Hindawi
Complexity
Volume 2019, Article ID 6101697, 10 pages
https://doi.org/10.1155/2019/6101697

Research Article

Delving into Android Malware Families with a Novel Neural Projection Method

Rafael Vega Vega,[1] **Héctor Quintián**,[1] **Carlos Cambra**,[2] **Nuño Basurto**,[2] **Álvaro Herrero**,[2] **and José Luis Calvo-Rolle**[1,3]

[1]*University of A Coruña, Departamento de Ingeniería Industrial, Avda. 19 de febrero s/n, 15495, Ferrol, A Coruña, Spain*
[2]*Grupo de Inteligencia Computacional Aplicada (GICAP), Departamento de Ingeniería Civil, Escuela Politécnica Superior, Universidad de Burgos, Av. Cantabria s/n, 09006, Burgos, Spain*
[3]*Research Institute of Applied Sciences in Cybersecurity (RIASC), Spain*

Correspondence should be addressed to Rafael Vega Vega; rafael.alejandro.vega.vega@udc.es

Received 5 December 2018; Accepted 23 January 2019; Published 2 June 2019

Guest Editor: Alicja Krzemień

Present research proposes the application of unsupervised and supervised machine-learning techniques to characterize Android malware families. More precisely, a novel unsupervised neural-projection method for dimensionality-reduction, namely, Beta Hebbian Learning (BHL), is applied to visually analyze such malware. Additionally, well-known supervised Decision Trees (DTs) are also applied for the first time in order to improve characterization of such families and compare the original features that are identified as the most important ones. The proposed techniques are validated when facing real-life Android malware data by means of the well-known and publicly available Malgenome dataset. Obtained results support the proposed approach, confirming the validity of BHL and DTs to gain deep knowledge on Android malware.

1. Introduction and Previous Work

Undoubtedly, smartphones are one of the emerging technologies that have revolutionized the use of computing systems. From the very beginning (late 1990s), more and more smartphones are sold every year and it is expected that the number of smartphone users passes the 2.7 billion mark by 2019 [1]. Although there is a variety of operating systems for such devices, Google's Android is the most widely used one [1] and, consequently, the number of Android users has permanently increased. Concurrently, the number of Android apps strongly increased in the last years but it started to decline from 3.6 million in March, 2017 (highest value), to 2.6 million in September, 2018 [2].

From the security standpoint, one of the main problems of smartphone apps is malware that is included in software in general and in these apps in particular. Furthermore, "users of mobile devices are increasingly subject to malicious activity pushing malware apps" [3]. It is true that some effort has been devoted by Google to remove and prevent malicious apps from Google Play Market, but malware is still there [3]. Moreover, malware Android apps are increasing; in the third trimester of 2018 there has been an increase of 1.7 million detections [4].

As it can be seen, privacy and security of smartphones still are open challenges [5] and many researchers are working on this topic. To better fight against malware and be able to develop an effective solution, understanding it and its nature is required [6]. In keeping with this idea, present paper proposes getting deeper knowledge about Android malware for its better detection. More precisely, both supervised (Decision Trees) and unsupervised (Neural Projection Method) machine-learning techniques are applied to increase our knowledge about the main families of Android malware. In order to validate the proposed techniques, they are applied to the well-known Malgenome dataset [7] that is open and real-life.

This pioneering work on collecting Android malware found some interesting statistics [6] motivating further analysis of malware: 36.7% of the collected apps leverage root-level

exploits to fully compromise the security of the smartphone and more than 90% of the apps tried to turn the smartphone into a botnet controlled through network or short messages.

To improve present knowledge of Android malware families, a novel neural-projection technique from the family of Exploratory Projection Pursuit (EPP) techniques, named Beta Hebbian Learning (BHL) [8], is applied. Obtained results are then compared to those from several different Decision Trees (DTs) [9] when trying to predict the malware family from apps features.

Each app (data sample) that was collected for the Malgenome dataset is defined as a set of certain features using a binary representation. Apps were grouped according to the family they belong to, and features were recalculated for the whole family, taking into account which features were present in the given apps. The generated high-dimensional space is then analysed by means of BHL in order to reveal the inner structure of the dataset. Obtained projections are consequently scrutinized to get further knowledge about the app features that define the organization of the data in different groups and subgroups. For comparison purposes, DTs have been additionally generated on the same features set, in order to know the features that better discriminate between the different malware families.

A variety of problems have been addressed by artificial neural networks in recent decades [10–14]. More precisely, neural projection models have been previously applied to a wide variety of security datasets, including network traffic [15, 16], SQL code [17, 18], and HTTP traffic [19]. Similarly, from a more general perspective, different machine learning solutions have been proposed to differentiate between legitimate and malicious apps [20–22].

Visualization techniques have been previously applied to this problem of analyzing malware [23–29]. However, few dimensionality-reduction techniques have been applied to Android apps in order to detect malware; Pythagoras tree fractal visualization is proposed in [25], being all apps scattered, as leaves in the tree. Graphs for deciding about malicious apps by depicting lists of malicious methods, needless permissions, and malicious strings were proposed in [26]. Biclustering on permission information was used to generate visualization in [27], while behavior-related dendrograms are generated out of malware traces in [28]. In the later, different pieces of information are analysed, including nodes related to the package name of the application, the Android components that has called the API call, and the names of functions and methods invoked by the application. Differentiating from previous work, in present paper, a novel neural projection technique is applied for the first time to the characterization of Android malware [8, 24, 30]. Apps are not analysed one by one, but family-level is considered instead. Additionally, DTs are applied for the first time in order to improve characterization of such families.

The rest of this paper is organized as follows; initially BHL and DTs are presented and the analyzed dataset is described in the following section. Then, the proposed experiments are introduced and the obtained results are analyzed in Section 3. Finally, conclusions and future work are presented in Section 4 of the paper.

2. Materials and Methods

In present research, the EPP BHL algorithm [8] has been applied to a dataset of malware families with the aim of identifying the internal structure of such dataset and finding families of malware with similar characteristics. The obtained results have been compared with a well-known prediction algorithm (DT) [9] to validate the BHL results regarding the most relevant features to briefly characterize Malware families.

2.1. Beta Hebbian Learning. The Beta Hebbian Learning technique (BHL) [8] is an unsupervised neural network from the family of EPP that employs the Beta distribution to update its learning rule and fit the Probability Density Function (PDF) of the residual with the distribution of a given dataset.

Thus, if the PDF of the residuals is known, the optimal cost function can be determined. By using $B(\alpha, \beta)$ parameters of the Beta distribution, the residual (e) can be drawn with the following PDF:

$$p(e) = e^{\alpha-1}(1-e)^{\beta-1} = (x - Wy)^{\alpha-1}(1 - x + Wy)^{\beta-1} \quad (1)$$

where α and β are used to adjust the shape of the PDF of the Beta distribution, x is the input of the network, e is the residual, W is the weight matrix, and y is the output of the network.

Then, by using the following, gradient descent is performed to maximize the likelihood of the weights:

$$\begin{aligned}\frac{\partial p}{\partial W} &= \left(e_j^{\alpha-2}\left(1-e_j\right)^{\beta-2}\left(-(\alpha-1)\left(1-e_j\right)+e_j(\beta-1)\right)\right)\\ &= \left(e_j^{\alpha-2}\left(1-e_j\right)^{\beta-2}\left(1-\alpha+e_j(\alpha+\beta-2)\right)\right)\end{aligned} \quad (2)$$

In the case of BHL, the learning rule allows for fitting the PDF of the residual, by maximizing the likelihood of such residual with the current distribution.

Therefore, the neural architecture for BHL is defined as follows:

$$Feedforward: y_i = \sum_{j=1}^{N} W_{ij}x_j, \quad \forall i \quad (3)$$

$$Feedback: e_j = x_j - \sum_{i=1}^{M} W_{ij}y_i \quad (4)$$

$$\begin{aligned}&Weightsupdate: \Delta W_{ij}\\ &= \eta\left(e_j^{\alpha-2}\left(1-e_j\right)^{\beta-2}\left(1-\alpha+e_j(\alpha+\beta-2)\right)\right)y_i\end{aligned} \quad (5)$$

2.2. Decision Trees. Decision Trees (DTs) [9] are machine-learning algorithms widely used for prediction that have proved their benefits in several real applications. They can be categorized as supervised nonparametric inductive learning techniques. They are based on the construction of diagrams from a dataset, in a similar way to prediction systems based

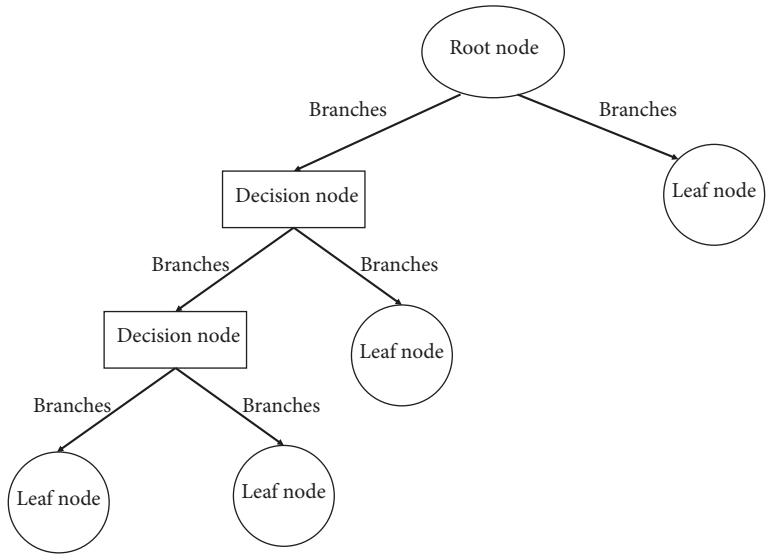

Figure 1: Structure of decision trees.

on rules, which serve to represent and categorize a series of conditions that occur repeatedly for the solution of a problem.

The main objective of a classification DT is to divide a dataset into groups of samples as similar as possible in relation to one of the features. They are made of three main elements: root node (contains all samples of the dataset), decision nodes (represent a decision or rule), and leaf nodes (final label). A dataset is then classified based on subdivisions of the DT nodes to reach one of the final (leaf) nodes whose label corresponds to a class (Figure 1).

Several algorithms have been proposed so far to build DTs and their efficiency has been proved. The most notable ones [31] are ID3 (Iterative Dichotomiser 3), C4.5 (successor of ID3), CART (Classification and Regression Tree), CHAID (CHi-squared Automatic Interaction Detector), MARS, and Conditional Inference Trees. Among all of them, CART has been selected in present work due to two main reasons: the binary nature of the dataset and the main objective of the study (to identify the most relevant features of the dataset) [31].

2.2.1. CART. The Classification and Regression Tree (CART) [9] is a binary tree, so each decision node has two binary branches determined by a splinting function obtained by processing variance function. In order to build the tree, this CART algorithm takes 4 main steps [9]:

(1) Build the decision tree splitting nodes according to a given function.

(2) Finish tree construction once the learning fits the stop criteria.

(3) Pruning the tree to avoid overfitting.

(4) Select the best tree after pruning process.

Originally, the splitting function used by CART is the Gini Index

$$Gini\,(S) = 1 - \sum_{i=1}^{n} p_i^2 \tag{6}$$

where S is the dataset, n is the number of classes in the dataset, and p is the probability of different classes. Therefore, a Gini index of *0* means a *100%* accuracy in predicting the class.

For comparison purposes, two other splitting functions have been applied in present paper: Deviance (7) and Twoing (8)

$$Deviance\,(S) = -\sum_{i=1}^{n} p_i \log_2 p_i \tag{7}$$

Twoing is a splitting function different from Gini and Deviance. Being L_i and R_i there is fraction of members of class i in the left and right child nodes after a split, respectively. P_L and P_R are the fractions of observations that split to the left and right, respectively. Therefore, the function to be maximized is the one in

$$P_L P_R \left(\sum_{i=1}^{n} \left| L_i - Ri \right| \right)^2 \tag{8}$$

On the other hand, in standard CART algorithm, the split feature that is selected for a decision node is the one that maximizes the split-criterion gain. Once again, for a more comprehensive comparison, two other criteria have been applied for selecting split features: curvature [32] and interaction [33]. These criteria can be defined as follows:

(i) Curvature: it is based on the null hypothesis of unassociated two features. With these criteria, the best split predictor feature is the one that minimizes the

significant p-values of curvature tests between each feature and the response variable. Such a selection is robust to the number of levels in individual features.

(ii) Interaction: it is based on the null hypothesis of no interaction between the label and the predictor features. Therefore, for deep decision trees, standard CART tends to miss important interactions between pairs of features when there are also many other less important features. By means of this criterion, the detection of such important interactions is improved.

2.3. Malgenome Dataset. The dataset used in this research has been obtained from the Android Malware Genome Project [7], which consists on 1260 Android malware samples grouped in a total of 49 malware families. It was collected from August 2010 to October 2011 and still is a standard benchmark dataset for Android Malware.

This dataset contains malware apps installed in user phones and based on 3 main attack strategies: repackaging, update attack, and drive-by download. Samples of this dataset were manually classified based on different aspects such as installation and activation mechanisms and malicious payloads nature. Collected malware was split in families that were obtained "by carefully examining the related security announcements, threat reports, and blog contents from existing mobile antivirus companies and active researchers as exhaustively as possible and diligently requesting malware samples from them or actively crawling from existing official and alternative Android Markets" [6].

The different families present in the dataset are ADRD, AnserverBot, Asroot, BaseBridge, BeanBot, BgServ, CoinPirate, Crusewin, DogWars, DroidCoupon, DroidDeluxe, DroidDream, DroidDreamLight, DroidKungFu1, DroidKungFu2, DroidKungFu3, DroidKungFu4, DroidKungFuSapp, DoidKungFuUpdate, Endofday, FakeNetflix, FakePlayer, GamblerSMS, Geinimi, GGTracker, GingerMaster, GoldDream, Gone60, GPSSMSSpy, HippoSMS, Jifake, jSMSHider, Kmin, Lovetrap, NickyBot, Nickyspy, Pjapps, Plankton, RogueLemon, RogueSPPush, SMSReplicator, SndApps, Spitmo, TapSnake, Walkinwat, YZHC, zHash, Zitmo, and Zsone.

Therefore, the final dataset is made of a total of 49 samples, one for each family of malware, defined by a total of 26 binary features divided in 6 categories (Table 1). A detailed description of each feature can be found in the original paper [6], and some previous work where this dataset is used can be found in [34–36].

3. Experiments and Results

This section presents the experiments performed and the results obtained in the validation process of the proposed solution.

Both BHL (Section 2.1) and DT (Section 2.2) algorithms have been applied to the previously described dataset (Section 2.3), in order to identify the features that define the internal structure of the data and that support the grouping of the different families of malware attacks. In the conducted experiments, firstly BHL is applied to identify groups of malware families with similar behaviour. This is done by visually inspecting the obtained BHL projections, and the most relevant features are consequently identified. Then, the dataset is analyzed by means of DTs to determine the level of importance of each feature, considering as the most relevant features those that are used in the decision nodes at lowest depth.

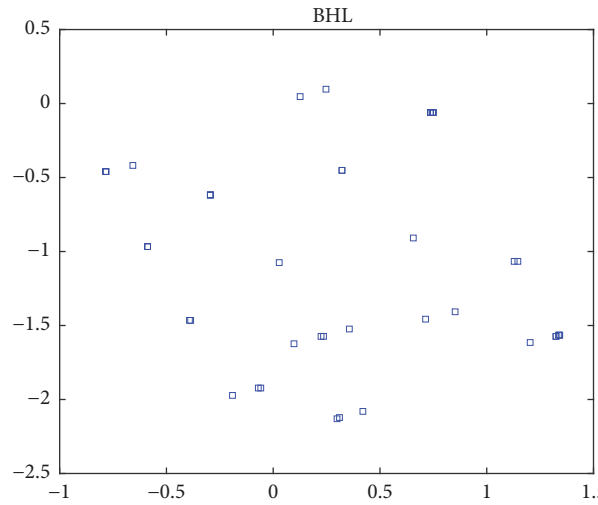

Figure 2: BHL: Projection of malware families.

In Figure 2 it is shown the best projection obtained by BHL using the following parameter values: $\alpha = 3$, $\beta = 4$, $numberofiterations = 1000$, and $learningrate = 0.05$. These parameter values were chosen in an experimental process of trial and error. As parameter tuning is a task that is very dependent on the dataset to be used, several initial experiments were conducted with a range of combinations of these parameter values.

Based on such projection, samples are grouped in 2 main clusters: G1 and G2 (Figure 3). Additionally, several subgroups (at a 3 level depth, i.e., G1 ⟶ G1A ⟶ G1A.1, G1A.2, G1A.3, and G1A.4) can be defined in these main groups.

Figure 4 presents the split of family groups in a schema that shows the results of thoroughly analyzing the allocation of families in groups. The most relevant features that have been identified, varying from one cluster to another, can be seen. As an example, data are split in G1 and G2 based on the features "Repackaging" and "Standalone." The complete lists of families assigned to each one of the groups are presented in Figures 5 and 6. Malware families are allocated in the same group as they are associated to similar characteristics and behaviour, and therefore there could be similar ways to deal with them.

Based on the analysis of BHL results, the most relevant features, in decreasing order of importance, are "Repackaging" and "Standalone," "Boot" and "Activation: SMS," and "Financial Charges: SMS."

BHL clearly outperforms other algorithms used in previous works [24, 29], providing a clearer visualization

Table 1: Features in the Malgenome Dataset.

Category 1: Installation	1.Repackaging, 2.Update, 3.Drive-by download, 4.Standalone
Category 2: Activation	5.Boot, 6.SMS, 7.Net, 8.Call, 9.USB, 10.PKG, 11.Batt, 12.SYS, 13.Main
Category 3: Privilege escalation	14.exploid, 15.RATC/zimperlich, 16.ginger break, 17.asroot, 18.encrypted
Category 4: Remote control	19.NET, 20.SMS
Category 5: Financial charges	21.phone call, 22.SMS, 23.block SMS
Category 6: Personal information stealing	24.SMS, 25.phone number, 26.user account

Table 2: Summary table of DT results: minimum depth of decision nodes for each one of the original features.

		Deviance			Gini			Twoing			
ID	Feature	Standard	Curvature	Interaction curvature	Standard	Curvature	Interaction curvature	Standard	Curvature	Interaction curvature	Average
1	Repackaging	1	2	4	1	2	6	1	2	4	2.56
5	BOOT	2	3	3	4	3	2	2	3	3	2.78
18	Encrypted		4	2		4			4	2	3.20
9	USB	6	3		5	3	4	6	3		4.29
3	Drive-by Download	5	5	4	2	5	6	5	5	4	4.56
24	SMS	4	3	5	10	3	6	3	3	5	4.67
26	User Account	6	1	8	3	1	8	6	1	8	4.67
2	Update	5	7	4	2	6	3	5	7	4	4.78
19	NET	6	2	8	9	2	3	4	2	8	4.89
6	SMS	3	6	5	8	7	3	3	6	4	5.00
10	PKG	6	5		4	5	4	6	5		5.00
22	SMS	3	6	4	10	6	4	3	6	4	5.11
4	Standalone	5	10	3	3	9	3	5	10	3	5.67
8	CALL	4		7	4		8	4		7	5.67
11	BATT	4	9		5	4	5	4	9		5.71
16	Ginger Break				6						6.00
15	RATC/Zimperlich	6	8	1	9	9	8	7	8	1	6.33
7	NET	5	8	6	10	9	2	5	8	6	6.56
14	Exploid	5	8	6	9	6	6	5	8	6	6.56
17	Asroot	7	4	7	11	4	9	8	4	7	6.78
23	Block SMS	3	8	5	9	9	9	4	8	6	6.78
25	Phone Number	2	11	2	12	12	11	2	11	2	7.22
12	SYS	5	10	6	10	11	7	5	10	6	7.78
21	Phone Call	4	9		12	13	7	4	9	5	7.88
13	MAIN	6	11	7	10	12	1	6	11	7	7.89
20	SMS				10		8				9.00

of the internal structure of the dataset. Groups obtained by BHL are more compact and well defined than the groups generated by other EPP techniques in the previous work.

In addition to the BHL experiments, experiments with DTs were additionally conducted in order to compare and validate the obtained results. As it has been previously mentioned, 3 different splitting functions have been applied in present paper: Gini, Deviance, and Twoing. In addition, 3 different criteria for selecting split features have been applied: Standard, Curvature, and Interaction.

As an example, one of the obtained DTs is shown in Figure 7. This is the tree generated from the Malgenome dataset when applying the standard CART split criteria and the Deviance function. It has been selected as it is the one with lowest depth. In the leaf nodes, labels refer to family numbers (alphabetically ordered as presented in Section 2.3).

To show the most interesting results from the different alternatives to build DT, information has been summarized in Table 2. For each combination of splitting function and selecting criteria, the minimum depth of decision nodes linked to each one of the original features is shown. That is,

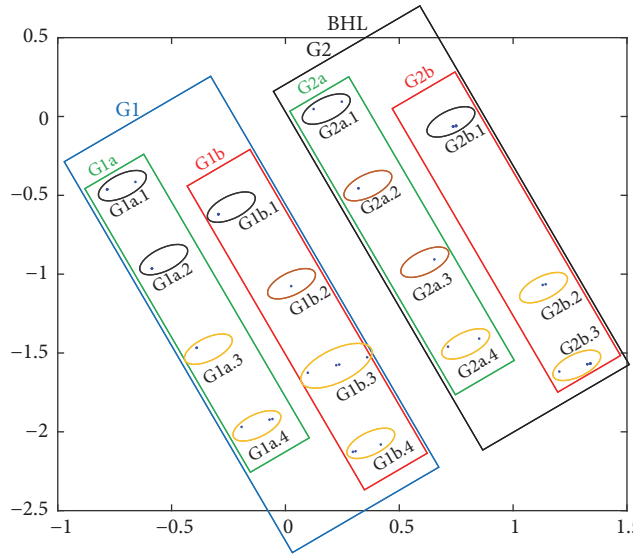

Figure 3: BHL: Labelling of clusters.

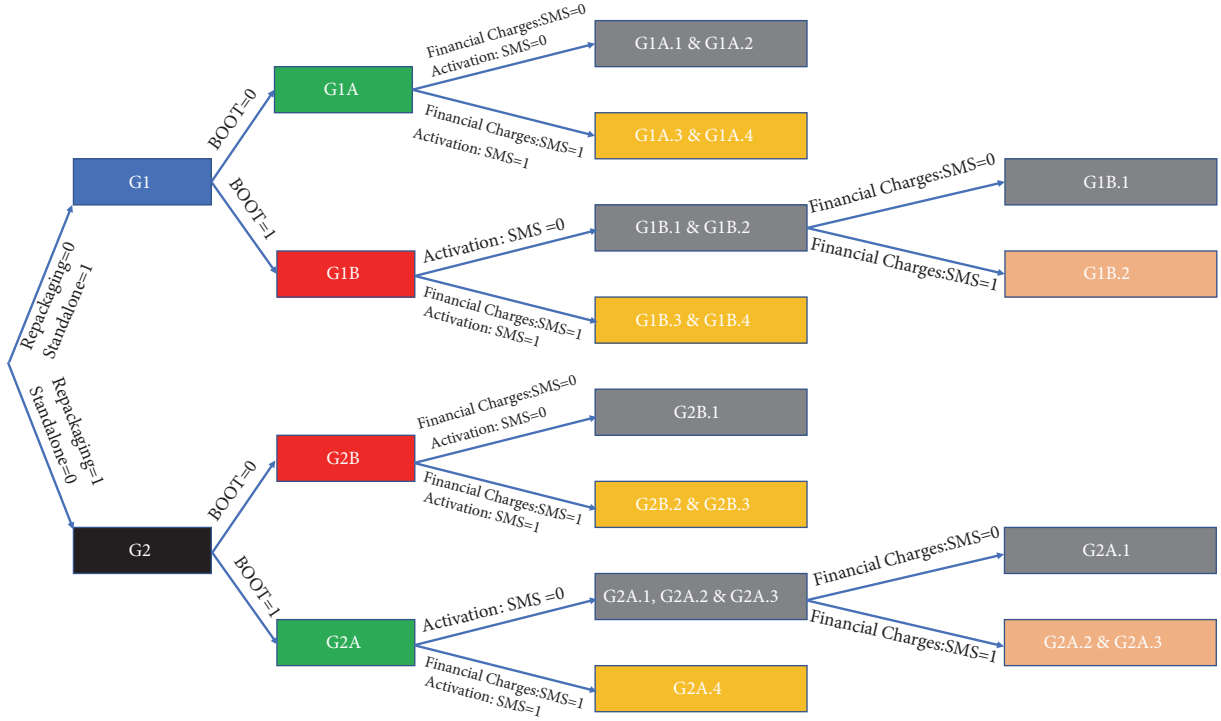

Figure 4: Schematic clustering and relevant features from BHL projection.

when the same feature appears in more than one node, the minimum depth of all these nodes is the one selected for the given feature. In the case a certain feature was not included in the DT, there is no value.

In this table it can be seen that results (slightly or significantly) vary when comparing the obtained results (by different splitting function and selecting criteria) for a certain feature. As general conclusions cannot be derived and to sum up all figures, the average depth value is calculated for each feature, that is, further analyzed.

When analyzing Figure 4 and Table 2, it can be seen that results from BHL and DT are coherent. In the case of BHL, it can be seen that Repackaging is identified as the most discriminative feature, because the two main groups in the dataset (G1 and G2) take complementary values for such feature. Coherently, Repackaging is the feature with the

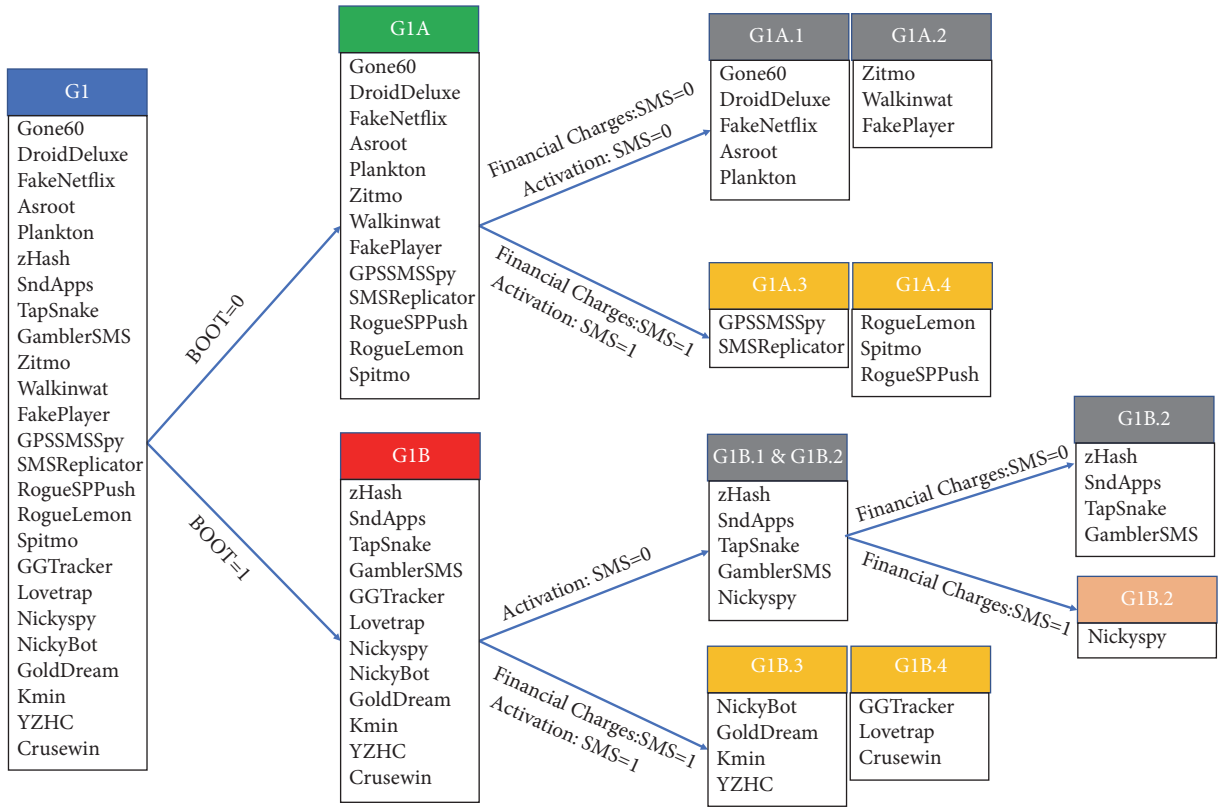

Figure 5: Families allocation in Group 1 and relevant features identified in BHL projection.

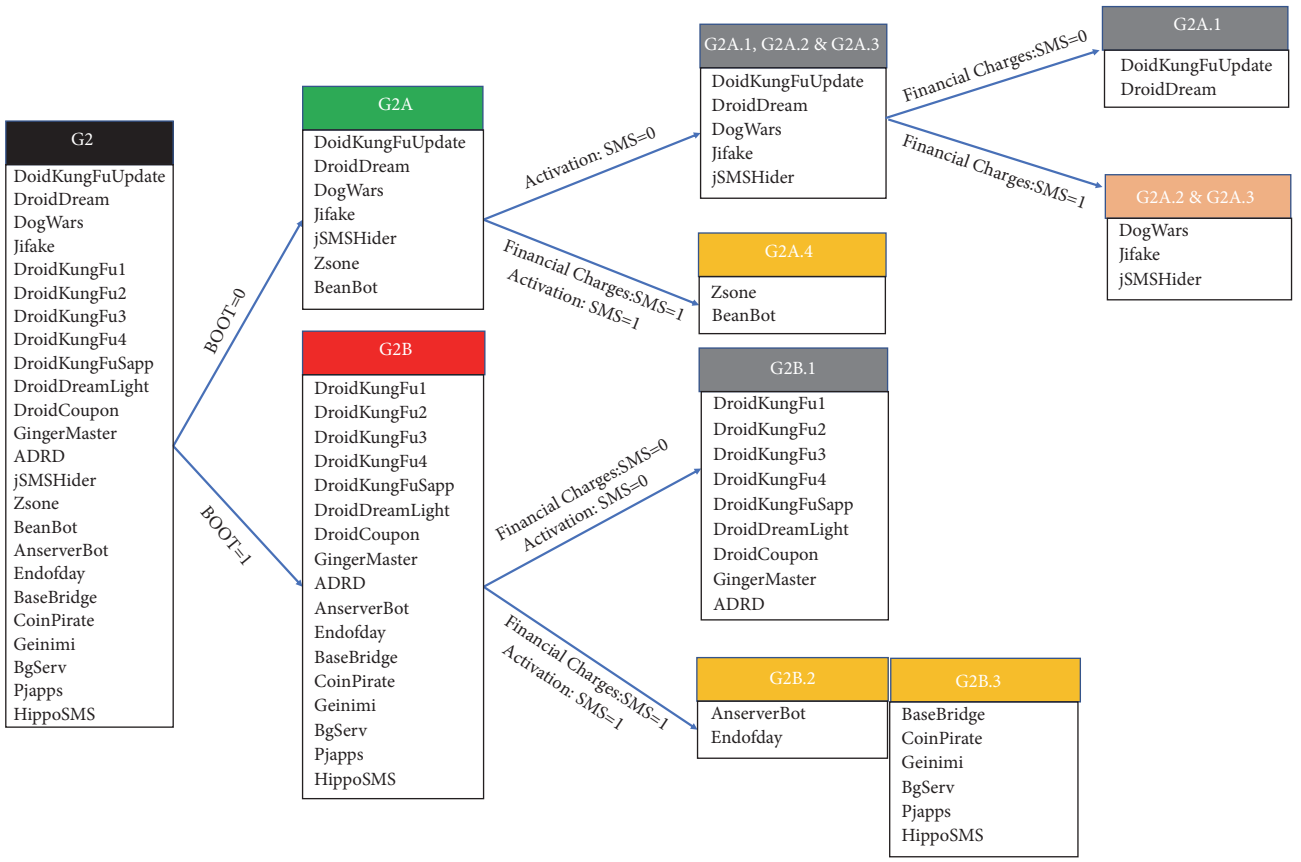

Figure 6: Families allocation in Group 2 and relevant features identified in BHL projection.

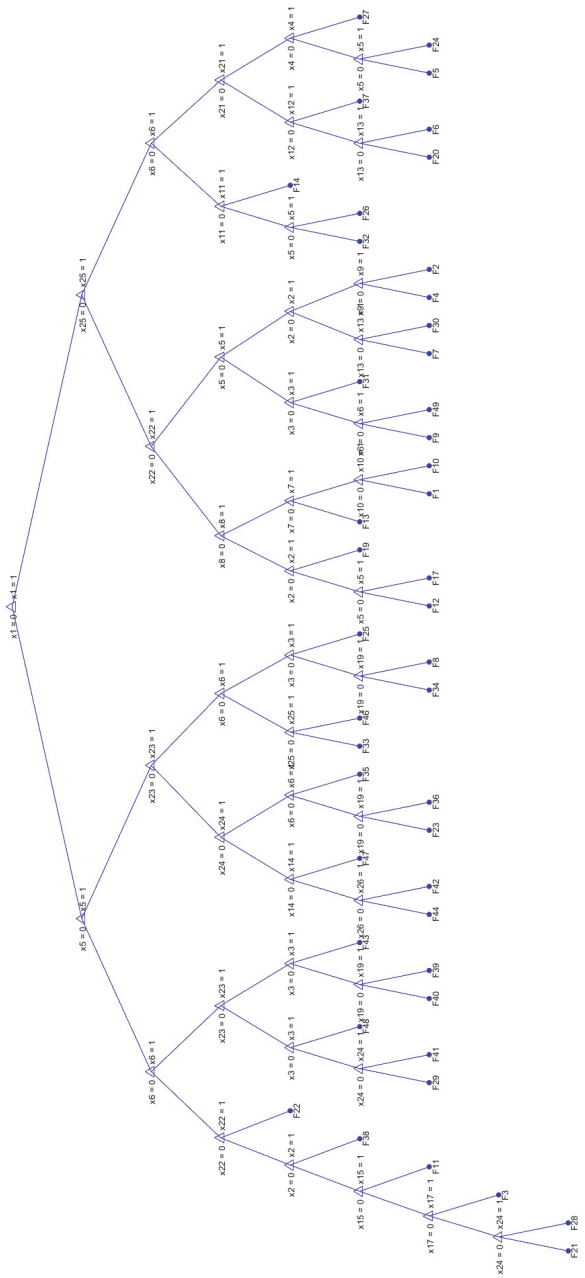

Figure 7: DT obtained with standard CART split criteria and Deviance function.

lowest mean depth, being included in all the generated trees. Furthermore, it was selected for the root node of 3 DTs. When analyzing subgroups in BHL projection (Figure 3), it can be seen that BOOT is the feature that drive the split in 1st-level subgroups (subgroups G1A and G1B in the case of group G1, and subgroups G2A and G2B in the case of group G2). In keeping with this idea, according to DTs results, Boot is the second feature with the lowest mean depth. For the next level of importance, the BHL projection identifies Financial Charges SMS and Activation SMS as the features that best explain the split in different subgroups. The two of them are also selected by DTs as ranked in the first half of features with a lowest mean depth, although some other features take lower values.

Additionally, from the DTs results (Table 2), Privilege escalation-Ginger Break and Remote control-SMS can be identified as the least relevant features. The former was not included in 7 (out of 9) DTs while the latter was not included in 6. Furthermore, Remote control-SMS is the feature with a highest value of the average depth, taking a value of 9. It means that these features are almost useless when characterizing malware families.

Results from present paper are consistent with those obtained in previous work [30] when applying Feature Selection (FS) to the same dataset. Installation-Repackaging, Activation-SMS, Activation-Boot, Remote Control-NET, and Financial Charges-SMS were identified as the 5 most relevant features in order to characterize malware families, according to a given method of filter-based FS: Minimum-Redundancy Maximum Relevance. This method is intended at obtaining the maximum relevance to the output while keeping redundancy of selected features to lowest levels. Complementarily, two evolutionary approaches to FS (GA-ICC-W and GA-I-W) identified Installation-Repackaging, Installation-Standalone, Activation-SMS, Remote Control-NET, and Financial Charges-SMS as the 5 most relevant ones. These methods perform the selection of features according to the Information Correlation Coefficient and the Mutual Information, respectively.

4. Conclusions and Future Work

In this paper, some machine learning techniques have been applied to Android malware data in order to analyse the features of such apps and subsequently identify the ones that better define the organization of malware families. As a result, detection and categorization of malware could be improved and sped up at the same time. Furthermore, by knowing about these features, malware apps could be identified more quickly and precisely and then removed from the official Android market.

From the obtained results some conclusions can be derived; first of all, the proposed machine-learning techniques probed to successfully address the given challenge. BHL has outperformed previous neural projection techniques that have been applied to the same data in clearly revealing the structure of the Malgenome dataset. Additionally, features identified as the most important ones by such EPP technique are also highlighted by DTs as being relevant to better differentiate between malware families.

Obtained results are consistent with those obtained by FS and hence validate present proposal. Future work will focus on the development of a Hybrid Intelligent System to integrate results from the previously validated machine-learning techniques. In addition, it will be applied to up-to-date malware datasets in order to check its performance when facing 0-day malware.

Data Availability

Dataset used in this research is available in [7]: Bibliography: 7. (2010) Malgenome Project [Online], available at http://www.malgenomeproject.org.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is partially supported by Instituto Nacional de Ciberseguridad (INCIBE) and developed by Research Institute of Applied Sciences in Cybersecurity (RIASC).

References

[1] Gartner, Global smartphone sales to end users from 1st quarter 2009, 2018 https://www.statista.com/statistics/266219/global-smartphone-sales-since-1st-quarter-2009-by-operating-system/.

[2] AppBrain, Android and google play statistics, https://www.appbrain.com/stats/stats-index.

[3] SOPHOSLABS, "Ltd., s., sophoslabs 2019 threat report," Tech. Rep., 2019.

[4] M. Labs, "Cybercrime tactics and techniques : Q3 2018," Tech. Rep., 2018.

[5] S. Arshad, M. A. Shah, A. Khan, and M. Ahmed, "Android malware detection & protection: A survey," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 2, 2016.

[6] Y. Zhou and X. Jiang, "Dissecting android malware: characterization and evolution," in *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, pp. 95–109, San Francisco, Calif, USA, May 2012.

[7] Y. Zhou, Malgenome project. 2010 http://www.malgenome-project.org.

[8] H. Quintían and E. Corchado, "Beta hebbian learning as a new method for exploratory projection pursuit," *International Journal of Neural Systems*, vol. 27, no. 6, Article ID 1750024, 2017.

[9] L. Breiman, *Classification and regression trees*, Routledge, London, UK, 2017.

[10] P. J. García Nieto, J. Martínez Torres, F. J. De Cos Juez, and F. Sánchez Lasheras, "Using multivariate adaptive regression splines and multilayer perceptron networks to evaluate paper manufactured using Eucalyptus globulus," *Applied Mathematics and Computation*, vol. 219, no. 2, pp. 755–763, 2012.

[11] M. Paliwal and U. A. Kumar, "Neural networks and statistical techniques: a review of applications," *Expert Systems with Applications*, vol. 36, no. 1, pp. 2–17, 2009.

[12] R. F. Garcia, J. L. C. Rolle, M. R. Gomez, and A. D. Catoira, "Expert condition monitoring on hydrostatic self-levitating bearings," *Expert Systems with Applications*, vol. 40, no. 8, pp. 2975–2984, 2013.

[13] C. C. Turrado, M. D. C. M. López, F. S. Lasheras, B. A. R. Gómez, J. L. C. Rollé, and F. J. D. C. Juez, "Missing data imputation of solar radiation data under different atmospheric conditions," *Sensors*, vol. 14, no. 11, pp. 20382–20399, 2014.

[14] J. L. Calvo Rolle, I. Machón González, and H. López García, "Neuro-robust controller for non-linear systems," *Dyna*, vol. 86, no. 3, pp. 308–317, 2011.

[15] Á. Herrero, E. Corchado, M. A. Pellicer, and A. Abraham, "Hybrid multi agent-neural network intrusion detection with mobile visualization," in *Innovations in Hybrid Intelligent Systems*, pp. 320–328, 2008.

[16] R. Sánchez, Á. Herrero, and E. Corchado, "Visualization and clustering for SNMP intrusion detection," *Cybernetics and Systems*, vol. 44, no. 6-7, pp. 505–532, 2013.

[17] C. Pinzón, Á. Herrero, J. F. De Paz, E. Corchado, and J. Bajo, "CBRid4SQL: A CBR intrusion detector for SQL injection

attacks," in *Proceedings of the 5th International Conference on Hybrid Artificial Intelligence Systems HAIS 2010 - Part II*, vol. 6077 of *Lecture Notes in Computer Science*, pp. 510–519, Springer Berlin Heidelberg, 2010.

[18] C. Pinzón, J. F. De Paz, J. Bajo, Á. Herrero, and E. Corchado, "AIIDA-SQL: An adaptive intelligent intrusion detector agent for detecting SQL injection attacks," in *Proceedings of the 2010 10th International Conference on Hybrid Intelligent Systems, HIS 2010*, pp. 73–78, USA, August 2010.

[19] D. Atienza, Á. Herrero, and E. Corchado, "Neural Analysis of HTTP Traffic for Web Attack Detection," in *Proceedings of the 8th International Conference on Computational Intelligence in Security for Information Systems CISIS 2015*, vol. 369 of *Advances in Intelligent Systems and Computing*, pp. 201–212, Springer International Publishing, 2015.

[20] L. Cen, C. S. Gates, L. Si, and N. Li, "A probabilistic discriminative model for android malware detection with decompiled source code," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 4, pp. 400–412, 2015.

[21] H.-J. Zhu, Z.-H. You, Z.-X. Zhu, W.-L. Shi, X. Chen, and L. Cheng, "DroidDet: effective and robust detection of android malware using static analysis along with rotation forest model," *Neurocomputing*, vol. 272, pp. 638–646, 2018.

[22] F. A. Narudin, A. Feizollah, N. B. Anuar, and A. Gani, "Evaluation of machine learning classifiers for mobile malware detection," *Soft Computing*, vol. 20, no. 1, pp. 343–357, 2016.

[23] M. Wagner, F. Fischer, R. Luh et al., "A Survey of Visualization Systems for Malware Analysis," in *Proceedings of the Eurographics Conference on Visualization (EuroVis) - STARs*, 2015.

[24] A. González, Á. Herrero, and E. Corchado, "Neural visualization of android malware families," in *International Joint Conference SOCO'16-CISIS'16-ICEUTE'16*, vol. 527 of *Advances in Intelligent Systems and Computing*, pp. 574–583, Springer International Publishing, Cham, 2017.

[25] A. Paturi, M. Cherukuri, J. Donahue, and S. Mukkamala, "Mobile malware visual analytics and similarities of Attack Toolkits (Malware gene analysis)," in *Proceedings of the 2013 International Conference on Collaboration Technologies and Systems, CTS 2013*, pp. 149–154, USA, May 2013.

[26] W. Park, K. Lee, K. Cho, and W. Ryu, "Analyzing and detecting method of Android malware via disassembling and visualization," in *Proceedings of the 2014 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 817-818, Busan, South Korea, October 2014.

[27] V. Moonsamy, J. Rong, and S. Liu, "Mining permission patterns for contrasting clean and malicious android applications," *Future Generation Computer Systems*, vol. 36, pp. 122–132, 2014.

[28] O. Somarriba, U. Zurutuza, R. Uribeetxeberria, L. Delosières, and S. Nadjm-Tehrani, "Detection and visualization of android malware behavior," *Journal of Electrical and Computer Engineering*, vol. 2016, Article ID 8034967, p. 17, 2016.

[29] R. Vega Vega, H. Quintián, J. L. Calvo-Rolle, Á. Herrero, and E. Corchado, "Gaining deep knowledge of Android malware families through dimensionality reduction techniques," *Logic Journal of the IGPL*, 2018.

[30] J. Sedano, S. González, C. Chira, Á. Herrero, E. Corchado, and J. R. Villar, "Key features for the characterization of Android malware families," *Logic Journal of the IGPL*, vol. 25, no. 1, pp. 54–66, 2017.

[31] S. Singh and P. Gupta, "Comparative study id3, cart and c4. 5 decision tree algorithm: a survey," *International Journal of Advanced Information Science and Technology*, vol. 27, no. 7, pp. 97–103, 2014.

[32] W.-Y. Loh and Y.-S. Shih, "Split selection methods for classification trees," *Statistica Sinica*, vol. 7, no. 4, pp. 815–840, 1997.

[33] W.-Y. Loh, "Regression trees with unbiased variable selection and interaction detection," *Statistica Sinica*, vol. 12, no. 2, pp. 361–386, 2002.

[34] L. Li, A. Bartel, T. F. Bissyande et al., "IccTA: Detecting Inter-Component Privacy Leaks in Android Apps," in *Proceedings of the 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE)*, vol. 1, pp. 280–291, Florence, Italy, May 2015.

[35] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, "Drebin: effective and explainable detection of android malware in your pocket," in *Proceedings of the 2014 Network and Distributed System Security (NDSS) Symposium*, vol. 14, pp. 23–26, 2014.

[36] G. Suarez-Tangil, S. K. Dash, M. Ahmadi, J. Kinder, G. Giacinto, and L. Cavallaro, "Droidsieve: Fast and accurate classification of obfuscated android malware," in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy (CODASPY)*, pp. 309–320, Scottsdale, Arizona, USA, 2017.