in this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J.; Alazab, A. A novel Ensemble of Hybrid Intrusion Detection System for Detecting Internet of Things Attacks. *Electronics* **2019**, *8*, 1210. [CrossRef]
2. Chung, Y.; Kim, N.; Park, C.; Lee, J. Improved Neighborhood Search for Collaborative Filtering. *Int. J. Fuzzy Log. Intell. Syst.* **2018**, *18*, 29–40. [CrossRef]

3. Vapnik, V.N.; Chervonenkis, A.Y. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Its Appl.* **1971**, *16*, 264–279. [CrossRef]
4. Liu, Y.; Xu, Z.; Li, C.G. Online semi-supervised support vector machine. *Inf. Sci.* **2018**, *439*, 125–141. [CrossRef]
5. Nan, S.; Sun, L.; Chen, B.; Lin, Z.; Toh, K.A. Density-dependent quantized least squares support vector machine for large data sets. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *28*, 94–106. [CrossRef] [PubMed]
6. Kim, J.S.; Jeong, J.S. Pattern Recognition of Ship Navigational Data Using Support Vector Machine. *Int. J. Fuzzy Log. Intell. Syst.* **2015**, *15*, 268–276. [CrossRef]
7. Maza, S.; Touahria, M. Feature selection algorithms in intrusion detection system: A survey. *KSII Trans. Internet Inf. Syst.* **2018**, *12*, 5079–5099.
8. Bostani, H.; Sheikhan, M. Hybrid of binary gravitational search algorithm and mutual information for feature selection in intrusion detection systems. *Soft Comput.* **2017**, *21*, 2307–2324. [CrossRef]
9. Xiao, X.; Zhang, S.; Mercaldo, F.; Hu, G.; Sangaiah, A.K. Android malware detection based on system call sequences and LSTM. *Multimed. Tools Appl.* **2019**, *78*, 3937–3999. [CrossRef]
10. Jiang, G.; Chen, H.; Ungureanu, C.; Yoshihira, K. Multiresolution abnormal trace detection using varied-length n-grams and automata. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2007**, *37*, 86–97. [CrossRef]
11. Laszka, A.; Abbas, W.; Sastry, S.S.; Vorobeychik, Y.; Koutsoukos, X. Optimal thresholds for intrusion detection systems. In Proceedings of the Symposium and Bootcamp on the Science of Security (HotSos'16:), Pittsburgh, PA, USA, 20–21 April 2016; pp. 72–81.
12. Tan, K.M.; Maxion, R.A. "Why 6?" Defining the operational limits of Stide, an anomaly-based intrusion detector. In Proceedings of the 2002 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 12–15 May 2002; pp. 188–201.
13. Alarifi, S.S.; Wolthusen, S.D. Detecting anomalies in IaaS environments through virtual machine host system call analysis. In Proceedings of the 2012 International Conference for Internet Technology and Secured Transactions, London, UK, 10–12 December 2012; pp. 211–218.
14. Xue, B.; Cervante, L.; Shang, L.; Browne, W.N.; Zhang, M. A multi-objective particle swarm optimisation for filter-based feature selection in classification problems. *Connect. Sci.* **2012**, *24*, 91–116. [CrossRef]
15. Qu, G.; Hariri, S.; Yousif, M. A new dependency and correlation analysis for features. *IEEE Trans. Knowl. Data Eng.* **2005**, *17*, 1199–1207. [CrossRef]
16. Luo, B.; Xia, J. A novel intrusion detection system based on feature generation with visualization strategy. *Expert Syst. Appl.* **2014**, *41*, 4139–4147. [CrossRef]
17. Nauman, M.; Azam, N.; Yao, J. A three-way decision making approach to malware analysis using probabilistic rough sets. *Inf. Sci.* **2016**, *374*, 193–209. [CrossRef]
18. Ahmed, M.; Mahmood, A.N.; Hu, J. A survey of network anomaly detection techniques. *J. Netw. Comput. Appl.* **2016**, *60*, 19–31. [CrossRef]
19. Pektas, A.; Acarman, T. Deep learning for effective Android malware detection using API call graph embeddings. *Soft Comput.* **2019**, *24*, 1027–1043. [CrossRef]
20. Forrest, S.; Hofmeyr, S.A.; Somayaji, A. The Evolution of System-Call Monitoring. The evolution of system-call monitoring. In Proceedings of the 24th Annual Computer Security Applications Conference, Anaheim, CA, USA, 8–12 December 2008; pp. 418–430.
21. Forrest, S.; Beauchemin, C.; Somayaji, A. Computer immunology. *Immunol. Rev.* **2007**, *216*, 176–197. [CrossRef]
22. Haxhibeqiri, J.; Moerman, I.; Hoebeke, J. Low overhead scheduling of LoRa transmissions for improved scalability. *IEEE Internet Things J.* **2018**, *6*, 3097–3109. [CrossRef]
23. Xu, L.; Zhang, D.; Alvarez, M.A.; Morales, J.A.; Ma, X.; Cavazos, J. Dynamic android malware classification using graph-based representations. In Proceedings of the 2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing, Beijing, China, 25–27 June 2016; pp. 220–231.
24. Patgiri, R. HFil: A High Accuracy Bloom Filter. In Proceedings of the IEEE 21st International Conference on High Performance Computing and Communications, Zhangjiajie, China, 10–12 August 2019; pp. 1–12.
25. Ye, Q.; Wu, X.; Yan, B. An Intrusion Detection Approach Based on System Call Sequences and Rules Extraction. In Proceedings of the IEEE 2010 International Conference on E-business and Information System Security (EBISS), Wuhan, China, 22–23 May 2010; pp. 1–4.

26. Glass-Vanderlan, T.R.; Iannacone, M.D.; Vincent, M.S.; Bridges, R.A. A Survey of Intrusion Detection Systems Leveraging Host Data. *ACM Comput. Surv.* **2018**, *52*, 1–40.
27. Zhang, H.; Xiao, X.; Mercaldo, F.; Ni, S.; Martinelli, F.; Sangaiah, A.K. Classification of ransomware families with machine learning based on N-gram of opcodes. *Future Gener. Comput. Syst.* **2019**, *90*, 211–212. [CrossRef]
28. Laurén, S.; Rauti, S.; Leppänen, V. Diversification of system calls in linux kernel. In Proceedings of the 16th International Conference on Computer Systems and Technologies, Dublin, Ireland, 25 June 2015; pp. 284–291.
29. Durán, F.; Salaün, G. Robust and reliable reconfiguration of cloud applications. *J. Syst. Softw.* **2016**, *122*, 524–537. [CrossRef]
30. Bigyan, P. Thesis-Generating Knowledgebase of Common Behavior and Workflow Patterns for Secure Systems. Master' Thesis, East Carolina University, Greenville, NC, USA, 2018.
31. Kumar, G.R.; Mangathayaru, N.; Narasimha, G. An approach for intrusion detection using text mining techniques. In Proceedings of the International Conference on Engineering & MIS (ICEMIS), Istanbul, Turkey, 24–26 September 2015; pp. 63–74.
32. Dimjašević, M.; Atzeni, S.; Ugrina, I.; Rakamaric, Z. Evaluation of android malware detection based on system calls. In Proceedings of the ACM on International Workshop on Security and Privacy Analytics, New Orleans, LA, USA, 11–12 March 2016; pp. 1–8.
33. Nissim, N.; Lapidot, Y.; Cohen, A.; Elovici, Y. Trusted system-calls analysis methodology aimed at detection of compromised virtual machines using sequential mining. *Knowl. Based Syst.* **2018**, *153*, 147–175. [CrossRef]
34. Zuzcak, M.; Sochor, T.; Zenka, M. Intrusion detection system for home windows based computers. *KSII Trans. Internet Inf. Syst.* **2019**, *13*, 4706–4726.
35. Wang, Q.H.; Ouyang, X.Q.; Zhan, J.C. A classification algorithm based on data clustering and data reduction for intrusion detection system over big data. *KSII Trans. Internet Inf. Syst.* **2019**, *13*, 3714–3732.
36. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *Comput. Sci.* **2017**, *5*, 16–29.
37. Pan, J.; Liu, S.; Sun, D.; Zhang, J.; Liu, Y.; Ren, J.; Li, Z.; Tang, J.; Lu, H.; Tai, Y.-W.; et al. Learning Dual Convolutional Neural Networks for Low-Level Vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake, UT, USA, 18–23 June 2018; pp. 3070–3079.
38. Wang, X.; Girshick, R.; Gupta, A.; He, K. Non-local neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake, UT, USA, 18–23 June 2018; pp. 7794–7803.
39. Collobert, R.; Weston, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 7–8 July 2008; pp. 160–167.
40. Hirschberg, J.; Manning, C.D. Advances in natural language processing. *Science* **2015**, *349*, 261–266. [CrossRef]
41. Goldberg, Y. A Primer on Neural Network Models for Natural Language Processing. *Comput. Sci.* **2015**, *4*, 67–77. [CrossRef]
42. Shah, B.; Trivedi, B.H. Artificial Neural Network based Intrusion Detection System: A Survey. *Int. J. Comput. Appl.* **2012**, *39*, 13–18. [CrossRef]
43. Staudemeyer, R.C.; Omlin, C.W. Evaluating performance of long short-term memory recurrent neural networks on intrusion detection data. In Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference (SAICSIT 13), East London, South Africa, 7–9 October 2013; p. 218.
44. Kim, G.; Yi, H.; Lee, J.; Paek, Y.; Yoon, S. LSTM-Based System-Call Language Modeling and Robust Ensemble Method for Designing Host-Based Intrusion Detection Systems. *arXiv* **2016**, arXiv:1611.01726.
45. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
46. Chen, L.; Sultana, S.; Sahita, R. HeNet: A Deep Learning Approach on Intel circled Processor Trace for Effective Exploit Detection. *arXiv* **2018**, arXiv:1801.02318.
47. Chen, Q.; Luley, R.; Wu, Q.; Bishop, M.; Linderman, R.W.; Qiu, Q. AnRAD: A neuromorphic anomaly detection framework for massive concurrent data streams. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 1622–1636. [CrossRef] [PubMed]
48. Naseer, S.; Saleem, Y. Enhanced network intrusion detection using deep convolutional neural networks. *Ksii Trans. Internet Inf. Syst.* **2018**, *12*, 5159–5178.
49. Ambusaidi, M.A.; He, X.; Nanda, P.; Tan, Z. Building an intrusion detection system using a filter-based feature selection algorithm. *IEEE Trans. Comput.* **2016**, *65*, 2986–2998. [CrossRef]

50. Creech, G.; Hu, J. A semantic approach to host-based intrusion detection systems using contiguous and discontiguous system call patterns. *IEEE Trans. Comput.* **2014**, *63*, 807–819. [CrossRef]
51. Khreich, W.; Murtaza, S.S.; Hamou-Lhadj, A.; Talhi, C. Combining heterogeneous anomaly detectors for improved software security. *J. Syst. Softw.* **2018**, *137*, 415–429. [CrossRef]
52. Abdlhamed, M.; Lifayat, K.; Shi, Q.; Hurst, W. Intrusion prediction systems. In *Information Fusion for Cyber-Security Analytics*; Springer International Publishing: Cham, Germany, 2019; pp. 1358–1363.
53. Lv, S.; Wang, J.; Yang, Y.; Liu, J. Intrusion prediction with system-call sequence-to-sequence model. *IEEE Access.* **2018**, *6*, 1358–1363. [CrossRef]
54. Zhang, Z.; Peng, Z.; Zhou, Z. The Study of Intrusion Prediction Based on HsMM. In Proceedings of the Asia Pacific Services Computing Conference, 2008 (APSCC '08), Yilan, Taiwan, 9–12 December 2008; pp. 1358–1363.
55. Watson, M.R.; Marnerides, A.K.; Mauthe, A.; Hutchison, D. Malware detection in cloud computing infrastructures. *IEEE Trans. Dependable Secur. Comput.* **2016**, *13*, 192–205. [CrossRef]
56. Xie, M.; Hu, J.; Slay, J. Evaluating host-based anomaly detection systems: Application of the one-class SVM algorithm to ADFA-LD. In Proceedings of the 2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'14), Xiamen, China, 19–21 August 2014; pp. 978–982.
57. Kang, Q.; Shi, L.; Zhou, M.; Wang, X.; Wu, Q.; Wei, Z. A distance-based weighted under sampling scheme for support vector machines and its application to imbalanced classification. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *99*, 4152–4165.
58. Chandra, M.A.; Bedi, S.S. Survey on SVM and their application in image classification. *Int. J. Inf. Technol.* **2018**, *3*, 1–11. [CrossRef]
59. Alabdallah, A.; Awad, M. Using weighted support vector machine to address the imbalanced classes problem of intrusion detection system. *KSII Trans. Internet Inf. Syst.* **2019**, *12*, 5143–5158.
60. Abe, S. Fuzzy support vector machines for multilabel classification. *Pattern Recognit.* **2015**, *48*, 2110–2117. [CrossRef]
61. Huang, H.P.; Liu, Y.H. Fuzzy support vector machines for pattern recognition and data mining. *Int. J. Fuzzy Syst.* **2002**, *4*, 826–835.
62. Deepak, G.; Bharat, R.; Parashjyoti, B. A fuzzy twin support vector machine based on information entropy for class imbalance learning. *Neural Comput. Appl.* **2018**, *31*, 7153–7164.
63. Chen, S.G.; Wu, X.J. A new fuzzy twin support vector machine for pattern classification. *Int. J. Mach. Learn. Cybern.* **2018**, *9*, 1553–1564. [CrossRef]
64. Wang, Y.; Wang, S.; Lai, K.K. A new fuzzy support vector machine to evaluate credit risk. *IEEE Trans. Fuzzy Syst.* **2006**, *13*, 820–831. [CrossRef]
65. An, W.; Liang, M. Fuzzy support vector machine based on within-class scatter for classification problems with outliers or noises. *Neurocomputing* **2013**, *110*, 101–110. [CrossRef]
66. Jiang, X.; Yi, Z.; Lv, J.C. Fuzzy SVM with a new fuzzy membership function. *Neural Comput. Appl.* **2006**, *15*, 268–276. [CrossRef]
67. Lin, C.F.; Wan, S.D. Fuzzy support vector machines. *IEEE Trans. Neural Netw.* **2002**, *13*, 464–471.
68. Xiang, Z.; Xiao-Ling, X.; Guang-You, X.U. Fuzzy Support Vector Machine Based on Affinity among Samples. *J. Softw.* **2006**, *17*, 951–958.
69. Zhang, X. Using class-center vectors to build support vector machines. In Proceedings of the IEEE Signal Processing Society Workshop, Madison, WI, USA, 25 August 1999; pp. 3–11.
70. Warrender, C.; Forrest, S.; Pearlmutter, B. Detecting intrusions using system calls: Alternative data models. In Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, USA, 14 May 1999; pp. 133–145.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).