Are Evolutionary Rule Learning Algorithms Appropriate for Malware Detection?

M. Zubair Shafiq, S. Momina Tabish, Muddassar Farooq
Next Generation Intelligent Networks Research Center (nexGIN RC)
National University of Computer & Emerging Sciences (FAST-NUCES)
Islamabad, 44000, Pakistan
{zubair.shafiq,momina.tabish,muddassar.farooq}@nexginrc.org

ABSTRACT

In this paper, we evaluate the performance of ten well-known evolutionary and non-evolutionary *rule learning algorithms*. The comparative study is performed on a real-world classification problem of detecting malicious executables. The executable dataset, used in this study, consists of a total of 189 attributes which are statically extracted from the executables of Microsoft Windows operating system. In our study, we evaluate the performance of rule learning algorithms with respect to four metrics: (1) classification accuracy, (2) the number of rules in the developed rule set, (3) the comprehensibility of the generated rules, and (4) the processing overhead of the rule learning process. The results of our study highlight important shortcomings in evolutionary rule learning classifiers that render them infeasible for deployment in a real-world malware detection system.

Categories and Subject Descriptors

D.4.6 [**Software**]: Security and Protection—*Invasive software*

General Terms

Algorithms, Experimentation, Security

Keywords

Genetics Based Machine Learning, Malware Detection

1. INTRODUCTION

The genetic based machine learning systems, more commonly known as GBML, have become significantly more sophisticated compared with primitive versions of the learning classifier systems (LCS) proposed by John Holland in 1970s. A number of LCS variants exist that include evolutionary rule learners, evolutionary neural networks and other hybrid schemes. But evolutionary rule learners have been the most popular amongst all. They can be further subdivided into Michigan style, Pittsburgh style, anticipatory, and iterative rule learners [1]. Most of the recent work in LCSs is focused on classical Michigan- and Pittsburgh-style evolutionary rule learners that include but are not limited to XCS, UCS, GAssist and GALE.

In this paper, we evaluate several well-known evolutionary rule learning classification algorithms (XCS, UCS, GAssist-ADI, GAssist-Intervalar, SLAVE) using a real-world classification problem of malware detection. We also compare these evolutionary rule learning algorithms with five other well-knwon non-evolutionary rule learning algorithms: RIPPER, SLIPPER, PART, C4.5 rules, RIONA. The malware detection problem has several stringent constraints other than classification accuracy, such as comprehensibility of the developed solution (for malware forensic experts) and processing overheads (for realtime deployment). Therefore, the malware detection problem is relatively more challenging because accuracy and efficiency of classification must be optimized simultaneously.

The executable dataset, used in this study, consists of $11,786$ executable files for the Microsoft Windows operating system in Portable Executable (PE) format [3]. A total of 189 attributes are statically extracted from the executables. The executable dataset consists of two types of executables: *benign* and *malicious*. We have collected $1,447$ benign PE files from the local area network of our virology lab. We have obtained $10,339$ malicious executables from a publicly available malware database called 'VX Heavens Virus Collection' [4].

In order to undertake a comprehensive study, we use four performance metrics for comparison: (1) classification accuracy, (2) the number of rules, (3) comprehensibility of the rules, and (4) processing overhead. To factor out implementation related bias in our study, we use the implementations of rule learning algorithms provided in a unified framework called Knowledge Extraction based on Evolutionary Learning (KEEL) [2].

2. RESULTS & CONCLUSION

In this section, we provide the results of our experiments and relevant discussions. We have used the default parameters of all algorithms as provided in KEEL, unless stated otherwise. We compare the performance of classifiers with the help of four performance metrics:

2.1 Classification Accuracy

In a typical two-class problems, such as malicious executable detection, the classification decision of an algorithm may fall into one of the following four categories: (1) true positive (TP), (2) true negative (TN), (3) false positive (FP), and (4) false negative (FN). A suitable metric for quantifying classification accuracy of an algorithm, also used in this study, is *accuracy* which is defined as:

Table 1: Classification accuracy of all algorithms

Algorithm	Average Accuracy	Number of rules	Execution time (sec)
Evolutionary Rule Based Algorithms			
XCS	0.9550± 0.0057	10,000.0	109.0
UCS	0.9847± 0.0034	9,652.6	16.3
GAssist-ADI	**0.9929±0.0013**	2.6	1,177.2
GAssist-Intervalar	0.9873± 0.0024	4.2	4,060.9
SLAVE	0.9917±0.0013	3.3	775.1
Non-Evolutionary Rule Based Algorithms			
RIPPER	0.9961± 0.0033	6.0	8.0
SLIPPER	**0.9975± 0.0023**	57.0	29.4
PART	0.9956± 0.0028	6.3	51.8
C4.5 rules	0.9962± 0.0031	7.4	37.0
RIONA	0.9899± 0.0052	-	2680.5

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Table 1 shows the classification accuracy of all algorithms (both evolutionary and non-evolutionary). We represent the best evolutionary and non-evolutionary classifier with a bold value. The best evolutionary rule learning algorithm is GAssit-ADI, closely followed by SLAVE which also has the average accuracy of more than 0.99. GAssist-Intervalar and UCS have accuracies in the range of approximately 0.98. XCS provides the worst accuracy amongst all classification algorithms used in our study. SLIPPER outperforms rest of the non-evolutionary rule learning algorithms. It is closely followed by C4.5 rules, RIPPER and PART respectively. The relatively worst classifier among non-evolutionary algorithm is RIONA that provides an accuracy of approximately 0.99. It is evident from Table 1 that non-evolutionary rule learning algorithms clearly outperform their evolutionary counterparts in terms of accuracy.

2.2 Number of Rules

The number of rules developed by an algorithm are dependent on the various input parameters. Sometimes, the maximum possible number of rules is explicitly limited via an input parameter. Table 1 shows that the number of rules in XCS and UCS are limited by the *MaxPopSize* (=10,000) parameter as they try to overfit. In comparison, other evolutionary rule learning algorithms utilize significantly fewer rules for classifying a malware. Non-evolutionary rule learning algorithms also generate significantly less number of rules. However, SLIPPER appears to be the only exception, which generates more than fifty rules on the average.

2.3 Comprehensibility of Rules

The comprehensibility of generated rules is another important metric as it provides great help to the 'malware forensic experts' in understanding the behavior and characteristics of a given malware. The comprehensibility of rules is a subjective term so we quantify it by using coarse grain Low, Medium, and High categories.

Both XCS and UCS develop conjunctive IF-THEN rules containing specified intervals for *every* attribute. Moreover, the number of rules in the final rule set are very large (see Table 1). Due to these obvious shortcomings, we say that the comprehensibility of the rules generated by XCS and UCS is *Low.* GAssist-ADI mostly develops rules by using single-sided ranges and has *default* rules as well. The comprehensibility of rules generated by GAssist-ADI can be categorized as *High.* GAssist-Intervalar, similar to XCS and UCS, generates rules that contain specified intervals for *every* attribute. However, the final rule set just contains only a few rules. So the comprehensibility of the generated rules is categorized as *Medium.* SLAVE uses fixed intervals (such as L1,L2,L3,L4) to generate the fuzzy rules. It also assigns weights to the individual rules. Overall, the comprehensibility of the rules generated by SLAVE can be ranked as *Medium.* The rules generated by all non-evolutionary rule learning algorithms are not only simpler to comprehend but are also smaller in number. Therefore, the comprehensibility of the rules generated by RIPPER, SLIPPER, PART and C4.5 rules can be categorized as *High.*

2.4 Processing Overheads

The processing overheads are reported in Table 1 as the average time (in seconds) required for training and testing per fold of the 10-fold cross validation process. It is evident from Table 1 that the processing overheads of all evolutionary rule learning algorithms, except UCS, are significantly large compared with non-evolutionary classifiers. Malware detection systems are deployed in real operating systems; therefore, the processing overhead is a crucial metric. GAssist-ADI and GAssist-Intervalar have very large processing overheads. The processing overheads of XCS and SLAVE are also high. UCS is the only evolutionary rule learning algorithm which has a processing overhead comparable with non-evolutionary algorithms.

All non-evolutionary rule learning algorithms, except RIONA, have acceptable processing overheads. The processing overhead of RIONA is large because it uses a combination of instance based learning and rule induction. For every given test instance, it first selects the neighboring instances similar to the nearest neighbor algorithm. After selection, it develops rule sets through rule induction. The complexity incurred due to exhaustive search of the neighboring instances for every test instance is a major performance bottleneck.

2.5 Conclusions

In this paper, we have compared ten rule learning algorithms for a real-world problem of executable malware detection. The results of our study show that non-evolutionary rule learning algorithms clearly outperform evolutionary rule learning algorithms for every performance metric.

3. REFERENCES

[1] J.H. Holland, L.B. Booker, M. Colombetti, M. Dorigo, D.E. Goldberg, S. Forrest, R.L. Riolo, R.E. Smith, P.L. Lanzi, W. Stolzmann, S.W. Wilson, "What Is a Learning Classifier System?", Internatinoal Workshop on Learning Classifier Systems (IWLCS), Volume 1813 of Lecture Notes in Artificial Intelligence, pp. 3-32, Springer, 2000.

[2] J. Alcala-Fdez, L. Sanchez, S. Garcia, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernandez, F. Herrera, "KEEL: a software tool to assess evolutionary algorithms for data mining problems", Soft Computing, Volume 13, pp. 307-318, Springer, 2009.

[3] Microsoft Portable Executable and Common Object File Format Specification, Windows Hardware Developer Central, Updated March 2008.

[4] VX Heavens Virus Collection, VX Heavens website, available at `http://vx.netlux.org`.