An Evaluation of Machine Learning Method for Intrusion Detection System Using LOF on Jubatus

Tadashi Ogino*
Okinawa National College of Technology, Okinawa, Japan.

* Corresponding author. Email: ogino@okinawa-ct.ac.jp
Manuscript submitted August 5, 2014; accepted March 8, 2015.
doi: 10.17706/jsw.10.6.748-756

Abstract: The network intrusion is becoming a big threat for a lot of companies, organization and so on. Recent intrusions are becoming more clever and difficult to detect. Many of today's intrusion detection systems are based on signature-based. They have good performance for known attacks, but theoretically they are not able to detect unknown attacks. On the other hand, an anomaly detection system can detect unknown attacks and is getting focus recently. We study an anomaly detection system as one application area of machine learning technology. In this paper, we study the effectiveness and the performance experiments of one of the major anomaly detection scales, LOF, on distributed online machine learning framework, Jubatus. After basic experiment, we propose a new machine learning method and show our new method has a better performance than the original method.

Key words: Anomaly detection, machine learning, Jubatus, LOF.

1. Introduction

As the internet is spreading in our daily life and in the business scene, the attacks using the internet are increasing. Although the technology of defense from such attacks is making progress rapidly, the attacks are becoming smarter.

There are mainly two different approaches for network intrusion detection technology [1], [2]. One is signature-based technology, and another is anomaly detection technology.

With signature-based technology, the system has set of attack patterns and compares them with actual transferred data. When the data match the attack patterns, it means the data is an attack. This system can detect all the data in the attack dataset, but cannot detect new attacks which are not included in the attack dataset.

The anomaly detection system has a normal behavior pattern profile about the defense system. When coming data are different from the normal pattern, they are taken as attacks. This system can detect new attacks. The problem is that this system has a possibility of alarming for normal data as attacks.

Majority of intrusion detection studies had been about signature-based technology. The anomaly detect technology has been getting focus recently. Studies in the area of machine learning, big-data analysis and so on, have been applied to anomaly detection studies. As a result, anomaly detect systems with feasible performance are being developed.

The objective of our research is to build an intrusion detection system combining current up-to-date big-data technologies. As a preliminary study, we evaluate the performance of anomaly detection algorithm

"LOF" [3] on the online machine learning framework "Jubatus" [4].

In this paper, after the system overview is explained in Section 2, how our system detects intrusions is shown in Section 3. Basic experiment is executed in Section 4 and its result is in Section 5. Our new learning method and its result are in Section 6 and Section 7. The future works are in Section 8. Section 9 is conclusion.

2. System Overvew

The basic process of our system is shown in Fig. 1 [5]. The system collects the system logs, i.e. traffic log, manipulation log and so on. All the collected logs are statically analyzed and learned as normal data. In this period, we estimate the system is not attacked. The system can detect new attacks after this learning period.

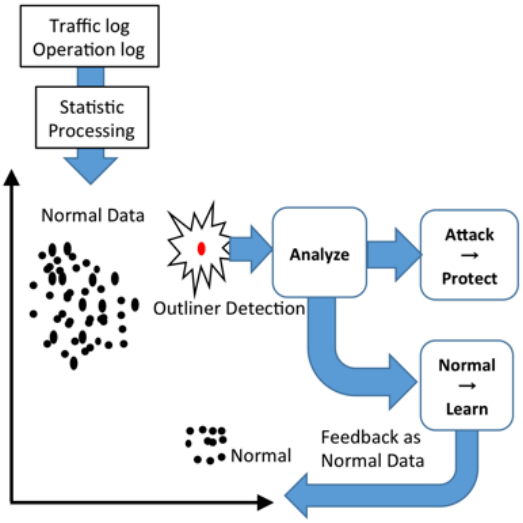

Fig. 1. Overview of system processing flow.

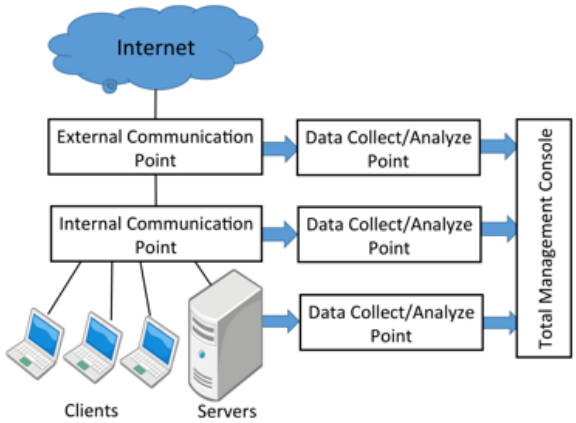

Fig. 2. System configuration.

After a suitable period, the system starts to analyze the collected data. When the system detects the outlier, it analyzes the data in more detail. The outlier is not necessarily the intrusion. When new tools or new services are introduced in the user system, it might produce new traffic or usage patterns. So it is necessary to check if the outliers are the results of normal usage or attacks. We suppose that these tasks are executed by the system administrators. When the outliers are intrusions from outside, they are blocked. On the other hand, when the outliers are results from proper but new procedures, they are permitted and learned as normal data. After some proper amounts of those data are learned, they are not classified as outlier data.

This feedback step of registering the proper but new procedure as normal data is necessary, since a lot of new services appear in a short time recently.

The specific objectives of this system are as follows.

1) The system can detect new unknown intrusions.
2) The system can detect intrusions in real time.
3) Not only the communication from/to outside, but also irregular communications/operations in the local area network can be detected.
4) Detected incidents can be investigated by system administrators and decided if they are actual intrusions or normal usage.

The total system configurations are shown in Fig. 2. Each component has the following functions.

1) External Communication Point: Communication points between externalnetworks, i.e.. internet and internal systems (LAN). For the purpose of BCP enhancement, more than two external

communication points might be equipped. Traffic logs from/to internet are collected from these points.

2) Internal Communication Point: Communication points between local clients and servers. Usually these points are built by routers/switches. Traffic logs in the local system are collected from these points.
3) Servers: File servers, print servers, application servers, etc. Server operation logs are collected.
4) lients: Each client in the system. Client operation logs are collected.
5) Data Collect/Analyze Point: Data Collect/Analyze Points collect all the traffic logs and operation logs, analyze all the data and find the abnormal data. In the case of servers/clients, data analysis function can be executed by themselves.
6) Total Management Console: All the found suspicious incidents are gathered. System administrators can invest and manage all the incidents.

This paper describes the evaluation of collecting traffic data at External Communication Point and detecting anomaly data.

3. Detection Method

This chapter explains how our system analyzes the traffic data and finds anomaly data in real time.

3.1. Anomaly Detection

A couple of studies have been conducted on anomaly detection from huge data sets. Most of them come from extensive studies for clustering. The main objectives of such studies are to find anomalies to remove them from the data set, since they are "noises" for clustering purpose. There are less studies about anomalies in order to process anomalies as main subject. In this paper, we use LOF (Local Outliner Factor) algorithm [3] from such 'outlier-oriented' studies. It is reported that LOF is superior to other outlier detect algorithm for detecting network intrusion [6]. Another reason to use LOF is that machine learning framework Jubatus has LOF algorithm for its standard repertoire. Jubatus will be explained later in this chapter.

3.2. LOF

There are a couple of definitions for outliers. Hawkins gave the definition of outliers as "an observation that derives so much from other observations as to arouse suspicion that it was generated by a different mechanism." [7].

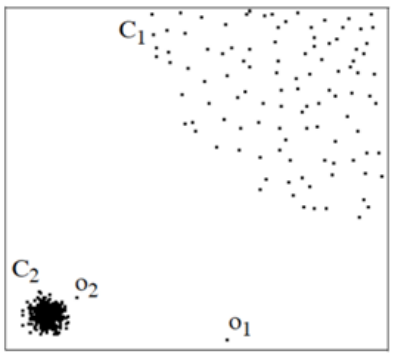

Fig. 3. 2d Dataset example.

Most of the cases, this definition is enough for detecting outliers. But, in the case of Fig. 3 [3], though o1 can be detected as outliner, o2 cannot be detected. LOF is designed to find such cases. The details of LOF algorithm are explained in the original paper.

3.3. Detection Examples

Some examples of detecting cyber attacks using outlier detect algorithm are as follows:

1) Example 1: When the time period of packet transmission is too short, there are possibilities of DoS attacks.
2) Example 2: When the length of the packet is too long, there are possibilities of BufferOverFlow attacks.

Even in the case of new unknown attacks, since there may exist some different parameters for normal traffic, the possibility to be detected as outliers might be high.

3.4. Jubatus

The amount of network traffic is increasing dramatically. In order to analyze those data in real time, it would be better to apply 'big data' analysis technology to manipulate huge data.

One of the most famous big data analysis tools is Hadoop [8]. Hadoop is an open source software tool. It is used in the wide areas such as recommendation, web search, text mining, etc. Hadoop is usually used under batch processing. It is not suitable to use it for real time analysis.

We use Jubatus [4] for our real time analysis platform. Jubatus is a distributed machine learning platform developed by NTT and PFI. It is developed for the purpose of real-time, deep analysis in a distributed environment.

4. Basic Experiment

For the purpose of preliminary study, we evaluate the performance of LOF algorithm running on Jubatus framework with KDD Cup 99 data [9].

4.1. Evaluation System Hardware

We use an aws (Amazon Web Service) t2.micro instance (1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory) with Ubuntu Server 14.04 as an evaluation platform. Jubatus is ver 0.6.0.

4.2. Evaluation Data

We use KDD Cup '99 data as the experiment traffic data. KDD Cup is a Data Mining and Knowledge Discovery competition organized by ACM. In 1999, the main topic was Network Intrusion Detector. The competition data of KDD Cup 99 can be used as the evaluation data for network intrusion detection system[10]. The data simulates the typical U.S. Air Force LAN. The raw data was 4GB of compressed TCPdump format from seven weeks of network traffic. This data was processed to 5 million connection records. Each connection record is labeled as either a normal, or an attack. An attack data has its attack type such as buffer_overflow, guess_passwd etc. Each record consists of 41 columns and the record size is around 100 bytes. They also supply 10% data, which has the same data distribution and 500 thousand records. We use this 10% KDD Data as an evaluation data.

Although we know that there are a couple of discussions about using KDD99 data as evaluation data [11], for now it is the only candidate for using network intrusion evaluation data.

4.3. Jubatus Parameters

Jubatus has LOF processing function. We use this function with no modification to the source code. KDD Cup 99 data is also used with no modification. It is treated as 41 dimensional data.

Jubatus Server configuration parameters are shown in Fig. 4.

4.4. Learning Method

In this experiment, we use a very simple learning method which is as follows:

Step 1: Use first fixed number of normal data as training data.

Step 2: Analyze the remaining data using LOF.

```
{
 "method" : "light_lof",
 "parameter" : {
  "nearest_neighbor_num" : 10,
  "reverse_nearest_neighbor_num" : 30,
  "method" : "euclid_lsh",
  "parameter" : {
   "hash_num" : 8
  }
 },

 "converter" : {
  "string_filter_types": {},
  "string_filter_rules": [],
  "num_filter_types": {},
  "num_filter_rules": [],
  "string_types": {},
  "string_rules": [{"key":"*",
   "type":"str",
   "global_weight" : "bin",
   "sample_weight" : "bin"}],
  "num_types": {},
  "num_rules": [{"key" : "*",
   "type" : "num"}]
 }
}
```

Fig. 4. Jubatus parameters.

4.5. Jubatus APIs

We use 2 Jubatus APIs for our system.

add(data) : Add data as a training data and returns LOF score.

calc_score(data) : Calculates and returns LOF score. Data is not added as a training data

As add( ) API has to update Jubatus internal data model, it takes more time than calc_score( ) API which just calculates LOF score for the data.

We measured the execution time for each 2 APIs.

The execution time of add( ) is measured as follows:

1) add( ) all the normal data
2) measure execution time for each add( ) API
3) calculate minimum, average and maximum of 100 consecutive API executions

The execution time of calc_score( ) is measured as follows:

1) add( ) fixed number of normal data
2) calc_score( ) all the remaining data
3) calculate the average time for calc_score()

5. Result of Basic Experiment

We evaluate the system with different number of training data, including from1000 to 10000.

5.1. API execution time

The execution time for 2 APIs depends on the number of training data. Theoretically, these times are in proportion to n_2, where n is the number of training data. As a matter of fact, Jubatus uses a couple of techniques to decrease execution time. Fig. 5 shows the actual execution time for add ( ) API. As the execution time changes according to the data itself and Jubatus internal model, the result is shown with minimum, average and maximum time.

Fig. 6 shows the execution time for calc_score() API. Be careful that units of each graph are different, seconds for Fig. 5 and milliseconds for Fig. 6.

Total execution time is shown in Fig. 7.

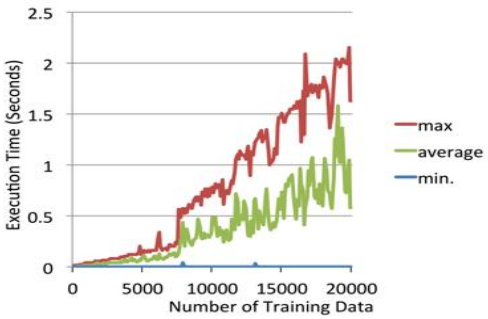

Fig. 5. Execution Time of add() API.

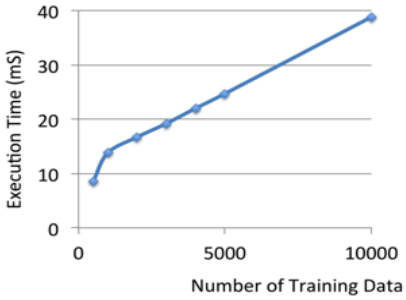

Fig. 6. Execution Time of calc_score() API

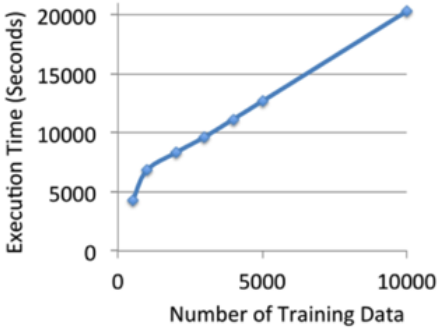

Fig. 7. Total Execution time.

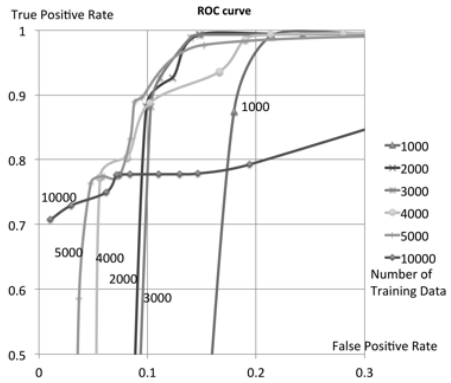

Fig. 8. ROC Curve for basic experiment.

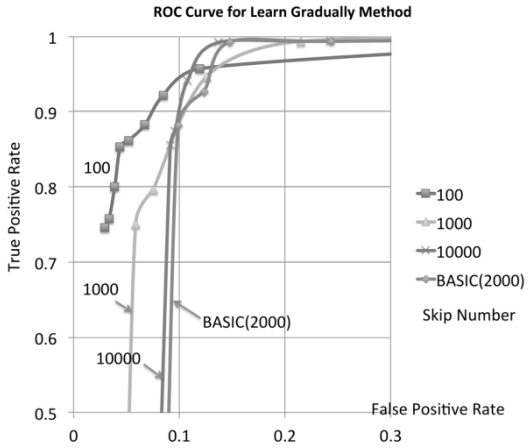

Fig. 9. ROC curve for learn gradually method.

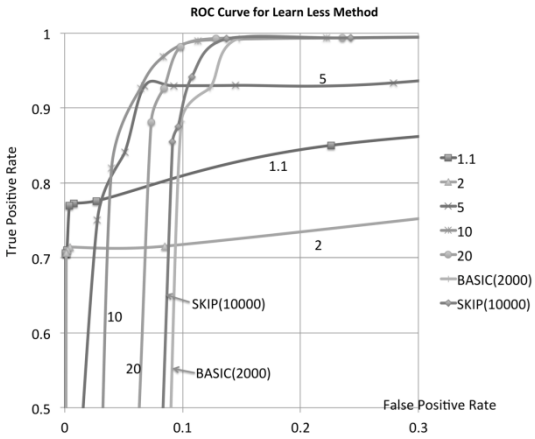

Fig. 10. ROC curve for learn less method.

5.2. ROC Curve

As our system uses LOF for analytic algorithm, the direct output of the each data is LOF score. We need to set a threshold to decide if the data is an attack or not. After we set a threshold, we can count True Negative (normal data under threshold), False Positive (normal data over threshold), False Negative (attack data under threshold) and True Positive (attack data over threshold). True Positive Rate and False Positive Rate are calculated with these data.

We can draw ROC curve with different threshold. Fig. 9 shows the ROC curve of the basic experiment.

From this result, the accuracy of detecting increases according to the size of training data as a rough trend.

On the other hand, the total execution time increases with more training data. In order to meet the conditions of these conflicting, we propose a new learning method in the next chapter.

6. Experiment of Learning Method

In order to improve the total performance of our system, we try the following 2 learning methods.

6.1. Learn Gradually Method

As learning all the data needs more time, we should decrease the number of training data. At the same time, we need to learn new data to follow the change of trend. We try the "Learn Gradually Method" which shown in the following steps.

Step 1: Train with the first fixed number of normal data (same as the simple learning).

Step 2: Train once for every fixed number (= skip number) of normal data.

Fig. 10 shows the ROC curve for this method under the same environment as Basic Experiment with skip number is from 100 to 10000. Label BASIC(2000) is the original simple learning method and the same data for label 2000 in Fig. 9. The result shows this method has a little better than the original method.

6.2. Learn Less Method

To achieve the accuracy of detecting and the less execution time at the same time, the training data should be limited to the valuable data. That means that only the meaningful data for the following analysis should be added. In our example, we assume the data with high LOF score is valuable.

When the LOF score for a new data is low, it means there are enough other data around the data and adding that data into the training data doesn't increase the total system accuracy so much. On the other hand, when the LOF score for a new data is high, it means there are less other data around the new data so adding this new data is important for increasing the total system performance. We call this method as "Learn Less Method".

A simple algorithm for "Learn Less Method" is as follows:

Step 1: Train with the first fixed number of normal data (same as the simple learning).

Step 2: Calculate the LOF score for each data.

If LOF score is over the threshold (we call this as "Value Threshold"), then add this data as training data.

We evaluate this method under the same environment as the basic experiment. Fig.11 is the ROC curve for this method with number of first training data is 2000 and the threshold is from 1.1 to 20. Label SKIP(10000) is the result of "Learn Gradually Method" with Skip Number is 10000. The result shows that the cases with Value Threshold of 5,10 or 20 have the better detection rates than the original simple method and "Learn Gradually Method".

7. Discussion

7.1. Execution Time

Our target is to detect cyber attacks in real time. KDD Cup '99 data consist of 7weeks traffic data and include 5 million traffic data. That means 1 traffic data is around 0.8 seconds. Jubatus calc_score( ) API is around dozens of milliseconds and is small enough compared to one traffic data time in our experiment. Jubatus add( ) API is a couple of seconds in the worst case and when this API is called we need some buffering function to keep the analyzing procedure continued. And the fact that KDD Cup 99 data was made 15 years before, we need to evaluate the overall system performance under current network traffic and current hardware.

In order to improve system performance, we have some candidates:

7.1.1. Parameter tuning

Jubatus and LOF algorithm have some tuning parameters. As we haven't tuned such parameters this time, tuning such parameters might improve total system performance.

7.1.2. Distributed Configration

As Jubatus is designed to run in a distributed environment with multiple servers, this can improve system performance.

7.1.3. Discard Data

In this experiment, we learned that the number of learned data affects the system performance dramatically. Our proposed method decreases the amount of training data increase, but not the total size of the training data. We need to discard some old or useless data from the training data. This will keep the total number of training data as fixed.

7.2. Detection Rate

As shown in the previous chapter, when we choose the suitable Value Threshold, our proposed "Learn Less Method" have better detection rates than original method and "Learn Gradually Mode". The problem is that finding suitable threshold and other parameters are difficult task and such parameters depend on the data features. We have to evaluate different dataset other than KDD Cup 99.

The detection rate of this experimental system is around 90% to 98% with False Positive Rate is a little under 10%. This number is not enough for actual network intrusion detection system. We need to study how to decrease False Positive Rate more.

8. Future Works

We plan to invest the following issues.

8.1. Automatic Parameter Tuning

The performance of the proposed system might change with other data set or other parameter tuning. We need to study how to find the best combination of the parameters automatically or manually with ease by administrators.

8.2. Better Machine Learning Algorithm

The number of learned data affects the total system performance. We need to find a better learning algorithm to continue to learn and not to decrease performance.

8.3. Administration User Interface

We need to develop the administrator interface for managing the incidents with ease.

9. Conclusion

In this paper, we evaluate the cyber attack detection system using LOF on Jubatus platform. Our evaluation shows the execution time of the system is small enough for building real time detection system. The detection rate can be improved with the proposed learning method. We need more studies to performance improvement and better learning strategy.

References

[1] Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, *41(3)*, 2009

[2] Jyothsna, V., Prasad, V. V. R., & Prasad, K. M. (2011). A review of anomaly based Intrusion detection system. *International Journal of Computer Applications*, *28(7)*.

[3] Breunig, M. M., Kriegel, H. P., Ng, R. T., & Sander, J. (2000). LOF: Identifying density-based local outliers. *Proceedings of the ACM SIGMOD 2000 International Conference On Management of Data.*

[4] Jubatus. Retrieved, from http://www. jubat.us/

[5] Ogino, T. (2014). An evaluation of intrusion detection system on Jubatus. *Proceedings of the 23rd International Conference on Systems Engineering* (pp. 359-364).

[6] Lazarevic, A., *et al*. A Comparative study of anomaly detection schemes in network intrusion detection. P*roceedings of SIAM Conference on Data Mining*.

[7] Hawkins, D. M. (1980). *Identification of Outliers*. Chapman and Hall.

[8] Hadoop. Retrieved, from http://www. hadoop.apache.org/

[9] KDDCup. Retrieved, from http://www. kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[10] Singh, R., & Singh, D. (2014). A review of network intrusion detection system based on KDD dataset. *International Journal of Engineering and Techno Science* (pp. 10-15).

[11] Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. *Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications* (pp. 53-58).

Tadashi Ogino received his doctor degree of electric engineering from University of Tokyo in 1988. His research interest is in distributed system, cloud computing and big data analysis.

He joined Mitsubishi Electric Corp. in 1988 and developed mid-range business servers.

He is now working at Okinawa National College of Technology as a professor. Dr. Ogino is a member of ACM, IEEE, IPSJ and IEICE.