Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 4214784, 9 pages
https://doi.org/10.1155/2021/4214784

Research Article

An Effective Algorithm for Intrusion Detection Using Random Shapelet Forest

Gongliang Li,[1,2] **Mingyong Yin**,[2] **Siyuan Jing**,[3] **and Bing Guo**[1]

[1]*School of Computer, Sichuan University, Chengdu, China 610000*
[2]*Institute of Computing Applications, China Academy of Engineering Physics, Mianyang, China 621000*
[3]*School of Artificial Intelligence, Leshan Normal University, Leshan, China 614000*

Correspondence should be addressed to Bing Guo; guobin@scu.edu.cn

Received 19 June 2021; Accepted 16 August 2021; Published 3 September 2021

Academic Editor: Lihua Yin

Detection of abnormal network traffic is an important issue when builds intrusion detection systems. An effective way to address this issue is time series mining, in which the network traffic is naturally represented as a set of time series. In this paper, we propose a novel efficient algorithm, called RSFID (Random Shapelet Forest for Intrusion Detection), to detect abnormal traffic flow patterns in periodic network packets. Firstly, the Fast Correlation-based Filter (FCBF) algorithm is employed to remove irrelevant features to decrease the overfitting as well as the time complexity. Then, a random forest which is built upon a set of shapelet candidates is used to classify the normal and abnormal traffic flow patterns. Specifically, the Symbolic Aggregate approXimation (SAX) and random sampling technique are adopted to mitigate the high time complexity caused by enumerating shapelet candidates. Experimental results show the effectiveness and efficiency of the proposed algorithm.

1. Introduction

Intrusion detection system (IDS) is an important part of modern network security protection infrastructure. It is aimed at analyzing the traffic packages online or offline to identify the intrusion behaviors from networks. However, some attacks are very difficult to be detected. For example, distributed denial of service (DDos) attack creates tens of thousands of zombie computers and orders them attack a target server at the same time. It not only fabricates source IP address to avoid detection, but also increases the traffic exponentially. Therefore, an efficient technique for detection of intrusion behaviors is required.

The basic principle of intrusion detection technology is to build a normal or abnormal behavior model through the analysis of relevant data which may be stored in security log or audit database and compare the model with the user behavior to identify the potentially harmful behavior [1]. It is obvious that the key to victory is the discovery of the effective behavior characteristics (or patterns) from relevant data. As an effective technology to search and mine hidden information from massive data, data mining is very suitable for intrusion detection. So far, a variety of data mining technologies, including classification, clustering, and anomaly detection, have been successfully applied in intrusion detection.

Classification is a popular technology in intrusion detection. Given a set of labeled instances, it learns a function which can assign a label to a new unlabeled instance. Lee and Stolfo [2] firstly extracted rules from audit data and used the rules for detection of abnormal behavior in network traffic. Gao et al. [3] employed the Apriori algorithm to extract traffic flow patterns from network data and subsequently used the K-means cluster algorithm to generate a detection model. Besides that, many popular techniques of classification were adopted in intrusion detection, such as K-nearest neighbor [4, 5], decision tree [6, 7], and support vector machine [8]. Recently, deep learning, which attracts lots of attentions from community, is employed in intrusion detection and achieves state-of-art performance [9, 10]. However,

the intrinsic defect of deep learning, a.k.a. lack of interpretability, prevents it to be a ready-made panacea.

In this paper, we employ time series classification (TSC) technique to detect abnormal behaviors based on the offline traffic flow data. Specifically, the adopted technique is called shapelet, which is a new primitive in the field of TSC. The contributions of this work include the following:

(1) We propose a novel TSC framework for intrusion detection which is composed of a feature selection algorithm (FCBF) and a shapelet-based random forest classifier

(2) The traffic flow data is represented by SAX and the shapelet candidates, which are used to train the classifier and are sampled randomly. By this way, the running time is greatly mitigated

(3) The proposed algorithm, called RSFID, is validated on several data sets of intrusion detection. The results prove that RSFID is effective to detect abnormal behavior in traffic flow. Since the intrinsic advantage of the shapelet-based method, i.e., good interpretability, our work provides a different solution to solve the problem of intrusion detection

The rest of the paper is organized as follows. Section 2 briefly introduces the development of IDS and recalls the basic knowledge of shapelet-based TSC. Section 3 explains the details of the RSFID algorithm, and the theoretical analysis of complexity is also given. Next, the experimental details are introduced, and the results are analyzed in deep in Section 4. Finally, Section 5 gives conclusions.

2. Background

2.1. Intrusion Detection and Time Series Classification. Intrusion detection is aimed at extracting patterns or characteristics of user's behaviors by analyzing the security log and then identifying the dangerous behavior in the system. The solutions can be divided into two types. The first is building a safe/normal behavior model as the evaluation criteria of user behavior. When the user behavior is obviously different from the safe/normal behavior model, it is considered to be an intrusion. The second is building an unsafe/abnormal model (a.k.a. intrusion behavior) based on a set of obtained data of intrusion. If the detected behavior is similar with the unsafe/abnormal model, we think it is an intrusion.

There are abundant ways to handle the intrusion detection problem, such as classification, clustering, and abnormal detection. Besides those, time series classification is considered to be a suitable solution because the traffic flow data is temporal ordered. Luo et al. [11] modeled the brain activity as time series and used the K-nearest neighbor algorithm to detect the abnormal. Chin et al. [12] evaluated abundant algorithms of anomaly detection which based on symbolic time series analysis. Recently, Wei et al. [13] proposed an assumption-free technique for anomaly detection using time series classification. Kim et al. [14] introduced a shapelet-based method to detect abnormal behavior in network traffic. However, the algorithm is based on exhaustive search; hence, it is too time consuming.

2.2. Shapelet-Based Time Series Classification. Shapelet refers to time series subsequences that are maximally representative of a class [15]. Due to the strong interpretability, it has attracted abundant attentions from the community. In the last decade, over a hundred papers have been published to develop this technique. Later, we will recall some basic knowledge in this field.

Definition 1 (Time series). The time series is denoted by a sequence of values $T = t_1, t_2, \cdots, t_{|T|}$, where $|T|$ is the length of time series. Data points $t_1, t_2, \cdots, t_{|T|}$ are typically arranged by temporal order and spaced at equal time interval.

Definition 2 (Time series data set). A time series data set D is a set of pairs of time series T_i and its corresponding label $c_i \in C$, i.e., $D = \{\langle T_1, c_1\rangle, \langle T_2, c_2\rangle, \cdots, \langle T_n, c_n\rangle\}$, where n is the number of time series in the data set and C is the set of labels.

Furthermore, since most of the time series data in real world are multidimensional, such as the monitoring data collected from Internet of Things system, the ECG monitoring system, and the IDS, we use $T_{i,j}$ to represent the j-th dimension of the i-th time series and the k-th position of $T_{i,j}$ can be written as $T_{i,j,k}$.

Definition 3 (Subsequence). A time series subsequence S is a contiguous sequence of a time series. Subsequence of length l of time series $T_{i,j}$ starting at position k can be denoted as $S_{i,j}^{k,l} = T_{i,j,k}, T_{i,j,k+1}, \cdots, T_{i,j,k+l-1}$. Furthermore, the overall subsequence of time series T with length l is denoted as $\Psi(T, l)$.

For simplicity, lots of concepts introduced below only explain the one-dimension time series and all of them can be naturally extended to multidimension.

Definition 4 (α distance and β distance). The α distance and β distance define the distance between two time series T_1, T_2 with the same length and the distance between a subsequence S and a time series T, respectively.

In this paper, we also use Euclidean distance to measure the two types of distance and the formulas are given below, where m is the length of two time series.

$$\begin{aligned} \mathrm{dist}_\alpha(T_1, T_2) &= \sqrt{\frac{1}{m}\sum_{i=1}^{m}(T_{1,i} - T_{2,i})}, \\ \mathrm{dist}_\beta(S, T) &= \min_{S' \in \Psi(T,|S|)} \mathrm{dist}_\alpha\left(S, S'\right). \end{aligned} \tag{1}$$

Shapelets which are maximally representative of a class are essentially a set of subsequence. Our purpose is to choose a subset of subsequences which have strong discriminatory

power to build a classifier. To measure the discriminatory power of a shapelet candidate, we give the definition of split and information gain (IG).

Definition 5 (Split). A split is a tuple $\eta = \langle S, \tau \rangle$, where S is a time series subsequence and τ is a distance threshold which can split the data set D into two subsets D_L and D_R.

Given a time series subsequence S, we can calculate the distance between S and all series in D, i.e., $\text{dist}_\beta(S, T_i)$. If $\text{dist}_\beta(S, T_i) \leq \tau$, the time series T_i will be added to D_L; otherwise it will be added to D_R.

Definition 6 (IG). The information gain of a split $\eta = \langle S, \tau \rangle$ can be calculated as follows:

$$\text{IG}(\eta) = E(D) - \frac{n_L}{n} E(D_L) - \frac{n_R}{n} E(D_R). \quad (2)$$

The symbols n_L and n_R denote the number of time series in D_L and D_R, respectively, and $E(D) = \sum_{i=1}^{|C|} (n_i/n) \log (n_i/n)$ is the entropy of data set.

Given a time series subsequence S and a data set of time series D, we can calculate the distance between S and all series in D and obtain a set of distance sorted in ascending order $\langle d_1, d_2 \cdots, d_n \rangle$. We say a split $\eta = \langle S, \tau \rangle$ is a shapelet candidate that there is no $\eta' = \langle S, \tau' \rangle$ that $\text{IG}(\eta') > \text{IG}(\eta)$. To distinguish the shapelet candidate with split, we use symbol $\theta = (S, \tau)$ to represent it. It is not difficult to find that there are infinite splits for a specific subsequence. To limit the search space, we only detect the mean value of any two adjacent distance value, i.e., $(d_i + d_{i+1})/2$.

Ye and Keogh [15] firstly introduced the concept of shapelet; meanwhile, they proposed a Brute-Force algorithm to search the best candidate to be the final shapelet embedded into a decision tree classifier. The algorithm suffers from two problems that the exhaustive search is too time-consuming, and the decision tree training is embedded in the search process. There are some solutions to address the first problem, including [15–17]. Due to the limit of page, we skip the introduction of these techniques. Next, we introduce an interesting technique, called shapelet transformation, which separates the shapelet searching and the classifier building by transforming the original time series data set to a new feature space [18].

Definition 7 (Shapelet transformation). Given a time series data set $D = \{T_1, T_2, \cdots, T_n\}$ and a feature space Σ consisted of a set of selected shapelet, i.e., $\Sigma = \{S_1, S_2, \cdots, S_k\}$, shapelet transformation is a matrix M with n rows and k columns, where $M_{i,j} = \text{dist}_\beta(S_j, T_i)$.

It is easy to find that, by shapelet transformation, the temporal characteristic in original time series has been removed. Hence, a large amount of classical data mining techniques can be applied to the time series mining. However, there are also some problems in this technique. For example, the process of shapelet selection is also time-consuming, and the selected shapelets are always be irrelevant and redundant [19–21].

3. The Proposed Method

3.1. The RSFID Algorithm. The idea of the RSFID algorithm (Random Shapelet Forest for Intrusion Detection) is descripted as Figure 1. There are five steps that learn a random shapelet forest (a.k.a. the classifier) from the original time series. Firstly, the raw data of network traffic requires to be represented by SAX [22]. Although there are some other techniques for presentation of time series data, such as PAA, APCA, and DFT, SAX has been proven to be the most efficient technique to compress time series data [23]. The details of SAX technique can be found in [22]. It must be noted that the traffic flow data not only contains real value, but also includes other data types. For example, the KDD CUP 99, which is a famous data set of intrusion detection, contains real value and nominal value. Therefore, the raw data must be preprocessed and converted to normalized real value. After that, the time series data is represented by a set of symbolic words.

The second step is in charge of randomly selecting a set of shapelet candidates. In [19], the authors have validated that random sampling is an effective technique which can greatly reduce the running time by 3~4 orders of magnitude than the Fast Shapelet (FS) algorithm, but without loss of accuracy. Different with [19], we combine the random sampling with SAX presentation which can further improve the scalability of the algorithm. The third step is merging shapelets extracted from the instances of different classes in the same dimension. During this step, part of self-similar shapelets will be removed to reduce the redundancy of the features. Then, the time series data are transformed to the new feature space. We should calculate the distance between shapelets and all series in data sets. In the fifth step, we adopt classical feature selection algorithm to reduce the dimension of new data sets, i.e., the matrix. Finally, we train a set of random forest classifiers for each dimension, which will be used to adjudge whether a network traffic is an intrusion attack or not.

The pseudo-code of the RSFID algorithm is given Algorithm 1. It is not difficult to obtain the idea of the proposed algorithm. From steps 3 to 9, it is composed of two loops. The first loop is aimed at generating m random forest classifiers, i.e., each forest corresponds to a dimension of the time series (a.k.a. network traffic data). For prediction of a new time series, the label is decided by the voting of all classifiers. The inner loop is for generation of p decision trees for the forest. There are two key steps in the inner loop. The function shapelet_sampling is to randomly sample r shapelets from the j-th dimension of the data set D', which is represented by the SAX method. Another function random_shapelet_tree is to generate a decision tree based on the obtained shapelets $S_{i,j}$ and D'. Next, we will explain the two functions in detail.

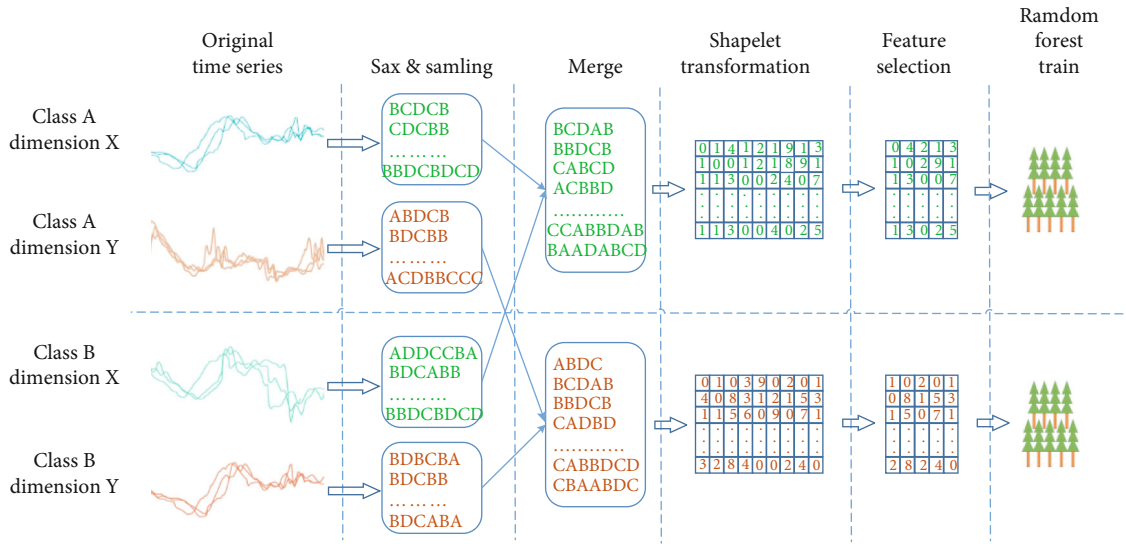

Figure 1: Description of the RSFID algorithm.

3.2. Shapelet Sampling. Since exhaustive search leads to exponential growth of training time, researchers tested the random sampling technique and the results show that it can reduce the running time by 3~4 orders of magnitude than the exhaustive search, without loss of accuracy [19]. However, the existing work does not consider the redundancy and diversity of the sampled shapelets. In this section, we firstly introduce definitions of self-similarity and utility, which are used to filter out nonsimilar shapelets with strong power of discrimination. Then, we explain the code of shapelet_sampling(D', j, r).

Definition 8 (Self-similarity) [23]. Given two subsequences of time series S_1 and S_2, let id_1 and id_2 be the index number of time series that we extract S_1, S_2 from, and $\mathrm{pos}_1, \mathrm{pos}_2$ and $\mathrm{len}_1, \mathrm{len}_2$ denote the start position and the length of S_1, S_2, respectively. We say S_1 and S_2 have self-similarity, when $\mathrm{id}_1 = \mathrm{id}_2 \wedge |\mathrm{pos}_1 - \mathrm{pos}_2| \le \sigma \wedge |\mathrm{len}_1 - \mathrm{len}_2| \le \lambda$.

Here, symbols σ and λ are two user-defined threshold. The former denotes the allowed distance between the starting positions of two shapelets, and the latter represents the allowed difference of two shapelet lengths. Next, we give the definition of utility.

Definition 9 (Utility). Given a shapelet candidate $\theta = \langle S, \tau \rangle$, c denotes the label of the instance that we extract θ from, $C(\cdot)$ is a function that returns the label of an instance. We denote the precision, recall, and utility as follows:

$$P(\theta) = \frac{\left\|\mathrm{dist}_\beta(S, T) \le \tau \wedge c = C(T)\right\|}{\left\|\mathrm{dist}_\beta(S, T) \le \tau\right\|}, T \in D,$$

$$R(\theta) = \frac{\left\|\mathrm{dist}_\beta(S, T) \le \tau \wedge c = C(T)\right\|}{\|c = C(T)\|}, T \in D, \quad (3)$$

$$\mathrm{Utility}(\theta) = \frac{2P(\theta)R(\theta)}{P(\theta) + R(\theta)}.$$

It is easy to find that utility is, essentially, the f-score integrated with precision value and recall value which is regarded as the quality score of a shapelet candidate. Next, we show the pseudo-code in Algorithm 2. In step 2, the algorithm refines the data set of time series that only keeps the j-th dimension of D. From steps 3 to 8, the algorithm randomly extracts a subsequence of a time series and generates a shapelet candidate θ. If the θ is self-similar with any candidates in Θ, it would discard it and resample a new one; otherwise, the θ would be added into the shapelet set Θ. The extraction will be repeated for $r \times \kappa$ times where κ is a coefficient for controlling the total number of shapelet candidates for evaluation. After that, we sort the shapelet candidates in Θ by their utility; then, we keep the top r best shapelets as the final choice.

3.3. Random Shapelet Tree Generation. The pseudo-code of the function random_shapelet_tree is shown in Algorithm 3. It is aimed at generating a decision tree based on a set of shapelets. The algorithm is a typical recursive algorithm which is usually adopted in tree generation. In the third step, the function bestShapelet is to find the best shapelet from Θ which has the highest information gain. If two or more shapelets have the same gain, we choose the one that maximizes the separation gap [16]. After that, we remove the selected shapelet from Θ in step 4. The function distribute is used to separate the instances in D into two groups, those with a distance $\mathrm{dist}_\beta(S, T_i) \le \tau$ and those with a distance $\mathrm{dist}_\beta(S, T_i) > \tau$. Then, we invoke random_shapelet_tree to generate the left subtree and right subtree based on D_L and D_R, respectively. Finally, the function makeLeaf returns a representation of a leaf in the generated tree by simply assigning the class label that occurs most frequently among the instances reaching the node, dealing with ties by selecting a label at random according to a uniform distribution.

3.4. Time Complexity Analysis. Since the time complexity of applying SAX to represent the original time series data is far

Input: D: a data set of time series; p: the number of trees in forest; r: the number of shapelet for each tree
Output: $\Omega = \{F_1, F_2, \cdots, F_m\}$: a set of random forests and each for one dimension.
1 $\Omega \longleftarrow \varnothing$;
2 $D' \longleftarrow \mathrm{SAX}(D)$;
3 for $j = 1$ to m do
4 $\quad F_j \longleftarrow \varnothing$;
5 $\quad$ for $i = 1$ to p do
6 $\quad\quad \Theta_{i,j} \longleftarrow$ shapelet_sampling(D', j, r);
7 $\quad\quad ST_{i,j} \longleftarrow$ random_shapelet_tree$(D', \Theta_{i,j})$;
8 $\quad\quad F_j \longleftarrow F_j \cup ST_{i,j}$;
9 $\quad \Omega \longleftarrow \Omega \cup F_j$;
10 return Ω;

Algorithm 1: RSFID (Random Shapelet Forest for Intrusion Detection).

Input: D: the data set of time series; j: The dimension of the time series; r: the number of shapelet for each tree
Output: Θ: a set of shapelets
1 $\Theta \longleftarrow \varnothing$;
2 $D' \longleftarrow$ refine(D, j);
3 for $i = 1$ to $r \times \kappa$ do
4 $\quad$ id $\longleftarrow$ rand $(1, |D'|)$, $l \longleftarrow$ rand $(3, \mathrm{len}(T_{\mathrm{id}}) - 3)$, pos $\longleftarrow$ rand $(1, \mathrm{len}(T_{\mathrm{id}}) - l + 1)$;
5 $\quad \theta \longleftarrow$ generateShapelet$(D'$, id, l, pos$)$;
6 $\quad$ if self_similar$(\theta, \Theta) =$ true do
7 $\quad\quad i \longleftarrow i - 1$; continue;
8 $\quad \Theta \longleftarrow \Theta \cup \theta$;
9 sort_by_utility(Θ);
10 $\Theta \longleftarrow$ select_best_top_$r(\Theta)$;
11 return Θ;

Algorithm 2: Shapelet_sampling ().

Input: D: the data set of time series; Θ: a set of shapelets; r: the number of shapelets
Output: ST: a shapelet tree
1 if isTerminal(D) do
2 $\quad$ return makeLeaf(D);
3 $\theta \longleftarrow$ bestShapelet(D, Θ);
4 $\Theta \longleftarrow \Theta/\theta$;
5 $(D_L, D_R) \longleftarrow$ distribute(D, θ) ;
6 $ST_L \longleftarrow$ random_shapelet_tree(D_L, Θ)
7 $ST_R \longleftarrow$ random_shapelet_tree(D_R, Θ)
8 Return (ST_L, ST_R);

Algorithm 3: Random_shapelet_tree ().

less than the generation of random shapelet forest, we only discuss the latter part. In Algorithm 2, the function generateShapelet requires to find the best split of a subsequence whose worst time complexity is $O(nl^2)$ where n is the number of instances in data set and l is the length of time series. Besides, the time complexity of the function self_similar is $O(r^2)$ which is far less than $O(nl^2)$. Therefore, the time complexity of shapelet_sampling is $O(r\kappa nl^2)$. In Algorithm 3, the function random_shapelet_tree requires to select the shapelet that has the highest information gain whose time complexity is $O(rn^2l^2)$. Then, it recursively builds the left subtree and the right subtree. The worst case is that the data set is separated into two subsets with equal size in each

Table 1: Description of UNIT.

	Normal	Attack	S-effect	Total
#training instance	962	198950	768	200680
#testing instance	942	198047	522	199511

Table 2: Description of KDD CUP 99.

	Normal	Probing	DoS	U2R	R2L	Total
#training instance	10000	4107	5467	52	1126	20752
#testing instance	60593	4166	229853	228	16189	3111029

Table 3: The precision and recall values of five algorithms on UNIT (%).

	1NN+DTW		NS		ST+CART		ST+SVM		IDRSF	
	Prec.	Recall	Prec.	Recall	Prec.	Recall	Prec.	Recall	Prec.	Recall
Attack	99.9	87.7	99.9	81.7	100.0	89.3	100.0	89.4	100.0	92.4
S-effect	31.6	77.2	46.7	75.1	64.7	86.6	65.3	87.7	87.4	98.1
Normal	3.7	92.5	2.0	76	3.7	86.8	4.1	94.9	5.7	96.4

Table 4: The precision and recall values of five algorithms on KDD CUP 99 (%).

	1NN+DTW		NS		ST+CART		ST+SVM		IDRSF	
	Prec.	Recall	Prec.	Recall	Prec.	Recall	Prec.	Recall	Prec.	Recall
Probing	82.7	75.2	69.5	71.4	81.7	83.2	91.1	80.4	90.9	87.5
DoS	91.9	86.3	88.2	84.0	98.5	90.2	98.9	94.6	99.9	97.6
U2R	7.5	7.1	11.1	4.2	18.3	15.1	29.6	14.4	48.5	14.0
R2L	26.2	15.9	27.7	3.8	48.8	12.8	65.3	12.2	77.5	13.2
Normal	59.2	87.4	46.2	75.8	63.3	92.5	76.0	92.3	75.7	99.4

iteration. Thus, the complexity is as follows:

$$O\left(rn^2l^2 + (r-1)\left(\frac{n}{2}\right)^2 l^2 + (r-2)\left(\frac{n}{2}\right)^2 l^2 + (r-3)\left(\frac{n}{4}\right)^2 l^2 + \cdots\right) \approx O(rn^2l^2). \tag{4}$$

Obviously, $O(r\kappa nl^2)$ is less than $O(rn^2l^2)$; hence, the overall time complexity of the IDRSF algorithm is $O(rpmn^2l^2)$. Recall that, the symbols r, p, and m represent the number of sampled shapelets, the number of trees in forest, and the number of dimensions in time series, respectively, and it is not difficult to finger out that its time complexity is far less than the time complexity of classical shapelet algorithm, i.e., $O(mn^2l^4)$.

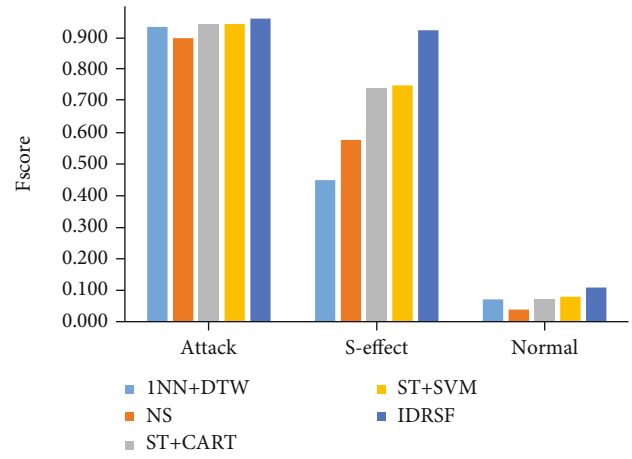

Figure 2: The f-score of five algorithms on UNIT.

4. Experiments

4.1. Data Sets and Parameter Setting. The data sets in the experiments include UNIT [24] and KDD CUP 99 [25], both of which are usually adopted in the field of network security. The UNIT data set includes 14 million records of network attack flows. The collected instances are divided into three groups, which are malicious traffic (attact), side-effect traffic (S-effect), and unknown traffic (normal). Because the size of the UNIT data set is too large to be handled, and meanwhile it lacks normal network traffic, Winter et al. [26] sampled part of instances according to the distribution of the original data set and supplement 1904 instances of normal network traffic. In this paper, we adopt Winter et al.'s data set. KDD CUP 99 is a famous data set for intrusion detection which has 5 million instances of network traffic. There are four different types of network attack in KDD CUP 99, which are labeled as Probing, DoS, U2R, and R2L, respectively. We also sampled 10 percent instances of the original data set and obtained a training data set with 494021 instances and a testing data set with 311029 instances. The details of UNIT and KDD CUP 99 data sets are given in Tables 1 and 2, respectively. Additionally, both data sets were preprocessed, including transform of nominal value to integer value and z-normalization.

The experiments were performed on a PC with Intel Core i7-8700 3.2 GHz CPU and 32 GB RAM. In the proposed algorithm, there are five parameters. We performed cross-validation to decide the parameter settings. The

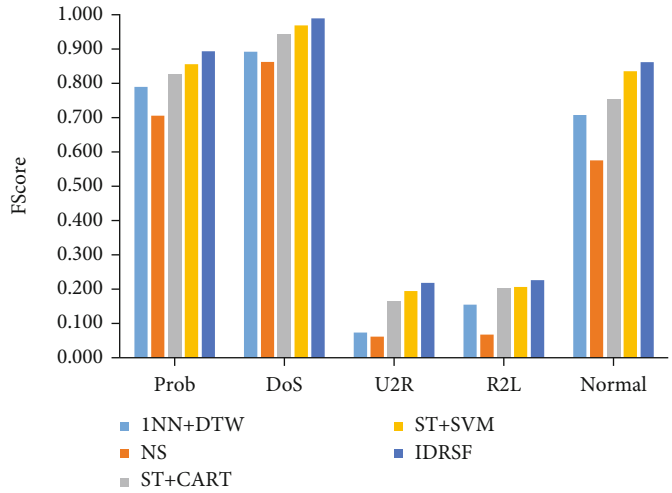

Figure 3: The f-score of five algorithms on KDD CUP 99.

number of shapelets sampled for each forest r is set to $|C| \times \sqrt{m}$, the number of trees p in forest is set to 500, the sampling coefficient κ is set to 1.2, and the two parameters, i.e., δ and λ, for self-similarity detection are set to 2 and 5, respectively.

4.2. Experimental Results and Analysis. To evaluate the effectiveness of the IDRSF algorithm, we choose four algorithms for comparison in the experiments. The first is a classical algorithm, named as 1NN+DTW, which employs one-nearest-neighbor classifier and dynamic time warping. Wang et al. [27] proved that the 1NN+DTW is a classic algorithm for time series classification which is hard to be defeated. Except 1NN+DTW, other three algorithms are all based on shapelet technique, including Naïve Shapelet (NS), Shapelet Transform-based CART (ST-CART) algorithm, and Shapelet Transform-based SVM (ST-SVM).

Additionally, we employed three metrics to evaluate the effectiveness, which are recall, precision, and f-score. It is well known that there are four possible results when predicting a new instance, i.e., true positive (TP), true negative (TN), false positive (FP), and false negative (FN). TP and TN refer to the correct prediction of normal behavior and attack behavior. FP and FN refer to the incorrect prediction. Then, the formulas of the three metrics are given below.

$$\begin{aligned} \text{Recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}}, \\ \text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}}, \\ \text{f-score} &= \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \end{aligned} \tag{5}$$

The precision and recall value of five algorithms on two data sets are given in Tables 3 and 4, respectively. In each table, the experimental results are listed according to the class label. We can find in Table 3 that all the precision values of five algorithms on class "normal" are very low, just from 2% to 5.7%. The rationale behind the result is the unbalance of the UNIT data set. The number of instances in class "normal" is only several hundreds, but tens of thousands of "attack" instances are assigned the label "normal" in the prediction. This dramatically decreases the precision value. The same phenomenon appears in Table 4, e.g., the precision value and the recall value on class "U2L."

For more intuitive comparison, the f-scores obtained by the five algorithms on two data sets are shown in Figures 2 and 3. From the two tables and the two figures, it is not difficult to find that the precision value and the recall value obtained by the IDRSF algorithm on two data sets are obviously better than other four algorithms. Moreover, we can see that the IDRSF algorithm usually performs better on the recall metric, and this is very important for an IDS system. Furthermore, we compare the results of the IDRSF algorithm with that of the state-of-art algorithm (named DSSVM) reported in [28], and we can find that the IDRSF is superior to the DSSVM algorithm. This proves the effectiveness of the proposed algorithm.

However, we cannot ignore that the precision and recall values obtained by the IDRSF algorithm on classes "U2L" and "R2L" are not satisfactory. The reason behind the results is that the testing instances of the two classes include lots of "new patterns" which not appears in the training data set. The shapelet-based technique is essentially a pattern-based method; therefore, it is not easy for the IDRSF to deal with this problem.

5. Conclusions

In this paper, we propose a novel algorithm, named IDRSF, to handle the intrusion detection problem. The algorithm is based on a new primitive "shapelet" in the field of TSC. The advantages of this technique not only include the better ability of classification than traditional techniques in TSC, but also have good interpretability which is not provided by the deep learning methods. The IDRSF algorithm is evaluated on two famous data sets of intrusion detection, i.e., UNIT and KDD CUP 99, and it is compared with four classical algorithms in the field of TSC in which three are based

on shapelet. Experimental results prove the effectiveness of the proposed algorithm. Next, we will try to extend this technique to further handle the unbalance data set and new patterns which not appear in the training set.

Data Availability

The data sets in the experiments include UNIT and KDD CUP 99, both of which are usually adopted in the field of network security [24, 25].

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors thank the anonymous reviewers for their constructive comments, which help the paper quality improved. This work is supported by the National Natural Science Foundation of China (Grant No. 61472050) and the State Key Program of National Natural Science Foundation of China (Grant No. 61332001).

References

[1] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *Journal of Network and Computer Applications*, vol. 84, pp. 25–37, 2017.

[2] W. Lee and S. J. Stolfo, "A framework for constructing features and models for intrusion detection systems," *ACM Transactions on Information and System Security*, vol. 3, no. 4, pp. 227–261, 2000.

[3] N. Gao, D. Feng, and J. Xiang, "A data mining based dos detection technique," *Chinese Journal of Computers*, vol. 29, no. 6, pp. 944–951, 2006.

[4] Y. Liao and V. R. Vemuri, "Use of K-nearest neighbor classifier for intrusion detection," *Computers & Security*, vol. 21, no. 5, pp. 439–448, 2002.

[5] M. Shafiq, Z. Tian, A. Bashir, X. du, and M. Guizani, "IoT malicious traffic identification using wrapper-based feature selection mechanisms," *Computers & Security*, vol. 94, p. 101863, 2020.

[6] F. Jiang, Y. Sui, and C. Cao, "An incremental decision tree based on rough sets and its application in intrusion detection," *Artificial Intelligence Review*, vol. 40, no. 4, pp. 517–530, 2013.

[7] S. S. Sivatha Sindhu, S. Geetha, and A. Kannan, "A decision tree based light weight intrusion detection," *Expert Systems with Applications*, vol. 39, no. 1, pp. 129–141, 2012.

[8] L. Khan, M. Awad, and B. M. Thuraisingham, "A new intrusion detection system using support vector machine and hierarchical clustering," *The VLDB Journal*, vol. 16, no. 4, pp. 507–521, 2007.

[9] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, "CorrAUC: a malicious bot-IoT traffic detection method in IoT network using machine learning techniques," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3242–3254, 2021.

[10] Z. Tian, C. Luo, J. Qiu, X. Du, and M. Guizani, "A distributed deep learning system for web attack detection on edge devices," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1963–1971, 2020.

[11] C. Luo, Z. Tan, G. Min, J. Gan, W. Shi, and Z. Tian, "A novel web attack detection system for internet of things via ensemble classification," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5810–5818, 2021.

[12] S. C. Chin, A. Ray, and V. Rajagopalan, "Symbolic time series analysis for anomaly detection: a comparative evaluation," *Signal Processing*, vol. 85, no. 9, pp. 1859–1868, 2005.

[13] L. Wei, N. Kumar, V. N. Lolla, and E. J. Keogh, "Assumption-free anomaly detection in time series," in *Proceedings of the 17th SSDBM*, pp. 237–240, Santa Barbara, CA, USA, 2005.

[14] Y. Kim, J. Sa, S. Kim, and S. Lee, "Shapelets-Based Intrusion Detection for Protection Traffic Flooding Attacks," in *Database Systems for Advanced Applications*, pp. 227–238, Springer, 2018.

[15] L. Ye and E. Keogh, "Time series shapelets: a new primitive for data mining," in *Proceedings of the 15th SIGKDD*, pp. 947–956, Paris, France, 2009.

[16] A. Mueen, E. J. Keogh, and N. E. Young, "Logical-shapelets: an expressive primitive for time series classification," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1154–1162, San Diego, CA, USA, 2011.

[17] E. J. Keogh and T. Rakthanmanon, "Fast shapelets: a scalable algorithm for discovering time series shapelets," in *Proceedings of the 2013 SIAM International Conference on Data Mining*, pp. 668–676, Austin, Texas, USA, 2013.

[18] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall, "Classification of time series by shapelet transformation," *Data Mining and Knowledge Discovery*, vol. 28, no. 4, pp. 851–881, 2014.

[19] I. Karlsson, P. Papapetrou, and H. Boström, "Generalized random shapelet forests," *Data Mining and Knowledge Discovery*, vol. 30, no. 5, pp. 1053–1085, 2016.

[20] J. Grabocka, M. Wistuba, and L. Schmidt-Thieme, "Fast classification of univariate and multivariate time series through shapelet discovery," *Knowledge and Information Systems*, vol. 49, no. 2, pp. 429–454, 2016.

[21] Z. C. Fang, P. Wang, and W. Wang, "Efficient learning interpretable shapelets for accurate time series classification," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pp. 497–508, Paris, France, 2018.

[22] J. Lin, E. J. Keogh, L. Wei, and S. Lonardi, "Experiencing SAX: a novel symbolic representation of time series," *Data Mining and Knowledge Discovery*, vol. 15, no. 2, pp. 107–144, 2007.

[23] W. H. Yan, G. L. Li, Z. D. Wu, S. Wang, and P. S. Yu, "Extracting diverse-shapelets for early classification on time series," *World Wide Web*, vol. 23, no. 6, pp. 3055–3081, 2020.

[24] M. Sheikhan and Z. Jadidi, "Flow-based anomaly detection in high speed links using modified GSA-optimized neural network," *Neural Computing and Applications*, vol. 24, no. 3-4, pp. 599–611, 2014.

[25] K. Siddique, Z. Akhtar, F. Aslam Khan, and Y. Kim, "KDD Cup 99 data sets: a perspective on the role of data sets in network intrusion detection research," *Computer*, vol. 52, no. 2, pp. 41–51, 2019.

[26] P. Winter, E. Hermann, and M. Zeilinger, "Inductive intrusion detection in flow-based network data using one-class vector support machine," in *2011 4th IFIP International Conference*

on New Technologies, Mobility and Security, pp. 1–5, Paris, France, 2011.

[27] X. Y. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh, "Experimental comparison of representation methods and distance measures for time series data," *Data Mining and Knowledge Discovery*, vol. 26, no. 2, pp. 275–309, 2013.

[28] C. Guo, Y. J. Zhou, Y. Ping, Z. Zhang, G. Liu, and Y. Yang, "A distance sum-based hybrid method for intrusion detection," *Applied Intelligence*, vol. 40, no. 1, pp. 178–188, 2014.