IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 1 February 2023, accepted 24 February 2023, date of publication 2 March 2023, date of current version 7 March 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3251354*

RESEARCH ARTICLE

# HC-DTTSVM: A Network Intrusion Detection Method Based on Decision Tree Twin Support Vector Machine and Hierarchical Clustering

**LI ZOU[1], XUEMEI LUO[1], YAN ZHANG[1], XIAO YANG[1], AND XIANGWEN WANG[2]**
[1]Gansu Provincial Meteorological Information and Technical Support and Equipment Center, Gansu Meteorological Bureau, Lanzhou 730020, China
[2]College of Computer Science and Engineering, Northwest Normal University, Lanzhou 730070, China

Corresponding author: Xiangwen Wang (wangxw2015@nwnu.edu.cn)

This work was supported in part by the Youth Science and Technology Fund Project of Gansu Province under Grant 20JR10RA097, and in part by the Scientific Research Ability Enhancement Program for Young Teachers of Northwest Normal University under Grant NWNU-LKQN2020-29.

**ABSTRACT** Network intrusion detection is an important technology in national cyberspace security strategy and has become a research hotspot in various cyberspace security issues in recent years. The development of effective and efficient intelligent network intrusion detection methods using advanced machine learning algorithms is of great importance for defending against various network intrusions in complex network environments. In this study, a network intrusion detection method based on decision tree twin support vector machine and hierarchical clustering, named HC-DTTWSVM, is proposed, which can effectively detect different categories of network intrusion. First, the hierarchical clustering algorithm is applied to construct the decision tree for network traffic data, where the bottom-up merging approach is used to maximize the separation of the upper nodes of the decision tree, which reduces the error accumulation in the construction of the decision tree. Then, twin support vector machines are embedded in the constructed decision tree to implement the network intrusion detection model, which can effectively detect the network intrusion category in a top-down manner. The detection performance of the proposed HC-DTTWSVM method is evaluated on NSL-KDD and UNSW-NB15 intrusion detection benchmark datasets. Experimental results show that HC-DTTWSVM can effectively detect different categories of network intrusion and achieves comparable detection performance compared to some of the recently proposed network intrusion detection methods.

**INDEX TERMS** Network intrusion detection, twin support vector machine, hierarchical clustering, decision tree.

## I. INTRODUCTION

The wide application of information technology and the rise and development of cyberspace have greatly contributed to the prosperity and progress of the economy and society, but also brought new cyberspace security risks and challenges [1], [2]. With the increasing improvement of computer networks and communication networks, cyberspace security faces complex and diverse security threats such as network intrusions and attacks [3], [4]. The issue of cyberspace security has been elevated to a national strategy in China [5]. Network intrusion detection is an important security defense technology in cyberspace security for monitoring and detecting security events. It helps to ensure network and data security by analyzing the characteristics of network traffic data, discovering and detecting malicious network behavior, and formulating reasonable defense strategies [6], [7]. According to the requirements of cyberspace security protection, it is necessary not only to quickly detect network intrusions in network traffic data, but also to accurately identify the specific categories of network intrusions, so as to take targeted defense strategies and response dispositions

The associate editor coordinating the review of this manuscript and approving it for publication was Long Xu.

to ensure network security. Network intrusion detection has become a popular research topic in cyberspace security in recent years. Due to the explosive growth of network traffic, designing efficient and robust network intrusion detection systems in a big data environment has become crucial but difficult [8], [9]. Therefore, it is important to study fast and accurate intelligent network intrusion detection methods to defend against various network intrusions in complex network environments.

The Twin Support Vector Machine (TWSVM) is a powerful machine learning algorithm based on traditional support vector machines (SVMs) [10]. TWSVM generates two hyperplanes so that the samples of each class are as close as possible to the hyperplane for that class and as far away as possible from the other hyperplane [11]. That is, TWSVM transforms a binary classification problem into a problem of obtaining two non-parallel hyperplanes by solving two quadratic programming problems of smaller size compared to traditional SVMs. Compared to the traditional SVMs, TWSVM has the following advantages [12], [13]: (1) TWSVM improves the computational efficiency by transforming the solution of large-scale classification problems into smaller quadratic programming problems. (2) TWSVM retains the advantages of traditional SVMs and fully considers the prior information in the class. It is less sensitive to the noise in the sample data and has a strong generalization ability. (3) TWSVM has a higher classification accuracy and a training speed that is nearly four times faster than the traditional SVMs, which has a great advantage in the binary classification problems.

TWSVMs are effective in solving binary classification problems, but usually cannot be used directly to solve multi-class classification problems [14], [15]. A common approach to solving multi-class classification problems using TWSVMs is to combine multiple TWSVMs to form an integrated classification model. A number of combinatorial strategies have been proposed for individual TWSVMs [15], [16], such as one-against-rest methods, one-against-one methods, and decision tree-based methods. One-against-rest TWSVM generates a hyperplane for each class of the sample data, so that each binary TWSVM classifier corresponds to a particular class. The classification hypothesis determines whether the given feature vector belongs to the selected class or to one of the remaining classes. One-against-rest TWSVM is conceptually simple, but its runtime complexity is quite high. One-against-one TWSVM is usually implemented using a winner-takes-all strategy, where each binary TWSVM classifier uses samples from only two classes. The final classification results are determined by a voting procedure. The main disadvantage of the one-against-one TWSVM is that its training process is complex and slow. Both one-against-rest TWSVM and one-against-one TWSVM suffer from the problem of unclassifiable region (ambiguous region). Decision tree TWSVM organizes binary TWSVM classifiers into a reasonable decision tree structure, where each non-leaf node is a binary TWSVM and the leaf nodes are labeled by the classes so that the final decision can be achieved directly. The decision tree TWSVM is faster than the one-against-rest TWSVM and the one-against-one TWSVM, and can solve the unclassifiable region problem well [15]. Therefore, in this paper, the decision tree TWSVM is applied to design a network intrusion detection method.

Network intrusion detection systems should not only detect network intrusions in network traffic data, but also accurately identify the specific categories of network intrusions, which is a typical multi-class classification problem. In this paper, we apply the hierarchical clustering and decision tree method to construct a TWSVM-based model that can be used for multi-class classification, and propose a network intrusion detection method based on the hierarchical clustering and decision tree TWSVM multi-class classification model for identifying different categories of network intrusions. The proposed multi-class TWSVM classification model is called HC-DTTWSVM. The key to HC-DTTWSVM is to determine the structure of the decision tree. We apply the hierarchical clustering algorithm to construct the decision tree, and merge the two most similar categories of network traffic data into one class by bottom-up stepwise merging until all the data are merged, and then we get the decision tree structure. This bottom-up merging method can maximize the separation of the upper nodes of the decision tree, improve the classification performance of the upper nodes, and effectively reduce the error accumulation in the classification process of the decision tree. When the proposed HC-DTTWSVM method is applied to network intrusion detection application, the top-down stepwise classification is used to identify the network intrusion category according to the constructed decision tree and the trained TWSVMs.

The main contributions of this paper can be summarized as follows:

- A network intrusion detection method based on decision tree TWSVM and hierarchical clustering is proposed, which can accurately identify the specific categories of network intrusions.
- The TWSVM algorithm is used to implement the classification model of network traffic data, which is fast and accurate compared with traditional SVMs.
- The decision tree method is used to construct the multi-class TWSVM classification model, which effectively overcomes the unclassifiable region and improves the computational complexity.
- The hierarchical clustering algorithm with bottom-up merging is used to construct the decision tree, which can reduce the error accumulation in the classification process of the decision tree.

The rest of this paper is organized as follows. In section II, the current network intrusion detection methods are reviewed, including traditional machine learning based network intrusion detection methods and deep learning based network intrusion detection methods. In section III, the basic theory of TWSVM is first introduced, and then the proposed HC-DTTWSVM model is described in detail, including its application in network intrusion detection. In section IV, the flexibility and performance of the proposed HC-DTTWSVM algorithm are demonstrated on NSL-KDD [17] and UNSW-NB15 [18] network intrusion detection benchmark datasets, and compared with other relevant methods. Finally, the conclusion of this paper and the future work are presented in section V.

## II. RELATED WORKS

With the rapid development of artificial intelligence technology, machine learning algorithms have been widely used in the design of network intrusion detection methods, and a series of research results have been achieved [19], [20], [21]. The application of modern machine learning algorithms to design efficient network intrusion detection methods is an important research direction in the field of cyberspace security [22]. According to the different machine learning algorithms used in network intrusion detection systems, the current network intrusion detection methods can be roughly divided into two categories: traditional machine learning-based network intrusion detection methods and deep learning-based network intrusion detection methods.

Traditional machine learning algorithms, such as SVMs, artificial neural networks, Bayesian networks, decision trees, logistic regression, and random forests, are widely used in the design of network intrusion detection systems [23], [24]. SVM and its variants are very important statistical machine learning algorithms. SVMs have the advantage of minimizing structural risk and having a relatively high generalization ability, and can efficiently solve many pattern classification problems [25], [26], [27]. By combining SVMs with other techniques, researchers have proposed many network intrusion detection methods. Teng et al. [28] presented a self-adaptive and collaborative network intrusion detection method using SVMs and decision trees. The experimental results on the KDD CUP 1999 dataset show that this method is effective in terms of detection precision rate and recall rate. Gu et al. [29] applied the Naïve Bayes approach to transform features of network traffic data to generate high-quality new data, and further built a network intrusion detection model using SVM, which achieved high detection accuracy. Jing and Chen [30] proposed a network intrusion detection method combining SVMs with a nonlinear scaling method, and achieved superior classification performance compared to other methods. Raman et al. [31] presented an SVM-based network intrusion detection method, in which the hypergraph method is used for feature selection of network traffic data and the genetic algorithm is used to optimize the SVM parameters. The experimental results show that this method can achieve high detection accuracy and low false alarm rate. Aburomman and Reaz [16] proposed a weighted one-against-rest SVM algorithm and used it to identify multiple attack categories in network intrusion detection. The experimental results on the NSL-KDD benchmark dataset show that this method outperforms the other related approaches in terms of overall accuracy. Ponmalar and Dhanakoti [32] proposed an intrusion detection approach using an ensemble SVM based on the chaos game optimization algorithm in a big data platform. Rashid et al. [33] proposed a tree-based stacking ensemble intrusion detection model that integrates decision tree, random forest, and extreme gradient boosting. In addition, researchers have also used other machine learning algorithms to design network intrusion detection methods, such as Bayesian networks [34], decision trees [35], and random forests [36]. Overall, these traditional machine learning-based network intrusion detection methods are relatively simple in methodology design and efficient in computation, making them more effective in real-time network intrusion detection applications.

Deep learning is an emerging machine learning technique and has been widely used in many areas of pattern recognition [37]. In recent years, many efficient deep learning models have emerged, such as deep belief networks, deep convolutional neural networks, recurrent neural networks, and deep generative networks. These models have been used to develop network intrusion detection systems [38], [39]. Shone et al. [40] proposed a novel deep learning classification model constructed using stacked non-symmetric deep auto-encoders that can correctly detect network intrusion categories. Hassan et al. [41] proposed an efficient network intrusion detection method in a big data environment by combining deep convolutional neural networks and weight-dropped long short-term memory networks to extract meaningful features from network traffic data, thus improving intrusion detection accuracy. Khan et al. [42] used convolutional neural networks and recurrent neural networks to capture local features and temporal features, respectively, and proposed a hybrid intrusion detection framework based on deep learning for predicting and classifying malicious network attacks. Cao et al. [43] proposed a network intrusion detection model that fuses a convolutional neural network and a gated recurrent unit, which can well solve the problems of low classification accuracy and class imbalance. Kasongo [44] implemented an intrusion detection framework using different types of recurrent neural networks, including long-short term memory, gated recurrent unit, and simple recurrent neural network. Similar methods are presented in [45] and [46]. Although these deep learning-based network intrusion detection methods can usually achieve high detection accuracy, the algorithm design is more complex, and the network training usually requires a large number of iterations, which requires more computational resources, and the training process is generally time-consuming.

## III. MATERIALS AND METHODS

### A. BASIC THEORY OF TWSVM

TWSVM is a widely used machine learning algorithm in the field of pattern recognition. It is very effective in solving binary classification problems. Given a training dataset of data samples $D = D_+ \cup D_-$, which consists of positive class data samples $D_+ = \{(x_i, 1)|x_i \in R^n, i = 1, \cdots, N_+\}$ and negative class data samples $D_- = \{(x_j, -1)|x_j \in R^n, j = 1, \cdots, N_-\}$, where $N_+$ and $N_-$ denote the number of positive and negative class data samples, respectively. In binary classification tasks, the goal of TWSVM is to solve the following two non-parallel hyperplanes in the $n$-dimensional feature space $R^n$:

$$\begin{cases} x^T w_+ + b_+ = 0 \\ x^T w_- + b_- = 0 \end{cases} \tag{1}$$

It requires that each hyperplane be closer to one of the two classes and as far away from the other as possible. A sample point is assigned to class $+1$ or $-1$ depending on its proximity to the two non-parallel hyperplanes.

The solution of these two non-parallel hyperplanes in eq.(1) can be transformed into the solution of the following two quadratic programming problems:

$$\begin{cases} \min\limits_{w_+, b_+, \xi_-} \dfrac{1}{2}\|X_+ w_+ + e_+ b_+\|^2 + c_+ e_-^T \xi_-, \\ \qquad s.t. \quad -(X_- w_+ + e_- b_+) + \xi_- \geq e_-, \xi_- \geq 0 \\ \min\limits_{w_-, b_-, \xi_+} \dfrac{1}{2}\|X_- w_- + e_- b_-\|^2 + c_- e_+^T \xi_+, \\ \qquad s.t. \quad -(X_+ w_- + e_+ b_-) + \xi_+ \geq e_+, \xi_+ \geq 0 \end{cases} \tag{2}$$

where $X_+$ and $X_-$ denote the attribute values of all samples in $D_+$ and $D_-$, respectively, $c_+$ and $c_-$ are penalty parameters, $\xi_+$ and $\xi_-$ are slack vectors, $e_+$ and $e_-$ are the vectors of the ones of appropriate dimensions. These two quadratic programming problems in eq.(2) can be solved by solving their dual problems in dual space. By introducing four Lagrangian multipliers $\alpha \in R^{N_-}$, $\beta \in R^{N_-}$, $\gamma \in R^{N_+}$, and $\eta \in R^{N_+}$, the Lagrangian functions corresponding to the quadratic programming problems in Eq.(2) can be expressed as follows:

According to the Karush-Kuhn-Tucker (KKT) conditions [13] and the Lagrangian functions in Eq.(3), as shown at the bottom of the next page, the Wolfe dual of the quadratic programming problems in Eq.(2) can be expressed as follows:

$$\begin{cases} \max\limits_{\alpha} e_-^T \alpha - \frac{1}{2}\alpha^T G(H^T H)^{-1} G^T \alpha, \quad s.t. \quad 0 \leq \alpha \leq c_+ \\ \max\limits_{\gamma} e_+^T \gamma - \frac{1}{2}\gamma^T H(G^T G)^{-1} H^T \gamma, \quad s.t. \quad 0 \leq \gamma \leq c_- \end{cases} \tag{4}$$

where $H = [X_+, e_+]$ and $G = [X_-, e_-]$. By solving the Wolfe dual in Eq.(4), the solution of the non-parallel hyperplane in Eq.(1) is obtained as:

$$\begin{cases} [w_+ \quad b_+]^T = -(H^T H)^{-1} G^T \alpha \\ [w_- \quad b_-]^T = (G^T G)^{-1} H^T \gamma \end{cases} \tag{5}$$

When TWSVM is used to solve the binary classification problem, the classification decision function for the input sample data $x$ is expressed as

$$f(x) = \arg\min_{k \in \{+,-\}} \frac{|x^T w_k + b_k|}{\|w_k\|} \tag{6}$$

That is, the sample data $x$ is assigned to the class corresponding to the hyperplane to which it is closest.

The main difference between TWSVM and SVM is that when solving a binary classification problem, SVM creates only one hyperplane, while TWSVM creates two non-parallel hyperplanes so that each hyperplane is as close as possible to the sample points of that class and far away from the sample points of the other class. Figure 1 illustrates the basic SVM and TWSVM in two-dimensional space. Assuming that there are $n$ training data samples, the computational complexity of SVM is $O(n^3)$. If the number of data samples in each class is approximately $n/2$, then the complexity of TWSVM is $O(2 \times (n/2)^3)$. The ratio of the computational complexity of SVM and TWSVM is $\frac{n^3}{2\times(n/2)^3} = 4$. In other words, TWSVM is theoretically four times faster than traditional SVM.

### B. MULTI-CLASS CLASSIFICATION ALGORITHM: HC-DTTWSVM

A single TWSVM cannot be used directly to solve multi-class classification problems. Embedding a single TWSVM in each non-leaf node of the decision tree is a commonly used method to design TWSVM algorithms for multi-class classification problems. Decision tree-based multi-class TWSVMs only need to build $K-1$ sub-classifiers for a $K$-class classification problem. When $K$ is very large, the number of non-leaf nodes and the number of levels of the tree will increase, and the complexity of the upper nodes will be very high [15]. That is, decision tree-based multi-class TWSVMs suffer from cumulative errors. Therefore, the key to using decision trees to design the TWSVM multi-class classification algorithm is to construct the structure of the decision tree with less classification error accumulation. On the one hand, if a misclassification occurs at a non-leaf node at a certain level, this error will be inherited by the classifier at the next level, resulting in error accumulation. On the other hand, the classification accuracy of the upper-level non-leaf nodes directly affects the performance of the whole classification model, so the structure of the decision tree and the upper-level classifier should be reasonably designed to reduce the downward transmission and accumulation of classification errors.

Here we use the hierarchical clustering algorithm and the decision tree algorithm to design our multi-class classification algorithm HC-DTTWSVM. The core idea of the HC-DTTWSVM algorithm is to first apply the hierarchical clustering algorithm to construct the structure of the decision tree, then embed individual TWSVMs into the non-leaf nodes of the constructed decision tree, and finally train these individual TWSVMs to optimize the classification performance of the whole model.

The first step of the HC-DTTWSVM algorithm is to apply the hierarchical clustering algorithm to construct the structure of the decision tree. For $K$ class classification problems, the training dataset $D = \{(x_i, y_i)|i = 1, \cdots, N\}, x_i \in R^n, y_i \in \{1, \cdots, K\}$ containing $N$ data samples belonging to

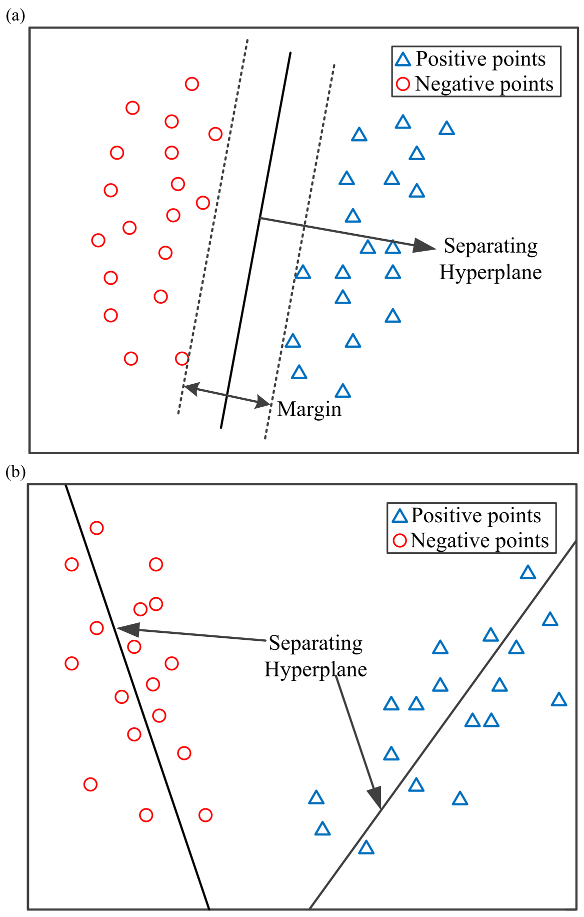

**FIGURE 1. Illustration diagram of (a) SVM and (b) TWSVM in two-dimensional space.**

$K$ classes is first divided into $K$ clusters $C = \{C_1, \cdots, C_K\}$ according to their labels, and the distance matrix $Dist$ between the $K$ clusters is computed using the Euclidean distance. Furthermore, the two most similar classes of the samples are merged into a new class, and the distance between the new class and the remaining classes is calculated. The cluster paths are also saved. The merging process is repeated until all the classes are merged into one class, and then the clustering dendrogram is obtained through the stored cluster paths. Finally, the decision tree structure is obtained according to the clustering dendrogram. This process of constructing the decision tree structure based on the hierarchical clustering algorithm is a bottom-up, step-by-step merging process based on the similarity between samples, which allows the upper-level nodes to have greater separation, so as to effectively reduce the accumulation of errors in the classification process. The process of constructing a decision tree using the hierarchical clustering algorithm is shown in Algorithm 1.

**Algorithm 1** Constructing a Decision Tree Using the Hierarchical Clustering Algorithm

```
Input: Training dataset $D = \{(x_i, y_i)|i = 1, \cdots, N\}, x_i \in R^n, y_i \in \{1, \cdots, K\}$
Output: Decision tree $DT$
 1: Divide $D$ into $K$ clusters $C = \{C_1, \cdots, C_K\}$ according to their labels
 2: $Dist \Leftarrow Euclidean(C)$    ▷ Calculate the Euclidean distance matrix $Dist$ between the clusters $C$
 3: $CP = \emptyset$    ▷ Initialize cluster path $CP$ as an empty set
 4: while $length(Dist) > 1$ do
 5:     $(i, j) \Leftarrow \arg\min_{i,j} Dist$    ▷ Find clusters $C_i$ and $C_j$ such that $Dist(i, j) = min(Dist)$
 6:     $k \Leftarrow 2K - length(Dist) + 1$
 7:     $C_k \Leftarrow C_i + C_j$    ▷ Merge $C_i$ and $C_j$ into $C_k$
 8:     $CP \Leftarrow Append(i, j, k)$    ▷ Append $(i, j, k)$ into $CP$ to update the cluster path
 9:     $C = C - \{C_i, C_j\} + C_k$    ▷ Remove $C_i$ and $C_j$ from $C$, and add $C_k$ into $C$
10:     $Dist \Leftarrow Euclidean(C)$    ▷ Recalculate the distance matrix $Dist$ between the updated clusters $C$
11: end while
12: $DG \Leftarrow dendrogram(CP)$    ▷ Get the dendrogram according to $CP$
13: $DT \Leftarrow decisiontree(DG)$    ▷ Construct decision tree $DT$ according to $DG$
14: return Decision tree $DT$
```

After the decision tree structure is obtained, each non-leaf node is embedded with a binary TWSVM and the leaf nodes are labeled by the classes. That is to say, the framework of the HC-DTTWSVM model consists of many individual TWSVMs embedded in the non-leaf nodes of the constructed decision tree. The main task remaining is to train these individual TWSVMs so that the overall classification performance of the model is optimal. For $K$-class classification problems, only $K - 1$ TWSVMs need to be trained. When training a TWSVM, the training dataset is first divided into two subsets according to their labels and the decision tree structure. Then the two subsets of sample data are fed into the TWSVM to calculate its two hyperplanes. The training process is repeated until all the embedded binary TWSVMs are trained. The training process of HC-DTTWSVM is shown in Algorithm 2. After training all TWSVMs, the trained HC-DTTWSVM model is obtained, which can be used for network intrusion detection.

$$\begin{cases} L_+(w_+, b_+, \xi_-, \alpha, \beta) = & \frac{1}{2}(X_+w_+ + e_+b_+)^T(X_+w_+ + e_+b_+) + c_+e_-^T\xi_- - \alpha^T(-(X_-w_+ + e_-b_+) + \xi_- - e_-) - \beta^T\xi_- \\ L_-(w_-, b_-, \xi_+, \gamma, \eta) = & \frac{1}{2}(X_-w_- + e_-b_-)^T(X_-w_- + e_-b_-) + c_-e_+^T\xi_+ - \gamma^T(-(X_+w_- + e_+b_-) + \xi_+ - e_+) - \eta^T\xi_+ \end{cases} \tag{3}$$

**Algorithm 2** Training HC-DTTWSVM

```
Input: Training dataset $D = \{(x_i, y_i)|i = 1, \cdots, N\}, x_i \in R^n, y_i \in \{1, \cdots, K\}$, untrained HC-DTTWSVM model
Output: Trained HC-DTTWSVM model
1: for each TWSVM do
2:     Split $D$ into 2 subsets $D = \{D^+, D^-\}$ according to their labels and the decision tree structure
3:     Calculate $\alpha$ and $\gamma$ using Eq.(4)
4:     Calculate the parameters of the hyperplanes using Eq.(5)
5:     Determine the hyperplanes using Eq.(1)
6: end for
7: return Trained HC-DTTWSVM model
```

### C. HC-DTTWSVM FOR NETWORK INTRUSION DETECTION

The proposed HC-DTTWSVM model is applied to the network intrusion detection task to accurately detect the specific categories of network intrusion. The general process of HC-DTTWSVM for network intrusion detection is shown in Figure 2. There are four main steps in the network intrusion detection process:

- Step 1: Data preprocessing. The network intrusion detection benchmark dataset usually contains non-numerical features, such as service types and protocol types. In order to detect network intrusion categories using the proposed HC-DTTWSVM algorithm, all the non-numerical features and the label information of the network traffic data must be converted into numbers. Meanwhile, the feature values need to be normalized to the interval [0, 1]. Finally, the sample data in the whole dataset are randomly divided into a training set and a testing set.
- Step 2: Training the HC-DTTWSVM model. The proposed HC-DTTWSVM multi-class classification model is trained with the network traffic data in the training set. Specifically: (1) Generate the decision tree structure using the training samples according to Algorithm 1; (2) Generate the untrained HC-DTTWSVM model by embedding a binary TWSVM in each non-leaf node and labeling the class information in each leaf node of the constructed decision tree; (3) Train the HC-DTTWSVM model using the samples in the training set according to Algorithm 2.
- Step 3: Testing the HC-DTTWSVM model. After training the HC-DTTWSVM model, the sample data in the testing set is used to test and verify the detection performance of the trained HC-DTTWSVM model. Starting from the top-level TWSVM classifier (the root node of the decision tree), the classification decision function $f(x)$ of a testing sample $x$ is calculated according to Eq.(6). If $f(x) > 0$, the sample $x$ is detected by TWSVMs in the left subtree of the decision tree (assuming the left subtree represents a positive class sample), otherwise, it is detected by TWSVMs in the right subtree of the decision tree. If sample $x$ is classified into a leaf node, the detected intrusion category of the sample $x$ is assigned as a label on that leaf node.
- Step 4: Performance evaluation. The detection results of all samples in the testing set are recorded, and the detection performance of the proposed HC-DTTWSVM model is evaluated by 6 metrics defined in Section IV-B (accuracy, precision, recall, false positive rate, F1-score, and G-mean).

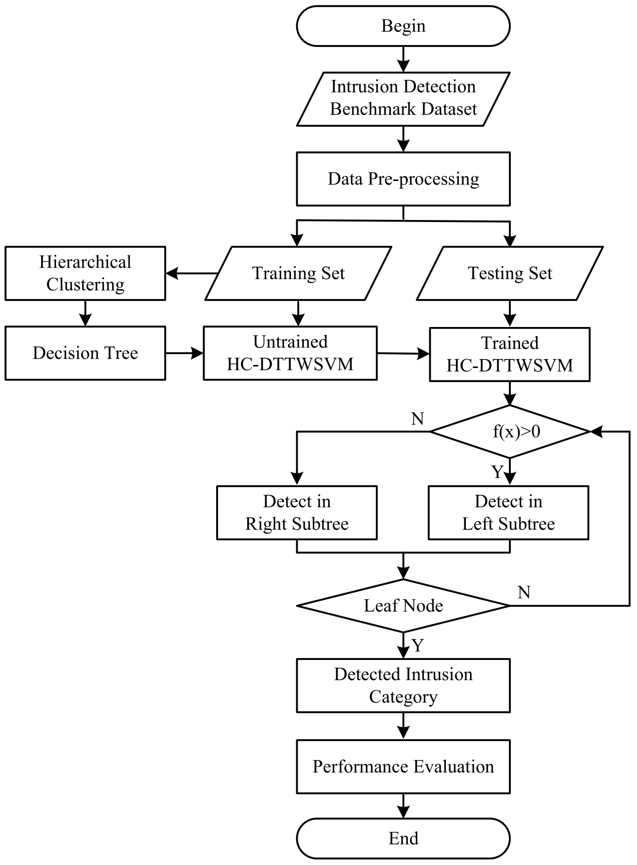

**FIGURE 2.** The general process of HC-DTTWSVM for network intrusion detection.

## IV. EXPERIMENTAL RESULTS

In this section, some experiments are performed to demonstrate the detection performance of the proposed HC-DTTWSVM algorithm on two network intrusion detection benchmark datasets, NSL-KDD [17] and UNSW-NB15 [18]. First, the details of these two datasets and the performance evaluation metrics of the algorithm are introduced. Then the experimental results of the proposed HC-DTTWSVM algorithm on the two datasets are presented and compared with some most recently proposed network intrusion detection methods.

### A. DATASET DESCRIPTION AND PARAMETER SETTINGS

The NSL-KDD and UNSW-NB15 datasets are currently two widely used network intrusion detection benchmark datasets.

The NSL-KDD dataset contains normal network traffic data and four categories of abnormal network traffic data, namely Probe, Denial of Service (DoS), User to Root (U2R), and Remote to Local (R2L). Each sample has 42 features. The UNSW-NB15 dataset contains normal network traffic data and nine categories of abnormal network traffic data, and each sample has 49 features. The samples in each dataset are divided into a training set and a testing set. In the NSL-KDD dataset, all the samples in the "KDDTrain+.TXT" file are used as the training set, and 18794 samples are randomly selected from the "KDDTest+.TXT" file to form the testing set. The UNSW-NB15 dataset has been configured as a training set and testing set, so we use all the samples in the "UNSW_NB15_training-set.csv" file and all the samples in the "UNSW_NB15_testing-set.csv" file to form the training and testing sets. For a fair comparison, as in many related studies [40], [47], [48], [49], we use the same method of splitting the training and testing sets. The data distributions in the NSL-KDD and UNSW-NB15 datasets are shown in Table 1 and Table 2, respectively.

All the non-numerical features and the label information of the samples in the two datasets need to be converted to numbers. The method is simply to replace the non-numeric features and sample label information with numeric values. For example, for the *protocol_type* attribute in the NSL-KDD dataset, the value *tcp* is changed to 1, *udp* to 2, and *icmp* to 3. The numerical labels of the samples in the NSL-KDD and UNSW-NB15 datasets are shown in Table 1 and Table 2, respectively. Additionally, all feature values are normalized to the interval [0, 1] using the min-max scaling method expressed as follows:

$$f_{normalized} = \frac{f - f_{min}}{f_{max} - f_{min}} \tag{7}$$

where $f_{normalized} \in [0, 1]$ is the normalized feature value, $f_{max}$ and $f_{min}$ are the maximum and minimum values of feature $f$.

In the HC-DTTWSVM method, the main hyperparameters are the kernel function and two penalty parameters. The kernel function is the Gaussian kernel, which is expressed as follows:

$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right) \tag{8}$$

The values of the parameter $\sigma$ in the Gaussian kernel and the penalty parameters $c_+$ and $c_-$ in Eq.(2) are determined by 5-fold cross-validation. The parameter values of each binary TWSVM in the HC-DTTWSVM model are given in the following subsections.

## B. PERFORMANCE EVALUATION

Intrusion detection accuracy and time efficiency are two important aspects when evaluating intrusion detection systems. The time efficiency of intrusion detection algorithms is affected by software and hardware platforms. The results of the intrusion detection methods used for comparison in this paper are mainly from the original literature, rather than from our own implementation. Therefore, we evaluate the performance of the proposed method only in terms of intrusion detection accuracy. As in many network intrusion detection studies [16], [43], [49], [50], we use accuracy, precision, recall, false positive rate (FAR), F1-score, and G-mean to evaluate the detection performance of the proposed HC-DTTWSVM algorithm. These six metrics are defined as follows:

**TABLE 1. The data distribution in the NSL-KDD dataset.**

| Category | Numerical label | Training set | Testing set |
|---|---|---|---|
| Normal | 1 | 67343 | 9711 |
| Probe | 2 | 11656 | 1106 |
| Dos | 3 | 45927 | 5741 |
| U2R | 4 | 52 | 37 |
| R2L | 5 | 995 | 2199 |
| Total | | 125973 | 18794 |

**TABLE 2. The data distribution in the UNSW-NB15 dataset.**

| Category | Numerical label | Training set | Testing set |
|---|---|---|---|
| Normal | 1 | 56000 | 37000 |
| Analysis | 2 | 2000 | 677 |
| Backdoor | 3 | 1746 | 583 |
| Dos | 4 | 12264 | 4089 |
| Exploits | 5 | 33393 | 11132 |
| Fuzzers | 6 | 18184 | 6062 |
| Generic | 7 | 40000 | 18871 |
| Reconnaissance | 8 | 10491 | 3496 |
| Shellcode | 9 | 1133 | 378 |
| Worms | 10 | 130 | 44 |
| Total | | 175341 | 82332 |

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

$$FAR = \frac{FP}{FP + TN} \tag{12}$$

$$F1\text{-}score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{13}$$

$$G\text{-}mean = \sqrt{Recall \times (1 - FAR)} \tag{14}$$

where *TP* (true positive) indicates the number of abnormal samples detected as abnormal samples, *TN* (true negative) indicates the number of normal samples detected as normal samples, *FP* (false positive) indicates the number of normal samples detected as abnormal samples, and *FN* (false negative) indicates the number of abnormal samples detected as normal samples.

The *Accuracy* represents the ratio of the number of correctly detected samples to the total number of samples. The *Precision* represents the ratio of the number of correctly detected abnormal samples to the total number of detected abnormal samples. The *Recall* represents the ratio of the number of correctly detected abnormal samples to the total number of actual abnormal samples. The *FAR* represents the ratio of the number of normal samples detected as abnormal samples to the total number of actual normal samples. The $F1$-*score* represents the harmonic mean between *Precision* and *Recall*, which is a comprehensive performance evaluation metric. The *G-mean* represents the geometric mean of *Recall* and *Specificity* (*Specificity* $= 1 - FAR$), which is a comprehensive performance evaluation metric for unbalanced datasets. The values of these six metrics are between 0 and 1. Large values of *Accuracy*, *Precision*, *Recall*, $F1$-*score*, *G-mean* and small value of *FAR* indicate that the model has better performance in network intrusion detection.

### C. RESULTS ON THE NSL-KDD DATASET

In the first experiment, the detection performance of HC-DTTWSVM is tested and verified on the NSL-KDD dataset. There are 5 classes of samples in this dataset, and their labels are converted to 1-5 (see Table 1). The modeling process of the HC-DTTWSVM algorithm on the NSL-KDD dataset is shown in Figure 3. Figure 3(a) illustrates the hierarchical clustering process. It can be seen that the two most similar classes of samples, class 4 (U2R) and class 5 (R2L), are merged into a new class 6 (U2R + R2L), and then the two most similar classes of samples in a new round, class 6 and class 1 (Normal) are merged into a new class 7. This bottom-up merging is repeated until all classes of samples are merged into class 9. After hierarchical clustering, the clustering dendrogram is obtained and used to construct the decision tree. Figure 3(b) shows the multi-class classification TWSVM model, where 4 binary TWSVMs need to be trained. It can be seen that the earliest merged class of samples is detected last in hierarchical clustering, and the last merged classes of samples are detected earliest. For example, class 3 (Dos) is detected earliest by TWSVM$_1$, while class 4 (U2R) and class 5 (R2L) are detected last by TWSVM$_4$. It just explains that the proposed HC-DTTWSVM algorithm constructs a decision tree by bottom-up merging, and realizes intrusion detection by top-down classification. After 5-fold cross-validation, the parameter values of each binary TWSVM for the NSL-KDD dataset are shown in Table 3.

The detection results of the HC-DTTWSVM algorithm on the NSL-KDD dataset are shown in Table 4. It can be seen that Dos samples have the highest precision at 98.04%, the highest F1-score at 96.08%, and the highest G-mean at 96.66%. Normal samples have the highest recall at 96.69%, and R2L samples have the lowest FAR at 0.21%. Normal samples have the highest FAR because there are too many Normal samples in the testing set. U2R samples and R2L samples have the lowest Recall because the fact that there are too few U2R samples and R2L samples in the training set compared to other sample classes, and the TWSVM$_4$ used to detect them is located at the bottom level of the decision tree. Dos samples have the highest precision and F1-score because the TWSVM$_1$ used to detect them is located at the top level of the decision tree. Overall, HC-DTTWSVM can detect different categories of network intrusions on the NSL-KDD dataset, and the detection performance on Dos samples is the best. The detection performance of the samples detected by the top-level TWSVMs in the decision tree is generally higher than that of the samples detected by the lower-level TWSVMs.

**TABLE 3. Parameter values of each binary TWSVM for the NSL-KDD dataset.**

| Classifier | $\sigma$ | $c_+$ | $c_-$ |
|---|---|---|---|
| TWSVM$_1$ | 1.25 | 2.25 | 2.25 |
| TWSVM$_2$ | 1.25 | 3.25 | 3.25 |
| TWSVM$_3$ | 1.25 | 1.25 | 1.25 |
| TWSVM$_4$ | 0.25 | 1.25 | 1.25 |

**TABLE 4. The detection results of HC-DTTWSVM for each class on the NSL-KDD dataset.**

| Category | Precision | Recall | FAR | F1-score | G-mean |
|---|---|---|---|---|---|
| Normal | 81.28% | 96.69% | 23.80% | 88.32% | 85.84% |
| Probe | 76.00% | 78.75% | 1.55% | 77.35% | 88.05% |
| Dos | 98.04% | 94.20% | 0.82% | 96.08% | 96.66% |
| U2R | 12.86% | 24.32% | 0.33% | 16.82% | 49.23% |
| R2L | 93.14% | 21.60% | 0.21% | 35.07% | 46.43% |

**TABLE 5. The detection accuracy of different algorithms for each class on the NSL-KDD dataset.**

| Methods | Normal | Probe | Dos | U2R | R2L |
|---|---|---|---|---|---|
| DBN [40] | 95.64% | 72.97% | 87.96% | 0.00% | 0.00% |
| S-NDAE [40] | 97.73% | 94.67% | 94.58% | 2.70% | 3.82% |
| DE-ELM [51] | 96.80% | 62.74% | 91.50% | 0.00% | 11.90% |
| OAO-SVM [16] | 93.93% | 78.25% | 78.49% | 11.90% | 12.51% |
| OAR-SVM [16] | 93.32% | 75.30% | 77.08% | 9.52% | 17.70% |
| DAG-SVM [16] | 91.71% | 80.72% | 48.30% | 0.00% | 43.83% |
| WOAR-SVM [16] | 89.29% | 89.23% | 84.21% | 33.33% | 30.05% |
| HC-DTTWSVM | 96.69% | 78.75% | 94.20% | 24.32% | 21.60% |

The detection accuracy of HC-DTTWSVM for each class of samples on the NSL-KDD dataset is compared with other machine learning algorithms and the results are shown in Table 5. The deep belief network (DBN) [40], stacked non-symmetric deep auto-encoder (S-NDAE) [40], differential evolution and extreme learning machine (DE-ELM) [51], one-against-one SVM (OAO-SVM) [16], one-against-rest SVM (OAR-SVM) [16], directed acyclic graph SVM (DAG-SVM) [16], and weighted one-against-rest SVM (WOAR-SVM) [16] algorithms are used for comparison. It can be seen that all these algorithms can detect Normal, Probe and Dos samples relatively accurately, but only the WOAR-SVM and HC-DTTWSVM algorithms can detect U2R and R2L samples more accurately at the same time. The detection accuracy of the HC-DTTWSVM algorithm for Normal, Probe, Dos, U2R and R2L samples is 96.69%, 78.75%, 94.20%, 24.32% and 21.60%, respectively. The detection accuracy of the HC-DTTWSVM algorithm for Normal and Dos samples is second only to the S-NDAE algorithm, but higher than that of other algorithms. The detection accuracy of the HC-DTTWSVM algorithm for U2R and R2L samples is second only to the WOAR-SVM algorithm, but significantly higher than that of other algorithms. This indicates that the proposed HC-DTTWSVM algorithm has some advantages in network intrusion detection.

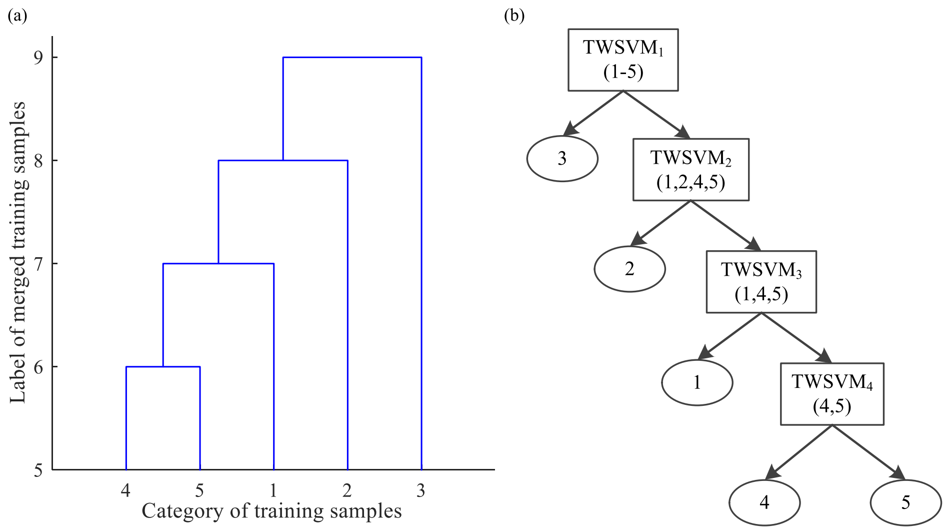

**FIGURE 3.** Modeling process of HC-DTTWSVM on the NSL-KDD dataset. (a) Hierarchical clustering dendrogram. (b) Multi-class classification TWSVM model.

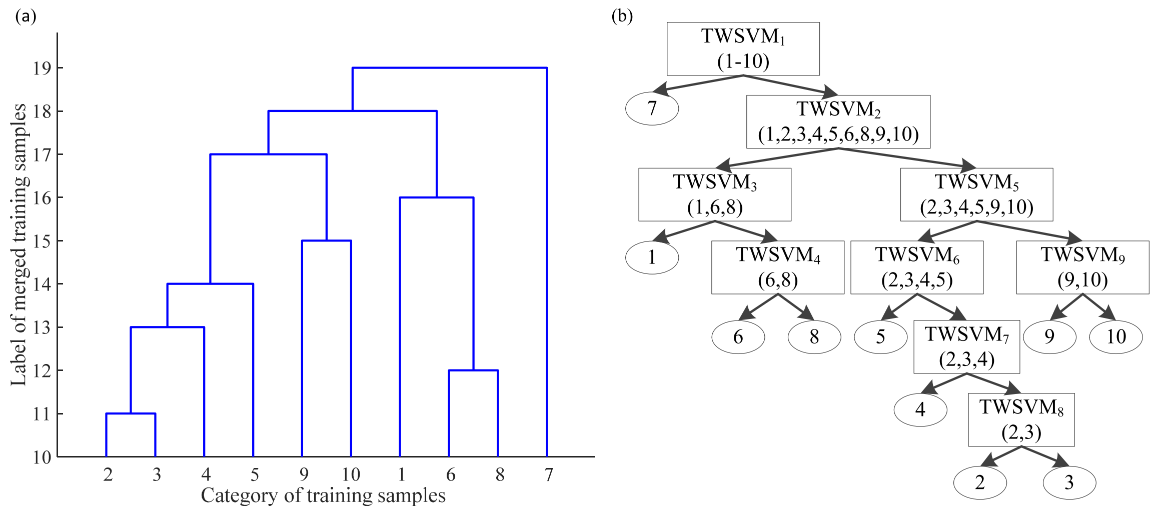

**FIGURE 4.** Modeling process of HC-DTTWSVM on the UNSW-NB15 dataset. (a) Hierarchical clustering dendrogram. (b) Multi-class classification TWSVM model.

Table 6 shows the comparison of the overall classification accuracy of different detection methods on the NSL-KDD dataset. Recently proposed methods for comparison include bidirectional long short-term memory (BLSTM) [45], BLSTM with attention mechanism (BAT) [45], BAT with multiple convolutional layers BAT-MC [45], bi-directional long short-term memory (BiLSTM) [46], convolution neural network and bi-directional long short-term memory (CNN-BiLSTM) [46], adversarial environment with synthetic minority over-sampling technique (AESMOTE) [52], deep autoencoder and deep neural network (DAE-DNN) [53], focal loss using deep neural network (FL-DNN) [54], focal loss using convolutional neural network (FL-CNN) [54], differential evolution and extreme learning machine (DE-ELM) [51], long short-term memory (LSTM) [50], deep neural network (DNN) [55], cost sensitive neural networks (CS-NN) [48], deep state action reward state action (Deep-SARSA) [49], recurrent neural network with extreme gradient boosting (RNN-XGBoost) [44], long short-term memory with extreme gradient boosting (LSTM-XGBoost) [44], and gated recurrent unit with extreme gradient boosting (GRU-XGBoost) [44]. It can be seen that the proposed HC-DTTWSVM algorithm achieves the highest detection accuracy of 85.95% on the entire NSL-KDD dataset. In general, the proposed HC-DTTWSVM algorithm achieves higher overall detection accuracy than other machine learning algorithms on the NSL-KDD dataset, indicating that the algorithm is effective in network intrusion detection.

### D. RESULTS ON THE UNSW-NB15 DATASET

The second experiment tests and verifies the detection performance of the proposed HC-DTTWSVM algorithm on the UNSW-NB15 dataset. There are 10 classes of samples

**TABLE 6. Comparison of the overall classification accuracy of different detection methods on the NSL-KDD dataset.**

| Methods | Year | Accuracy |
|---|---|---|
| BLSTM [45] | 2020 | 79.40% |
| BAT [45] | 2020 | 82.56% |
| BAT-MC [45] | 2020 | 84.25% |
| BiLSTM [46] | 2020 | 79.43% |
| CNN-BiLSTM [46] | 2020 | 83.58% |
| AESMOTE [52] | 2020 | 82.09% |
| DAE-DNN [53] | 2021 | 83.33% |
| FL-DNN [54] | 2021 | 75.13% |
| FL-CNN [54] | 2021 | 75.26% |
| DE-ELM [51] | 2022 | 80.15% |
| LSTM [50] | 2022 | 79.00% |
| DNN [55] | 2022 | 81.87% |
| CS-NN [48] | 2022 | 85.56% |
| Deep-SARSA [49] | 2023 | 84.36% |
| RNN-XGBoost [44] | 2023 | 84.03% |
| LSTM-XGBoost [44] | 2023 | 85.93% |
| GRU-XGBoost [44] | 2023 | 85.65% |
| HC-DTTWSVM | 2023 | 85.95% |

**TABLE 7. Parameter values of each binary TWSVM for the UNSW-NB15 dataset.**

| Classifier | $\sigma$ | $c_+$ | $c_-$ |
|---|---|---|---|
| TWSVM$_1$ | 0.25 | 1.25 | 0.25 |
| TWSVM$_2$ | 1.25 | 1.25 | 1.25 |
| TWSVM$_3$ | 3.25 | 0.25 | 0.25 |
| TWSVM$_4$ | 1.25 | 1.25 | 1.25 |
| TWSVM$_5$ | 1.25 | 0.25 | 1.25 |
| TWSVM$_6$ | 1.25 | 1.25 | 1.25 |
| TWSVM$_7$ | 0.25 | 1.25 | 1.25 |
| TWSVM$_8$ | 0.25 | 0.25 | 1.25 |
| TWSVM$_9$ | 0.25 | 1.25 | 1.25 |

in this dataset, and their labels are converted to 1-10 (see Table 2). Figure 4 shows the modeling process of HC-DTTWSVM on the UNSW-NB15 dataset. The hierarchical clustering process illustrated in Figure 4(a) shows that Analysis samples (class 2) and Backdoor samples (class 3) are merged first while Generic samples (class 7) are merged last. Figure 4(b) shows the multi-class classification TWSVM model constructed from the clustering dendrogram. There are 9 binary TWSVMs need to be trained. It can be seen that Generic samples (class 7) are first detected by TWSVM$_1$, and then the remaining 9 classes of samples are divided into two classes by TWSVM$_2$, and finally Shellcode samples (class 9) and Worms samples (class 10) are detected by TWSVM$_9$. After 5-fold cross-validation, the parameter values of each binary TWSVM for the UNSW-NB15 dataset are shown in Table 7.

Table 8 shows the detection results of the HC-DTTWSVM algorithm on the UNSW-NB15 dataset. It can be seen that the Generic samples have the highest Precision, Recall, F1-score, and G-mean, which are 94.04%, 98.18%, 96.06% and 98.17%, respectively. This is because the TWSVM$_1$ used to detect the Generic samples is located at the top level of the decision tree. The Worms samples have the lowest FAR, which is 0.05%. This is because there are too few Worms samples in the testing set. In general, the proposed HC-DTTWSVM can better detect Normal, Exploits and

**TABLE 8. The detection results of HC-DTTWSVM for each class on the UNSW-NB15 dataset.**

| Category | Precision | Recall | FAR | F1-score | G-mean |
|---|---|---|---|---|---|
| Normal | 86.21% | 88.84% | 11.60% | 87.51% | 88.62% |
| Analysis | 10.57% | 12.56% | 0.88% | 11.48% | 35.28% |
| Backdoor | 39.02% | 8.23% | 0.09% | 13.60% | 28.68% |
| Dos | 40.82% | 64.25% | 4.87% | 49.92% | 78.18% |
| Exploits | 72.50% | 80.17% | 4.75% | 76.14% | 87.39% |
| Fuzzers | 59.78% | 29.23% | 1.56% | 39.26% | 53.64% |
| Generic | 94.04% | 98.18% | 1.85% | 96.06% | 98.17% |
| Reconnaissance | 55.70% | 52.83% | 1.86% | 54.23% | 72.01% |
| Shellcode | 41.83% | 38.62% | 0.25% | 40.17% | 62.07% |
| Worms | 20.00% | 25.00% | 0.05% | 22.22% | 49.99% |

Generic samples, but has a lower detection accuracy for Analysis and Backdoor attacks.

Table 9 shows the detection accuracy of different intrusion detection algorithms on the UNSW-NB15 dataset. The detection accuracy of the proposed HC-DTTWSVM algorithm is compared with the decision tree algorithm implemented by the C5.0 algorithm [56], the Naïve Bayes algorithm [56], the SVM algorithm [30] and the Deep SARSA algorithm [49]. It can be seen that the proposed HC-DTTWSVM algorithm can achieve the highest accuracy for Analysis, Dos, and Generic samples. In particular, the detection accuracy of the proposed HC-DTTWSVM algorithm for Analysis and Dos attacks is much higher than that of other algorithms. This indicates that the proposed HC-DTTWSVM algorithm has some advantages on large and complex network intrusion detection dataset.

The overall detection accuracy of the proposed HC-DTTWSVM algorithm on the UNSW-NB15 dataset is compared with various machine learning algorithms, and the results are shown in Table 10. Recently proposed methods for comparison include feed-forward deep neural network (FFDNN) [57], multi-layer perceptron neural network (ANN-MLP) [58], non-dominated sorting genetic algorithm II and logistic regression (NSGA2-LR) [59], decision tree with extreme gradient boosting (DT-XGBoost) [47], artificial neural network with extreme gradient boosting (ANN-XGBoost) [47], logistic regression with extreme gradient boosting (LR-XGBoost) [47], k nearest neighbour with extreme gradient boosting (kNN-XGBoost) [47], support vector machine with extreme gradient boosting (SVM-XGBoost) [47], convolution neural network and bi-directional long short-term memory (CNN-BiLSTM) [46], genetic algorithm with extra-trees (GA-ET) [60], deep belief network (DBN) [61], deep belief network with kernel-based extreme learning machine (DBN-KELM) [61], deep belief network with enhanced grey wolf optimizer and kernel-based extreme learning machine (DBN-EGWO-KELM) [61], one-dimensional convolutional neural network 1D-CNN [62], multi-label detection model based on deep learning (MLD) [63], recurrent neural network with extreme gradient boosting (RNN-XGBoost) [44], long short-term memory with extreme gradient boosting (LSTM-XGBoost) [44], and gated recurrent unit with extreme gradient boosting

**TABLE 9.** Accuracy of different detection algorithms for each class on the UNSW-NB15 dataset.

| Category | Decision Trees [56] | Naïve Bayes [56] | SVM [30] | Deep SARSA [49] | HC-DTTSVM |
|---|---|---|---|---|---|
| Normal | 74.93% | 64.54% | 75.60% | 94.65% | 88.84% |
| Analysis | 0.00% | 0.00% | 0.00% | 9.00% | 12.56% |
| Backdoor | 4.97% | 22.47% | 3.30% | 10.50% | 8.23% |
| Dos | 8.83% | 0.00% | 4.60% | 26.74% | 64.25% |
| Exploits | 90.08% | 24.97% | 92.70% | 81.35% | 80.17% |
| Fuzzers | 55.24% | 36.28% | 47.10% | 60.24% | 29.23% |
| Generic | 96.96% | 96.29% | 95.80% | 98.09% | 98.18% |
| Reconnaissance | 80.77% | 49.57% | 78.30% | 76.55% | 52.83% |
| Shellcode | 60.84% | 1.32% | 53.20% | 59.62% | 38.62% |
| Worms | 72.72% | 38.64% | 9.10% | 10.69% | 25.00% |

**TABLE 10.** Comparison of the overall classification accuracy of different detection methods on the UNSW-NB15 dataset.

| Methods | Year | Accuracy |
|---|---|---|
| FFDNN [57] | 2020 | 77.16% |
| ANN-MLP [58] | 2020 | 76.96% |
| NSGA2-LR [59] | 2020 | 64.23% |
| DT-XGBoost [47] | 2020 | 67.57% |
| ANN-XGBoost [47] | 2020 | 77.51% |
| LR-XGBoost [47] | 2020 | 65.29% |
| kNN-XGBoost [47] | 2020 | 72.30% |
| SVM-XGBoost [47] | 2020 | 53.95% |
| CNN-BiLSTM [46] | 2020 | 77.16% |
| GA-ET [60] | 2021 | 77.64% |
| DBN [61] | 2021 | 68.60% |
| DBN-KELM [61] | 2021 | 75.60% |
| DBN-EGWO-KELM [61] | 2021 | 79.50% |
| 1D-CNN [62] | 2022 | 76.30% |
| MLD [63] | 2022 | 79.87% |
| RNN-XGBoost [44] | 2023 | 74.19% |
| LSTM-XGBoost [44] | 2023 | 73.01% |
| GRU-XGBoost [44] | 2023 | 78.40% |
| HC-DTTWSVM | 2023 | 81.21% |

(GRU-XGBoost) [44]. It can be seen that the proposed HC-DTTWSVM algorithm achieves the highest detection accuracy of 81.21% on the entire UNSW-NB15 dataset. Overall, the comparison results indicate that the proposed HC-DTTWSVM algorithm is effective in network intrusion detection and can achieve higher detection accuracy than some most recently proposed network intrusion detection methods.

## V. CONCLUSION

Network intrusion detection is an important research content in cyberspace security. In this paper, we proposed a twin support vector machine based multi-class classification algorithm named HC-DTTWSVM for network intrusion detection. The HC-DTTWSVM algorithm uses the hierarchical clustering algorithm to construct the decision tree in a bottom-up merging manner, and the twin support vector machines embedded in the constructed decision tree to realize multi-class classification in a top-down manner. The HC-DTTWSVM algorithm can reduce the error accumulation in decision tree construction, and only needs to train $K - 1$ TWSVMs for $K$ class classification problems. The HC-DTTWSVM algorithm is used to solve the network intrusion detection problem, and its detection performance is validated on the NSL-KDD and UNSW-NB15 datasets. Experimental results show that the proposed HC-DTTWSVM algorithm can accurately identify different categories of network intrusions and achieve higher overall detection accuracy than some recently proposed network intrusion detection methods on the entire dataset.

In future work, we will further improve the construction of decision trees and TWSVM models to design more efficient TWSVM-based multi-class classification models with greater generalization capability, and evaluate the model performance with the cross-validation scheme, and further apply them to network intrusion detection and other pattern recognition problems.

## REFERENCES

[1] M. Humayun, M. Niazi, N. Jhanjhi, M. Alshayeb, and S. Mahmood, "Cyber security threats and vulnerabilities: A systematic mapping study," *Arabian J. Sci. Eng.*, vol. 45, no. 4, pp. 3171–3189, Apr. 2020.

[2] J.-X. Wu, J.-H. Li, and X.-S. Ji, "Security for cyberspace: Challenges and opportunities," *Frontiers Inf. Technol. Electron. Eng.*, vol. 19, no. 12, pp. 1459–1461, Dec. 2018.

[3] S. Wang, "Research on the method of network space security countermeasure drill," *Int. J. Commun. Syst.*, vol. 35, no. 5, p. e4654, Mar. 2022.

[4] P. L. Goethals and M. E. Hunt, "A review of scientific research in defensive cyberspace operation tools and technologies," *J. Cyber Secur. Technol.*, vol. 3, no. 1, pp. 1–46, Jan. 2019.

[5] T. M. Cheung, "The rise of China as a cybersecurity industrial power: Balancing national security, geopolitical, and development priorities," *J. Cyber Policy*, vol. 3, no. 3, pp. 306–326, Sep. 2018.

[6] Z. Yang, X. Liu, T. Li, D. Wu, J. Wang, Y. Zhao, and H. Han, "A systematic literature review of methods and datasets for anomaly-based network intrusion detection," *Comput. Secur.*, vol. 116, May 2022, Art. no. 102675.

[7] M. Mehmood, T. Javed, J. Nebhen, S. Abbas, R. Abid, G. R. Bojja, and M. Rizwan, "A hybrid approach for network intrusion detection," *Comput., Materials Continua*, vol. 70, no. 1, pp. 91–107, 2022.

[8] A. Thakkar and R. Lohiya, "A survey on intrusion detection system: Feature selection, model, performance measures, application perspective, challenges, and future research directions," *Artif. Intell. Rev.*, vol. 55, no. 1, pp. 453–563, Jan. 2022.

[9] L. Yi, M. Yin, and M. Darbandi, "A deep and systematic review of the intrusion detection systems in the fog environment," *Trans. Emerg. Telecommun. Technol.*, vol. 34, no. 1, p. e4632, Jan. 2023.

[10] R. Khemchandani and S. Chandra, "Twin support vector machines for pattern classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 5, pp. 905–910, May 2007.

[11] S. Ding, N. Zhang, X. Zhang, and F. Wu, "Twin support vector machine: Theory, algorithm and applications," *Neural Comput. Appl.*, vol. 28, no. 11, pp. 3119–3130, Nov. 2017.

[12] S. Ding, J. Yu, B. Qi, and H. Huang, "An overview on twin support vector machines," *Artif. Intell. Rev.*, vol. 42, no. 2, pp. 245–252, Aug. 2014.

[13] H. Huajuan, W. Xiuxi, and Z. Yongquan, "Twin support vector machines: A survey," *Neurocomputing*, vol. 300, pp. 34–43, Jul. 2018.

[14] D. Tomar and S. Agarwal, "A comparison on multi-class classification methods based on least squares twin support vector machine," *Knowl.-Based Syst.*, vol. 81, pp. 131–147, Jun. 2015.

[15] S. Ding, X. Zhao, J. Zhang, X. Zhang, and Y. Xue, "A review on multi-class TWSVM," *Artif. Intell. Rev.*, vol. 52, no. 2, pp. 775–801, Aug. 2019.

[16] A. A. Aburomman and M. B. I. Reaz, "A novel weighted support vector machines multiclass classifier based on differential evolution for intrusion detection systems," *Inf. Sci.*, vol. 414, pp. 225–246, Nov. 2017.

[17] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Jul. 2009, pp. 1–6.

[18] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.

[19] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *Appl. Sci.*, vol. 9, no. 20, p. 4396, Oct. 2019.

[20] R. Magán-Carrión, D. Urda, I. Díaz-Cano, and B. Dorronsoro, "Towards a reliable comparison and evaluation of network intrusion detection systems based on machine learning approaches," *Appl. Sci.*, vol. 10, no. 5, p. 1775, Mar. 2020.

[21] Q.-V. Dang, "Using machine learning for intrusion detection systems," *Comput. Informat.*, vol. 41, no. 1, pp. 12–33, 2022.

[22] A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi, and R. Ahmad, "Machine learning and deep learning approaches for cybersecuriy: A review," *IEEE Access*, vol. 10, pp. 19572–19585, 2022.

[23] A. Das, S. A. Ajila, and C. H. Lung, "A comprehensive analysis of accuracies of machine learning algorithms for network intrusion detection," in *Proc. Int. Conf. Mach. Learn. Netw.* Cham, Switzerland: Springer, 2019, pp. 40–57.

[24] T. Saranya, S. Sridevi, C. Deisy, T. D. Chung, and M. A. Khan, "Performance analysis of machine learning algorithms in intrusion detection system: A review," *Proc. Comput. Sci.*, vol. 171, pp. 1251–1260, Jan. 2020.

[25] B. Kumar, O. P. Vyas, and R. Vyas, "A comprehensive review on the variants of support vector machines," *Modern Phys. Lett. B*, vol. 33, no. 25, Sep. 2019, Art. no. 1950303.

[26] J. Liu, J. Feng, and X. Gao, "Fault diagnosis of rod pumping wells based on support vector machine optimized by improved chicken swarm optimization," *IEEE Access*, vol. 7, pp. 171598–171608, 2019.

[27] L. C. Padierna, C. Villasenor-Mora, and S. A. Lopez Juarez, "Biomedical classification problems automatically solved by computational intelligence methods," *IEEE Access*, vol. 8, pp. 101104–101117, 2020.

[28] S. Teng, N. Wu, H. Zhu, L. Teng, and W. Zhang, "SVM-DT-based adaptive and collaborative intrusion detection," *IEEE/CAA J. Autom. Sin.*, vol. 5, no. 1, pp. 108–118, Jan. 2018.

[29] J. Gu and S. Lu, "An effective intrusion detection approach using SVM with naïve Bayes feature embedding," *Comput. Secur.*, vol. 103, Apr. 2021, Art. no. 102158.

[30] D. Jing and H.-B. Chen, "SVM based network intrusion detection for the UNSW-NB15 dataset," in *Proc. IEEE 13th Int. Conf. ASIC (ASICON)*, Oct. 2019, pp. 1–4.

[31] M. R. G. Raman, N. Somu, K. Kirthivasan, R. Liscano, and V. S. S. Sriram, "An efficient intrusion detection system based on hypergraph—Genetic algorithm for parameter optimization and feature selection in support vector machine," *Knowl.-Based Syst.*, vol. 134, pp. 1–12, Oct. 2017.

[32] A. Ponmalar and V. Dhanakoti, "An intrusion detection approach using ensemble support vector machine based chaos game optimization algorithm in big data platform," *Appl. Soft Comput.*, vol. 116, Feb. 2022, Art. no. 108295.

[33] M. Rashid, J. Kamruzzaman, T. Imam, S. Wibowo, and S. Gordon, "A tree-based stacking ensemble technique with feature selection for network intrusion detection," *Int. J. Speech Technol.*, vol. 52, no. 9, pp. 9768–9781, Jul. 2022.

[34] Q. Shi, "A framework of intrusion detection system based on Bayesian network in IoT," *Int. J. Performability Eng.*, vol. 14, pp. 2280–2288, Oct. 2018.

[35] M. A. Ferrag, L. Maglaras, A. Ahmim, M. Derdour, and H. Janicke, "RDTIDS: Rules and decision tree-based intrusion detection system for Internet-of-Things networks," *Future Internet*, vol. 12, no. 3, p. 44, Mar. 2020.

[36] N. Farnaaz and M. A. Jabbar, "Random forest modeling for network intrusion detection system," *Proc. Comput. Sci.*, vol. 89, pp. 213–217, May 2016.

[37] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[38] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *J. Inf. Secur. Appl.*, vol. 50, Feb. 2020, Art. no. 102419.

[39] S.-W. Lee, H. M. Sidqi, M. Mohammadi, S. Rashidi, A. M. Rahmani, M. Masdari, and M. Hosseinzadeh, "Towards secure intrusion detection systems using deep learning techniques: Comprehensive analysis and review," *J. Netw. Comput. Appl.*, vol. 187, Aug. 2021, Art. no. 103111.

[40] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.

[41] M. M. Hassan, A. Gumaei, A. Alsanad, M. Alrubaian, and G. Fortino, "A hybrid deep learning model for efficient intrusion detection in big data environment," *Inf. Sci.*, vol. 513, pp. 386–396, Mar. 2020.

[42] M. A. Khan, "HCRNNIDS: Hybrid convolutional recurrent neural network-based network intrusion detection system," *Processes*, vol. 9, no. 5, p. 834, May 2021.

[43] B. Cao, C. Li, Y. Song, Y. Qin, and C. Chen, "Network intrusion detection model based on CNN and GRU," *Appl. Sci.*, vol. 12, no. 9, p. 4184, Apr. 2022.

[44] S. M. Kasongo, "A deep learning technique for intrusion detection system using a recurrent neural networks based framework," *Comput. Commun.*, vol. 199, pp. 113–125, Feb. 2023.

[45] T. Su, H. Sun, J. Zhu, S. Wang, and Y. Li, "BAT: Deep learning methods on network intrusion detection using NSL-KDD dataset," *IEEE Access*, vol. 8, pp. 29575–29585, 2020.

[46] K. Jiang, W. Wang, A. Wang, and H. Wu, "Network intrusion detection combined hybrid sampling with deep hierarchical network," *IEEE Access*, vol. 8, pp. 32464–32476, 2022.

[47] S. M. Kasongo and Y. Sun, "Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset," *J. Big Data*, vol. 7, no. 1, p. 105, Dec. 2020.

[48] M. Rani, "Effective network intrusion detection by addressing class imbalance with deep neural networks multimedia tools and applications," *Multimedia Tools Appl.*, vol. 81, no. 6, pp. 8499–8518, Mar. 2022.

[49] S. Mohamed and R. Ejbali, "Deep SARSA-based reinforcement learning approach for anomaly network intrusion detection system," *Int. J. Inf. Secur.*, vol. 22, no. 1, pp. 235–247, Feb. 2023.

[50] E. Mushtaq, A. Zameer, M. Umer, and A. A. Abbasi, "A two-stage intrusion detection system with auto-encoder and LSTMs," *Appl. Soft Comput.*, vol. 121, May 2022, Art. no. 108768.

[51] W. L. Al-Yaseen, A. K. Idrees, and F. H. Almasoudy, "Wrapper feature selection method based differential evolution and extreme learning machine for intrusion detection system," *Pattern Recognit.*, vol. 132, Dec. 2022, Art. no. 108912.

[52] X. Ma and W. Shi, "AESMOTE: Adversarial reinforcement learning with SMOTE for anomaly detection," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 943–956, Apr. 2021.

[53] Y. N. Kunang, S. Nurmaini, D. Stiawan, and B. Y. Suprapto, "Attack classification of an intrusion detection system using deep learning and hyperparameter optimization," *J. Inf. Secur. Appl.*, vol. 58, May 2021, Art. no. 102804.

[54] M. Mulyanto, M. Faisal, S. W. Prakosa, and J.-S. Leu, "Effectiveness of focal loss for minority classification in network intrusion detection systems," *Symmetry*, vol. 13, no. 1, p. 4, Dec. 2020.

[55] S. P. Thirimanne, L. Jayawardana, L. Yasakethu, P. Liyanaarachchi, and C. Hewage, "Deep neural network based real-time intrusion detection system," *Social Netw. Comput. Sci.*, vol. 3, no. 2, p. 145, Mar. 2022.

[56] S. Meftah, T. Rachidi, and N. Assem, "Network based intrusion detection using the UNSW-NB15 dataset," *Int. J. Comput. Digit. Syst.*, vol. 8, no. 5, pp. 478–487, 2019.

[57] S. M. Kasongo and Y. Sun, "A deep learning method with wrapper based feature extraction for wireless intrusion detection system," *Comput. Secur.*, vol. 92, May 2020, Art. no. 101752.

[58] J. O. Mebawondu, O. D. Alowolodu, J. O. Mebawondu, and A. O. Adetunmbi, "Network intrusion detection system using supervised learning paradigm," *Sci. Afr.*, vol. 9, Sep. 2020, Art. no. e00497.

[59] C. Khammassi and S. Krichen, "A NSGA2-LR wrapper approach for feature selection in network intrusion detection," *Comput. Netw.*, vol. 172, May 2020, Art. no. 107183.
[60] S. M. Kasongo, "An advanced intrusion detection system for IIoT based on GA and tree based algorithms," *IEEE Access*, vol. 9, pp. 113199–113212, 2021.
[61] Z. Wang, Y. Zeng, Y. Liu, and D. Li, "Deep belief network integrating improved kernel-based extreme learning machine for network intrusion detection," *IEEE Access*, vol. 9, pp. 16062–16091, 2021.
[62] M. K. Hooshmand and D. Hosahalli, "Network anomaly detection using deep learning techniques," *CAAI Trans. Intell. Technol.*, vol. 7, no. 2, pp. 228–243, 2022.
[63] J. Xie, S. Li, Y. Zhang, P. Sun, and H. Xu, "Analysis and detection against network attacks in the overlapping phenomenon of behavior attribute," *Comput. Secur.*, vol. 121, Oct. 2022, Art. no. 102867.

**LI ZOU** received the master's degree in computer technology from Northwest Normal University, Lanzhou, China, in 2015. She is currently an Engineer with Gansu Provincial Meteorological Information and Technical Support and Equipment Center, Gansu Meteorological Bureau. Her current research interests include data mining, machine learning, and cyberspace security.

**XUEMEI LUO** is currently a Senior Engineer with Gansu Provincial Meteorological Information and Technical Support and Equipment Center, Gansu Meteorological Bureau. Her current research interests include network resource management, cyberspace security, and high-performance computing.

**YAN ZHANG** is currently an Engineer with Gansu Provincial Meteorological Information and Technical Support and Equipment Center, Gansu Meteorological Bureau. Her current research interests include computer communication and machine learning.

**XIAO YANG** is currently an Engineer with Gansu Provincial Meteorological Information and Technical Support and Equipment Center, Gansu Meteorological Bureau. His current research interests include machine learning and big data analytics.

**XIANGWEN WANG** received the Ph.D. degree in computer application technology from Lanzhou University, Lanzhou, China, in 2022. He is currently an Engineer with the College of Computer Science and Engineering, Northwest Normal University. His current research interests include neural networks, machine learning, and bioinformatics.