Key words:
machine learning, ensemble classification, malware detection, computer security

Anna BEDNARSKA*

ENSEMBLE CLASSIFICATION FOR ANDROID-BASED MALWARE DETECTION

This paper describes the real-time malware (malicious software) detection for Android operating systems. It is commonly known that malicious software may cause risky and expensive damages. Due to continuously increasing amount of downloaded and uploaded data it is important to check every piece of it before allowing for an access to our mobile device. Manual checking of every piece of software is impossible. Therefore fast, intelligent filters which may almost in real-time scan downloading packages are of high interest to users. We propose to build a detection system based on ensemble classifiers. In this paper, we present accuracy and running time comparisons of Random Forest, Bagging and Boosting classification committees, and tune the proper number of base trees in each of them. Experiments carried out on a large real-life dataset confirm the usefulness of our proposal.

1. INTRODUCTION

All of mobile devices using Android operating systems are more susceptible to various attacks. Nowadays, malware is powerful way to infringe privacy, damage equipment or to obtain undesirable access to user's data. It also may cause financial abuses and frauds due to obtaining access to making phone calls or sending messages. Malware is a simple piece of code added, changed or deleted from the system, that intentionally cause unwanted behavior of the device [1]. It can be hide in every piece of downloading data. Unfortunately, in spite of increase of popularity and functionality of mobile devices, users still care less about their safety than about safety of the personal computers. It should be changed due to the fact that these days smartphones are keeping as many private data as computers. They are also more prone to attacks and

* Department of Systems and Computer Networks, Faculty of Electronics, Wroclaw University of Technology, Wyb. Wyspiańskiego 27, 50-370, Wrocław, Poland.
E-mail: 201004@student.pwr.edu.pl

adversary access user's behavior like location or talks. The most popular malware detection systems are based on checking by special application every piece of transferring data. It is burdensome for user, therefore it is worth to automatize this process. Nowadays, malware detection methods are using parameters about known malware which are keeping as sequences of bytes. The main disadvantage is that may only recognize known malwares and are unable adapt to new assaults. It is also worth to take into account that it is possible to detect the presence of malware only after its installation on a device and in most cases this is too late. Machine learning algorithms offer a new solution for automatic and real-time malware detection before executing malware detection [2]. It may become very effective and dynamically adjusted tool which allow classification between software and malware [3].
In this paper, we will consider behavior of ensemble classification methods based on classification trees committees for designing an automatic malware filter. We will analyze their accuracies as well as executing times.

2. MALWARE ANALYSYS AND DETECTION

In this paper we will concentrate on malware attack in the Andriod operating systems which are still a weak point in its security and which becomes one of the most targeted and easy accessible operating systems.
We use a large dataset based on real software files and we divided it in two parts- safe and damaging software. Two groups of features were identified. First one was binary structure of the file's headers called Portable Executable (PE) features. They are collecting from known fragments of binary executable files parts: dynamic link library, COFF file header, optional header, second header, resource directory and checksum. The second group of checked features was API call-based features examined between the software and Android operating system. API allows applications to use resources of Android and that's the reason why it is commonly used as a way of performing malicious attacks. This two types of features may potentially offer a highly discriminative information about the nature of analyzed application.

3. ENSEMBLE CLASSIFER

An ensemble classifiers may give better results than using an individual one [4]. Even if each classifier is not the best one, combined in an ensemble may outperform each individual member. There are two kinds of ensembles: dependent and parallel.

- **Dependent committees** are those which use a sequence of classifiers and the result of first has impact on operations in next one (see Figure 1).

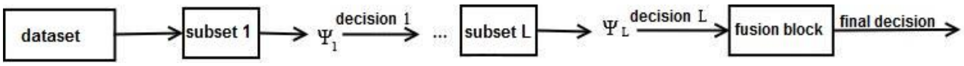

Fig. 1. *Dependent committee*

- **Parallel committees** use independent classifiers and combine their outputs using voting or discriminant-based methods (see Figure 2).

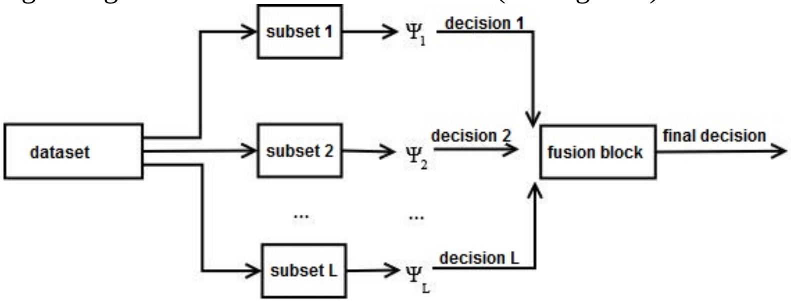

Fig. 2. *Parallel committee*

In our research we used Bagging, Boosting and Random Forest ensembles [5]:

- In **Bagging** on each run, algorithm use the training set to build of a sample of ***m*** elements randomly chosen by sampling with replacement from the original training set. That method is called bootstarp and it built subsets which contains about 63,2% of the original dataset [6]. Using bagging we may choose number of bags - classifiers used in its parallel structure.
- In **Boosting** we use amplification due to dependent structure. Output of the previous classifier is the input of next one to correct previous misclassifications. Each of classified object is labeled by a weight- if classification was correct the weight decreases. If classifier was mistaken, we increase this object's weight in order to boost its importance for the next classifier. In each iteration we use the same data subset. Boosting is complex algorithm and it is difficult to use it in distributed systems, however it is one of the best classifiers for majority of problems.
- **Random Forest** algorithm starts with bagging and learn each subset using tree. In each tree node there is random subset of data set. It uses full and not pruned trees. The main advantage is that it is processing data almost in real-time.

4. EXPERIMENTAL STUDY

During our experiment we were comparing different ensemble classifiers to choose which one may assure real-time and high accuracy malware detection. We used large real-life malware dataset.

4.1. DATASET AND SET-UP

In our experiment we used malware dataset which contains 376 attributes of 2000 objects. Each of them was labeled by 0 what means software or 1-malware. Dataset was balanced.
We used 10-fold cross validation for training / testing procedure. We analyzed behavior of bagging, boosting and random forest ensembles each built for 5, 25, 50, 75 and 100 built in trees. We compare results with single full and pruned tree accuracy using gini index method.

4.2. RESULTS

Firstly, we need to tune the number of base classifiers for our ensembles. Outcome of this procedure is presented in Figure 3.

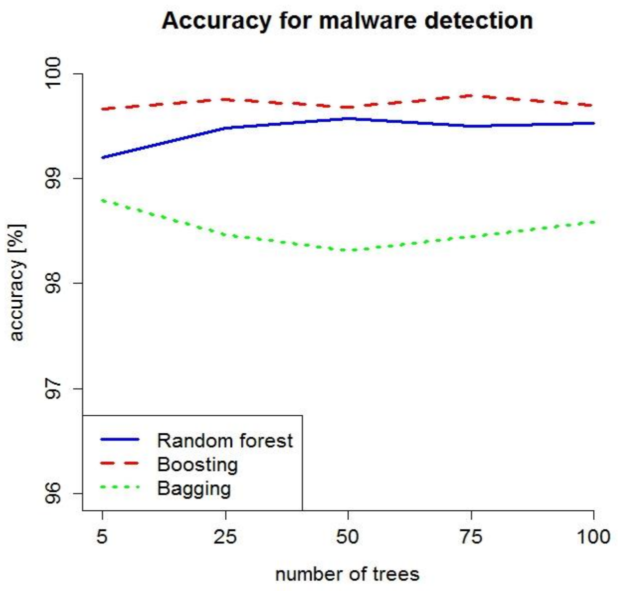

Fig. 3. *Comparison of accuracy for malware detection*

As we can see, all of the ensemble classifiers obtained satisfying results from about 98% to more than 99% accuracy. Increasing number of built in trees has almost no impact for accuracy and amount of 25 seems to be optimal for each of tested methods. However, let us take a closer look for a execution time which is presented in Table 1.

Table 1. *Comparison of training times [s] for different ensemble methods.*

Number of trees / *Type of classifier*	*5*	*25*	*50*	*75*	*100*
Random Forest	**0.440**	**1.045**	**1.790**	**2.578**	**3.343**
Bagging	5.470	27.638	54.800	83.820	111.921
Boosting	4.425	18.949	37.277	55.184	73.677

Comparing accuracy of best settings for each ensemble and single both full and pruned CART with gini index we noticed that combining classifiers we reach about 2% better accuracy. It confirms drift of using ensemble in our research. In Figure 4 we can also see stability of classification.

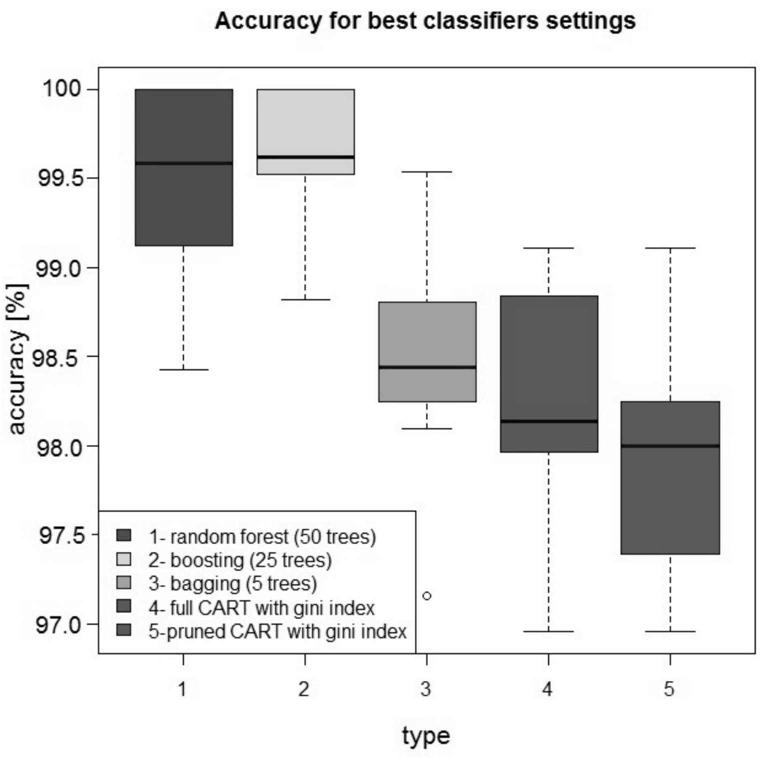

Fig. 4. *Comparison of best classifiers settings*

As we can see, random forest execution time is clearly better than for both bagging and boosting algorithms. The increase is not as much dependent on amount of using trees and it still gives almost real-time feedback to the user.

4.3. RESULTS

On the grounds of presented results, we decided that to solve malware detection problem it is worth to use Random Forest algorithm. It has realized all of our requirements: high accuracy, fast response. Random Forest malware filter is able to work with Android system without impeding user's work. It can also recognize malware before installation on mobile device.

5. CONCLUSIONS

In this paper we have dealt with the problem of an automatic malware detection. We have proposed a highly accurate ensemble learning approach for this task. This is an important step towards increasing a security of mobile devices, as the proposed classifier embedded within the Android system will allow for early detection and removal of malicious software. By using Portable Executable features we are able to detect suspicious application before their installation, thus preventing them from accessing crucial system files.
In the future work, we would like to test presented methods for imbalanced big datasets and also check their efficiency in working with data streams to process real life datasets.

REFERENCES

[1] B. KRAWCZYK, M. WOŹNIAK. Evolution Cost-Sensitive Ensemble for Malware Detection. *International Joint Conference SOCO'14-CISIS'14-ICEUTE'14: 433-442, 2014.*

[2] S. SHEEN, R. ANITHA, P. SIRISHA. Malware detection by pruning of parallel ensembles using harmony search. *Pattern Recognition Letters 34(14): 1679-1686, 2013.*

[3] K. RIECK, P. TRINIUS, C. WILLEMS, AND T. HOLZ. Automatic analysis of malware behavior using machine learning. *Journal of Computer Security, 19(4):639-668, 2011.*

[4] T. G. DIETTERICH. Ensemble Methods in Machine Learning. *Multiple Classifier Systems: 1-15, 2000.*

[5] M. WOŹNIAK, M. GRAÑA, E. CORCHADO. A survey of multiple classifier systems as hybrid systems. *Information Fusion 16: 3-17, 2014.*

[6] L. KUNCHEVA, M. SKURICHINA, R. P. W. DUIN. An experimental study on diversity for bagging and boosting with linear classifiers. *Information Fusion 3(4): 245-258, 2002.*