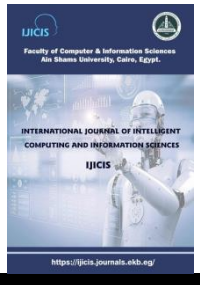
IJICIS, Vol.23, No.1, 106-114
DOI: 10.21608/ijicis.2023.167424.1223

International Journal of Intelligent Computing and Information Sciences

https://ijicis.journals.ekb.eg/



A RULE LEARNING APPROACH FOR BUILDING AN EXPERT SYSTEM TO DETECT NETWORK INTRUSIONS

Omar Galal

Computer Engineering Department,
Faculty of Engineering,
Cairo University,
Giza, Egypt
omar.galal@eng.cu.edu.eg

Ahmed Nasr

Computer Engineering Department,
Faculty of Engineering,
Cairo University,
Giza, Egypt
ahmed.nasr9677@gmail.com

Lydia Wahid Rizkallah

Computer Engineering Department,
Faculty of Engineering,
Cairo University,
Giza, Egypt
lydiawahid@cu.edu.eg

Received 2022-10-08; Revised 2023-03-07; Accepted 2023-03-08

Abstract: *Network intrusion detection is the problem of detecting suspicious requests through networks. In recent years, many researchers focus on addressing this problem in the context of machine learning. Although machine learning algorithms are powerful, most of them lack the power of interpretability. Expert systems, on the other hand, are knowledge-based systems designed to simulate the problem-solving behavior of human experts. Expert systems possess the advantage of interpretability through an explanation mechanism that justifies their line of reasoning, however, they need the availability of a domain expert. This paper proposes the use of rule learning approaches to gain the best of both fields, being interpretable as the expert system and learnable through collected datasets without the need for explicit expertise. A separate and conquer rule learning approach is proposed for network intrusion detection. Our results show that the separate and conquer approach achieves a 0.99 weighted average F1-score on the test set which makes it very comparative to both decision trees and classical machine learning approaches. We also show that rules produced using separate and conquer are much simpler than decision trees and more interpretable.*

Keywords: *Intrusion Detection, Expert Systems, Rule Learning, Separate and Conquer, Divide and Conquer*

1. Introduction

The Internet is currently one of the life basics for a very large number of people worldwide. People all around the globe use the internet by using services provided by web servers. While web servers and networking grow, new types of attacks occur that aim to make the server down or unavailable to users. A great area of research in ethical hacking is dedicated to building systems that are able to secure web servers from suspicious requests and all known types of attacks .

Sometimes it is required to build lightweight systems that are able to detect intrusions from metadata of incoming requests to servers, not from its actual content. Network intrusion detection has been addressed by many classical machine learning (ML) approaches. Although machine learning approaches are attractive due to their ability to learn hidden and complex functions that are very complex to be learned by humans, interpretability is a major concern for them. When it comes to network intrusion detection, interpretability is important to understand why some request is considered suspicious by the machine learning model. Of course, machine learning algorithms are not the same in terms of interpretability, and a lot of research is also done to determine the interpretability of each algorithm**.[1]**

On the other hand, expert systems are known for their interpretability [2] since they can provide explanations mechanisms and provide their own line of reasoning. To build an expert system, a human expert in the network intrusion detection domain is required to transfer his knowledge and translate it into programmable rules to build the expert system. Sometimes it is hard to get an expert in a specific domain that is able to do so. Here comes the area of intersection between expert systems and machine learning **.**

In this paper, we address the network intrusion detection problem by rule learning approaches that produce interpretable expert systems without the need to get any human expert. We also compare rule learning expert systems approaches with machine learning baselines that we trained ourselves. The rest of the paper is organized as follows: Section 2 includes background about algorithms used in our work. Section 3 includes the approaches that are used in literature that tackle the network intrusion detection problem. The proposed approach is explained in Section 4. Section 5 includes the results and analysis. Finally, the conclusion is found in Section 6.

2. Background

Expert systems are computer programs that are designed to have similar problem-solving behavior to human experts. Expert systems have been applied successfully to solve many problems such as academic advising [3], COVID-19 diagnosis [4], diagnosing and treating anxiety [5], and many more.

There are many types of expert systems. A very common type is the rule-based expert system. A rule-based expert system contains a set of rules that are used for inference. Each rule consists of conditions and conclusions. If conditions are satisfied, the rule is said to fire achieving the conclusions that are added in the form of facts. Constructing the rules is dependent on the application we are working on. In the era of machine learning and big data, a new direction arises to use the data collected in a field to learn some rules that are interpretable enough to seem as if a human expert is the producer of these rules. This approach is referred to as rule learning. Mainly there are two main approaches to rule learning, divide and conquer and separate and conquer.

The divide and conquer approach is widely used to generate decision trees. It is also known as Top-Down Induction of Decision Trees (TDIDT) as its main purpose is to generate rules in a tree form. The produced decision tree can then be turned into a rule set by taking every path from the root to the leaves where each path will form a single rule. Decision trees have two major problems. First, they are complex and hard to understand and that affects their interpretability feature. Also, an ordered rule set with at most n conditions is more interpretable and expressive than a decision tree with a depth of n. Second, Decision trees suffer from the replicated sub-tree problem where the decision tree is often forced to learn identical subtrees which also affects its interpretability.

To overcome the problems of decision trees, the separate and conquer approach became very popular. The aim of this approach is to directly generate the if-then rules from the training set. It is also known as the covering approach as it learns one rule that covers some of the training examples then the next rule is learned from the remaining training examples. Examples of this approach include Prism and Ripper.

For the sake of comparison, several machine learning algorithms are used. Logistic regression has been employed as an example. It is widely used for estimating conditional probabilities for binary classes and multi-classes. Another algorithm that is used is the Support Vector Machine (SVM), which is a widely used margin classifier, designed to maximize the margin between the decision plane and a set of data points called the support vectors. Another commonly used ML algorithm is Naive Bayes, which is a simple probabilistic model, based on Bayes Rule for estimating probabilistic posteriors, the model assumes the independence between data. Neural Networks (Multi-Layer Perceptrons) are another example of commonly used ML algorithms. Neural networks are powerful ML models, and they can perform different tasks, including classification. One of the major disadvantages of Neural Networks is their lack of interpretability.

3. Literature Review

The problem of network intrusion detection has been addressed from different perspectives. Some studies addressed the problem from a machine learning perspective such as using deep belief networks (DBN) [6]. In [7] authors propose a new deep reinforcement learning algorithm to tackle the problem. In [8] authors propose a new ensemble system by combining multiple AdaBoost classifiers. In [9] authors use Neural Networks (NN) with feature selection. The authors arrived at the conclusion that NN outperforms Support Vector Machines (SVM) when dealing with network intrusion detection.

Some studies addressed the problem by combining rule-based expert systems with clustering techniques [10]. The authors build the rule base by hand coding the rules and they use it to identify the known attack patterns whereas the clustering techniques are used to identify new attacks which are then added to the rule base. Some studies designed fuzzy rule-based expert systems for intrusion detection and cybersecurity [11]. The authors build the expert system by interviewing security experts and gaining the necessary knowledge from them. Some studies tackled the problem using genetic algorithms [12].

Other rule learning approaches use inductive logic programming (ILP) to detect intrusions by learning sophisticated relational theories by using logic programs as expressive representations for instances, background knowledge (BK), and hypotheses. Given positive and negative examples, an ILP system creates a hypothesis to explain examples in terms of BK. Association rule learning is a data mining technique also used to detect network intrusions. It mines examples to discover interesting patterns among the different features and represents them as rules. It uses breadth-first search to find the frequent itemsets. Rough set theory is another approach that can deal with uncertain knowledge, it is a formal approximation of a crisp set resulting in an upper and a lower approximation sets. Rules are derived from these two sets. Lower approximation rules are unquestionably valid, whereas upper approximation rules are not necessarily true [13].

Other techniques that handle intrusion detection are based on anomaly detection methods. Some anomaly detection methods use statistical techniques to model normal behavior and any events that violate such models are considered suspicious. There are several statistical anomaly detection techniques such as threshold measures, correlation, mean and standard deviation. Misuse detection techniques are another set of techniques used to detect intrusions. One approach is state transition analysis which views an attack as a series of attacker-performed actions that transform a system's starting state into a compromised state [14].

4. Proposed Approach

4.1. Dataset

The majority of the researchers who studied performance testing lack important issues. They did not care about providing effective testing approaches to automate the whole performance testing process. Tables 2, and 3 show a summary of the previously mentioned studies from the literature. Table 2 presents the comparison between performance testing approaches for mobile applications testing. Table 3 presents the comparison between performance testing approaches for web-based application testing. Table 4 presents the comparison between performance testing approaches for web-based application testing. The comparison in both tables 2, 3, and 4 is based on the following: (i) the proposed automatic test cases generation approach; (ii) the proposed automatic test execution approach; (iii) support the parallel test execution; (iv) the proposed automatic test result from analysis approach. Additionally, table 5 shows the comparison between the advantages and drawbacks of the top performance testing tools in the market.

4.2. Data Exploration and Preprocessing

We started with Exploratory Data Analysis (EDA) phase. The dataset contains nearly 50k normal examples, 100k suspicious ones, and 20k unknown. The dataset has a “flags” column which is a string of flags that are necessary for networking. If some flag is not used it is replaced by a ‘.’. This column is replaced by five other columns – a column for each flag – so if a flag is present in the string, its value is set in its corresponding column. Some other columns such as “attack description”, “Flows”, “Tos”, and “attack type” have a single value so they are dropped since they are of no value. Some columns are also dropped being unnecessary for classification such as the attack ID. The “Bytes” column is an object containing characters, so it is converted to its numerical value. All categorical features such as class, source IP, destination IP, and proto were encoded using label encoding. We split the dataset into 70 percent training and 30 percent testing. All preprocessing is done in Python using Scikit-Learn.

4.3. Rule Learning Approaches

Our main approach for intrusion detection rules learning is via the separate and conquer approach detailed in [16]. It is a covering algorithm where rules are iteratively added to the rules set that is initially an empty set. Each rule in the rules set has to cover some examples from the dataset to be included. The algorithm proceeds by adding rules until all the dataset has been covered by the rules in the rules set, hence the name separate and conquer, since the examples in the dataset covered by already existing rules are isolated (i.e. separated) and the algorithm conquers the remaining uncovered examples by finding new rules to cover

them until all examples are covered by the rules set. Each example in the dataset is guaranteed to be covered by at least one rule. Each induced rule consists of two parts: premise (IF part) and conclusion (THEN part) and it has the following form:

IF condition1
AND condition2
…..
AND conditionN
THEN class is c

Each condition has the form Aj =aj where Aj is attribute j and aj is the value given to attribute j which is part of this attribute's domain. The equality operator is used with nominal attributes, however, for numerical attributes operators such as ≥ and ≤ are used.
A rule is said to cover an example in the dataset if all conditions are satisfied i.e., the value given to the attributes in the rule is the same (in the case of nominal attributes) as the values of the corresponding attributes in the example. The consequent of the rule gives the classification (the class) of the examples covered by that rule. We will now discuss how each rule is learned. A rule is learned by adding conditions to an initially empty rule premise. This addition is done by searching exhaustively for the condition whose addition will produce a high-quality rule (based on a certain quality measure). When many conditions lead to the same rule quality, then the condition that makes the rule cover more examples is selected. This procedure can lead to generating too specific rules, therefore, if the addition of a condition causes the rule to cover examples less than a predefined threshold, then the such condition will be discarded. The procedure stops when no condition can be added that satisfies the such threshold.

We also trained a decision tree on our training set using Scikit-Learn. To achieve a fair comparison between rule learning via separate and conquer and divide and conquer approaches, we limited the depth of the decision tree to seven. This is done because the separate-and-conquer approach produced rules with a maximum of seven conditions per rule as it will be shown in the results section. All training is done on Google Colab.

4.4. Classical Machine Learning Approaches

All the ML algorithms are trained on the data using Scikit-learn. The training is done on Google Colab. For the logistic regression, to adapt it to be used for multi-class, we used the "one Vs all" schema, which converts the multi-class classification problem to multiple binary class problems, where in each problem we have one class vs. the rest.

For the SVM algorithm, we use the linear kernel (no special/nonlinear kernel used) with regularization coefficient C equal to 1, and with one iteration for the solver. Finally, for the Multi-Layer Perceptrons, we use the implementation of Scikit-learn, and we use the following parameters: one hidden layer network with 100 hidden units, a learning rate of 0.001, L2 regularization coefficient equals 0.0001, a momentum of 0.9. Moreover, the network uses ReLu activation function and stochastic gradient descent optimizer.

5. Results and Analysis

We evaluated and compared the aforementioned algorithms after training on the dataset. We used the weighted average F1-score for each algorithm as our evaluation metric. The F1 score can be calculated as follows:

$$\boldsymbol{F1\ Score\ =\ 2\ \times\ \frac{Precision\ \times Recall}{Precision\ +\ Recall}}$$

We used the weighted average F1-score as our evaluation metric as it takes into account the weight of each class in the dataset. Figure 1 shows the weighted average F1-score for all tried models. We notice that the performance of the different algorithms varies on the test set, for example, ML models such as Logistic regression, Linear SVM, and Naive Bayes score 60%, 87%, 88% respectively. Neural Networks approach scores a remarkably high score of 99% but it lacks results interpretability. Other rule-based models such as Decision Trees and Separate & Conquer scored remarkably high weighted average F1-score of 99.9% and 99%, respectively.

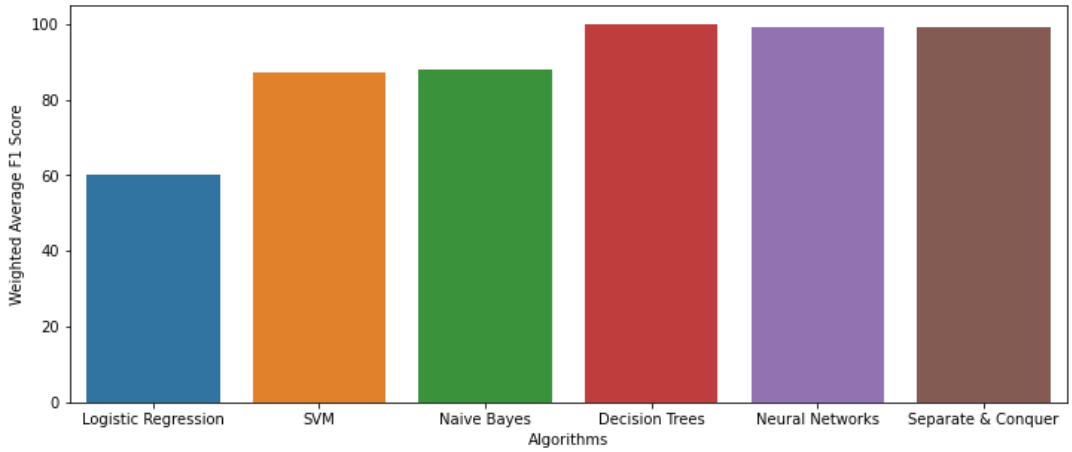


Figure. 1: Weighted average F1-score of the considered algorithms

We have also used other metrics for the sake of comparison. Table 1 includes the metrics used in our evaluation for each of the aforementioned algorithms. We used macro and weighted averages as illustrated in Table 1. Macro average just takes the average value of a given metric across the different class labels, while weighted average takes in consideration the class imbalances.

Table 1. Comparisons of the used algorithms with respect to evaluation metrics

	Macro average			**Weighted average**			**Accuracy**
	Precision	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**	
Separate and conquer Rule Learning	0.96	**0.99**	0.97	**0.99**	0.98	**0.99**	0.98
Decision Tree	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**
Logistic regression	0.49	0.43	0.44	0.59	0.62	0.6	0.62
Naive Bayes	0.81	0.69	0.69	0.88	0.91	0.88	0.91
SVM	0.96	0.68	0.67	0.92	0.91	0.87	0.91

Multi-Layer Perceptron	**0.99**	0.98	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**

As seen in Table 1 multi-layer perceptron, decision trees (divide and conquer), and separate and conquer approaches have achieved the best results in the different metrics. However, multi-layer perceptron lacks the interpretability characteristic and explanation mechanism which makes it less than an optimal solution when the explanation of the results produced is a major aspect of the application at hand; it fails to explain its own line of reasoning. On the other hand, decision trees and separate-and-conquer approaches possess interpretability characteristics and both achieved comparable results.

To compare the rules produced by the separate-and-conquer approach with the decision tree we will next investigate their results. The separate and conquer approach produced 17 rules. Each rule is in the form of the If-Then rule. Each rule has some conditions ANDed together learned from the dataset. Eleven rules out of 17 have only three conditions. Five rules have seven conditions. One rule has four conditions. The divide and conquer approach produced a decision tree with a max depth of seven for a fair comparison with the separate and conquer approach. The decision tree contains 45 rules or paths from the root to each leaf. Table 2 shows the details of the number of induced rules for each approach and the number of conditions per rule. It can be easily seen that the separate-and-conquer approach is much simpler than the divide-and-conquer one regarding both the number of total rules and the number of conditions per rule. This supports the argument that decision trees become more complex and which decreases their interpretability.

Table 2. Number of rules and conditions produced from rule learning approaches

Separate and Conquer		**Divide and Conquer**	
Num. of Rules	Num. of conditions per rule	Num. of Rules	Num. of conditions per rule
11	3	24	7
5	7	6	6
1	4	9	5
-	-	5	4
-	-	1	3
Total Rules = 17		Total Rules = 45	

To address the interpretability of rules produced from the separate and conquer approach, we will take two examples from the 17 output rules to understand them. The first rule we will consider is as follows:

IF Bytes = (-inf, 11636.50) AND Src IP Addr = (-inf, 10537.50) AND Dst IP Addr = (-inf, 10476.50) THEN class = { Suspicious }

The rule says that if the size of the request is below some threshold, the source IP address is within some range, and the destination IP address is within some range then the request is suspicious. This can be translated from the dataset requests that come from some regions to another one with small packet sizes, these requests were suspicious.

We will next consider another rule which is as follows:

IF Packets = <4.50, inf) AND Src IP Addr = <10191.50, inf) AND Dst IP Addr = <10135, inf) THEN class = { Normal }

The above rule says that if the number of sent packets is above some threshold, the source IP address is within some range, and the destination IP address is within some range then the request is normal. This suggests that given the information in the dataset, the requests sent from some region to another with a number of packets exceeding some limit then these requests are normal. From these two examples, we can conclude that it is difficult for a human expert to see these relations given huge datasets. We can also conclude that learned rules are very interpretable and have an understanding nature that can be used further by domain-specific experts.

6. Conclusion

In this paper, we addressed the network intrusion detection problem from a rule-based expert systems perspective. We used separate-and-conquer rule learning as our main approach to learn the rules that construct the knowledge base of our expert system. We also used the divide and conquer approach to build a decision tree that can also be transformed into a set of rules. We showed the importance of interpretability which makes the choice of rule learning an appealing one. We compared both approaches with some classical machine learning baselines. Our results showed that rule learning achieves comparative results with machine learning approaches. We also compared the separate and conquer approach with the divide and conquer approach in terms of interpretability and showed that separate and conquer is much simpler and more interpretable in terms of the rules produced.

References

1. Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., & Kagal, L. (2018, October). Explaining explanations: An overview of interpretability of machine learning. In 2018 IEEE 5th International Conference on data science and advanced analytics (DSAA) (pp. 80-89). IEEE.
2. Cao, Y., Zhou, Z., Hu, C., He, W., & Tang, S. (2020). On the Interpretability of Belief Rule-Based Expert Systems. IEEE Transactions on Fuzzy Systems, 29(11), 3489-3503
3. El-Sayed, R., Seddik, S., Rizkallah, L.W. (2022). Expert Systems in Academic Advising. In: Hassanien, A.E., Snášel, V., Chang, KC., Darwish, A., Gaber, T. (eds) Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2021. AISI 2021. Lecture Notes on Data Engineering and Communications Technologies, vol 100. Springer, Cham. https://doi.org/10.1007/978-3-030-89701-7_18.
4. Fawzi, R., Ghazy, M., Rizkallah, L.W. (2022). Designing Knowledge-Based Systems for COVID-19 Diagnosis. In: , et al. Hybrid Intelligent Systems. HIS 2021. Lecture Notes in Networks and Systems, vol 420. Springer, Cham. https://doi.org/10.1007/978-3-030-96305-7_7
5. Hsu, C. C., & Lin, C. C. (2020, December). Framework and conceptual design of rule base for building SWI-Prolog-based expert systems to diagnose and treat anxiety. In 2020 International Conference on Pervasive Artificial Intelligence (ICPAI) (pp. 54-57). IEEE.
6. Sohn, I. (2021). Deep belief network based intrusion detection techniques: A survey. Expert Systems with Applications, 167, 114170.

7. Lopez-Martin, M., Carro, B., & Sanchez-Esguevillas, A. (2020). Application of deep reinforcement learning to intrusion detection for supervised problems. Expert Systems with Applications, 141, 112963.
8. Zhou, Y., Mazzuchi, T. A., & Sarkani, S. (2020). M-AdaBoost-A based ensemble system for network intrusion detection. Expert Systems with Applications, 162, 113864.
9. Taher, K. A., Jisan, B. M. Y., & Rahman, M. M. (2019, January). Network intrusion detection using supervised machine learning technique with feature selection. In 2019 International conference on robotics, electrical and signal processing techniques (ICREST) (pp. 643-646). IEEE.
10. Aneetha, A. S., Indhu, T. S., & Bose, S. (2012, October). Hybrid network intrusion detection system using expert rule based approach. In Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology (pp. 47-51).
11. Goztepe, K. (2012). Designing fuzzy rule based expert system for cyber security. International Journal of Information Security Science, 1(1), 13-19.
12. Hoque, M. S., Mukit, M., Bikas, M., & Naser, A. (2012). An implementation of intrusion detection system using genetic algorithm. arXiv preprint arXiv:1204.1336.
13. Liu, Q., Hagenmeyer, V., & Keller, H. B. (2021). A review of rule learning-based intrusion detection systems and their prospects in smart grids. IEEE Access, 9, 57542-57564.
14. Verwoerd, T., & Hunt, R. (2002). Intrusion detection techniques and approaches. Computer communications, 25(15), 1356-1365.
15. Ring, M., Wunderlich, S., Gruedl, D., Landes, D., Hotho, A. (2017). Flow-based benchmark data sets for intrusion detection. In: Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS), pp. 361-369. ACPI
16. Gudyś, A., Sikora, M., & Wróbel, Ł. (2020). RuleKit: A comprehensive suite for rule-based learning. Knowledge-Based Systems, 194, 105480.