Implementation of Multiple Classifier System on MapReduce Framework for Intrusion Detection

Masataka Mizukoshi  
Graduate School of Information Science and Technology  
Hokkaido University Sapporo  
Sapporo 060-0811, Japan  
Email: m.mizukoshi@ist.hokudai.ac.jp

Bando Shintaro  
Graduate School of Information Science and Technology  
Hokkaido University Sapporo  
Sapporo 060-0811, Japan  
Email: dontforget3-7@ec.hokudai.ac.jp

Martin Schlueter  
Information Initiative Center  
Hokkaido University Sapporo  
Sapporo 060-0811, Japan  
Email: schlueter@midaco-solver.com

Masaharu Munetomo  
Information Initiative Center  
Hokkaido University Sapporo  
Sapporo 060-0811, Japan  
Email: munetomo@iic.hokudai.ac.jp

Abstract—Since the data volume from various facilities keeps growing rapidly in recent years, “big data” processing frameworks such as Hadoop have been developed as a scalable architecture to process large amount of data in cloud computing environment. We focus on intrusion detection problems which require large amount of data to be processed in order to detect malicious attacks. In this paper we discuss a Hadoop implementation of a multiple classifier system to enhance performances of the learning process in intrusion detection.

I. INTRODUCTION

Currently, the amount of data that has to be processed by many companies has reached Petabytes and is expected to keep growing in the future. Under these circumstances, it is critical that these large amounts of data be efficiently treated. This requires that useful information be gleaned from such data. To do this, various Machine Learning approaches are being proposed. This is also true in the field of Intrusion Detection Systems (IDS). Many machine learning approaches have been proposed in the field of IDS. These approaches include Support Vector Machines (SVM), Neural Networks (NN), and classifier systems (see [1]-[5]). IDSs need to to analyze large-scale data quickly and respond promptly to any new intrusion detected. However, if machine learning is applied to large scale data, time becomes a critical issue. Here, we suggest an early learning technique by executing multiple classifier systems simultaneously (parallel) on Hadoop MapReduce framework (see [8],[9]). Classifier systems are valid for a variety of input environments. A multiple classifier system extends this concept to facilitate the processing of big data. Recently, classifier systems have been studied extensively as a powerful way to learn large-scale data. In this paper, a proposed implementation of such a multiple classifier system on the Hadoop MapReduce Framework is presented. Hadoop is an open source software for large-scale distributed data processing that enables users to easily perform distributed processing (see [10]).

II. CLASSIFIER SYSTEMS

Classifier systems are the learning process of rule-based machine learning system. They are composed of a set of rules, called a classifier, which consists of IF/THEN statements. The method by which a classifier system learns is described below (Fig. 1).

A. Flow of learning in a Classifier System

Classifier systems can roughly be divided into five modules: i) action module, ii) renumeration assignment module, iii) discovery module, iv) effector module, and v) detector module. A classifier system first converts the input from the environment to adapt it for use by the system. In the action module, the system determines the classifier that should act on the input data. In the renumeration assignment module, the classifiers chosen in the action module return some rewards for the classifiers. In the discovery module, a Genetic Algorithm (GA) is executed in order to receive a better population of classifiers.

In the action module, the system selects classifiers that match the input from the environment and move them into a message list. Classifiers have their own level of intensity. Classifiers that have a high intensity perform in the message list. When selecting an action classifier, we consider not only intensity but also the number of hashes (يقة). The method by which classifiers are chosen depends on the nature of the problem to be dealt with.

In the renumeration assignment module, a reward is assigned to the classifier population. The reward allocation method heavily influences learning efficiency. Let us now look at a typical reward allocation algorithm, called the bucket brigade algorithm.

If the reward is only assigned to the classifier that acted, a large deviation appears when learning with many test cases. It is therefore impossible to perform learning by considering the
turn in which classifiers acted. The bucket brigade algorithm was proposed as a way to overcome this problem. With the bucket brigade algorithm, the classifiers perform an auction market using the intensity of the classifiers and acquire their reward in this way. First, those classifiers that match the input from the environment are put in a message list. In this message list, the classifiers that act are chosen and classifiers pay intensity for the classifier leading to a matching. Only those classifiers that acted are kept in the message list, classifiers that did not act are excluded from the list. Therefore, those classifiers that remain in the list cause the next match. Thus, by causing matching continuously, renumeration can also be assigned to the classifier relevant to the classifier outputted to the environment.

In the discovery module, a GA is executed on a classifier population. GAs are approximative optimization algorithms inspired by the process of biological evolution. In GAs, the mechanisms of selection, intersection, and mutation are represented by operations on bit strings. In a classifier system, a classifier is considered to be a gene sequence (represented as a bit string) and a GA is used to manipulate this gene sequence. By performing the GA, a classifier population’s flexibility is maintained and refined.

III. FLOW OF LEARNING IN A MULTIPLE CLASSIFIER SYSTEM

In a multiple classifier system (MCS), input data is distributed to each machine and a classifier system is executed by each machine. One classifier population is made by the classifier from which it is shifted to execution by each machine and is summarized in one place. Although learning big data is attained in a short time using this method, the accuracy of the learning depends on the composition of the population collected in one place. Thus, the method that is used to conduct the merge is very important. The efficiency of learning changes significantly according to which classifier is entered into a population and which classifier is removed.

A. Merge technique

In the merge technique, classifiers, which are sent sequentially, are sorted by the system according to intensity and number of hashes (hashes) in order to make a high-quality population. The decision as to whether the system leaves a classifier that has a high correspondence power or a classifier that has high intensity is dependent on the learning objective or contents. The performance of the population changes in accordance with the kind of population constituted. There is no absolute way that is good to constitute a population from any kind of technique since many methods have been proposed. Figure 2 illustrates the system-wide flow in a typical MCS (Fig. 2).

B. Bottlenecks in a multiple classifier system

It is required that an optimal classifier group be generated by a suitable merge method in an MCS, as mentioned above. Moreover, in order to obtain a better classifier and for group flexibility, it is necessary to generate to some extent a large group. In order that a problem does not arise here, a search may be conducted by the merge technique for a group that contains a new classifier. At this point the merge may take time, so that the collective size becomes large.

IV. INTRUSION DETECTION SYSTEM (IDS)

Intrusion detection is an adversarial classification task. In recent years, it has become necessary to use big data for learning of intrusion detection. To date, many types of machine learning techniques have been proposed for IDS. Nowadays, MCS are used for learning of intrusion detection,
which leaves a track record. Intrusion detection has two types of detection methods: anomaly detection and misuse detection. Misuse detection is based on a comparison of the current connection pattern with known attack patterns. In contrast to misuse detection, anomaly detection learns from normal patterns of connection. The two types of detection methods also have different respective features. Misuse detection has a low false positive rate and a high detection rate for known attack patterns. However, against new attack patterns it receives it is weak. Anomaly detection can detect new attacks, but it has a high false negative rate. In intrusion detection by machine learning, the anomaly method is used in many cases. The purpose of an IDS is to suppress the false positive rate and keep the false negative rate low. In our proposed system, the two types of detection methods, anomaly detection and misuse detection, are utilized by dividing the gathering as a result of the classifier system.

V. Proposed implementation

A. Hadoop

Hadoop is an open source large-scale distributed processing software that uses the MapReduce Framework. Hadoop performs Terabyte to Petabyte levels of big data processing that ordinarily takes a lot of time and is not dependent on an expensive computer as the work is distributed over thousands of nodes comprising ordinary servers that can be obtained cheaply. By using the MapReduce Framework, underlying details and troublesome problems, such as data distribution, which must be solved in the usual distributed processing, failure processing, and load balancing are taken care of. Thus, parallel distributed processing can be performed easily. Hadoop comprises MapReduce and HDFS: Data distribution, node management, etc. are enabled by the Hadoop distributed file system (HDFS). Hadoop simplifies the construction of distributed processing programs. The typical layout of the Hadoop MapReduce Framework is depicted in Fig. 3.

B. MapReduce Framework

A MapReduce Framework is divided into three phases: Map, Shuffle, and Reduce. By passing data to each phase using the value from the key/value pair, processing is performed by each phase. Users can carry out distributed processing by describing the processing in the Map and Reduce functions. In Map processing, the input data are received as key/value pairs and processing by analyzing the contents generates middle data in the form of key/value pairs. The generated data is gathered by key in Shuffle processing, and is sent to Reduce processing. In Reduce processing, the value processed for every key is generated and output as a result.

C. Mounting of the multiple classifier system on Hadoop

In Hadoop, classifier systems are individually mounted in Map processing and merge is performed in Reduce processing. The contents of mounting within each phase of processing is explained below.

VI. Experiment

A. Object problem

In our experiment, a classifier system was built to study data for intrusion detection using the network communication

![Fig. 3. Typical layout of the Hadoop MapReduce Framework](image-url)
data of KDD Cup 1999 Data (international data-mining contest). The contents of the data are the study data used by a teacher which is accompanied by data for result judging. The environment was set up by binary coding this data to 0 and 1.

B. Execution environment

One hundred and thirty virtual machines (VM) were launched on a CloudStack cloud platform and Hadoop version 1.0.4 mounted on them.

1) Key setup: An execution classifier was tagged according to the kind of input data it processes. Keys were setup according to attacks from 22 kinds of intrusion attack data. The data were divided according to intrusion data and groups made.

2) The merge method performed: In the classifier system using Hadoop, the fall of the merge speed by hypertrophy of a global model is avoidable by increasing the number of Reducers. Therefore, even if the group becomes very large, computation time is seldom influenced. The maximum global size was set at 500. The groups were setup such that an individual with a higher specialty nature (individuals with a small number of hashes, '#'s) might remain since it is highly possible to setup a maximum. In this experiment, classifiers were sorted using conditions 1 and 2:
   1) Keep those classifiers with a low number of hashes, #'s, in the system.
   2) If two classifiers have an identical number of hashes, #'s, keep the one with the higher intensity in the system.

VII. EXPERIMENTAL RESULTS

The system was mounted and two or more Mappers started learning in the multiple classifier system. Learning with a simple classifier system was compared with learning distributed with the MCS using the method outlined in Section IV.

The number of nodes used by the Mapper was varied from 3 to 93 and execution time measured. The resulting execution time is depicted in Fig. 4. Block count was also changed in increments of 8 MB, 16 MB, 32 MB,... and so on in Hadoop, when distributing a file. Like the upper graph, it turned out that execution time decreased linearly. As shown in Fig. 5, execution time decreased as the number of Reducers increased. However, the change is not linear because the execution time resulting from the change in the number of Reducers is influenced by the items of learning data.

Next, we examined the change in the accuracy of learning in terms of the change in the number of nodes. Here, the accuracy is determined by the probability of the classifier group generated to learn and judge a result correctly according to the test data.

Figure 6 illustrates learning accuracy with respect to the number of Mappers. If the number of Mappers is increased by too much, a state will be reached where there is hardly any learning. Also, if a file is divided too much, learning by each machine will become insufficient, and as a result, the overall learning rate will fall. In this experiment, the size of the learning data used was 740 MB. The accuracy of the learning became low for sizes smaller than 64 MB. Thus, it

![Fig. 4. Execution time versus number of Mappers](image)

![Fig. 5. Execution time versus number of Reducers](image)

![Fig. 6. Accuracy of learning versus number of Mappers.](image)
was necessary to search the MCS on Hadoop to determine the optimal file assignment size.
Next, learning with a simple classifier system was compared with learning using our proposed MCS technique.
The column on the right of the graph shows learning with

![Graph showing accuracy of learning in simple and multiple classifier systems](image)

Fig. 7. Accuracy of learning in simple and multiple classifier systems

the simple classifier system, while that on the left represents learning by our proposed MCS. As can be seen in the graph, the accuracy of the overall learning fell in the MCS. Since a classifier system is what is originally arisen and learned in one classifier group, when it merges, as the whole group’s work, it is inferior to the simple classifier system.
The pie chart depicted in Fig. 8 illustrates the items of learning data used in the experiment. In the learning data used (740 MB), attack intrusion data of a very rare kind comprising only about 20 KB (less than 1 % of the whole), such as guess passed and nmap, was included. Next, we examined how the learning rate varied with the small amount of attack data (Fig. 9). It was difficult for the above data to update the learning intensity of the simple classifier system. Because there was the possibility of obtaining learning by other inputs, leaking and being removed from a group was high, so there was hardly any learning. However, in our proposed technique, since the classifier was gathered by distributing classifiers that received and reacted to each small input in learning to each Reducer,

![Pie chart showing learning data items](image)

Fig. 8. Learning data items, displayed according to their proportional likelihood to be attacked.

![Graph showing learning rate variation with small amount of attack data](image)

Fig. 9. Left: Learning of multiple classifier system. Right: Learning of simple classifier system

it was possible to keep the learning rate high.

VIII. CONCLUSION

Learning by a multiple classifier system, compared with learning with a simple classifier system, can significantly reduce the learning time and the effectiveness of the learning method on large-scale problem data sets. Moreover, a time reduction in the merge process was gained when executing the proposed multiplier classifier system on the Hadoop Map Reduce framework. However, it turns out that some accuracy of learning is lost when using a MCS. In order to raise the accuracy of learning, we have to consider a better merging method and to further study the method of classifiers. When learning data was distributed, It turned out, that the optimal distribution of data is problematic. Learning by each machine is inefficient, if the processed data is too small. Also, one has to be careful with splitting data where the distribution of important of data is unknown. To consider that Hadoop divides a data file automatically, it seems that this issue becomes a big subject to the proposed technique.

By the proposed technique, it was possible to learn without leaking small input data. Even in large scale data, the proposed technique was able to learn in a reasonable fast time without leaking small input data.

Recently, an improvement of the MCS technique, such as sharing classifiers and other parameters between individual machines, has been proposed (see [2]). Such improvements seems also to be promising for our proposed technique and should be investigated further in the future.

ACKNOWLEDGEMENT

This work is supported by KAKENHI (No.22500196), Japan Society for the Promotion of Science.

REFERENCES


[3] Battista Biggio, Goirgio Fumera, Fabio Roli. Multiple Classifier System for Adversarial Classification Tasks. Dept. of Electrical and Electronic Eng, Univ. of Cagliari Piazza d’Armi, 09123 Cagliari, Italy.


[8] Henrique Santos, Manuel Filipe Santos and Wesley Mathew, University of Minho, Portugal, Supervised Learning Classifier System For Grid Data Mining. 2009


