OBNAI: An Outlier Detection-Based Framework for Supporting Network-Attack Identification over 5G Environment

Yi Shen(✉), Yangfu Liu, Jiyuan Ren, Zhe Wang, and Zhen Luo

Northeast Branch of State Grid Corporation of China, Shenyang, Liaoning, China
183835533@qq.com

Abstract. With the development of 5G, network attacking becomes more and more easy. Many system vulnerability are utilized to be attacked via 5G technology. It leads that the network attack frequency turn to high, and the network attack strength turns to strong. Among all network attack identification methods, outlier detection is one of the most important one. It aims to find data which is much different from most of the others. In this paper, we propose an outlier detection based framework to support network-attack identification. It first uses a novel algorithm to construct core point set so as support efficiently outlier detection. Next, it uses a novel index named ZB-Tree to manage these core points. Thirdly, we propose a predictive IP-table to handle and predict suspicious IP addresses. In this way, we could identify most suspicious IP addresses based on the position relationships among different base stations. Theoretical analysis and extensive experimental results demonstrate the effectiveness of the proposed algorithms.

Keywords: IP-Table · Path selection · ZB-Tree · 5G

1 Introduction

The Internet of Things (IoT) and the fifth generation of wireless technology(short for 5G) are restructuring the digital world. However, one problem is network attack becomes more and more seriously. Many system vulnerability are utilized to attack Internet via 5G technology. It leads that both the network attack frequency and network attack strength all turn to high.

In order to avoid network attack as much as possible, many algorithms are used for network-attack identification, such as machine learning, cluster, outlier detection, and so on. These algorithms can be used for identifying different kind of attacks. Among all of them, outlier detection is one type of powerful algorithms. The key idea behind them is an object is regarded as an outlier if it conforms to the unexpected behavior. Since the behavior of attacking traffic

J. Liu et al. (Eds.): MobiCASE 2020, LNICST 341, pp. 68–79, 2020.
https://doi.org/10.1007/978-3-030-64214-3_5

is very different from common traffic, but common traffic usually has similar behaviors, we could use distance relationship among traffics to identify attacking traffics.

In this paper, we study the problem of network attack identification based on outlier detection [1–3]. In order to effectively and efficiently support this problem, we should overcome the above challenges. Firstly, since the speed of network traffic is fast, it is difficult to efficiently find network attack in the premise that most attack could be identified. Secondly, under 5G environment, since the distance among different base station is near, attacker could use different IP to attack hosts via changing their position. Thus, it is difficult to check which traffics are attack traffic via their IP address.

In order to overcome the above challenges, we propose a novel framework named OBI. It first uses historical data to construct a set of new points. In this paper, we call these points as core points. Based on these points, when processing newly arrived traffics, we evaluate the "distance relationship" between core points and newly arrived traffics. If we can find few core points whose distance to these newly arrived traffics are near, we regard them as non-attacking traffic. Otherwise, we check whether these traffic is from an IP address which had generated attacking traffics. If so, we regard it as an attacking traffics. Otherwise, we use an ELM-based algorithm for further identifying. Above all, the contribution of this paper are as follows:

- We propose a novel algorithm to construct core point set. It first selects a group of representative points from the whole data set. Next, we construct a group of core points based on the position relationship among these representative points. We find the overall cost is $\mathcal{O}(N)$.
- We propose a novel index named ZB-Tree to manage these core points. The key idea behind it is we first compute the Z-address of each core point. Next, we use a B-Tree to maintain Z-addresses of these core points. The benefit is we could use, as few as, cost to support range query.
- We propose a predictive IP-table to handle and predict suspicious IP addresses. When an attack is generated from a given 5G base station i, we map it into a hash table for one thing. For another, we predict the new attacking IP addresses based on the location of base stations i. In this way, we can identify most suspicious IP addresses based on the position relationships among different base stations.

The rest of this paper is organized as follows: Sect. 2 gives background, Sect. 3 presents the framework. Section 4 evaluates the proposed methods with extensive experiments. Section 5 is the conclusion.

2 Background

In this section, we review the algorithms about network attack based on outlier detection. Thereafter, we introduce a novel machine learning technique called ELM. Table 1 summarizes the mathematical notations used in the paper.

2.1 Related Works

In this section, we review the algorithms about network attack based on outlier detection. Due to the importance of network attack based on outlier detection, many researchers have studied this problem.

Table 1. The Summary of Notations

Notation	Definition
D	the historical data set
$\mathcal{I}$	the IP address Table
$\mathcal{E}$	the ZB-Tree
$\mathcal{M}$	the machine learning based idtendifyer
G	the grid
c	a cell c of G
G	multi-resolution grid
O_c	a set of objects contained in c
$\|O_c\|$	the number of objects contained in c

Jamshidi Y et al. studied the application of detecting large-scale attacks by the K-nearest neighbor algorithm which based on lattice theory£the experiments show that algorithm has a high detection rate of attack data.

Kuang L et al. proposed a KNN algorithm combined by Strangeness and Isolation indicators in the application of internet intrusion. It uses the two indicators to calculate the outlier score of the sample, and then combine the two interest group degree score to judge whether the sample to be tested is an attack sample. Experiments show that compared to using the KNN algorithm with a single indicator£the combination of the two indicators has a better detection rate and lower false alarm rate.

Zhang J et al. proposed a network anomaly detection algorithm based on random forest algorithm law which uses a random forest algorithm to find outliers. Experiments show that algorithm has higher detection rates and lower false alarm rates than the other three network anomaly detection algorithms.

Bhuyan M H et al. proposed a network anomaly detection algorithm (NADO) based on outlier detection. The algorithm first uses the K-Means algorithm to cluster the training data set, and find the cluster center of each cluster as reference samples. Next, they calculate the distance of each sample to be test the reference sample as the sample outlier degree score value, if this value is greater than the given threshold then output the attack. Experiments show that algorithm has a better detection rate compared with C4.5 and ID3.

Yan Shaohua et al. studied the application of improved LOF algorithm in intrusion detection. Experiments show that when $K = 120$ and $\varepsilon = 1.0$, the algorithm has a higher detection rate on all types of attacks in KDDCUP99.

Guo Chun proposed a network anomaly detection algorithm named NADCP. The algorithm first uses the K-Means algorithm to cluster the training data set, and find the cluster center of each cluster as reference samples. Combined with the idea of over-sampling to calculate the outlier score of the sample, if the outlier of the sample to be tested is greater than a certain threshold, the output is an attack.

Intrusion detection algorithms usually use the idea of nearest neighbors search [4–6], which makes the algorithm should overcome the challenges of neighbor parameters-setting reasonable neighbor parameters can make the algorithm have better detection test results, but if the parameter settings are unreasonable, poor test results may be obtained. In addition, such methods often need manually set the outlier threshold, and the setting of the outlier threshold directly affects the detection rate and false alarm rate of the algorithm. Therefore, how to set a reasonable outlier threshold is also a major challenge for such algorithms.

2.2 Extreme Learning Machine

In this section, we introduce an efficiently machine learning algorithm named (ELM)(short for extreme learning machine). It is developed by Huang et al. ELM is based on a generalized single-hidden-layer feed. Compared with traditional neural networks, its hidden-layer nodes are randomly chosen. It leads that ELM [7,8] has the ability of providing us with good generalization performance at thousands of times faster than traditional popular learning algorithms(e.g.,. SVM, neural networks, and so on). In addition, it has good *universal approximation capability* and *classification capability*.

$$\beta_{k+1} = \beta_k + \mathbf{P}_{k+1}\mathbf{H}_{k+1}^T(\mathbf{T}_{k+1} - \mathbf{H}_{k+1}\beta_k) \tag{1}$$

$$\mathbf{P}_{k+1} = \mathbf{P}_k - \mathbf{P}_k\mathbf{H}_{k+1}^{\mathbf{T}}(I + \mathbf{H}_{k+1}\mathbf{H}_k\mathbf{H}_{k+1}^T)^{-1}\mathbf{H}_{k+1}\mathbf{P}_k \tag{2}$$

Compared with the traditional ELM, Huang et al. also propose the online sequential extreme [7] learning machine (abbreviated as OS-ELM). it can learn data one-by-one or chunk-by-chunk with fixed or varied size. Thus, it is suitable for processing streaming data.

$$\mathbf{H}_{k+1} = \begin{bmatrix} \mathbf{G}(\mathbf{a}_1, b_1, \mathbf{x}_{\sum N_j+1}) & \dots & \mathbf{G}(\mathbf{a}_N, b_N, \mathbf{x}_{\sum N_j+1}) \\ \vdots & \dots & \vdots \\ \mathbf{G}(\mathbf{a}_1, b_1, \mathbf{x}_{\sum N_j+1}) & \dots & \mathbf{G}(\mathbf{a}_1, b_1, \mathbf{x}_{\sum N_j+1}) \end{bmatrix}_{N_{k+1}\times l} \tag{3}$$

Given a set of samples (x_i,t_i), where $x_i = [x_{i1}, x_{i2},...,x_{in}]^T \in \mathcal{R}^n$ and $t_i = [t_{i1}, t_{i2},...,t_{in}]^T \in \mathcal{R}^m$, OS-ELM first selects the activation function, the hidden node number and so on. Then, OS-ELM is employed in a two-phase method including: (i) initialization (ii) sequential learning.

$$\mathbf{T}_{k+1} = \left[\mathbf{T}_{(\sum_{j=0}^{k} N_j)+1} \dots \mathbf{T}_{(\sum_{j=0}^{k} N_j)}\right]_{N_{k+1}\times m}^{T} \tag{4}$$

In the initialization phase, OS-ELM uses a small set of samples for training. The second phase employs the learning in a chunk-by-chunk way. To be more specific, in the k-th chunk of new training data, OS-ELM firstly computes the partial hidden layer and the output matrix $\mathbf{H}_{k+1}$. Lastly, OS-ELM computes the output weight matrix β_{k+1}. Here, β_{k+1},$\mathbf{H}_{k+1}$, and $\mathbf{T}_{k+1}$ are computed according to Eq. 5 to Eq. 4.

3 The Framework ODNAI

In this section, we propose an outlier detection based framework to identify network attack. We first discuss the basic idea. Secondly, we discuss how to construct the cluster. Thirdly, we discuss the network attack based on outlier detection.

3.1 The Basic Idea

Formally, let O be a set of historical network data. We carefully select part of them as *verification objects*. Our goal is using these objects to evaluate which newly arrived objects are outliers. In order to find high quality objects, we propose a *local-density* based algorithm, that is, if a region contains many objects, we select few of them as verification objects. Based on these verification objects, we further select fewer objects as *core-object*s. In Sect. 3.2, we will discuss the *core-object* set construction algorithm.

Algorithm 1: The Framework Overview

Input: IPBase $\mathcal{I}$, CSet $\mathcal{C}$, object o
Output: type c

```
1  bool rst r ←searchCore(C, o) ;
2  if rst is not outlier then
3      return;
4  else
5      bool bIn r ←searchIP(I, o) ;
6      if bIn is false then
7          bool bIn r ←ELMID(E, o) ;
8          if rst is true then
9              I ← insertion(I, o);
10             I ← prediction(I, o);
11     return;
```

The function of *core-object*s is when a newly arrived object o traffic into the system, we compute the distance between them and *core-object*s. If we find that the distance between o and *core-object*s are all longer than a threshold, we regard

it as an outlier, and further evaluate whether it is a network attack based on the IP-table. We will discuss the *core-object*s maintenance and how to quickly evaluate whether o is a network attack in Sect. 3.3.

The IP-Database maintains a group of suspicious IP address. In the 5G environment, since network attackers could continuously change their locations so as to exchange their IP address, suspicious IP in the IP-Database may be frequently changed. In this paper, we propose a hash based index, which is used to maintain suspicious IP addresses. In addition, we should predict new suspicious IP addresses that may appeared in the future.

If a traffic is identified as an attack traffic, but it is not contained in the IP cache, we use a machine learning algorithm named ELM for further identifying it. Here, ELM is proposed by Huang et al. It is based on a generalized single-hidden-layer feed. Compared with neural networks, its hidden-layer nodes are randomly chosen instead of iteratively tuned. The benefit of ELM is it could provide us with good generalization performance at thousands of times faster than traditional popular learning algorithms. Thus, it is suitable for identifying large scale attacking traffic. Since ELM is widely used in many application, we do not explain the ELM-based identifying algorithm in this paper.

As is shown in Algorithm 1, when an object traffics into the system, we first evaluate whether it is an outlier(See line 1). If not, we regard it as an inlier, and do not process it. Otherwise, we should check whether its corresponding IP address is contained in the suspicious IP addresses table. If the answer is yes, we regard it as an attack traffic. Otherwise, we use a machine learning based method to further process it. If the identification result shows that o is a attack traffic, we insert the corresponding address of o into the suspicious IP addresses table $\mathcal{I}$. In addition, we predict which base station will be used by attackers based on the location information of the current attacker.

3.2 The Outlier Based Identification

In this section, we first discuss how to construct the core set. Next, we discuss searching on core object set.

The Core Set Construction. In this section, we select a set of *non-attack* objects to construct Core Set. Let D be a set of *non-attack* objects with size N. The core set construction contains two steps, which are *mapping* and *selection*. In the first step, we construct a grid G, and map all the objects into G according to their coordinates. Here, the side-length of each $c \in G$ equals to r.

After mapping, we compute the number of objects in each cell. Based on the computational result, we find all cells in G whose scores are larger than k. Here, the score of a cell equals to the number of objects contained in it. For these cells, we randomly select k objects in each cell, and delete the others. Lastly, let O_c be a set of objects contained in the cell c. We compute the center of objects in c, use computing result as core points.

Take an example in Fig. 1. Black points refers to the historical objects. We first partition the whole space into $8 * 8$ cells, and then map these objects into

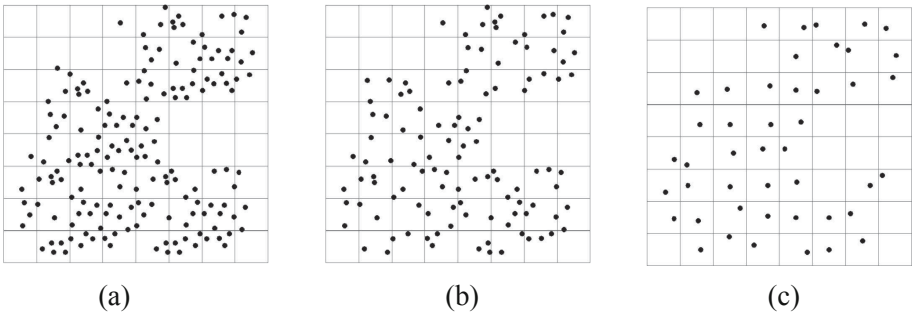

Fig. 1. Problem Definition($k = 3$)

these cells. Let the threshold k be 3. We select all these cells with scores larger than 3, and randomly delete parts of objects contained in them, i.e., the grey cells. The deletion result is shown in Fig. 1(b). Lastly, we compute the core points based on these selected objects. The result is shown in Fig. 1(c).

Indexing Core Set. Once the core set is constructed, we are going to index these core objects. Since core objects are generated by grid, we propose a Z-address based index named ZB-Tree (Fig. 2).

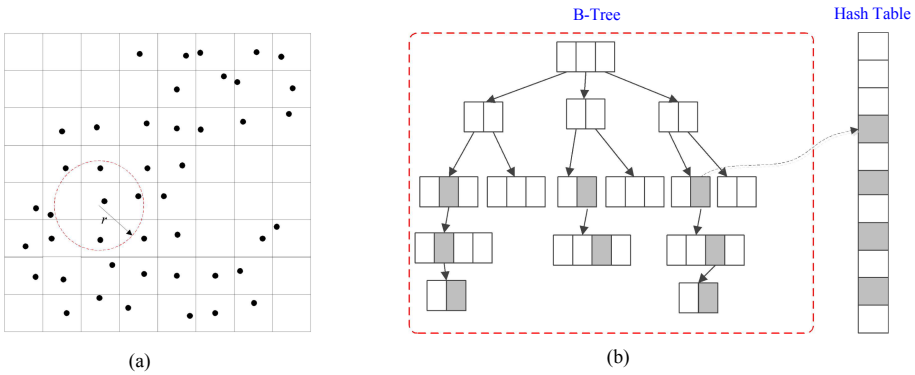

Fig. 2. Indexing Core Points

As is depicted in Fig. 3, ZB-Tree is a Z-curve [2,3] based two-level index. The first level of ZHB-Tree is a B-Tree, which is used to maintain location relationship among all non-empty cells. Since the Z-address of each non-empty cell is an integer, we use a B-Tree to maintain these non-empty cells. The second level of ZHB-Tree is a hash table, which is used to maintain the Z-address of these cells. Since the concept of Z-address has been discussed in many works, we skip the details for saving space.

Searching on ZB-Tree. In this section, we discuss the searching algorithm. Let o_{in} be a newly arrived object. The intuition behind it is if o_{in} is contained in the non-empty cell, it means the distance between it and many objects contained in the historical data set is near. Thus, it should be regarded as a non-attack traffic. In addition, if there exists few non-empty neighbour cells of o_{in}, we should assert that the distance between it and many objects contained in the historical data set is near. Under these cases, we also could assert that o_{in} is not an attack traffic.

Specially, we first compute the Z-address of o. Next, we access the hash table to find whether existing a non-empty cell whose Z-address is maintained by the hash table H. If the answer is yes, the searching algorithm is terminated. Otherwise, we compute the Z-addresses of o's neighbour cells. Based on the computing result, we search on the B-Tree to find whether these neighbour cells are maintained by this B-Tree. If so, we also regard o as a non-attack traffic. Otherwise, we regard it as an attack traffic.

3.3 The IP-Cache Algorithm

As is depicted in Sect. 3.1, if a traffic is identified as an attack traffic, its corresponding IP address is maintained by the hash table H. Since the attack host may be frequency changed, we only maintain the IP addresses of hosts they generate attack traffic in a few hours. In order to achieve this goal, we use a B-Tree to maintain the last attack moment of each attack IP address. In this way, we could know which attack host could be removed from the IP-Cache table based on its last attacking moment.

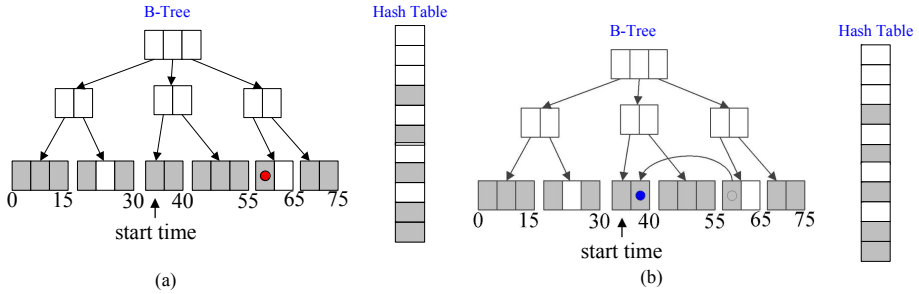

Fig. 3. Indexing Core Points

The IP-Cache Structure. Specially, we use a hash table to maintain the IP address for one thing. For another, we use a B-Tree to maintain the last attack moment of each attack IP address. When a new attack is appeared, we insert the moment of corresponding IP address according to the start time. When an traffic is repeatedly attacking the host, we remove the reference of IP address from its new position. In particularly, if a given IP address has not attack to the

host period of time, we remove it from both the hash table and the B-Tree at the same time.

Take an example in Fig. 3. We use the hash table H to maintain attack IP. In addition, we use a B-Tree to maintain the reference of these IP based on their last attacking moment. In addition, the current moment is 35, and we monitor the attacking moment in the last 75 min. The red points refers to an IP which has attack the host 75–30 min before. When a new attacking is appeared, since its corresponding IP address equals to the IP address of the red point, we remove it to another node, i.e., the node with time stamp region [30,40].

The Prediction Algorithm Based on IP Cache. In 5G environment, since the distance among different base-station is short, attacker has the ability of frequently changing IP address. Aiming to this problem, we propose a prediction algorithm that is used for predicting which base-stations attackers may use.

Specially, we first find the location of the base-station b based on the IP address. Next, we submit a range query to find another base-stations whose distance to is nearer than 1km, and obtain these base-stations' IP address. Lastly, we insert these IP addresses into the IP Cache. Since the range query algorithm is simple, we skip the details for saving space.

4 Experimental Evaluation

In this section, we conduct extensive experiments to demonstrate the efficiency of OBNAI. The experiments are based on one real data set and one synthetic dataset respectively. In the following, we first explain the settings of our experiments, and then report our findings.

4.1 Experimental Setting

Data Set. In total, two datasets are used in our experiments, two real data set namely KDDCUP99 and NSL-KDD. The data set is generated from the MIT Lincoln Laboratory, which is used for network detection evaluation. This data set contains roughly 5000000 tuples via simulating the network environment over 7 weeks. Among them, we use 2000000 tuples as training data. In order to apply these two data sets under intrusion detection research, Sal stolfo and Wenke Lee used data mining to analyze its characteristics. Therefore, it is the most widely used data set.

NSL-KDD is oriented from KDDCUP99. Compared with KDDCUP99, it removes redundant tuples from KDDCUP99. In addition, it makes tuple amount difference among different type of tuples small. In this way, it could solve the problem of in-balance. The benefit is it could make the accuracy of training result turns to high.

Metrics. In our experiments, we measure the following metrics by varying different parameters of the system, which are *detection rate*(short for DR), *false*

alarm rate(short for FAR) and the *accuracy*(short for ACC). Here, they are computed based on the following three equations. Here, P refers to the non-attacked tuple amount, N refers to attacked tuple amount, TP refers to true positive, FN refers to false negative, and FP refers to false positive. In addition, $TP + FN$ equals to P, and $FP + TN$ equals to N.

$$DR = \frac{TN}{N} \tag{5}$$

$$FAR = \frac{FN}{P} \tag{6}$$

$$ACC = \frac{TP + TN}{P + N} \tag{7}$$

Comparisons. we compare the results of OBNAI with a baseline algorithm named BNAI. Compared with OBNAI, it use a baseline outlier detection algorithm. In addition, it does not predict which IP address may generate attack traffics in the future.

4.2 Algorithm Performance

We first evaluate the impact of parameter r to these two algorithms' running time. We find that, with the increasing with r, the running time of these two algorithms are all turning to high. However, compared with the baseline algorithm, the running time of OBNAI increases much lower. The reason behind it is, we construct a small number of core points. In this way, we could use these points for evaluating which ones could be regarded as outliers. Compared with the baseline algorithm, its running time must be lower. Another reason is we use predict algorithm to evaluate which IP address may generate attacked traffics. Thus, we could avoid using machine learning algorithm for further identifying (Figs. 4 and 5).

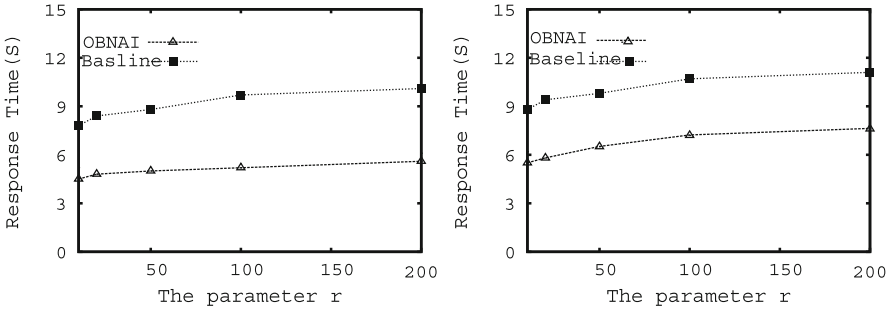

Fig. 4. Running time comparison under different r

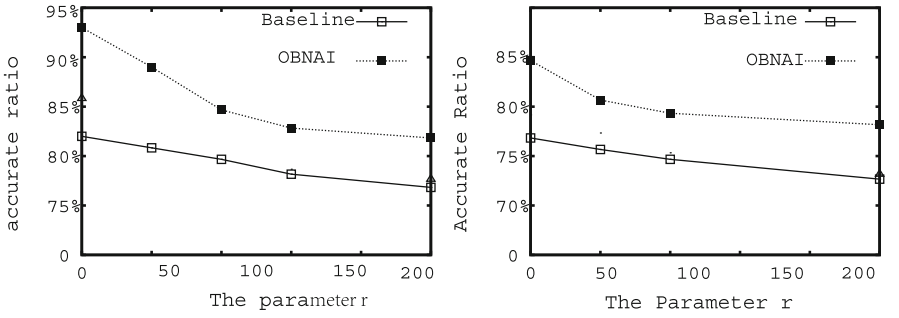

Fig. 5. Accurate ratio comparison under different r

Next, we evaluate the impact of r to the accurate ratio. We find that with the increasing of r, the accurate ratio of these two algorithms all turn to low. The reason behind is when r turns to large, many inliers are wrongly regarded as outliers. However, we also find that when r is large enough, the accurate ratio of OBNAI is not reduced a lot. The reason behind it is, we use a machine learning algorithm for further identifying attacked traffics. Therefore, even a record is wrongly identified a attacked traffic, we could use machine learn algorithm for further verification.

5 Conclusions

In this paper, we propose an outlier detection based framework to support network-attack identification. It first uses a data compression algorithm to reduce data set scale, and then use a Z-order based B-Tree to maintain these core points. The benefit is we could use, as small as, cost for outlier detection. Based on the detecting result, we propose a predictive IP-table based algorithm to find suspicious IP addresses. Last of all, we use a machine learning algorithm named ELM for further verifying suspicious traffics.

References

1. Tran, L., Fan, L., Shahabi, C.: Distance-based outlier detection in data streams. PVLDB **9**(12), 1089–1100 (2016)
2. Juhua, P., Wang, Y., Liu, X., Zhang, X.: STLP-OD: Spatial and temporal label propagation for traffic outlier detection. IEEE Access **7**, 63036–63044 (2019)
3. Zhu, J., Wang, Y., Zhou, D., Gao, F.: Batch process modeling and monitoring with local outlier factor. IEEE Trans. Control Syst. Technol. **27**(4), 1552–1565 (2019)
4. Georgiadis, D., Kontaki, M., Gounaris, A., Papadopoulos, A.N., Tsichlas, K., Manolopoulos, Y.. Continuous outlier detection in data streams: an extensible framework and state-of-the-art algorithms. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, 22–27 June 2013, pp. 1061–1064 (2013)

5. Hawkins, D.M.: Identification of Outliers. Monographs on Applied Probability and Statistics. Springer, Heidelberg (1980). https://doi.org/10.1007/978-94-015-3994-4
6. Knorr, E.M., Ng, R.T.: Algorithms for mining distance-based outliers in large datasets. In: Proceedings of 24rd International Conference on Very Large Data Bases, VLDB 1998, New York City, New York, USA, 24–27 August 1998, pp. 392–403 (1998)
7. Rong, H.J., Huang, G.B., Sundararajan, N., Saratchandran, P.: Online sequential fuzzy extreme learning machine for function approximation and classification problems. IEEE Trans. Syst. Man Cybern. **39**, 1067–1072 (2009)
8. Huang, G.-B., Zhou, H., Ding, X., Zhang, R.: Extreme learning machine for regression and multiclass classification. IEEE Trans. Syst. Man Cybern. **42**, 513–529 (2012)