Advances in Computer Science Research, volume 87

3rd International Conference on Mechatronics Engineering and Information Technology (ICMEIT 2019)

Research of Android Malware Detection based on ACO Optimized Xgboost Parameters Approach

Jie Ling[a], Xuejing Wang [b, *], Yu Sun[c]

Faculty of Computer, Guangdong University of Technology, Guangzhou, China

[a]jling@gdut.edu.cn, [b, *] gdut_wxj@163.com, [c]gdut6282015@126.com

Abstract. In order to deal with low efficiency and accuracy of detection caused by the improper selection of Xgboost parameters in Android malware detection. In this paper, we introduce Ant Colony Optimization (ACO) into Xgboost parameters optimization and propose an approach based on ACO optimize Xgboost parameters in Android malware detection. Selecting features such as permissions, intents and APIs in AndroidManifest.xml and smali files and extra the optimal feature subset, then apply to the proposed method. The experimental results show that the proposed method effectively improves accuracy of detection and reduces false positive rate compared with the Xgboost algorithm optimized by Genetic Algorithm (GA) and Particle Swarm Optimization (PSO).

Keywords: ACO, Xgboost, Android, malware, detection classification.

1. Introduction

Android operating system has become the most popular smartphone operating system due to its open source, compatibility and market openness. A variety of applications provide great convenience for people's lives, but many types of Android malware have followed. The security problems brought by Android smart terminals are becoming more and more serious, causing many problems such as user privacy leakage and economic loss, which brings a lot of troubles to users. Therefore, it is of great significance to the research of malware detection. How to effectively detect malware has become a hot research topic.

At present, there are some methods in android malware detection, mainly including static detection, dynamic detection, and hybrid detection methods combining static detection and dynamic detection [1]. With the widespread use of machine learning algorithms, many researchers have attempted to use machine learning methods for Android malware detection research. In [2], a set of Android malware detection methods based on privilege correlation is proposed. The Naive Bayes is improved to form a classifier, which achieve the initial rapid detection of malware. But it only uses a single feature which is likely to have a poor result in detection. Reference [3] proposed an Android malware detection method based on the permission frequent pattern mining algorithm PApriori. Reference [4] proposed an Android malware detection method based on Support Vector Machine (SVM), which uses dangerous permission combinations and vulnerable API calls as feature attributes, and establishes SVM classifier to automatically distinguish malware and benign software. Reference [5] proposed a sandbox-based Android malware dynamic analysis scheme, which designs and implements a recoverable Android sandbox through virtual machine, and uses the reverse tool to insert API monitoring code into the Android installation package. The software run in the box and the API call information is monitored, so as to simulate the real running process of the application in the sandbox. The effect of detecting the malware is achieved, but cost expensive resources. Reference [6] proposed an efficient, system-wide information flow tracking tool, TaintDroid, which can simultaneously track the diffusion path of multiple sensitive data of Android applications and achieve the tracking of multiple sensitive data leakage sources. Reference [7] designed and implemented the dynamic monitoring framework of Android application behavior. The experiments evaluated the four algorithms of SVM, decision tree, k-nearest neighbor and naive Bayes. In [8], the dynamic and static methods are used to extract three types of features, designs a Triple Hybrid Ensemble Algorithm (THEA) for three types of features and realizes Androdect tool by THEA algorithm, which is complicated in terms of technical implementation.

The current research results have many shortcomings in terms of detection accuracy, false positive rate and implementation complexity. This paper proposes an Android malware detection approach based on Xgboost that optimized by ACO, which processes the Android application through reverse engineering, extracts features such as Permission, Intent and API, and feature selection algorithm is employed to obtain the optimal features subset. Training the classifier by using the ten-fold cross-validation method. Experimental results show that proposed method effectively improves accuracy and reduces false positive rate of android malware detection.

2. Related Algorithms and Parameter Optimization

2.1 Xgboost Algorithm

Xgboost (eXreme Gradient Boosting) was developed on the basis of the Gradient Boosting Decision Tree, which was proposed by Tianqi Chen in 2015 [9]. Compared with the traditional GBDT algorithm, it is more advanced, the traditional GBDT mainly uses the first-order derivative information, and Xgboost uses the second-order Taylor expansion on the loss function. In the Xgboost integrated learning framework, the parameter shrinkage and the minimum sample weight threshold (min_child_weight) are the main factor influence the algorithm performance, shrinkage and min_child_weight too small may lead to over-fitting , shrinkage too large may cause the algorithm to not converge. Excessive min_child_weight will result in the classification performance of the algorithm for linear indivisible data. We take the objective function logloss as an example for theoretical introduction.

Perform the second-order expansion of the Taylor function of the objective function:

$$Obj^{(t)} = \sum_{i=1}^{n} \omega\left(y_i, y_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + C$$

$$\approx \sum_{i=1}^{n}\left[\omega\left(y_i, y_i^{(t-1)}\right) + \partial_{y_i^{(t-1)}}\omega\left(y_i, y_i^{(t-1)}\right)f_t(x_i) + \frac{1}{2}\partial^2_{y_i^{(t-1)}}\omega\left(y_i, y_i^{(t-1)}\right)f_t^2(x_i)\right] + \Omega(f_t) + C \quad (1)$$

Where $\partial_{y_i^{(t-1)}}\omega\left(y_i, y_i^{(t-1)}\right)$ is the first derivative of each sample, $\partial^2_{y_i^{(t-1)}}\omega\left(y_i, y_i^{(t-1)}\right)$ is the second derivative of each sample.

Since the objective function is log loss, there is

$$\partial_{y_i^{(t-1)}}\omega\left(y_i, y_i^{(t-1)}\right) = -y_i\left(1 - \frac{1}{1+e^{-y_i^{(t-1)}}}\right) + (1 + y_i)\frac{1}{1+e^{-y_i^{(t-1)}}} \quad (2)$$

$$\partial^2_{y_i^{(t-1)}}\omega\left(y_i, y_i^{(t-1)}\right) = \frac{e^{-y_i^{(t-1)}}}{(1+e^{-y_i^{(t-1)}})^2} \quad (3)$$

2.2 Ant Colony Optimization (ACO)

The ACO is an approximate optimization algorithm firstly proposed by Marco Dorigo to find the optimal path, which is inspired by the behavior of ant colonies finding the shortest path in the foraging process [10]. Ants release pheromones on their path to transmit information. Other ants in the ant colony will perceive pheromones and walk along a higher path of pheromone concentration. Each passing ant will release pheromones. Thus a positive feedback mechanism is formed. After a period of time, the entire ant colony will follow the optimal path to the food source.

The ACO is an intelligent optimization algorithm, and has better superiority in discrete optimization problems. In order to adapt the ACO to the solution of this problem, this paper takes the classification accuracy of Xgboost as the objective function value:

$\max\{F(s_1,\omega_1),F(s_2,\omega_2),\cdots F(s_m,\omega_m)\}$. Recorded as $\max \text{fitness} = \max\{F(X)\}$, Where $X = \{x_1,x_2,\cdots,x_m\}$, x_i represents the ant.

In this paper, the chaotic sequence [11] is used to initialize the ant colony, and its objective function is calculated. x_j^k is the position vector of the jth ant of the kth iteration. The larger the target function is, the larger the position pheromone concentration is, and the current target value x_{best}^k is saved. Ants as well as their pheromone maximum is τ_{max}^k.

For the ant colony transfer rule, the ant individual with large transition probability can search the local directional step in the solution space, which can effectively avoid falling into the local optimal solution and shorten the search time.

$$\mathrm{P}(x_j^k) = \exp\left(-\mathrm{S} * \left(\mathrm{F}(x_{best}^k) - \mathrm{F}(x_j^k)\right)\right) \tag{4}$$

Where S is the standard deviation of the fitness function and is calculated as follows:

$$\mathrm{S} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(F(x_i) - F_{ave})} \tag{5}$$

In the formula (5), m is the number of ants, and F_{ave} is the average fitness value;

If $\mathrm{P}(x_i) < P_0$, where P_0 is a constant, $0 < P_0 < 1$, then the ant searches at a local location nearby, and the formula is as follows:

$$x_{j-new}^k = x_{j-old}^k \pm \mathrm{a} \tag{6}$$

Where x_{j-new}^k is the position after the move, x_{j-old}^k is the position before the move, and a is the move step, as defined below:

$$\mathrm{a} = \frac{1}{(k+1)^2} \tag{7}$$

If $\mathrm{P}(x_i) > P_0$, the ant searches in the solution space.

2.3 Xgboost Parameter Optimization based on ACO

Aiming at the problem of parameter setting in Xgboost algorithm, proposes an method that ACO optimize Xgboost algorithm. Following the basic description of the algorithm, we update parameters of the Xgboost by the ant colony optimization algorithm for each iteration, until we get the max iteration and output the final Optimal parameters of Xgboost, and apply to the Xgboost classification model. Fig.1 is The process of the parameter optimization.

The specific implementation steps are as follows:

Step 1: Set the upper and lower limits of the shrinkage in the child node and min_child_weight, the maximum number of iterations MaxIter, the ant colony size M, the information evaporation coefficient Rho;

Step 2: Initialize the shrinkage and min_child_weight as the position vector of each ant;

Step 3: Perform an ant colony search operation;

Step 4: Input the data set into the Xgboost classifier for training;

Step 5: Calculate the objective function value and pheromone value of each ant with the Xgboost classifier to find the current optimal ant;

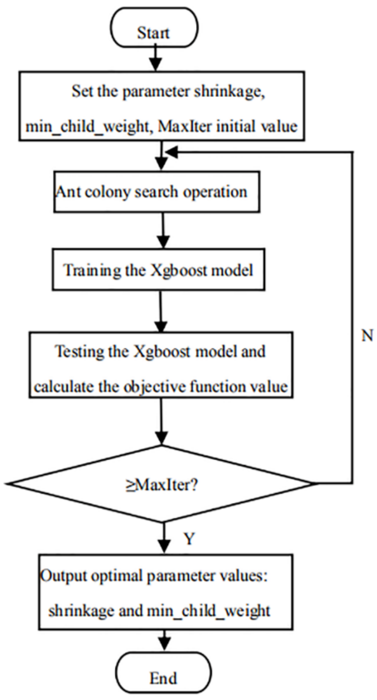
Start
Set the parameter shrinkage, min_child_weight, MaxIter initial value
Ant colony search operation
Training the Xgboost model
Testing the Xgboost model and calculate the objective function value
≥MaxIter?
N
Y
Output optimal parameter values: shrinkage and min_child_weight
End

Fig. 1. The process of optimizing the Xgboost parameters by ACO

Step 6: Determine whether the termination condition is satisfied: if the number of iterations is greater than MaxIter, output the ant colony optimal value and the corresponding shrinkage and min_child_weight values;

Step 7: Update the pheromone. If the termination condition is not satisfied, go to step 3.

3. Principles of Detection Methods

3.1 Method Design

The process of Android malware detection method in this paper is shown in Fig.2. The method mainly consists of three stages: feature extraction, feature optimization and detection classification. The detailed processing is as follows:

1) Decompiling apk files to get AndroidManifest.xml files and smali files.

2)Extracting attributes such as Permission and Intent of the AndroidManifest.xml file and the API of smali file. According to the extracted attributes, the number of occurrences of each feature is counted and sorted, and the attributeswith high frequency are combined into a feature set.

3) Comparing the attribute values extracted with each apk file and generate feature vector.

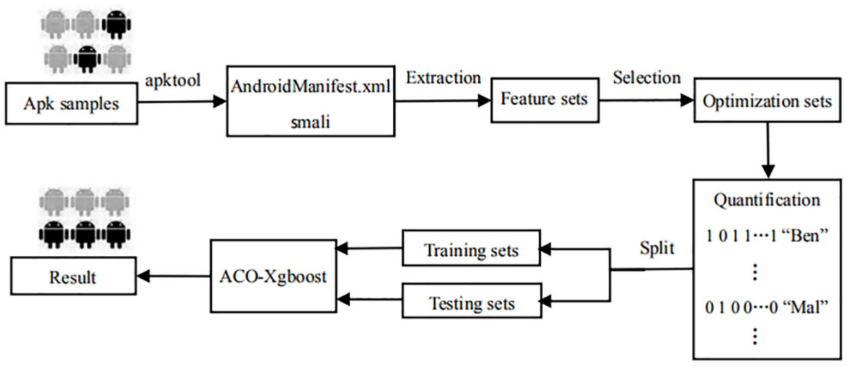

Fig. 2. Detection method process

4) Using the feature selection algorithm to perform feature selection and sorting on the generated feature vector and selecting more representative features according to the sorting result and generate an optimal feature subset.

5) According to the ten-fold cross-validation, the optimal feature subset is divided into training set and test set in a 9:1 ratio, put into the proposed method for training and testing, and compute the accuracy and false positive rate.

3.2 Feature Vector Extraction

In the feature extraction stage, extracting the Permission, Intent and API in the apk file as features, and then constructs the feature vector by quantizing, finally merges the feature vectors of all Android software samples into a data set. The process of obtaining Permission, Intent, and API is shown in Fig.3. The specific process design is as follows:

Step 1: Using apktool [12] to decompile all the apk files and generate a folder containing the AndroidManifest.xml and smali files.

Step 2: Parsing AndroidManifest.xml and count the Permission and Intent. Parsing the smali file and get the API feature.

Step 3: The elements in the feature vector can take the value of "1" or "0". The "1" indicates that the apk application contains the corresponding attribute, and the "0" indicates that the corresponding attribute is not included. Then, add a flag at the end of the feature vector, the flag can take the value "Ben" or "Mal", "Ben" indicates that the application is benign software, and "Mal" indicates that the apk application is malware.

Step 4: Constructing a feature vector using the quantified result and the category label ('Ben' and 'Mal'). Each feature

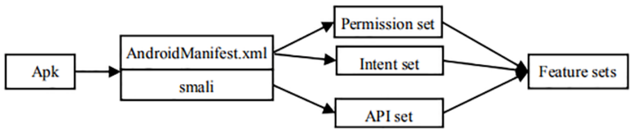

Fig. 3. Feature extraction process

vector represents a sample, and all feature vectors are combined into experimental data.

4. Experiment Result and Analysis

4.1 Sample Source and Experimental Environment

A total of 3,260 samples are collected in this experiment, including 2000 benign software that from Google's official application market, Google Play [13], including all application categories. The malware comes from the Android malware genome project [14], a total of 1,260 samples covering all categories of malicious application families.

In the experiment, the host is configured as Intel Core i5-3570 CPU @ 3.40 GHz, memory is 4GB, feature extraction and algorithm are implemented by Python programming, and feature filtering is done in Weka [15] environment.

4.2 Feature Vector Extraction and Optimization

According to the steps of the feature vector extraction stage, the AndroidManifest.xml and smali files are obtained by decompiling and parsing the apk file. The attribute value information of the tag items <uses-permission>, <intent-filter> <action> and <intent-filter><category> in benign software and malware is obtained from the AndroidManifest.xml file. The set of API features in benign software and malware is parsed from the smali file. The IG algorithm is used to

Table 1. Common feature statistics

Num	Permission	Intent	API
1	READ_SMS	PACKAGE_REMOVED	getSubscriberId( )
2	WRITE_SMS	BATTERY_CHANGED_ACTION	getDeviceId( )
3	SEND_SMS	MEDIA _ MOUNTED	getNETWORKCountryIso( )
4	CHANGE_WIFI_STATE	ACTION _ POWER _DISCONNECTED	getLatitude( )
5	RECORD_AUDIO	SIG _ STR	getLogitude( )
6	VIBRATE	PACKAGE _ ADDED	getOutputStream( )
7	CAMERA	USER _PRESENT	getInputStream( )
8	WAKE_LOCK	VIEW	sendTextMessage( )
9	CHANGE_NETWORK_STATE	CATEGORY.DEFAULT	sendDataMessage( )
10	BROWSABLE	BROWSABLE	startService( )

optimize the feature attributes. Among them, some common Permission, Intent and high-risk API are shown in Table 1.

4.3 Evaluation Indicator

(1) TP (true positive): The number of malicious samples that are correctly detected as malicious samples.
(2) FN (false negative): The number of malicious samples that are mistaken for normal samples.
(3) TN (true negative): The number of normal samples correctly classified as normal samples.
(4) FP (false positive): The normal sample is falsely reported as the number of malicious samples.
Based on the above concepts, the following evaluation indicators are derived:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$
$$\text{RecallRate} = \frac{TP}{TP + FN}$$
$$\text{FPR} = \frac{FP}{TN + FP}$$
$$\text{FNR} = \frac{FN}{TP + FN}$$
$$\text{Precision} = \frac{TP}{TP + FP}$$

4.4 Comparative Experiment and Analysis

This paper designed an experiment to verify the effectiveness of Android malware detection method based on ACO optimize Xgboost parameters. In order to verify the efficiency and accuracy of the method, we use the ten-fold cross-validation method to classify the sample set. The obtained optimal application sample set is divided into 10 parts, one part is taken as a testing sample, and the other 9 parts are training samples in turn, the experimental result is taken as 10 average values.

In order to verify the good performance of ACO optimizing Xgboost parameters, this paper also uses Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) to optimize the parameters of Xgboost. Comparing Android malware classification result by Xgboost classifier with different optimization algorithms. The evaluation indicators used in this paper for classification detection performance are ACC and FPR. The experimental results are shown in Fig.4 and Table 2.

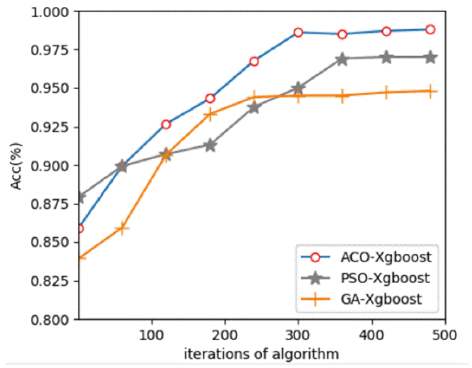

Fig.4. ACC of Xgboost with different optimization algorithms

Table 2. ACC and FPR of Xgboost with different optimization algorithms

Algorithm	ACC	FPR
GA-Xgboost	0.9480	0.091
PSO-Xgboost	0.9692	0.063
ACO-Xgboost	0.9880	0.008

As shown in Fig. 4, the convergence speed and the search performance of the ACO optimizing Xgboost parameters are better than those of GA and PSO. In Table II, the proposed method ACO-Xgboost has an accuracy of 98.80% in android malware detection, which is 4% and 1.88% higher than GA-Xgboost and PSO-Xgboost classification methods respectively. With the improvement of detection accuracy, the false positive rate is naturally reduced. The false positive rate of proposed method is only 0.8%, which is lower than the other two methods.

5. Conclusion

Aiming at the problem that the classification efficiency and accuracy is affected by the improper parameter's selection of Xgboost algorithm. In this paper, we optimize the Xgboost parameters by ACO algorithm, and propose a method based on ACO optimizing parameters of Xgboost in android malware detection. By selecting three feature attributes of Permission, Intent and API, and obtaining the optimal feature subset through feature selection algorithm, apply them into the proposed method with the optimal parameter combination to detect and classify Android malware. Compared with the Xgboost algorithm based on GA optimization and PSO optimization, the proposed method accuracy is higher and false alarm rate is lower than the other two methods respectively.

Acknowledgements

This work is supported by the science and technology project of Guangdong Province (No. 2017B090906003) and the project of Guangzhou Science and Technology (No.201802010043, 201807010058).

References

[1]. Qing SH. Research progress on Android security. Ruan Jian Xue Bao/Journal of Software, 2016,27(1):45−71(in Chinese). http://www.jos.org.cn/1000- 9825/4914.html.

[2]. ZHANG R, YANG J Y. Android malware detection based on permission relevance[J]. Journal of Computer Applications, 2014, 34(5): 1322-1325.

[3]. Yang Huan, Zhang Yuqing, Hu Yuhuai, et al. Android malware detection method based on permission sequential pattern mining algorithm[J]. Transactions, 2013(S1): 106-115.

[4]. Li W, Ge J, Dai G. Detecting Malware for Android Platform: An SVM-Based Approach [C]// IEEE International Conference on Cyber Security and Cloud Computing. Piscataway, New Jersey, USA: IEEE Press, 2015: 464-469.

[5]. ZHAO Y, HU L, XIONG H, et al. Dynamic Analysis Scheme of Android Malware based on Sandbox[J]. NetinfoSecurity, 2014, (12):21-26.

[6]. Enck W, Gilbert P, Han S, et al. TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones [C]// ACM Transactions on Computer Systems. New York, New York, USA: ACM Press, 2014: 393-407.

[7]. Sun Runkang, Peng Guojun, Li Jingwen, et al. Behavior oriented method of Android malware detection and its effectiveness[J]. Journal of Computer Applications, 2016, 36(4): 973-978.

[8]. Yang Huan, Zhang Yuqing, Hu Yuhuai, et al. A Malware Behavior detection system of Android Application based on Multi-class Features[J]. Chinese Journal of Computers, 2014, 37(1): 15-27.

[9]. Chen T, Guestrin C. Xgboost: A scalable tree boosting system[C]//Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. ACM, 2016: 785-794.

[10]. Duan Haipin, Zhang Xiangyin, Xu Chunfang. Bio-inspired computing [M]. Beijing: Science Press, 2011.

[11]. Meng Hongji, Zheng Peng, Mei Guohui, et al. Particle Swarm Optimization Algorithm Based on Chaotic Sequences[J]. Control and Decision, 2006, 21(3): 263-266.

[12]. FENG S Q. Android software security and reverse analysis[M]. Beijing: PTPRESS, 2013.

[13]. Https://play.google.com/store.

[14]. JIANG X, ZHOU Y. Dissecting Android malware: characterization and evolution [C]// IEEE Symposium on Security & Privacy. New Jersey, USA: IEEE, 2012: 95-109.

[15]. YUAN M Y. Data Mining and Machine Learning: WEKA Application Technology and Practice [M]. Beijing: Tsinghua University Press, 2016: 329-344.