Research Article

An Approach Based on the Improved SVM Algorithm for Identifying Malware in Network Traffic

Bo Liu , Jinfu Chen , Songling Qin, Zufa Zhang, Yisong Liu, Lingling Zhao, and Jingyi Chen

School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, China

Correspondence should be addressed to Jinfu Chen; jinfuchen@ujs.edu.cn

Received 3 February 2021; Revised 28 March 2021; Accepted 20 April 2021; Published 30 April 2021

Academic Editor: Entao Luo

Copyright © 2021 Bo Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Due to the growth and popularity of the internet, cyber security remains, and will continue, to be an important issue. There are many network traffic classification methods or malware identification approaches that have been proposed to solve this problem. However, the existing methods are not well suited to help security experts effectively solve this challenge due to their low accuracy and high false positive rate. To this end, we employ a machine learning-based classification approach to identify malware. The approach extracts features from network traffic and reduces the dimensionality of the features, which can effectively improve the accuracy of identification. Furthermore, we propose an improved SVM algorithm for classifying the network traffic dubbed Optimized Facile Support Vector Machine (OFSVM). The OFSVM algorithm solves the problem that the original SVM algorithm is not satisfactory for classification from two aspects, i.e., parameter optimization and kernel function selection. Therefore, in this paper, we present an approach for identifying malware in network traffic, called Network Traffic Malware Identification (NTMI). To evaluate the effectiveness of the NTMI approach proposed in this paper, we collect four real network traffic datasets and use a publicly available dataset CAIDA for our experiments. Evaluation results suggest that the NTMI approach can lead to higher accuracy while achieving a lower false positive rate compared with other identification methods. On average, the NTMI approach achieves an accuracy of 92.5% and a false positive rate of 5.527%.

1. Introduction

With the growth of the internet, network attacks are becoming increasingly frequent, and cyber security has become a problem that security experts urgently need to solve. Since we cannot prevent the generation of network attacks, an alternative approach is to automatically identify malware in network traffic. There are many network traffic classification methods for identifying malware in network traffic. However, these methods have two major drawbacks, i.e., incurring low accuracy and leading to high false positive rate. For this purpose, we propose an identification approach for malware in network traffic.

For machine learning-based classifiers, the first step is to extract the features of the data. However, not every feature will have the same impact on the classification. This means that some features are easier to use for classification, while

others play a minimal role. Additionally, for large-scale data, we extract a very large number of features, which is not conducive to classification. Therefore, we need to preprocess the data. In this paper, we apply the stratified sampling technique to sample some data from the original dataset as experimental samples. The samples extracted by this technique will restore the features of the dataset to the maximum extent and will not generate too many redundant features. We first apply ReliefF algorithm [1] to extract the features of the network traffic in this paper. To further improve the accuracy of the classifier, we perform a dimensionality reduction operation on the extracted features. This is equivalent to turning a complex problem with high dimensionality into a simple problem with low dimensionality, which helps in classification.

Each machine learning-based classifier has its own suitable application scenario. In [2], Shafiq et al. verified that the

decision tree gives the best classification results for network traffic. However, Cao et al. [3] improved the SVM algorithm to obtain higher accuracy than the decision tree. In this paper, we improve the SVM algorithm for classifying network traffic in terms of both parameter optimization and kernel function selection. In our independent experiments, on average, the radial basis function kernel can achieve an accuracy of 88.3%. Therefore, we finally choose radial basis function kernel as the kernel function of the SVM algorithm. This leads to the design and implementation of an improved SVM algorithm, called OFSVM. The accuracy of the sigmoid kernel is comparable to that of the radial basis function kernel. However, the overall result is not as good as the radial basis function kernel, i.e., with the average results, 85.92% for the sigmoid kernel vs. 88.3% for the radial basis function kernel.

This paper proposes an approach for identifying malware in network traffic by preprocessing the original dataset and applying feature extraction as well as feature dimensionality reduction to extract the major features and finally classify them using the OFSVM algorithm proposed in this paper to identify malware in network traffic. To show the usefulness of the NTMI approach, we apply it to four network traffic datasets we collected and a public common dataset. Furthermore, we compared the approach with existing identification methods (namely, SVM [4], LA-SVM [5], naive Bayes [6], and decision tree [7]). Experimental results show that the NTMI approach can achieve higher accuracy and lower false positive rate. Considering four datasets, on average, the NTMI can achieve 92.5% accuracy and 5.527% FPR.

In this paper, we present a malware identification approach and make three contributions.

First, we propose an improved SVM algorithm for more accurate classification of network traffic from two aspects: parameter optimization and kernel function selection. Additionally, we compare it with SVM algorithms using other kernel functions. Considering all together, it can be concluded that the algorithm proposed in this paper achieves the highest accuracy.

Second, we sample the original dataset and process the data using feature dimensionality reduction methods. The purpose of all these operations is to extract the major features of the data to reduce the impact of redundant and minor features on the classification results. The classification is then performed using the OFSVM algorithm proposed in this paper. Evaluation results demonstrate that the NTMI approach can lead to the highest identification performance compared to other methods.

Third, to show the usefulness of the NTMI approach, we capture real network traffic at different times of the week and selected 10% of the data in the public dataset CAIDA as the experimental data.

The rest of the paper is organized as follows. In Section 2, we present some previous methods related to network traffic classification, and in Section 3, we describe the design and implementation of the NTMI approach proposed in this paper. Section 4 introduces the experimental setup and discusses the experimental results. The conclusions are summarized in Section 5.

2. Related Work

Many scholars have already conducted research in network traffic classification or identification of malware in network traffic. Each method has its advantages and is worthy of later scholars. In this section, we present some related preliminary works.

Shafiq et al. [2] discussed network traffic classification techniques and captured real-time internet datasets. Additionally, they applied feature extraction tools to extract features and classified network traffic using four machine learning methods: support vector machine, C4.5 decision tree, naive Bayes, and Bayes net. Experimental results suggest that C4.5 decision tree can obtain a more accurate classification result compared to other classifiers. In [8], Yang et al. found that the parameters transmitted by the application layer vary according to protocols and proposed utilizing decision trees based on the minimum partition distance to perform classification. Experiments show that intercepting the first 4 or 6 packets can shorten the classification time and have higher accuracy, thus proving the effectiveness of the method. Soysal and Schmidt [9] utilized three supervised machine learning algorithms, namely, Bayes networks, decision trees, and multilayer perceptrons, for flow-based classification of network traffic. Additionally, they investigated the effect of the amount and composition of training data on the traffic classification performance. Experiments show that ML algorithms such as Bayes networks and decision trees are suitable for high-speed internet traffic classification and emphasize the importance of correctly classifying training examples.

Liu et al. [10] employed the *K*-means clustering algorithm to build classifiers using statistical information as input vectors. Experimental results on different datasets suggest that the method can obtain an overall accuracy of up to 80%, which increases to more than 90% after log transformation. It is demonstrated experimentally that the *K*-means performs well for traffic classification. Shrivastav and Tiwari [11] proposed a semisupervised method for classifying network traffic, which can design classifiers from training data consisting of only a few labeled traffic and many unlabeled flows. The method uses the *K*-means clustering algorithm to partition the training dataset into disjoint clusters and perform classification. Experiments demonstrate that the test error rate depends on the number of clusters randomly used in the training phase. Furthermore, the accuracy of the classifier ranges from 70% to 96% for various datasets.

Teufl et al. [12] proposed a framework to simplify empirical model selection and feature extraction, called Intelligent Feature-based Classification Tool (InFeCT). InFeCT analyzes network traffic to check whether the data in the traffic violate a certain rule and extracts the best set of features from the data to build a traffic classification model to classify the network traffic. Bekerman et al. [13] proposed an end-to-end surveillance-based system to detect unknown malware using network traffic classification. The classification method extracts behavioral features and applies feature selection methods to identify the most meaningful features

while reducing the data dimensionality. The accuracy of the proposed method is experimentally demonstrated to be effective in both sandbox and real networks, and the method can detect most modern malware as well as new and unknown malware. In [14], Mu and Wu proposed a parallelized network traffic classification method based on the hidden Markov model using packet-level properties in network traffic flows. Experimental results suggest that the classification method can obtain a high accuracy, giving more than 90% accuracy on the collected dataset. The most common technique used in network traffic classification is the machine learning approach.

Sethi and Behera [15] proposed using Deep Packet Inspection algorithm for internet traffic classification. The method achieves the classification of network traffic by analyzing and processing the data based on parameters such as the data to be searched, the time of searching, available bandwidth, the number of accessing users, and architecture of the network system, using clustering methods in machine learning and signature techniques. Rezaei and Liu [16] proposed a general deep learning-based framework for traffic classification and introduced common deep learning methods as well as their application in traffic classification tasks. Lim et al. [17] proposed using the convolutional neural network and residual network for network traffic classification. Experimental results show that the use of deep learning models for network traffic classification is effective and that the residual network outperforms the convolutional neural network for classification.

3. Research Methodology

The objective of this paper is to identify malware in network traffic. For this purpose, we present a malware identification approach dubbed Network Traffic Malware Identification (NTMI). The approach consists of three steps. NTMI first extracts features from network traffic, then reduces the dimensionality of the extracted features, and finally utilizes the improved SVM algorithm for classification to identify malware in network traffic.

3.1. Feature Extraction. The first step in classifying network traffic using machine learning techniques is to extract the features of the network traffic. In this section, we first describe how to collect, sample, and normalize the data.

3.1.1. Data Collection. To identify malware in network traffic, we first extract features from the collected network traffic data. We use the NetFlow tool [18] to collect the network traffic data. It is a lightweight tool that monitors all traffic passing through a port during a specified time period and then gets the packet version, number, buffer size, and other information.

3.1.2. Data Sampling. Furthermore, instead of extracting features directly from the collected network traffic, we first perform data sampling to select a better subset. The aim of

data sampling is to select some data as a subset of the entire dataset and sample it for observation because the subset inherits the features of the original dataset, thus allowing the evaluation of the whole dataset. Data sampling is divided into three categories: systematic sampling, random sampling, and stratified sampling. Systematic sampling is the sampling of a portion of the total sample according to a certain sampling distance. Random sampling refers to the random selection of some sample data from the entire sample data. Stratified sampling means that the entire data sample set is first stratified according to specified rules, and then some data are randomly selected from each stratum according to a specified proportion. In this paper, we take stratified sampling to select the sample data.

3.1.3. Data Normalization. If the selected dataset is anomalous, it will eventually affect the effectiveness of malware identification. Therefore, we need to normalize the dataset. By specifying the feature attributes of all data in a range, the normalized data can reduce the training time and improve the classification performance. If they are out of the specified interval range, the data will be excluded, thus helping to classify the network traffic, and the identification model built on this basis will improve the efficiency of identifying malware.

3.1.4. Traffic Feature Extraction. To achieve the identification of malware in network traffic, it is indispensable to extract the feature attributes of the data transmitted in the network and build a malware identification model. Through studying the behavior of network attacks, we find that malware has some common features which help us to identify them. An attacker will attack multiple commonly used ports or ports that have been closed in a short period of time, for example, anomalous packets that send only SYN or FIN packets, a great number of false connections or REJ packets, and plenty of network traffic packets. If common features of network traffic data packets can be extracted, it will improve the accuracy of malware identification.

Commonly used feature extraction methods are SNMP protocol technology [19] and probe technology [20]. SNMP protocol technology monitors the network links, but the features obtained by this technology are too few to be classified. Probe technology can be applied to the links of network traffic to obtain the traffic features quickly. However, it is not suitable for large-scale traffic feature extraction and is too time consuming. Additionally, the technique focuses on the extraction of protocol features, which cannot accurately parse the packet messages' information. This paper applies the ReliefF algorithm [1] for feature extraction, which is superior to the aforementioned common feature extraction methods. This technology compares the correlation of sample types and feature attributes on the processed dataset. The weight will keep increasing as the correlation becomes higher, and a threshold is set. If the weight corresponding to the correlation between the feature attribute and the sample type exceeds the set threshold, we keep the feature attribute; otherwise, we discard the attribute.

Furthermore, if more than one feature attribute appears in a packet, the feature attribute that appears most frequently is selected. The specific feature selection process is as follows:

- (i) Select some samples s randomly from the dataset D in a stratified sampling manner
- (ii) Select y samples r of the same type D_a nearest to the sample s
- (iii) Select y samples t from different types of D_b
- (iv) Calculate the distance between sample s and sample r as D_{sr} and the distance between sample s and sample t as D_{st}

If $D_{sr} > D_{st}$, it means that the feature attribute is problematic and cannot be utilized for classification, and we set a smaller weight value; otherwise, this feature attribute is helpful for classification, and we set a larger weight value. The formula for calculating the feature weights is shown in equation (1), where $w(x)$ is the corresponding weight, $d(x, r, t)$ represents the Euclidean distance of sample r , sample t , and feature x , D_j is the j -th sample data in the dataset, and n refers to the calculation of the weight size in n data for feature extraction. Loop through the above process, and the final computed weights are compared with the set threshold. If the requirements are met, the feature attribute is retained; otherwise, it is discarded. Finally, we can get the set S of extracted feature attributes.

$$w(x) = \sum_{i=1}^n \frac{d(x, r, t)}{y^n} + \sum_{D_j \in D} \frac{(1/(1 - p(D_j))) \sum_{i=1}^n d(x, r, t)}{y^n}. \quad (1)$$

Through the ReliefF algorithm [1], Table 1 records some of the extracted network traffic feature attributes.

3.2. Feature Dimensionality Reduction. After feature extraction is performed on network traffic, a certain traffic data packet contains a variety of feature attributes, which poses a complex high-dimensional feature space problem for the classification of network traffic. Some redundant features not only lead to increased learning complexity of the classification algorithm but also cause overfitting and local optimization problems. When the proportion of key features for malware identification is small, the final identification result will be poor. To solve the above problems, this study chooses key feature combinations to achieve dimensionality reduction of traffic features to help build the corresponding malware identification model.

The extracted feature attributes are first added to the set S . We propose utilizing the filter feature dimensionality reduction method with the help of the information gain technology [21], i.e., $E_{IG} = \text{evaluate}(F_{\text{filter}}, S)$. The algorithm is an evaluation of the information gain on the set S of feature attributes. By evaluating the impact of each feature attribute on the subsequent classification, it is determined whether to update the value of E_{IG} and the feature attribute set S , where the information gain value Test_{IG} of the candidate feature subset is calculated. When $\text{Test}_{IG} > \text{Test}_{\text{best}}$, the evaluation

value and feature subset will be updated; otherwise, they will not be updated. And then, the heuristic search strategy [22] is used to sort the feature attributes to obtain the feature attribute set S_1 . The process is repeated until the specified number of times is reached. Based on this, we employ the wrapper method [23] for secondary feature selection, and the heuristic sequence forward search method is used to obtain the feature attribute set S_2 . After the feature dimensionality reduction, it not only reduces the time and computational complexity but also improves the classification effect.

When using the wrapper method, equation (2) calculates the correlation of the traffic feature attributes to perform a secondary selection of the feature attributes, where n represents the number of feature attributes for all initial selections, k_{∂} represents the feature attribute coefficient, m_{ri} represents the average of the traffic feature attribute for the i -th packet, $h_{r\partial}^2$ is the corresponding variance, and m_r represents the average of the traffic feature attribute r .

$$F_r = \frac{1}{k_{\partial}} \cdot \frac{\sum_{i=1}^n k_{\partial} (m_{ri} - m_r)^2}{\sum_{i=1}^n k_{\partial} h_{r\partial}^2}. \quad (2)$$

After performing feature dimensionality reduction by using the above method, the previously extracted feature set can be further simplified to eliminate redundant features. Additionally, some of the features selected in this way are almost uncorrelated with each other, which is more conducive to classification. The final feature dimensionality reduction set is presented in Table 2.

We obtain a subset of feature attributes after feature dimensionality reduction. However, these feature attributes with different units and measurement criteria are not related to each other. Therefore, this study proposes to normalize the subset of feature attributes.

The specific normalization process is as follows. We utilize min-max normalization to process the data. Linear transformation on the acquired feature subset is performed to convert the target dataset into between 0 and 1 [11], using the conversion function as follows:

$$f' = \frac{f - \min}{\max - \min}. \quad (3)$$

In this formula, $\min$ refers to the minimum value of the sample data, and $\max$ refers to the maximum value of the sample data. However, this method has the disadvantage that continuing to add data to it during the target transformation will cause $\max$ and $\min$ to be changed, thus affecting the normalization criteria. Therefore, before the normalization process, it is necessary to ensure that the dataset will remain constant.

3.3. OFSVM Algorithm. This research will improve the existing support vector machine (SVM) classification method [23] and finally realize the improvement of the classification of the program in the network traffic. The first section introduces the shortcomings of the current SVM algorithm for classifying network traffic, and the second section proposes the improvement of the algorithm.

TABLE 1: Extracted feature attributes.

Feature name	Feature description
origin_ip	Source IP address
destination_ip	Destination IP address
port_number	Port number
duration	Connection duration
protocol_type	Protocol type
service	Type of network service of the destination host
flag	Connection normal or error state, and this field is discrete type
src_bytes	Number of bytes of data from the source host to the destination host
dst_bytes	Number of bytes of data from the destination host to the source host
wrong_fragment	Number of wrong fragments, and this field is continuous type
urgent	Number of urgent packages, and this field is continuous type
dst_host_srv_error_rate	Percentage of connections with SYN errors
hot	Number of accesses to sensitive files and directories on the system
mark_status	Mark status
packet_rate	Packet sending rate
max_pktLens	Maximum message length
min_pktLens	Minimum message length
num_compromised	Number of occurrences of compromised condition
num_access_files	Number of access control files
same_srv_rate	Percentage of connections with the same service as the current connection
dst_host_srv_count	Number of connections with the same destination host service as the current connection

TABLE 2: Features after dimensionality reduction.

Feature name	Feature description
origin_ip	Source IP address
destination_ip	Destination IP address
duration	Connection duration
flag	Connection normal or error state, and this field is discrete type
src_bytes	Number of bytes of data from the source host to the destination host
dst_bytes	Number of bytes of data from the destination host to the source host
wrong_fragment	Number of wrong fragments, and this field is continuous type
mark_status	Mark status
packet_rate	Packet sending rate
max_pktLens	Maximum message length
min_pktLens	Minimum message length
same_srv_rate	Percentage of connections with the same service as the current connection
dst_host_srv_count	Number of connections with the same destination host service as the current connection

3.3.1. Existing SVM Algorithm and Its Shortcomings.

SVM is a model for binary classification which is mainly used to find the maximum interval in the feature space. The objective of the SVM is to find a hyperplane in all sample data so that the distance between the nearest data on both sides and the plane is the largest. The SVM algorithm can divide the data in the training set by separated hyperplanes, of which there may be an infinite number, but the one that makes the maximum interval is selected.

In network traffic, we assume that the current network traffic set is $N = \{N_1, N_2, \dots, N_k\}$, and its corresponding feature set is $F = \{F_1, F_2, \dots, F_k\}$. Then, the SVM algorithm is used to construct the network traffic classification model and implement the classification of network traffic, i.e., malware or nonmalware. The SVM classification method can choose a relatively optimal classification plane to build a model for the classification and can complete a relatively

stable classification under the condition of unknown sample classification. There is a lot of noise in the real network environment and a lot of unprocessed redundant features in the sample data, both of which lead to low accuracy of the classification results. In this paper, we propose to optimize the SVM algorithm in terms of parameter optimization and suitable kernel functions. The method utilizes grid search parameter optimization [24] to prevent overfitting in order to find the optimal solution. Additionally, we introduced fuzzy factors to improve the accuracy of the classification [25]. This study uses the distance from the sample to the classification hyperplane to design the fuzzy factor because this approach removes the effect of noise while reducing the effect of classification plane shape on accuracy. And then, we use the feature validity [26] to eliminate the effect of redundant features. Finally, considering the importance of kernel function parameters on the classification

performance, this paper chooses the radial basis kernel function [27] to optimize the SVM algorithm.

3.3.2. Parameter Optimization. Several researchers have already improved the classification capability of the SVM algorithm, for example, genetic algorithm [28], particle swarm algorithm [29], and artificial fish swarm algorithm [30]. However, these classification algorithms still have some deficiencies in terms of stability and accuracy. Therefore, we present a new algorithm to improve the SVM algorithm, called OFSVM algorithm. This study will fully consider the complexity of real network traffic and the reasons for the decline in identification accuracy.

Parametric optimization of the SVM is mainly to find a convergent optimal solution in a finite number of searches using some search strategy in a space of many parameters. In this step, we consider two important parameters: the kernel function parameter and the penalty parameter. Among them, the penalty parameter will play a decisive role in the generalization capability of the SVM hyperplane, which is mainly used to indicate the fault tolerance when constructing the hyperplane. And the kernel function parameter will determine the scope of action, which will also affect the generalization capability of the SVM. Therefore, with the aim of finding the optimal parameter combination in a limited number of searches, we propose to employ grid search to optimize parameters to improve the SVM algorithm.

The principle of grid search used in this paper is as follows, and it consists of four main steps, which we have briefly summarized:

- (i) Delineating the k -dimensional parameter space, where grid nodes are used to represent the candidate parameters
- (ii) Sampling at the specified step and generating the corresponding set
 $P(c_i) = \{P(c_1) \times P(c_2) \times \dots \times P(c_k)\}$
- (iii) Setting the range of the parameter c_i to generate grids with different orientations
- (iv) Evaluating each grid node c_i according to the specified evaluation method, and outputting the final approximate optimal solution

In this process, the incremental is first set to be times the default step size q , i.e., $q \cdot t$. This step is to reduce the search time and the density of the generated grid. Then, all sample data are searched iteratively to obtain the optimal combination of parameters. To express the fault tolerance of the sample data when constructing the classification plane, a penalty parameter P is introduced and compared with the set overfitting threshold f . When P is less than f , narrow the search space and set the step size of the search to half of the initial step size, and search again. The reduction of step length is to expand the density of the grid and thus achieve a more accurate search. If P exceeds f , expand the search space and adjust the search direction for another search. The purpose of this step is to optimize the parameters and

prevent overfitting. Looping through the sample data until the penalty parameter P is within the critical range, the optimal parameter combination value is outputted. The algorithm has a large searchable space, and the nodes are uncorrelated with each other, so it is more generalizable.

To further improve the classification accuracy, fuzzy factors are first introduced. In this study, the distance from the sample to the classification hyperplane will be used to design the fuzzy factor, which will reduce the impact of the classification plane shape on the classification accuracy. On this basis, the corresponding classification hyperplane is constructed firstly. And then, the distance from each sample node to the hyperplane is calculated so that the fuzzy factor can be used to eliminate the effect of excess noise. Accordingly, it is proposed to construct feature validity to eliminate the influence of redundant features on classification accuracy.

For each sample point s_i , there is a corresponding fuzzy factor e_i , which represents the uncertainty of the sample distribution, where $0 \leq e_i \leq 1$. R^+ and R^- represent the mean point of the positive and negative samples, and normal vector can be expressed by $\alpha = \sqrt{R^+ - R^-}$. According to the method in [31], the corresponding hyperplane can be expressed as $(s - R)^2 \cos \alpha^T = 0$. In this way, the distance from the sample point to the hyperplane is described in equation (4), and then the maximum distance d_1 from the positive sample point to the hyperplane can be obtained if and only if R is R^+ . In the same way, when R is R^- , d_2 is the maximum distance from the negative sample point to the hyperplane. Then, the regulatory factor ∂ is used to make $0 < e_i \leq 1$. The fuzzy factor is shown in equation (5), where the value of d is d_1 and d_2 , respectively, in different positive and negative samples. Thus, the effect of excess noise on the classification accuracy is eliminated by using different fuzzy factors. Because the impact of different features on the classification is not considered, this paper proposes to introduce feature validity to eliminate the impact of weakly correlated features on classification accuracy.

$$d = \frac{(s_i - R)^2 \cos \alpha^T}{\|\alpha\|^2}, \quad (4)$$

$$e_i = \frac{\cos \alpha^T}{d \partial} - \ln \partial. \quad (5)$$

In [26], for each feature i of the sample data, it has a corresponding feature validity h_s^i , which can indicate the degree of influence of a certain feature used for classification. The greater the classification capability of feature i , the greater its feature validity h_s^i . In feature set S , the classification effect of each feature is judged by calculating the enhanced learning capability of each feature. If the training sample set S has a total number of $|S|$ and there are p feature attributes in a sample, the feature validity can be expressed as equation (6). When the enhancement learning value of a certain feature i is relatively large, its feature validity is relatively large, that is, its contribution to the classification will be relatively high. Finally, considering the importance of kernel function parameters on the classification

performance, this study optimizes the SVM algorithm by selecting the appropriate kernel functions.

$$h_s^i = \frac{|S|^2 \ln p}{\sum_{i=1}^n |S| - p^2}. \quad (6)$$

3.3.3. Appropriate Kernel Functions. The kernel function is mainly used to map the original nonlinear sample data into the feature space and then convert the nonlinear sample into a linear classifiable problem by means of the constructed optimal classification plane, thus avoiding the huge amount of calculation of the high-dimensional feature space. Assume the input space is $P \in R^n$, and the corresponding feature space is F . When there is a mapping function $\gamma(y) = Y \rightarrow P$ and any y_i and y_j belonging to Y satisfy $K(y_i, y_j) = \gamma(y_i)^T \gamma(y_j)$, there is a kernel function K . The kernel function needs to satisfy Mercer's theorem [32], that is, for any vector in the input space, the corresponding kernel matrix should be a positive semi-definite matrix. After selecting the appropriate kernel function, the linear classification can be completed without increasing the complexity. Therefore, the classification effect of the SVM is greatly related to the kernel function. In this paper, we choose the radial basis kernel function as the kernel function. This function has good performance in the local range, and it can achieve a high classification efficiency for the sample points in the dataset. Furthermore, it is not constrained by the number of samples and feature dimensions. And the radial basis kernel function has fewer parameters, which makes the kernel function have a lower complexity. Algorithm 1 describes the improved algorithm OFSVM.

By improving the SVM algorithm in the above manner, the error is relatively small, and the identification of malware in network traffic is further improved. The input of the algorithm is the set of feature attributes to be trained as support vectors. The algorithm applies grid search for correlation search, which expands the search space and search density, and then completes the accurate search. The distance between each sample and the class is used as a fuzzy factor, and the proposed feature validity is used to eliminate the effect of redundant features on the classification accuracy. It also depends on the radial basis kernel function verified by experiments and elaborated in Section 4.3. The kernel function has a higher accuracy and is more stable and finally generates a classifier model. Its time complexity is $O(n * m)$, where n is the number of input sample feature attributes and m is the number of kernel function operations.

3.4. NTMI Approach. In the previous sections, we introduced how to extract the features of network traffic and reduce the dimensionality of the extracted features, respectively, and then overviewed the solutions for identifying malware in network traffic. Additionally, to address the inaccuracy of the traditional SVM classification, we present the OFSVM algorithm for classifying network traffic in terms of parameter optimization and appropriate kernel functions. In this section, we detail the identification model

building process and propose the identification approach for malware in network traffic, i.e., NTMI.

To identify malware, the first step is to solve the problem of accurate classification in network traffic. We first apply the NetFlow tool [18] to collect real network traffic. Second, the collected network traffic data are sampled and normalized to obtain a more valuable dataset for the experiment, while the processed data are more convenient for feature extraction. Third, we utilize the ReliefF algorithm [1] to extract the features of the data packets in the network traffic. Meanwhile, the extracted features still contain some redundant feature attributes. These feature attributes will greatly reduce the accuracy of network traffic classification. Therefore, we propose to reduce the dimensionality of the above extracted feature set. Feature dimensionality reduction consists of a total of 4 steps:

- (i) Calculating and evaluating each feature using the information gain technology
- (ii) Sorting the feature set
- (iii) Selecting secondary features using the wrapper method
- (iv) Calculating the correlation of the features adopting the heuristic sequence forward search method

Next, the obtained feature subset needs to be normalized, and all feature attributes are converted into numerical values which are then put into a matrix array to calculate the minimum Euclidean distance. Then, the OFSVM algorithm is used to train for a better classifier with the processed network traffic testing set as the input. This classifier can classify normal programs and malware in network traffic and finally identify malware in network traffic. Algorithm 2 describes the specific NTMI approach.

The input of the algorithm is the set of collected traffic packets, and the final output is the dataset of malware in network traffic. The time complexity in the process of feature extraction and feature dimensionality reduction is greater than the time for feature normalization, so the final time complexity of the algorithm is $O(p * k)$, where p is the number of data packets normalized and k is the number of extracted feature attributes, which can be approximated as $O(n^2)$. The NTMI approach is less costly compared to several other classification methods in the experimental section.

4. Experiments and Discussion

To verify the effectiveness of the NTMI approach for identifying malware in network traffic, we compare it with existing identification methods, i.e., SVM [4], LA-SVM [5], naive Bayes model (NBM) [6], and decision tree model (DTM) [7]. We conduct experiments on each of the five datasets selected for this paper. To avoid errors caused by a single experiment, we perform 100 experiments for each method separately and calculated the average accuracy and average false positive rate of the method as the final experimental results.

Input: *executedDataM*//the set of processed feature attributes
Output: *generatedClassifier*//the generated optimized classifier

- (1) Construct *fuzzyFactor* = *null*;//calculate the distance between each sample and the class as a fuzzy factor to improve the classification accuracy
- (2) Construct *executedDefaultStep* = *q*, *executedSearchStep* = *null*;//control search time and grid density
- (3) Construct *executedPenaltyParameter*;//express the fault tolerance of the sample data when constructing the classification plane of SVM
- (4) Construct *executedOverfittingThreshold* = *f*;//judge whether the penalty parameter is within the critical range
- (5) *representCandidateParameters*()//use grid nodes to represent candidate parameters
- (6) set the range of parameters to generate grids in different directions;
- (7) **for each** sample *i* in *executedDataM* **do**
- (8) Construct *executedSearchStep* = *q.t*;// the incremental step is *t* times the default step *q*
- (9) *constructTraverseSearch*()//perform traversal search on all samples
- (10) divide into *i*-dimensional parameter space among *i* parameters;
- (11) **if** (*executedPenaltyParameter*(*i*) < *executedOverfittingThreshold*) **then**
- (12) *executedSearchStep* = *2/q*;// reduce the step size to increase the grid density for a more accurate search
- (13) *constructTraverseSearch*()//perform traversal search on all samples
- (14) **else**
- (15) expand the search space and adjust the search direction;
- (16) *constructTraverseSearch*()//perform traversal search on all samples
- (17) **end if**
- (18) *panel* = *createClassificationHyperplane*()// construct the corresponding classification hyperplane
- (19) *calculateDistance*(*M*[*i*], *panel*)// calculate the distance between each sample node and the hyperplane as a fuzzy factor
- (20) *computeFeatureValidity*(*i*)// calculate the feature *i* of each sample data, which has a feature validity, and determine the classification effect of each feature
- (21) *useRadialBasisKernel*()// the kernel function has lower complexity and higher classification efficiency
- (22) **end for**
- (23) *generateClassification*()// generate the optimized classifier
- (24) return *generatedClassifier*;

ALGORITHM 1: OFSVM algorithm.

Input: *executedOriginalData*// the set of collected traffic data packets
Output: *identifyMaliciousData*// the set of identified malware

- (1) Construct *executedOriginalFeatureSet* = *null*// store feature attributes extracted from network traffic packets
- (2) Construct *identifyMaliciousData* = *null*;// the set of identified malware
- (3) Construct *executedNormalizationData* = *null*;// store normalized data
- (4) *executedOriginalData* = *collectNetworkFlow*()// use NetFlow to collect data packets for assignment
- (5) **for each** data package *p* in *executedOriginalData_{train}* **do**
- (6) *executedNormalizationData* = *dataNormalization*()// to complete data sampling and normalization
- (7) **end for**
- (8) **for each** data package *p* in *executedNormalizationData* **do**
- (9) *executedOriginalFeatureSet* = *useReliefFCompleteFeatureExtracted* (*executedNormalizationData_p*);
- (10) **for each** feature *k* in *executedOriginalFeatureSet* **do**
- (11) *temp* = *compare*(*executedOriginalFeatureSet_k*, ∂)// compare each extracted feature attribute *k* with a threshold ∂ and return the value *temp*
- (12) **if** (*temp* == 1) **then**
- (13) *deleteFeature*(*executedOriginalFeatureSet_k*)// delete this feature attribute
- (14) **end if**
- (15) **end for**
- (16) *executedFirstFeatureSet* = *outputFeatureExtraction*()// retain the feature attributes extracted from each packet
- (17) **end for**
- (18) **for each** feature *j* in *executedFirstFeatureSet* **do**
- (19) use information gain technology to calculate and evaluate each feature;
- (20) *normalizedFeature* = *sencondExtraction*(*executedFirstFeatureSet_j*); // sort feature attributes and use Wrapper for second feature extraction
- (21) **end for**
- (22) *realizeUnit*()// convert to unitless values and keep the data at the same order of magnitude
- (23) *classifyModel* = *useOFSVMAlgorim*(*normalizedFeature*)// generate the classification model
- (24) *identifyMalware* (*classifyModel*, *executedOriginalData_{test}*);
- (25) return *identifyMaliciousData*;

ALGORITHM 2: NTMI approach.

4.1. Experimental Datasets. We capture four sets of network traffic data at different periods within a week, called NTDS1, NTDS2, NTDS3, and NTDS4, respectively. These four datasets have a relatively large randomness so that the performance of the proposed method can be better judged. Meanwhile, we adopt the public dataset CAIDA [33] to train and test the above methods. Due to the overwhelming amount of data in this dataset, we randomly select 10% of the data for the experiment. Table 3 summarizes the specific information of the network traffic datasets.

4.2. Experimental Metrics. We use accuracy and false positive rate (FPR) as experimental metrics for the experiment. Equations (7) and (8) show the calculation of accuracy and FPR. In equations (7) and (8), TP represents the number of samples that are correctly identified as malicious traffic. FP indicates the number of samples that are misclassified as the normal traffic, referring to abnormal traffic but mistakenly considered normal. FN denotes the number of samples that are misreported, i.e., normal traffic is misidentified as abnormal traffic. And TN means that the classification result is consistent with expectations, i.e., nonmalicious traffic is classified as normal traffic.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \quad (7)$$

$$\text{FPR} = \frac{FP}{FP + TN}. \quad (8)$$

4.3. Experimental Result. In this paper, experiments are conducted using the above five feature-processed datasets to detect the effect of different kernel functions on the classification effectiveness of the proposed OFSVM algorithm. According to Table 4, for our collected datasets and the publicly available dataset CAIDA, the linear kernel and polynomial kernel have poorer accuracy than the sigmoid kernel and the radial basis function (RBF) kernel, which is mainly due to nonlinear features and higher-dimensional features. Furthermore, it can be seen from Table 4 that the sigmoid kernel function is relatively stable, which is only slightly worse than the RBF kernel. The kernel function has relatively high requirements on the parameters. From the average value of the classification accuracy of the five datasets, the classification effect of the linear kernel function is the worst (i.e., 65.92% for average accuracy), and the RBF kernel performs best (i.e., 88.3% for average accuracy), which is more stable and more suitable for nonlinear high-dimensional feature space with a lower complexity. Therefore, this study selects the RBF kernel as the kernel function of classification.

4.4. Experimental Discussion. Table 5 records the accuracy and FPR of the NTMI approach and other identification methods. From the perspective of accuracy, the NTMI approach is the algorithm with the highest accuracy, followed by DTM, NBM, LA-SVM, and SVM. In terms of

TABLE 3: Network traffic datasets.

Dataset	Training set		Testing set	
	Nonmalware	Malware	Nonmalware	Malware
NTDS1	17,296	5,690	7,435	4,709
NTDS2	15,735	3,972	8,395	3,674
NTDS3	12,947	4,468	7,083	4,238
NTDS4	19,392	4,938	8,974	3,950
CAIDA	170,874	9,763	29,047	11,933

TABLE 4: Accuracy of OFSVM using different kernel functions.

Dataset	Accuracy (%)			
	Polynomial	Sigmoid	Linear	RBF
NTDS1	75.4	87.9	67.4	90.3
NTDS2	86.3	89.3	55.3	88.4
NTDS3	53.8	82.4	64.2	89.3
NTDS4	91.4	87.3	77.3	87.1
CAIDA	72.3	82.7	65.4	86.4
Average	75.84	85.92	65.92	88.3

TABLE 5: Comparison of accuracy and FPR on four collected datasets.

Dataset	Metric (%)	SVM	LA-SVM	NBM	DTM	NTMI
NTDS1	Accuracy	82.6	84.2	83.2	87.2	92.4
	FPR	12.45	11.90	9.56	9.47	5.78
NTDS2	Accuracy	82.8	84.8	84.3	87.5	92.5
	FPR	11.34	10.58	9.73	9.36	5.74
NTDS3	Accuracy	84.3	84.4	86.2	85.9	92.0
	FPR	10.87	9.94	9.46	10.18	5.41
NTDS4	Accuracy	84.6	85.6	86.2	86.6	93.1
	FPR	11.08	8.69	10.14	9.92	5.18
Average	Accuracy	83.57	84.75	84.97	86.8	92.5
	FPR	11.435	10.277	9.722	9.732	5.527

average results, we can observe that the NTMI approach is optimized by 8.93% compared to the SVM algorithm; NTMI is optimized by 7.75% compared to the LA-SVM algorithm; NTMI approach is optimized by 7.56% compared to the NBM algorithm; and NTMI is optimized by 5.7% compared to the DTM algorithm. As can be seen from Table 5, the NTMI approach is more accurate than the other four methods in identifying malware in network traffic.

For the FPR of these five identification methods, our proposed NTMI approach can achieve the lowest false positives. On average, the FPR of the NTMI approach is only 5.527%. The lowest FPR among other methods is 9.722% for NBM. Therefore, the NTMI approach proposed in this paper is the most effective for the identification of malware in network traffic, both in terms of accuracy and FPR.

To illustrate the effectiveness of the NTMI approach more visually, we plot the accuracy and FPR curves of these five identification methods. Figures 1–4 depict the accuracy curves of these methods. As can be seen from these figures, compared with the NTMI approach, the identification performance of the other four methods decreases significantly as the number of data packets continues to increase, and it is expected that when the number of data packets

FIGURE 1: Comparison of accuracy on NTDS1.

FIGURE 3: Comparison of accuracy on NTDS3.

FIGURE 2: Comparison of accuracy on NTDS2.

FIGURE 4: Comparison of accuracy on NTDS4.

expands exponentially, the accuracy of their identification will drop again.

As can be seen from Figures 5–8, compared with the other four methods, the NTMI approach has the lowest FPR in malware identification. The approach has not only high accuracy but also low FPR, which also shows that its identification effect is the best. With the increasing number of data packets, the FPR of the current five methods shows an increasing trend, but the FPR of the NTMI approach tends to be relatively stable. When the number of testing sets increases to about 12,000, it eventually stabilizes at 5.527%. Meanwhile, the NTMI approach consumes less time overhead when identifying malware. Therefore, the NTMI approach

proposed in this paper has better identification effectiveness and more stable performance in both accuracy and FPR.

To better verify that the NTMI approach proposed in this paper has good universality, this study selects the widely used dataset CAIDA for experiments. Figures 9 and 10 summarize the accuracy and FPR of the above five identification methods, and the following conclusions can be drawn.

We select 10% of the dataset for training and testing because the public dataset is large, and the final testing dataset is close to about 40,000. As can be seen from Figure 9, compared with other methods, the accuracy of the NTMI

FIGURE 5: Comparison of FPRs on NTDS1.

FIGURE 7: Comparison of FPRs on NTDS3.

FIGURE 6: Comparison of FPRs on NTDS2.

FIGURE 8: Comparison of FPRs on NTDS4.

approach still performs well. The accuracy of the NTMI approach can reach 91.7%, while the highest accuracy achievable by other classification methods is 78.6%. As the number of data packets continues to increase, the accuracy of the remaining identification methods has dropped significantly.

As can be seen from Figure 10, the FPR of the NTMI approach is lower than that of the other four methods for larger public datasets of network traffic and tends to be stable, remaining around 6%. Compared with the NTMI approach, the other four methods have in common that the FPR is increasing, which makes the identification

performance more unreliable. Additionally, the final time overhead of the NTMI approach is between 30 s and 50 s faster than other methods. As can be seen from Table 5, for the accuracy and FPR, the results of the NTMI approach on the datasets collected in this paper are very similar to the public dataset CAIDA, which also proves the proposed approach is feasible in the real network environment. Ultimately, we conclude that the NTMI approach can achieve the highest identification performance and is practical.

FIGURE 9: Comparison of accuracy on CAIDA.

FIGURE 11: Accuracy of feature dimensionality reduction.

FIGURE 10: Comparison of FPRs on CAIDA.

FIGURE 12: FPRs of feature dimensionality reduction.

4.5. Effectiveness of NTMI. To better measure the effect of the proposed method on the classification performance in this paper, we compare the accuracy and FPRs of the NTMI approach when performing classification and without feature dimensionality reduction, respectively. NTMI-NFDR is denoted as an identification approach that does not perform feature dimensionality reduction. We perform comparison experiments of NTMI and NTMI-NFDR on the public dataset CAIDA. Figures 11 and 12 plot the experimental results. We can observe that the NTMI approach performs better in terms of accuracy and FPR after feature dimensionality reduction on the extracted feature attribute set,

which also verifies that the feature dimensionality reduction method extracted in this paper is feasible.

5. Conclusion

To identify malware in network traffic, we first sample and normalize the data. Secondly, we extract features from the processed data and reduce the dimensionality, thus eliminating the impact of some redundant features on the classification performance of network traffic. And then, we present the OFSVM algorithm based on the SVM algorithm for classifying network traffic and improving the accuracy of classification in network traffic. The OFSVM algorithm improves the SVM algorithm in terms of both parameter

optimization and kernel function selection. Eventually, we propose the NTMI approach for identifying malware in network traffic. Experimental results show that the NTMI approach can achieve higher accuracy and lower FPR compared with other identification methods.

To verify the effectiveness of the NTMI approach, we compare it with four other classification methods, i.e., SVM, LA-SVM, NBM, and DTM. Additionally, we evaluate these five methods on five datasets. Evaluation results suggest that the algorithm proposed in this paper outperforms the other four classification methods in terms of both accuracy and false positive rate. Its average accuracy reaches 92.5%, while the average false positive rate is only 5.527%. In the publicly available dataset CAIDA, our proposed NTMI approach achieves the highest accuracy and the lowest false positive rate, i.e., 91.7% for accuracy and 6.42% for FPR. Therefore, the experimental results can prove the effectiveness of the algorithm.

However, the NTMI approach proposed in this paper is currently not a completely perfect classifier. Future research will consider whether it is possible to further detect which vulnerabilities are exploited by the identified malware. This could help security experts to be able to effectively identify the type of attack and provide solutions quickly.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

A preliminary version of this paper was presented at the 7th International Conference on Dependable Systems and Their Applications (DSA 2020) (Qin et al., 2020).

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was partly supported by the National Key R&D Program of China (Grant no. 2020YFB1005500), the National Natural Science Foundation of China (NSFC) (Grant nos. U1836116 and 61872167), and the Leading-edge Technology Program of Jiangsu Natural Science Foundation (Grant no. BK20202001).

References

- [1] M. Robnik-Šikonja and I. Kononenko, "Theoretical and empirical analysis of Relief F and R Relief F," *Machine Learning*, vol. 53, no. 1, pp. 23–69, 2003.
- [2] M. Shafiq, X. Yu, A. A. Laghari et al., "Network Traffic Classification techniques and comparative analysis using Machine Learning algorithms," in *Proceedings of the 2nd IEEE International Conference on Computer and Communications (ICCC)*, pp. 2451–2455, Chengdu, China, October 2016.
- [3] J. Cao, Z. Fang, G. Qu, H. Sun, and D. Zhang, "An accurate traffic classification model based on support vector machines," *International Journal of Network Management*, vol. 27, no. 1, 2017.
- [4] Y. Zhang, B. Li, H. Lu, A. Irie, and X. Ruan, "Sample-specific SVM learning for person Re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1278–1287, Las Vegas, NV, USA, June 2016.
- [5] Z. Ghaemi, A. Alimohammadi, and M. Farnaghi, "LaSVM-based big data learning system for dynamic prediction of air pollution in Tehran," *Environmental Monitoring and Assessment*, vol. 190, no. 5, pp. 1–17, 2018.
- [6] I. Rish, "An empirical study of the naive Bayes classifier," in *Proceedings of the IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, vol. 3, no. 22, pp. 41–46, Seattle, WA, USA, August 2001.
- [7] B. Kamiński, M. Jakubczyk, and P. Szufel, "A framework for sensitivity analysis of decision trees," *Central European Journal of Operations Research*, vol. 26, no. 1, pp. 135–159, 2018.
- [8] Z. Yang, L. Z. Li, Q. J. Ji, and Y. Q. Zhu, "Network traffic classification using decision tree based on minimum partition distance," *Journal of China Institute of Communications*, vol. 33, no. 3, pp. 90–102, 2012.
- [9] M. Soysal and E. G. Schmidt, "Machine learning algorithms for accurate flow-based network traffic classification: evaluation and comparison," *Performance Evaluation*, vol. 67, no. 6, pp. 451–467, 2010.
- [10] Y. Liu, W. Li, and Y. Li, "Network traffic classification using K-means clustering," in *Proceedings of the 2nd International Multi-Symposiums on Computer and Computational Sciences (IMSCCS)*, pp. 360–365, Iowa City, IA, USA, August 2007.
- [11] A. Shrivastav and A. Tiwari, "Network traffic classification using semi-supervised approach," in *Proceedings of the 2nd International Conference on Machine Learning and Computing (LCMLC)*, pp. 345–349, Bangalore, India, February 2010.
- [12] P. Teufl, U. Payer, M. Amling et al., "InFeCT - network traffic classification," in *Proceedings of the 7th International Conference on Networking (ICN)*, pp. 439–444, Cancun, Mexico, April 2008.
- [13] D. Bekerman, B. Shapira, L. Rokach, and A. Bar, "Unknown malware detection using network traffic classification," in *Proceedings of the 2015 IEEE Conference on Communications and Network Security (CNS)*, pp. 134–142, Florence, Italy, September 2015.
- [14] X. Mu and W. Wu, "A parallelized network traffic classification based on hidden Markov model," in *Proceedings of the 2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pp. 107–112, Beijing, China, October 2011.
- [15] P. C. Sethi and P. K. Behera, "Internet traffic classification for faster and secured network service," *International Journal of Computer Applications*, vol. 131, no. 4, pp. 15–20, 2015.
- [16] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: an overview," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 76–81, 2019.
- [17] H. Lim, J. Kim, J. Heo, K. Kim, Y. Hong, and Y. Han, "Packet-based network traffic classification using deep learning," in *Proceedings of the 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIC)*, pp. 46–51, Okinawa, Japan, February 2019.

- [18] "NetFlow," 2020, <https://www.manageengine.com/products/netflow/>.
- [19] D. Harrington, R. Presuhn, and B. Wijnen, "An architecture for describing simple network management protocol (SNMP) management frameworks," *RFC3411, STD*, vol. 62, 2002.
- [20] W. Guo, D. Y. Yao, Y. Fu et al., "Study on the method for regional traffic flow feature extraction and traffic status evaluation," *Journal of Highway and Transportation Research and Development*, vol. 22, no. 7, pp. 101–104, 2005.
- [21] S. Lei, "A feature selection method based on information gain and genetic algorithm," in *Proceedings of the 2012 International Conference on Computer Science and Electronics Engineering (ICCSEE)*, vol. 2, pp. 355–358, Hangzhou, China, March 2012.
- [22] H. Kaindl and G. Kainz, "Bidirectional heuristic search reconsidered," *Journal of Artificial Intelligence Research*, vol. 7, pp. 283–317, 1997.
- [23] S. Maldonado and R. Weber, "A wrapper method for feature selection using Support Vector Machines," *Information Sciences*, vol. 179, no. 13, pp. 2208–2217, 2009.
- [24] H. A. Fayed and A. F. Atiya, "Speed up grid-search for parameter selection of support vector machines," *Applied Soft Computing*, vol. 80, pp. 202–210, 2019.
- [25] C. F. Lin and S. D. Wang, "Fuzzy support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 464–471, 2002.
- [26] A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, no. 2003, pp. 1157–1182, 2003.
- [27] B. Scholkopf, K. K. Kah-Kay Sung, C. J. C. Burges et al., "Comparing support vector machines with Gaussian kernels to radial basis function classifiers," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2758–2765, 1997.
- [28] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, vol. 4, no. 2, pp. 65–85, 2014.
- [29] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, vol. 5, pp. 4104–4108, Orlando, FL, USA, October 1997.
- [30] M. Neshat, G. Sepidnam, M. Sargolzaei, and A. N. Toosi, "Artificial fish swarm algorithm: a survey of the state-of-the-art, hybridization, combinatorial and indicative applications," *Artificial Intelligence Review*, vol. 42, no. 4, pp. 965–997, 2012.
- [31] J. H. Sun, T. H. Jeng, C. C. Chen, H. C. Huang, and K. S. Chou, "MD-Miner: behavior-based tracking of network traffic for malware-control domain detection," in *Proceedings of the IEEE 3rd International Conference on Big Data Computing Service and Applications (BigDataService)*, pp. 96–105, San Francisco, CA, USA, April 2017.
- [32] J. Mercer, *Functions Of positive and Negative type and Their connection with the Theory Of integral Equations*, pp. 4–415, Philosophical Transactions of the Royal Society, London, UK, 1909.
- [33] J. Caida, <https://www.caida.org/home/>, 2020.