Comparative Study of K-means and Mini Batch K-means Clustering Algorithms in Android Malware Detection Using Network Traffic Analysis

[1]Ali Feizollah, [2]Nor Badrul Anuar, [3]Rosli Salleh and [4]Fairuz Amalina
Computer System and Technology Department.
University of Malaya, Kuala Lumpur, Malaysia
ali.feizollah@siswa.um.edu.my[1], badrul@um.edu.my[2],
rosli_salleh@um.edu.my[3] and fairuzamalina@siswa.um.edu.my[4]

***Abstract*— This paper evaluates performance of two clustering algorithms, namely k-means and mini batch k-means, in the Android malware detection. Network traffic generated by the Android applications, normal and malicious, is analyzed for detection purpose. We have used MalGenome data sample for this work to build the dataset. We chose 800 samples out of 1260 Android malware samples. In addition, we collected numerous normal applications from the official Android market. The results show that mini batch k-means algorithm performs better than k-means algorithm in the Android malware detection.**

Keywords-Android; clustering; dynamic analysis; malware.

I. INTRODUCTION

The security of mobile devices has been of a big concern recently due to the sensitivity of stored data such as mobile banking credentials, credit card numbers and personal emails. Google's Android dominates the market share of mobile operating systems. In the third quarter of 2013, 236 million units of mobile devices were shipped, out of which 79.3% were running Android operating system. Apple's iOS ranked second with 13.2% of market share [1]. Different venders such as Samsung, HTC, LG and recently Nokia use Android as their main operating system for their devices. The popularity of the Android lies in the fact that it is an open source operating system and thus it is free. Therefore, the overall price of their product decreases, making such devices appealing to end users [2].

Furthermore, there has been an increase in the number of Android malwares. Not only is the Android popular among users, but also it is targeted by attackers more than other mobile operating systems. In February 2014, Symantec reported that they discovered an average of 272 new malwares and five new families per month targeting Android operating system in 2013 [3]. In addition, it is believed that the speed of Android malware growth is not slowing down, but it is accelerating [3]. Based on a threat report from Symantec, the number of Android malwares increased by 200% between April 2013 and June 2013 reaching almost 300,000 malwares [4]. It is believed that 2014 is an important year for mobile malware, and security issue in such area is of a big concern [4].

Thus, there is a vital need to confront such an important security issue. Researchers have been studying malwares and their behavior in order to find an effective way to tackle them. Machine learning methods are widely used to develop intelligent systems for malware detection. There are two schemes in machine learning: classification and clustering.

Classification is a method of teaching a machine to make decision after it is trained via labelled data. The data labelling is a time consuming and laborious task that involves distinguishing normal data from malicious data done by an expert. The labelled data is then used to train a classification algorithm after which it is able to identify malicious data automatically based on what it learned.

Clustering, on the other hand, does not require data labelling. Clustering algorithms sift through data and discover a pattern based on which they categorize the data to groups. The clustering methods are more difficult than classification since heavy mathematical calculations are required.

In order to conduct a research work, a collection of data is needed. In this work, we used MalGenome data samples. It is an assemblage of 1,260 Android malwares gathered in the period of August 2010 and October 2011 [5]. They are categorized based on their families, which is defined as grouping malwares with resembling behavior. In addition, we collected clean applications from Google Play for analysis in this work.

978-1-4799-6444-4/14/$31.00 ©2014 IEEE

We analyzed the network traffic generated by each application. Quintessentially, most of the applications, and in particular malicious ones, require network connectivity in order to communicate with their servers. The malicious applications use the network connection to leak data from the mobile device to the attacker [6]. Thus, analyzing the network traffic is a plausible feature for this paper.

In this paper, we use the MalGenome data sample for our experiment. Network traffic of each application is collected and then analyzed using clustering methods. We chose k-means and mini batch k-means algorithms. In the result section, we discuss the performance of each algorithm along with the difference between them.

This paper is organized as follows. The next section discusses some of the research works related to this paper. Methodology of the experiment is explained in details in the section 3. The clustering algorithms employed in this paper are discussed in the section 4 along with background information about the algorithms. The empirical results are presented in the section 5 including evaluation measures and their definition. Finally, section 6 concludes this paper with an overview of the overall work.

II. Related Works

The classification method has been used widely in the Android malware detection. The labelling task is necessary for classification, which is a burdensome task. However, the clustering method does not require labelling of the data and it has not been used a lot for the Android malware detection compared to the classification method.

The authors of [7] conducted an experiment using clustering method. They extracted permissions from Android applications. The permissions were extracted from over 18,000 Android application to build a dataset. They used k-means algorithm on the collected data for clustering. The results show positive effect of k-means on the data.

Sahs and Khan [8] extracted the Android file's permissions and control flow graphs (CFG) [9] and applied the support vector machine (SVM), a classification algorithm, on them to make the system learn the pattern of malicious applications against the normal ones. The results show high accuracy as much as 93%.

Another study [10] introduced CHEX system in which the analysis was done on the code of Android applications to detect suspicious components in the application which can be used to hijack the application so that the attacker would be able to control the application and performs malicious activities. CHEX was evaluated with 5,486 Android applications and found 254 potential component hijacking vulnerabilities. They managed to reach 81% of true positive rate.

In this paper, we analyze the network traffic generated by Android applications, malicious and normal ones, and use two clustering algorithms for detection purpose. Additionally, the results of the two clustering algorithms are evaluated using appropriate measures.

III. Methodology

This paper evaluates the effectiveness of the k-means and mini batch k-means in Android malware detection. Furthermore, the results are compared in order to find the best performance. The architecture entails three stages.

The first stage is to install applications, normal and malicious, on the mobile device. The device is connected to the internet via 3G connection to allow applications to communicate over the net. Upon installation of the application, the generated network traffic is collected with a specific application. At the end of this stage, we have a dataset of massive number of packets generated by normal and malicious applications.

The second stage involves sifting through the pool of data, which are collected from stage one, and evaluate the best network features for this paper. We discuss the full process in a dedicated section. At the end of this stage, the dataset is ready to use in the clustering algorithms.

The final stage involves utilizing k-means and mini batch k-means algorithms on the refined dataset, which is ready from the second stage. Output of this stage is results along with calculation of the evaluation measurements to assess performance of the algorithms. Figure 1 depicts the proposed architecture of this paper.

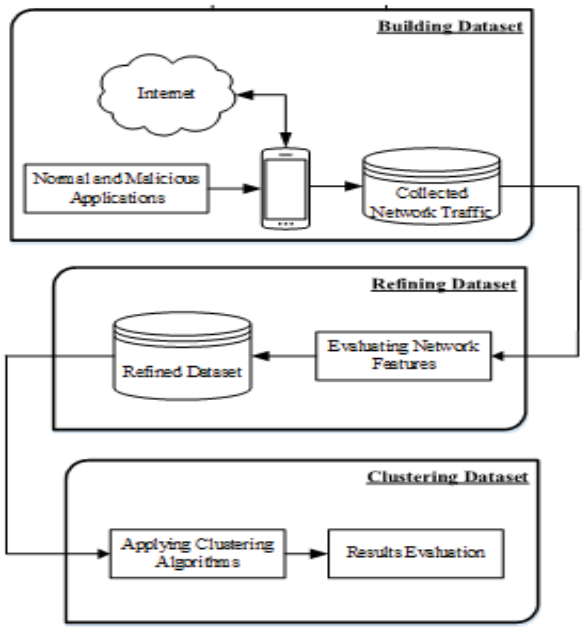

Fig.1. The Proposed Architecture

The following sections discuss each stage of the architecture in details.

A. Building Dataset

The first stage of the architecture involves collecting generated network traffic from the applications, normal or malicious, on the device. To do so, we have installed a dedicated application on the device to capture the network traffic of a specific application. We used 800 samples out of 1260 samples available in the MalGenome data samples for this experiment and we downloaded top applications from the official market. Each application is installed on the mobile device separately and is run for 30 minutes. Unlike other research works where the analysis is done on an emulator, we experimented on real devices. Needless to say that some malwares do not represent their malicious behavior on the emulator. It is worth noting that the user interacts with the application throughout the experiment while the network traffic is being captured. After collecting network traffic, they are all converged as a dataset, which contains normal and malicious traffic. The dataset is passed to the second stage for further processing.

B. Refining Dataset

In the second stage, the dataset from the previous stage is refined by scouring through the pool of network features. Based on the open source interconnection (OSI) [11] layers, every packet has lots of features such as source IP address, destination IP address, source port, destination port, size of the packet, used protocol in the packet, etc. We chose six features, which are frame length, frame number, connection duration, relative duration (time since the first frame), source port and destination port. The selected features are then extracted from the dataset using tshark program and the refined dataset is passed to the next stage.

C. Clustering Dataset

The final stage deals with clustering the refined dataset. We have applied two clustering algorithms on the dataset. The clustering algorithms involve finding a pattern in unlabeled data and divide the data into two categories. The output of this stage is the results of the experiment. The following section describe the clustering algorithms employed in this paper.

IV. CLUSTERING ALGORITHMS

We have used two clustering algorithms in this experiment. K-means and mini batch k-means are used and the results are produced at the end of the experiment.

A. K-means Algorithm

K-means clustering algorithm [12] is widely used among researchers. It divides the data into k clusters. It first defines k points, each for every cluster, which are known as centroids. The centroids are considered the center of a cluster and k-means algorithm chooses them randomly. It then takes each data and calculates its distance from the centroids. The data belong to the cluster with the shortest distance to the centroid. The aim of k-means is to minimize the sum of squares within the clusters that is defined in the equation 1.

$$J = \sum_{j=1}^{k} \sum_{i=1}^{x} \left\| X_i^{(j)} - C_j \right\|^2 \quad (1)$$

In the above formula, $\left\| X_i^{(j)} - C_j \right\|^2$ is a chosen distance between data $X_i^{(j)}$ and a cluster center C_j. The aim is to minimize the sum of squares within the clusters as much as possible. Minimizing it leads to having denser clusters with obvious separation between clusters which is the optimal result.

B. Mini Batch K-means Algorithm

The mini batch k-means clustering algorithm [13] is the modified version of the k-means algorithm. It uses mini-batches to reduce the computation time in large datasets. In addition, it attempts to optimize the result of the clustering. To achieve this, the mini batch k-means takes mini-batches as an input, which are subsets of the whole dataset, randomly. The mini batch k-means is considered faster than k-means and it is normally used for the large datasets.

V. RESULTS AND DISCUSSIONS

The results of our experiment are presented in this section. The experiment involves employing k-means and mini batch k-means clustering algorithms on the dataset of network traffic generated by Android applications. The experiment was done using customized Python [14] scripts.

A. Evaluation Metrics

We express the results of the experiment using evaluation metrics. They are as follows.

- Accuracy: It expresses how accurate the results of an algorithm are. In other words, the accuracy measures the closeness of predicted results to the actual results.
- Homogeneity: It is defined as each cluster contains only members of a single class. In other words, a degree describes how homogeneous a cluster is. It has a value between 0 and 1 where 1 is the optimal result [15].
- Completeness: All members of a given class are assigned to the same cluster. It has a value between 0 and 1. The value of 1 represents the perfect completeness [15].
- V-measure: It is a harmonic mean of the homogeneity and completeness. It is entropy-based measure to calculate

how homogeneity and completeness satisfied their criteria [16].

- Inertia: It is the distance of each point in the cluster from the cluster centroid. In this case, the less the inertia, the better the cluster is, since it signifies the coherence of the cluster [17].

B. Empirical Results

We conducted two experiments using k-means and mini batch k-means clustering algorithms. Table 1 shows the accuracy of the algorithms.

TABLE 1. ACCURACY RESULTS OF THE EXPERIMENTS

	K-means	Mini Batch K-means
Accuracy	0.48	0.62

It is evident that the accuracy of mini batch k-means is higher than the k-means algorithm. The accuracy of 0.62 for the mini batch k-means is interpreted as correctly categorizing 62 out of 100 instances of data, which is improved compare to 0.48 in the k-means algorithm.

Furthermore, we calculated the area under the ROC curve (AUC) for the two algorithms. Fig.2 and Fig.3 show the ROC curve and the value of AUC.

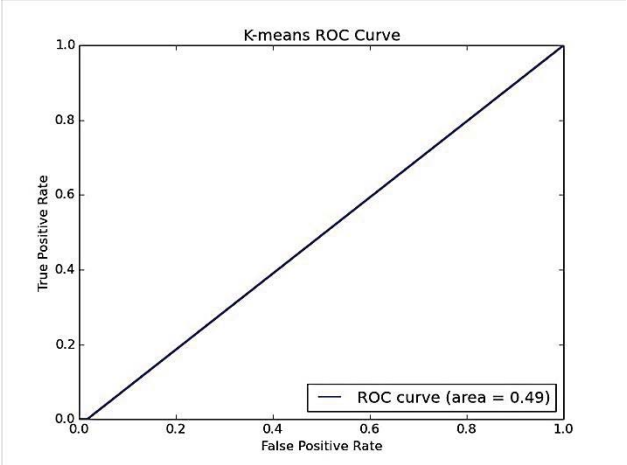

Fig.2. K-means ROC Curve

As the value of AUC goes up, the performance of the algorithm becomes better. Thus, the performance of the mini batch k-means algorithm is better than the k-means algorithm.

C. Additional Results

In addition to the accuracy and the ROC curve, we calculated several evaluation metrics, specifically for evaluating clustering algorithms. In order to use the algorithms, they require input parameters like init and n_init.

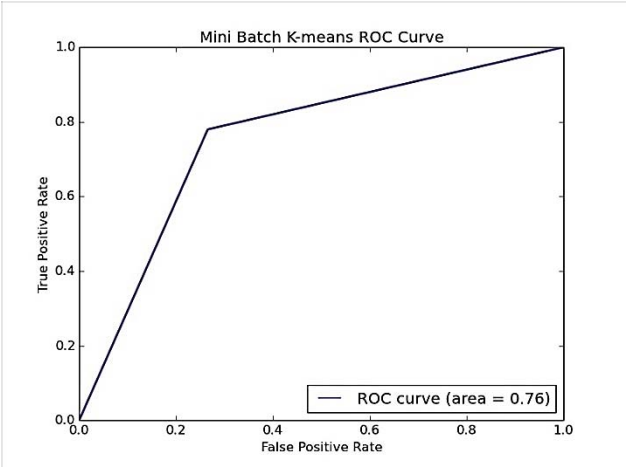

Fig.3. Mini Batch K-means ROC Curve

The init parameter was set to k-means++ which lets the algorithm selects the center of the clusters in a smart way to speed up the clustering process [18].

The n_init is the number of random initialization that the algorithm performs [18]. In experiments, we tested the algorithms for 5, 10, 15 and 20 as the value of n_init to evaluate the performance of the algorithms. Fig 4 shows the results from two experiments based on the n_init against the inertia.

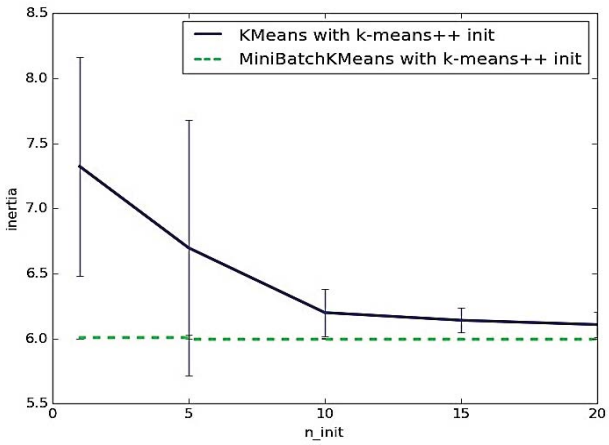

Fig 4. Evaluation of the Experiment based on n_init against inertia

As mentioned earlier, the less the inertia value, the better it performs. Based on the figure above, the green line, which is the mini batch k-means algorithm, performed better than the k-means algorithm. Additionally, it is inferred that as the value of n_init goes up, the performance of k-means algorithms becomes better.

Furthermore, we calculated the values of homogeneity, completeness and v-measure for the two experiments. It is worth reiterating that as their value approaches to 1.0, the algorithm has better result. The homogeneity value for k-means algorithm is 0.08 whereas it is 0.13 for the mini batch k-means algorithm. The value of completeness is 0.16 for k-means and 0.18 for the mini batch k-means algorithm. The values of v-measure are 0.11 and 0.15 for k-means and mini batch k-means algorithms respectively.

Additionally, we calculated the training time for the two clustering algorithms. As expected, the mini batch k-means clustering algorithm performed faster than the k-means algorithm. The time plays a crucial role in real time systems. Fig.5 shows the training time in two algorithms.

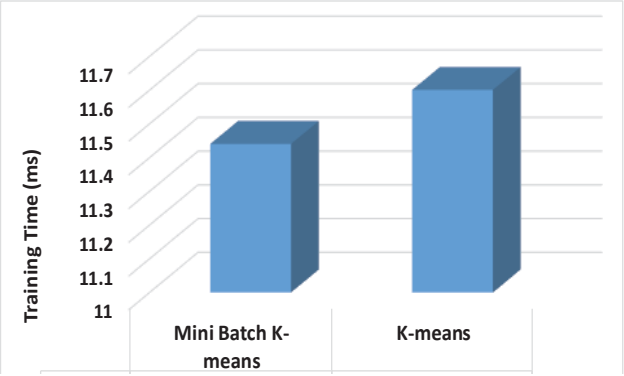

Fig.5. Comparison of Training Time

Overall, the results of mini batch k-means were better than k-means algorithm in terms of performance and training time.

VI. Conclusion

This paper evaluated the performance of two clustering algorithms namely k-means and mini batch k-means for the Android malware detection using the collected network traffic. The network traffic generated by the applications was captured on the mobile devices. The Android applications were collected from the MalGenome data sample. The results show that the mini batch k-means clustering outperformed the k-means clustering algorithm. Additionally, it shows that the analysis of network traffic of applications is effective in Android malware detection.

Acknowledgment

This work was supported in part by the Ministry of Higher Education, Malaysia, under Grant FRGS FP034-2012A and the Ministry of Science, Technology and Innovation, under Grant eScienceFund 01-01-03-SF0914.

References

[1] ZDNET (2013), "History rhymes: Android dominates smartphones like Windows dominated PCs", Available at: http://www.zdnet.com/history-rhymes-android-dominates-smartphones-like-windows-dominated-pcs-7000019130/ (Accessed: 1st March 2014).

[2] Gizmodo (2013), "Android Is Popular Because It's Cheap, Not Because It's Good", Available at: http://gizmodo.com/5977625/android-is-popular-because-its-cheap-not-because-its-good (Accessed: 1st March 2014).

[3] Symantec (2014), "The Future of Mobile Malware", Available at: http://www.symantec.com/connect/blogs/future-mobile-malware (Accessed: 1st March 2014).

[4] Symantec (2013), "Android Madware and Malware Trends", Available at: http://www.symantec.com/connect/blogs/android-madware-and-malware-trends (Accessed: 1st November 2013).

[5] Yajin Z and Xuxian J (2012), "Dissecting Android Malware: Characterization and Evolution", Proceedings of the 2012 IEEE Symposium on Security and Privacy (SP), San Fransico, USA, pp. 95-109.

[6] Feizollah A, Anuar NB, Salleh R, Amalina F, Ma'arof RuR and Shamshirband S (2013), "A Study Of Machine Learning Classifiers for Anomaly-Based Mobile Botnet Detection", Malaysian Journal of Computer Science, Vol. 26 No. 4, pp. 251-265.

[7] Samra AAA, Kangbin Y and Ghanem OA (2013), "Analysis of Clustering Technique in Android Malware Detection", Proceedings of the 2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), Taichung, Taiwan, pp. 729-733.

[8] Sahs J and Khan L (2012), "A Machine Learning Approach to Android Malware Detection", Proceedings of the 2012 European Intelligence and Security Informatics Conference (EISIC), Odense, Denmark, pp. 141-147.

[9] Allen FE (1970), "Control flow analysis", SIGPLAN Not., Vol. 5 No. 7, pp. 1-19.

[10] Lu L, Li Z, Wu Z, Lee W and Jiang G (2012), "CHEX: statically vetting Android apps for component hijacking vulnerabilities", Proceedings of the the 2012 ACM conference on Computer and communications security, Raleigh, North Carolina, USA, pp. 229-240.

[11] Cisco (2012), "OSI (Open Source Interconnection) 7 Layer Model", Available at: https://learningnetwork.cisco.com/docs/DOC-15624 (Accessed: 1st March 2014).

[12] Shameem MUS and Ferdous R (2009), "An efficient k-means algorithm integrated with Jaccard distance measure for document clustering", Proceedings of the First Asian Himalayas International Conference on Internet, 2009, Kathmandu, Nepal, pp. 1-6.

[13] Sculley D (2010), "Web-scale k-means clustering", Proceedings of the 19th international conference on world wide web, Raleigh, North Carolina, USA, pp. 1177-1178.

[14] Python (2014), "python", Available at: www.python.org (Accessed: 1st March 2014).

[15] Siless V, Medina S, Varoquaux G and Thirion B (2013), "A Comparison of Metrics and Algorithms for Fiber Clustering", Proceedings of the 2013 International Workshop on Pattern Recognition in Neuroimaging (PRNI), Philadelphia, USA, pp. 190-193.

[16] Rosenberg A and Hirschberg J (2007), "V-Measure: A conditional entropy-based external cluster evaluation measure", Proceedings of the Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Prague, Czech Republic, pp. 410–420.

[17] Li SX and Rowley TJ (2002), "Inertia and evaluation mechanisms in interorganizational partner selection: Syndicate formation among US investment banks", Academy of Management Journal, Vol. 45 No. 6, pp. 1104-1119.

[18] Scikit-learn (2013), "sklearn.cluster.MiniBatchKMeans", Available at: http://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html (Accessed: 1st March 2014).