Combining KNN and Decision Tree Algorithms to Improve Intrusion Detection System Performance

Sayyed Majid Mazinani and Kazem Fathi

Abstract—**Two types of algorithms are realized which have been used within the supervised of model of intrusion detection systems. These algorithms are either of type eager or lazy as far as their performance is concerned. At the learning phase, the lazy algorithms are fairly simple; however, the eager algorithms are highly effective. On the other hand the classification phase is in at most contrast with learning phase. The aim of this research is, taking the advantages of both lazy and eager algorithms to achieve a hybrid algorithm. This approach necessitates employing an eager algorithm of Decision Tree, on the training set, which has led to the creation of a set of Decisions. This set of Decisions is applied on the training set, which results in having a set of binary vectors. In order to enhance the training set these binary vectors were added as new attributes. After that with the lazy algorithm of nearest neighbors, we have classified the samples. The outcome of test results from existing algorithms has been compared with our proposed algorithm. The results show that the proposed algorithm outperforms where the volume of samples are high. The performance of the hybrid algorithm is also remarkable within platforms, with limited or very high processing resources.**

Index Terms—**Intrusion detection system, machine learning, classification.**

I. Introduction

The growth of attacks on networks continues to increase. Attacks range from relatively benign ping sweeps to sophisticated techniques exploiting security vulnerabilities [1].

Intrusion detection is the task of detecting and responding to computer misuses [2]. In other words, an IDS is typically a computer system, which monitors activity to identify malicious activities [3].

II. Machine Learning

Machine learning is an answer for how to construct computer programs that automatically learn with experience [4]. Machine learning algorithms automatically extract knowledge from machine readable information [5]. Two types of machine learning are supervised learning and unsupervised learning. Supervised learning uses labeled training data, while unsupervised learning uses unlabeled training data. Supervised algorithms can classify samples into specific classes [6].

Eager learning is a form of supervised learning, at this form there is a learning module, classification module and a model. Eager learning algorithms invest most of their effort in the learning phase. Usually these algorithms construct a representation of the target function from training instances. Classification of new instances is usually done by rules that uses the model [7].

Opposite of eager learning there is also lazy learning as a different form of supervised learning. In different contexts, memory-based learning algorithms are named lazy [8].

In lazy learning during the learning phase, all input samples are stored and no other attempt will be made [9]. The search for the optimal hypothesis takes place during the classification phase [10].

III. Some Machine Learning Algorithms

Decision Tree is a representation of how to make a decision according to a particular attribute set. Any given Decision Tree is completely deterministic [11]. The decision tree nodes have some attributes, with the branches involving alternative values of those attributes. The leaves represent the classes. To make a decision by a decision tree, select a set of values for an attribute and start at the root of the tree. Decision Tree is a form of supervised learning.

The naive Bayes model is a heavily simplified Bayesian probability model. In simple terms, a naive Bayes classifier assumes that the presence or absence of a particular feature is unrelated to the presence or absence of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 3" in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of the presence or absence of the other features.

For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without believing in Bayesian probability or using any Bayesian methods. Rule induction is a form of eager learning. In this form during the learning phase, the rules can be induced from the training samples. The goal of rule induction is generally to induce a set of rules from data that captures all generalizable knowledge within that data, and that is as small as possible at the same time. The rules that are extracted during the learning phase can easily be applied during the classification phase when new unseen test data are classified.

Manuscript received March 20; revised July 15, 2015.

Sayyed Majid Mazinani is with Imam Reza International University, Mashhad, Iran (e-mail: smajidmazinani@hotmail.com).

Kazem Fathi is with Islamic Azad University, Sari Branch, Iran (e-mail: kzm_fathi@yahoo.com).

There are several instance-based learning algorithms. One of the best known is k-Nearest Neighbor (k-NN). The learning phase of k-NN is simply storing training samples. During the classification phase, k-NN use of a similarity-based search strategy to determine an optimal hypothesis function. New instances will be compared to the stored instances and will be classified the same class label as the k most similar stored instances.

IV. Proposed Hybrid Algorithms

After eager and lazy algorithms, hybrids are considered. Used hybrids are a combination of the k-NN classifier and Decision Tree. Goal of constructing hybrids is to create relationships between memory-based learning and eager learning. Combining eager and lazy learners will produce machine learners that put effort in both the learning and classification phase. The hybrid will use the hypothesis as induced by Decision Tree, and the one created during memory-based learning.

We combined k-NN with rule-induction. Although rules seams fully different from instances that are being used in K-NN, there is a relation between them. Decisions represent a subset of training instances that match with specific conditions. Therefore, k-NN classification can be applied to Decisions [12].

In proposed Replacement Based Decisions (RBD) to create hybrid, Decisions are induced from the training set using Decision Tree algorithm. Then the decisions are transformed into vectors, representing the binary decision-features. The vectors that are produced would seem as new training set [12]. When using RBD, the binary rule-features replace the original features in the instances. From the k-NN view, RBD attempts to reduce the sensitivity of K-NN to noise and irrelevant features.

After RBD we propose a second type of hybrid, named Adding Based Replacement (ABD). When using this hybrid, the initial features of the instance are not replaced by the binary decision-features. These binary decision-features are added to initial features. Therefore, this new algorithm is a k-NN classifier with extra added features. The rule-features is not noise filtering step, but we expect that the rule-features can help to repair K-NNs sensitivity to noise. Setting more weights on more important features in k-NN helps to achieve more accuracy.

V. Attribute Selection

In general there are some reasons to use feature subset selection for machine learning algorithms. First of all, it can cause more performance. Secondly, it provides a reduced hypothesis search space, and third, attribute selection reduces the storages.

The first feature selection method that we used, is information gain. In order to define information gain precisely, a measure commonly used in information theory, called entropy.

The heuristic method, by which CFS measures "goodness" of feature subsets, takes into account the usefulness of individual features for predicting the class. CFS evaluates and hence ranks feature subsets rather than others [13].

TABLE I: Confusion Matrix

	Predicted Negative	Predicted Positive
Actual Negative	*a*	*b*
Actual Positive	*c*	*d*

VI. Dataset Details

The data used are from the University of California, Irvine Knowledge Discovery and Data Mining (UCI KDD) website. The data files give us the information to create and train the algorithms. Neither [14] nor [15], the main references in this research for this dataset, mention what is attacking. The testing data for the 10 percent of data set contains 311029 examples. These examples contain 60593 normal items and 250436 attack items. Therefore, this data is most likely atypical because it contains more attacks than normal data. The training dataset contains 494020 items. There are 97277 normal items and 396743 attack items.

VII. Evaluation Methods

To evaluate the algorithms Weka is used. Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes.

To evaluate the performance of algorithms, different metrics were used. Metrics are calculated using the confusion matrix, which shows the predicted and actual classifications. False positive and false negative are two types of errors that can be made. The number in "*b*" in the confusion matrix corresponds with the type I errors. The number in "*c*" equals the number of type II errors made. Both false positives and false negatives have to be reduced [16], accuracy has to be as close to 1 as possible. The accuracy is the proportion of the total number of predictions that were correct, and is measured by: $\text{Accuracy}=\frac{a+d}{a+b+c+d}$. Precision is the proportion of the predicted positive cases that were correct; $\text{Precision}=\frac{d}{d+b}$. Another metric to evaluate algorithms is recall, it is the proportion of alerts that were correctly identified; $\text{Recall}=\frac{d}{d+c}$. Because the equation mentioned above for accuracy may not be an adequate performance measure, the F-Measure was used. The sparseness of the positive examples could cause the average classification accuracy of the testing set to be unreliable [17]. For this reason, the weighted harmonic mean of precision and recall, with given β, can be calculated as: $F=\frac{(\beta^2+1)\times\text{precision}\times\text{recall}}{\beta^2\times\text{precision}+\text{recall}}$.

VIII. Results

The results of the experiments are summarized in Table II;

each cell is mean of 7 experiments.

TABLE II: EXPERIMENTS SUMMARIZED RESULTS

		Precision	Recall	F-Measure
RIPPER	Full	0.985	1	0.992
	CFS	0.977	0.992	0.984
	IG-10	0.994	0.999	0.996
	IG-6	0.990	0.998	0.994
	IG-2	0.990	0.993	0.991
KNN	Full	0.983	0.998	0.99
	CFS	0.973	0.979	0.976
	IG-10	0.984	0.997	0.99
	IG-6	0.994	0.999	0.996
	IG-2	0.984	0.990	0.987
Naive Bayes	Full	0.965	0.998	0.981
	CFS	0.976	0.928	0.951
	IG-10	0.976	0.988	0.982
	IG-6	0.976	0.989	0.982
	IG-2	0.972	0.937	0.954
Decision Tree	Full	0.970	0.993	0.982
	CFS	0.971	0.993	0.982
	IG-10	0.994	0.999	0.996
	IG-6	0.971	0.993	0.982
	IG-2	0.972	0.937	0.954
RBD	Full	0.986	0.998	0.992
	CFS	0.975	0.998	0.986
	IG-10	0.997	0.999	0.998
	IG-6	0.997	0.999	0.998
	IG-2	0.993	0.996	0.994
ABD	Full	0.987	0.994	0.99
	CFS	0.978	0.999	0.988
	IG-10	0.991	0.997	0.994
	IG-6	0.996	0.998	0.997
	IG-2	0.989	0.986	0.996

It can be concluded that the results indicates the RBD, ABD and Rule Induction algorithms have far better performance in comparison with other algorithms. In order to find out which of these three algorithms are superior to others, their F-measure parameters are compared as is depicted in Fig. 1.

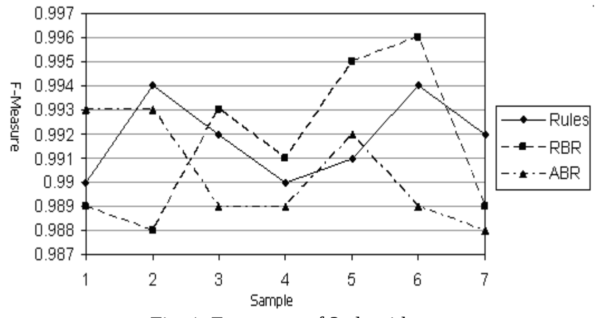

Fig. 1. F-measure of 3 algorithms.

For the next stage of analysis the three parameters of F-measure, Precision and Recall are compared for the RBD, ABD and Rule Induction algorithms. This comparison is done under the condition where few attributes are available. For example when the attributes are filtered with Infogian-2 method (Figs. 2-4).

Fig. 5 shows the F-measure for RBD, ABD and Rule Induction algorithms where all attributes have been filtered with CFS method.

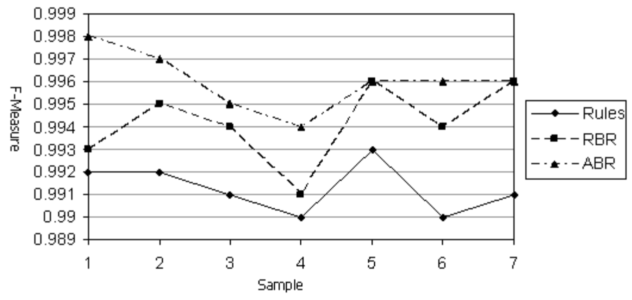

Fig. 2. F-measure of 3 algorithms — Info-2.

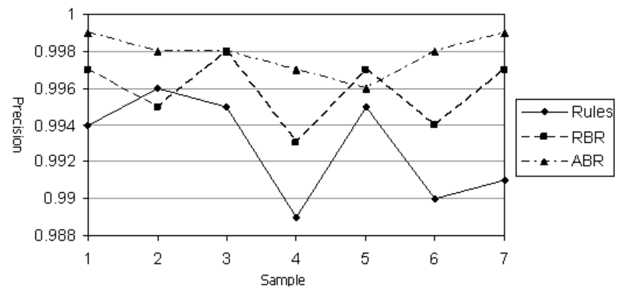

Fig 3. Precision of 3 algorithms — Info-2.

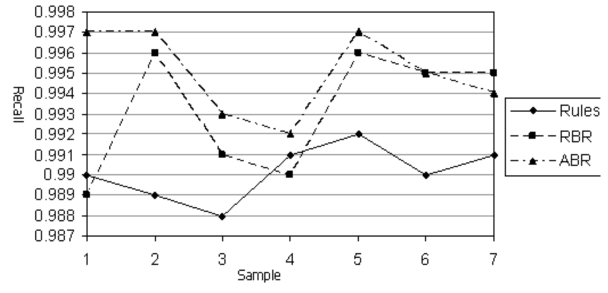

Fig. 4. Recall of 3 algorithms — Info-2.

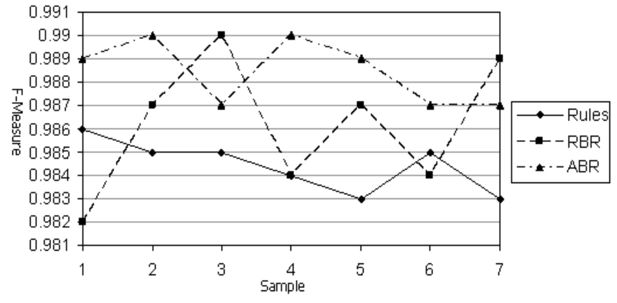

Fig. 5. F-measure of 3 algorithms — CFS.

Eventually it can be concluded that the Rule Induction algorithm in most instances has far superior performance under the condition where there are high volume attributes and highly susceptible to any attacks. Under circumstances where the volume of training set is not very big and the processing capability is not very high (in common environments), the Rule Induction algorithm is suggested. However when there are limitations on any or some parameters, either we are forced to select some of more effective attributes and have a high volume of training set and high processing capability is available, the ABD algorithm is preferable. Finally cause to the ABD is sensitive to process power, when there is limitation of the processing power, RBD algorithm performs better than the ABD algorithm.

REFERENCES

[1] T. Jackson, J. Levine, J. Grizzard, and H. Owen, "An investigation of a compromised host on a honeynet being used to increase the security of a large enterprise network," in *Proc. the 2004 IEEE Workshop on Information Assurance and Security*, 2004.

[2] J. X. Huang, J. Miao, and B. He, "High performance query expansion using adaptive co –training," *Information Processing & Management* vol. 49, no. 2, pp. 441–453, 2013.

[3] Y. Liu, X. Yu, and J. X. Huang, "An combining integrated sampling with SVM ensembles for learning from imbalanced datasets," *Information Processing &Management*, vol. 47, no. 4, pp. 617–631, 2011.

[4] S. X. Wu and W. Banzhaf, "The use of computational intelligence in intrusion detection systems: A review," *Applied Soft Computing*, vol. 10, pp. 1–35, 2010.

[5] Y. X. Wang, "A Sort of multi-agent cooperation distributed based intrusion detection system," *Modem Computer*, 2008

[6] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach (International Edition)*, Pearson US Imports & PHIPEs, November 2002.

[7] I. Hendrickx, *Local Classification and Global Estimation*, Koninklijke drukkerij Broese & Peereboom, 2005.

[8] A. van den Bosch and W. Daelemans, "Do not forget: Full memory in memory-based learning of ord pronunciation," NeMLaP3/CoNLL98, pp. 195–204, 1998.

[9] T. Mitchell, *Machine Learning*, McGraw-Hill International Editions, 1997b.

[10] W. Daelemans, A. van den Bosch, and J. Zavrel, "Forgetting exceptions is harmful in language learning," *Machine Learning*, vol. 34, pp. 11–41, 1999.

[11] D. Fisher, L. Xu, J. Carnes, Y. Reich, S. Fenves, J. Chen, R. Shiavi, G. Biswas, and J. Weinberg, *Applying Ai Clustering to Engineering Tasks*, pp. 51–60, Dec. 1993.

[12] A. van den Bosch, "Feature transformation through rule induction, a case study with the k-nn classifier," in *Proc. the Workshop on Advances in Inductive Rule Learning at the ECML/PKD*, pp. 1–15, 2004.

[13] W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch, *TiMBL: Tilburg Memory-Based Learning-Reference*, 2005.

[14] S. Hettich and S. D. Bay. Kdd cup 1999 data. [Online]. Available: http://kdd.ics.uci.edu//databases/kddcup99/kddcup99.html.irvine

[15] P. A. C. P. Stolfo *et al*., "Cost-based modeling for fraud and intrusion detection: results from the jam project," in *Proc. DARPA Information Survivability Conference and Exposition*, 2000, vol. 2, pp. 130–144.

[16] V. Gowadia, C. Farkas, and M. Valtorta, "Paid: A probabilistic agent-based intrusion detection system," *Computers & Security*, vol. 24, pp. 529–545, 2005.

[17] M. Kubat and S. Matwin, "Addressing the curse of imbalanced training sets: one-sided selection," in *Proc. the 14th International Conference on Machine Learning*, Morgan Kaufmann, 1997, pp. 179–186.

Sayyed Majid Mazinani was born in Mashhad, Iran on January 28, 1971. He received his bachelor degree in electronics from Ferdowsi University, Mashhad, Iran in 1994 and his master degree in remote sensing and image processing from Tarbiat Modarres University, Tehran, Iran in 1997. He worked in IRIB from 1999 to 2004. He also received his PhD in wireless sensor networks from Ferdowsi University, Mashhad, Iran in 2009. He is currently an assistant professor at the Faculty of Engineering in Imam Reza University, Mashhad, Iran. He was the head of Department of Electrical and Computer Engineering from 2009 to 2012. His research interests include computer networks, wireless sensor networks and smart grids.