Gaining Deep Knowledge of Android Malware Families through Dimensionality Reduction Techniques

Rafael Vega Vega[1], Héctor Quintián[1], José Luís Calvo-Rolle[1], Álvaro Herrero[2], and Emilio Corchado[3]

[2]Department of Industrial Engineering, University of Coruña, Spain
Avda. 19 de febrero S/N 15.405, Ferrol - Coruña, Spain
ravega@udc.es, hector.quintian@udc.es, jlcalvo@udc.es

[2]Department of Civil Engineering, University of Burgos, Spain
Avenida de Cantabria s/n, 09006 Burgos, Spain
ahcosio@ubu.es

[3]Departamento de Informática y Automática, Universidad de Salamanca
Plaza de la Merced, s/n, 37008 Salamanca, Spain
escorchado@usal.es

Abstract.
This research proposes the analysis and subsequent characterization of Android malware families, by means of low dimensional visualizations using dimensional reduction techniques. The well-known Malgenome dataset, coming from the Android Malware Genome Project, has been thoroughly analysed through six dimensionality reduction techniques: Principal Component Analysis, Maximum Likelihood Hebbian Learning, Cooperative Maximum Likelihood Hebbian Learning, Curvilinear Component Analysis, Isomap and Self Organizing Map. Results obtained enable a clear visual analysis of the structure of this high-dimensionality dataset, letting us gain deep knowledge about the nature of such Android malware families. Interesting conclusions are obtained from the real-life dataset under analysis.

Keywords: Android Malware, Malware Families, Dimensionality Reduction, Artificial Neural Networks

1 Introduction

Since the first smartphones came onto the market in the late 1990s, sales on that sector have increased constantly to the present-day. Among all the available operating systems, Google's Android has been, and increasingly is, the most popular mobile platform [1]. The number of Android units sold in Q1 2017 worldwide raised to 379.98 million out of 432.79 million units, that is a share of 87.79%. It is not only the number of devices but also the number of apps; those available at Google Play (Android's official store) constantly increase, up to more than 3.4 million that are available nowadays [2]. With regard to the security issue, the number of malicious Android apps has greatly risen in the last four years; from the half million of them that were identified in 2013 to the nearly 3.5 million in 2017 [3] . Furthermore, it has been forecast that increase in malware for Android devices is expected to continue [3], [4]. This operating system is an appealing target for bad-intentioned apps, mainly because of its open mentality, in contrast to iOS or some other operating systems.

Smartphone security and privacy still are nowadays major concerns although great efforts have been devoted over past years [5]. In order to address these issues, it is required to understand the malware and its nature. Otherwise, it will not be possible to practically develop an effective solution [6]. According to this idea of gaining deeper knowledge about malware nature, present study is focused on the analysis of Android malware families. To do so, Malgenome (a real-life publicly-available) dataset [7] has been analyzed by means of several Dimensionality Reduction Techniques (DRTs). From the samples contained in such dataset, several alarming statistics were found [6], that motivate further research on Android malware. That is the case of the 36.7% of the collected samples that leverage root-level exploits to fully compromise the security of the whole system or the fact that more than 90% of the samples turn the compromised phones into a botnet controlled through network or short messages.

To characterize malware families, present study proposes a comprehensive comparison of many DRTs, that are able to visualize a high-dimensionality dataset (further described in section 2), to gain deep

knowledge of Android malware families. Each individual from the Malgenome dataset (a malware app) encodes the subset of selected features by using a binary representation (details on section 3). These individuals are grouped by families and then visualized trying to identify patterns that exist across dimensional boundaries in the high dimensional dataset by changing the spatial coordinates of malware family data. The main goal is to obtain an intuitive visualization of the malware families to draw conclusions about the structure of the dataset and to characterize malware families subsequently.

Neural networks have been applied to a wide variety of fields in recent decades [8]–[12]; additionally, neural DRTs have been previously applied to massive security datasets, such as those generated by network traffic [13], [14], SQL code [15], [16], honeynets [17], and HTTP traffic [18]. In present paper, such methods are applied to a new problem, related to the characterization and knowing of malware families. On the other hand, several different techniques have been used to differentiate between legitimate and malicious Android apps, such as machine learning [19]–[21], knowledge discovery [22], and weighted similarity matching of logs [23], among others as well as hybridization approaches [24]. Although some visualization techniques have been applied to the detection of malware in general terms [25], few dimensionality-reduction proposals for Android malware detection are available at present time. In [26] Pythagoras tree fractal is used to visualize the malware data, being all apps scattered, as leaves in the tree. Authors of [27] proposed graphs for deciding about malware by depicting lists of malicious methods, needless permissions and malicious strings. In [28], visualization obtained from biclustering on permission information is described. Behavior-related dendrograms are generated out of malware traces in [29], comprising 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. Unlike previous work, Android malware families (instead of malware apps) are visualized by DRTs in present paper. Up to the authors knowledge, this is the first time that dimensionality-reduction models are applied to visualize Android malware.

The rest of this paper is organized as follows: the applied neural methods are described in section 2, the setup of experiments for the Android Malware Genome dataset is described in section 3, together with the results obtained and the conclusions of the study that are stated in section 4.

2 Dimensionality Reduction Techniques

This work proposes the application of several DRTs for the visualization of Android malware data. Visualization techniques are considered a viable approach to information seeking, as humans are able to recognize different features and to detect anomalies by means of visual inspection [30]. The underlying operational assumption of the proposed approach is mainly grounded in the ability to render the high-dimensional traffic data in a consistent yet low-dimensional representation [17], [18], [25]. In most cases, security visualization tools have to deal with massive datasets with a high dimensionality, to obtain a low-dimensional space for presentation [13], [15], [17], [18], [31], [32].

This problem of identifying patterns that exist across dimensional boundaries in high dimensional datasets can be solved by changing the spatial coordinates of data. Projection methods project high-dimensional data points onto a lower dimensional space in order to identify "interesting" directions in terms of any specific index or projection. Having identified the most interesting projections, the data are then projected onto a lower dimensional subspace plotted in two or three dimensions, which makes it possible to examine the structure with the naked eye [30].

2.1 Principal Component Analysis

Principal Component Analysis (PCA) is a well-known statistical model, introduced in [33], that describes the variation in a set of multivariate data in terms of a set of uncorrelated variables each, of which is a linear combination of the original variables. From a geometrical point of view, this goal mainly consists of a rotation of the axes of the original coordinate system to a new set of orthogonal axes that are ordered in terms of the amount of variance of the original data they account for.

PCA can be performed by means of neural models such as those described in [34] or [35]. It should be noted that even if we are able to characterize the data with a few variables, it does not follow that an interpretation will ensue.

2.2 Maximum Likelihood Hebbian Learning

Maximum Likelihood Hebbian Learning [30] which is based on Exploration Projection Pursuit (EPP). The statistical method of EPP [30], [36], [37] was designed for solving the complex problem of identifying structure in high dimensional data by projecting it onto a lower dimensional subspace in which its structure is searched for by eye. To that end, an "index" must be defined to measure the varying degrees of interest associated with each projection. Subsequently, the data is transformed by maximizing the index and the associated interest. From a statistical point of view the most interesting directions are those that are as non-Gaussian as possible.

2.3 Cooperative Maximum Likelihood Hebbian Learning

The Cooperative MLHL (CMLHL) model [38] extends the MLHL model, by adding lateral connections between neurons in the output layer of the model. Considering an N-dimensional input vector (x), and an M-dimensional output vector (y), with W_{ij} being the weight (linking input neuron j to output neuron i), then CMLHL can be expressed as defined in equations 1-4.

1. Feed-forward step:

$$y_i = \sum_{j=1}^{\mathbf{N}} W_{ij} x_j, \forall i \tag{1}$$

2. Lateral activation passing:

$$y_i(t+1) = \left[y_i(t) + \tau(b - Ay) \right]^+ \tag{2}$$

3. Feedback step:

$$e_j = x_j - \sum_{i=1}^{M} W_{ij} y_i, \forall j \tag{3}$$

4. Weight change:

$$\Delta W_{ij} = \eta . y_i . sign(e_j) | e_j |^{p-1} \tag{4}$$

Where: η is the learning rate, τ is the "strength" of the lateral connections, b the bias parameter, p a parameter related to the energy function and A a symmetric matrix used to modify the response to the data. The effect of this matrix is based on the relation between the distances separating the output neurons.

2.4 ISOMAP Algorithm

ISOMAP nonlinear DRT [39] attempts to preserve pairwise geodesic (or curvilinear) distance between data points. Geodesic distance is the distance between two points measured over the manifold. ISOMAP defines the geodesic distance as the sum of edge weights along the shortest path between two nodes. The doubly-centered geodesic distance matrix K in ISOMAP is of the form given by equation 5.

$$K = \frac{1}{2} H \ ^2 H \tag{5}$$

Where $D^2=D^2_{ij}$ means the element wise square of the geodesic distance matrix $D=[D_{ij}]$, and H is the centring matrix, given by equation 6.

$$H = I_n - \frac{1}{N} e_N e_N^T \tag{6}$$

In which $e_N = [1 \dots 1]^T \in R^N$

The top N eigenvectors of the geodesic distance matrix represent the coordinates in the new n-dimensional Euclidean space.

2.5 Curvilinear Component Analysis Algorithm

Curvilinear Component Analysis (CCA) [40], [41] is a non-linear projection method that preserves distance relationships in both input and output spaces. CCA is a useful method for redundant and non-linear data structure representation and can be used in dimensionality reduction. CCA is useful with highly non-linear data, where PCA or any other linear method fails to give suitable information.

CCA brings some improvements to other methods like Sammon's Mapping [42], although when unfolding a nonlinear structure, Sammon's Mapping cannot reproduce all distances. One way to get round this problem consists in favoring local topology: CCA tries to reproduce short distances firstly, long distances being secondary. Formally, this reasoning led to the following error function (without normalization) defined in equation 7.

$$E_{CCA} = \sum_{i,j=1}^{N} \left(d_{i,j}^{n} - d_{i,j}^{p}\right)^2 F_{\lambda}\left(d_{i,j}^{p}\right) \quad (7)$$

In comparison with E_{Sammon}, E_{CCA} has an additional weighting function F depending on d_{ij}^{p} and on parameter λ. The F factor is a decreasing function of its argument, so it is used to favour local topology preservation. For example, F could be a step function of (λ-d).

2.6 Self Organizing Maps

Among the great variety of tools for multidimensional data visualization, several of the most widely used are those belonging to the family of the topology preserving maps [43]–[48]. Probably the best known among these algorithms is the Self-Organizing Map (SOM) [43], [45], [49], [50]. It is based on a type of unsupervised learning called competitive learning; an adaptive process in which the units in a neural network gradually become sensitive to different input categories or sets of samples in a specific domain of the input space. The main feature of the SOM algorithm is its topology preservation. When not only the winning unit, but also its neighbors on the lattice are allowed to learn, neighboring units gradually specialize to represent similar inputs, and the representations become ordered on the map lattice.

An input vector *(x)* is presented to the network and the node of the network in which the weights (W_i) are closest (in terms of Euclidean distance) to x, is chosen:

$$c = \arg\min(\|x - W_i\|) \quad (8)$$

The weights of the winning node and the nodes close to it are then updated to move closer to the input vector. There is also a learning rate parameter that usually decreases as the training process progresses. The weight update rule for inputs is defined as follows:

$$\Delta W_i = \eta h_{ci}(x - W_i), \quad \forall i \in N \quad (9)$$

Where, W_i is the weight vector associated with neuron i, x is the input vector, and h is the neighborhood function.

3 Experiments & Results

As previously mentioned, several different DRTs (see Section 2) have been applied to analyze Android malware. Present section introduces the analyzed dataset as well as the main obtained results.

3.1 Malgenome Dataset

The Malgenome dataset [6], coming from the Android Malware Genome Project [7], has been analysed in present study. It is the first large collection of Android malware (1,260 samples) that was split in malware families (49 different ones). It covered the majority of existing Android malware, collected from the beginning of the project in August 2010.

Data related to many different apps from a variety of Android app repositories were accumulated over more than one year. Additionally, malware apps were thoroughly characterized based on their detailed behavior breakdown, including the installation, activation, and payloads.

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 defined families 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*. Samples of 14 of the malware families were obtained from the official Android market, while samples of 44 of the families came from unofficial markets. As some families are present in both markets (official and unofficial), the final dataset to be analysed consists of 49 samples (one for each family) and each sample is described by 26 different features derived from a study of each one of the apps. The features are divided into six categories, as can be seen in Table 1.

Table 1. Features describing each one of the malware families in the Malgenome dataset.

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

The features describing each family take the values of 0 (if that feature is not present in that family) or 1 (if the feature is present).

3.2 Results

For comparison purposes, some different projection models have been applied, whose results are shown below.

PCA Projection

Fig. 1 shows the principal component projection (components 1 and 2), obtained by applying PCA to the previously described data.

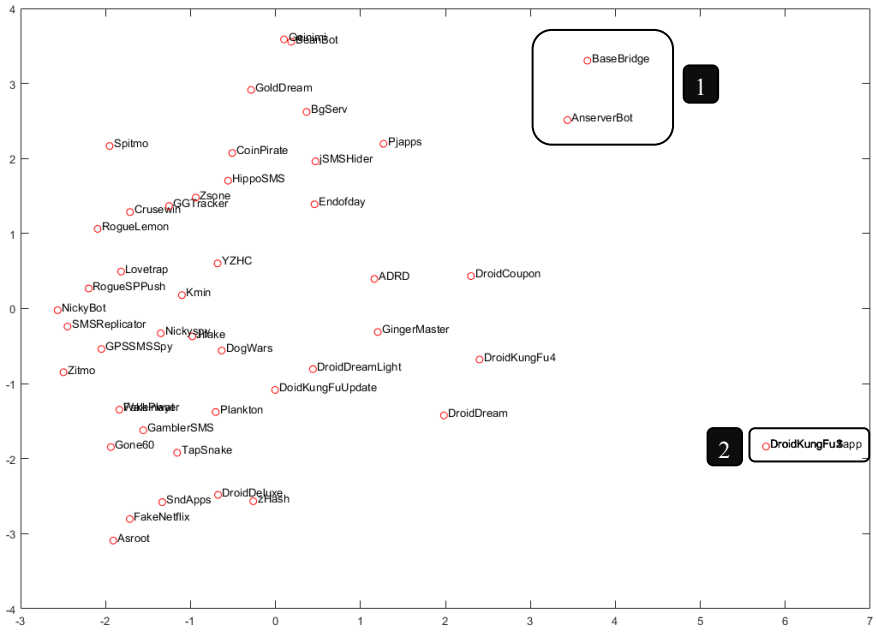

Fig. 1. PCA projection of Malgenome families.

In Fig 1 it can be seen that most of the malware families are grouped in a main group (left side of the figure) while just a few families can be identified away from this cluster (groups 1 and 2). Group 1 gathers two families (*BaseBridge* and *AnserverBot*), that are the only two families in the dataset that combine repackaging and update installation. Group 2 gathers four families (*DroidKungFu1, DroidKungFu2, DroidKungFu3* and *DroidKungFuSapp*) that are the only ones in the dataset presenting the encrypted privilege escalation.

Additionally, this first projection let us identify that some families are projected at the very same place. By getting back to the data we have realized that these families take the very same values for all the features. This is the case of *Walkinwat* and *FakePlayer* on the one hand and for *DroidKungFu1, DroidKungFu2, DroidKungFu3* and *DroidKungFuSapp* on the other hand. It means that, by taking into account the features in the analysed dataset, it will not be possible to distinguish *Walkinwat* from *FakePlayer* malware or any of the 4 mentioned variants of *DroidKungFu* malware.

MLHL Projection

Fig. 2 shows the MLHL projection of the analyzed data (two main components). MLHL projection shows the structure of the data in a way that a kind of ordering can be seen in the dataset. However, as it is more clearly shown in the CMLHL projection (Fig. 3), MLHL is not further described.

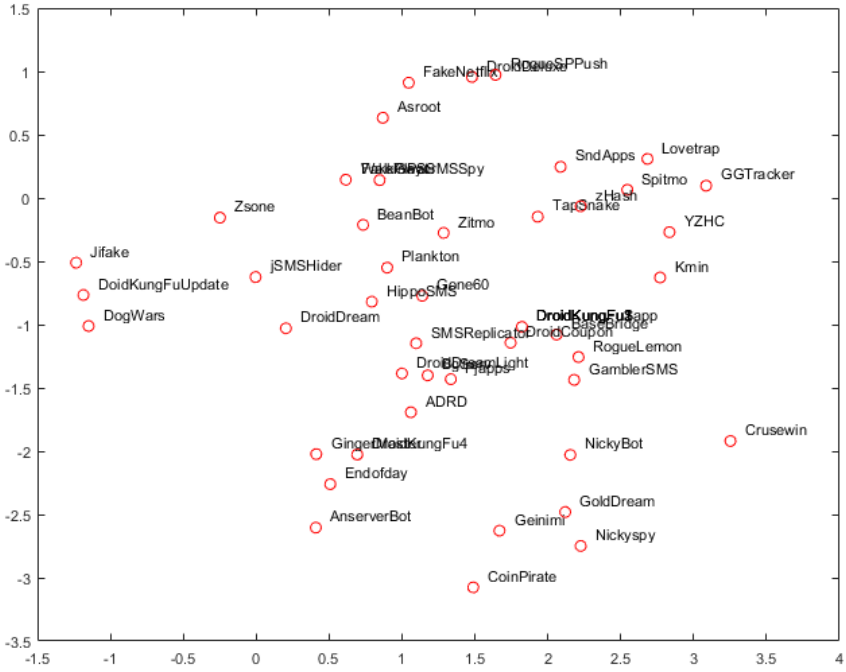

Fig. 2. MLHL projection of Malgenome families.

The parameter values of the MLHL model for the projections shown in Fig. 2 are: Number of output dimensions: 3. Number of iterations: 100, learning rate: 0. 2872, p: 0.4852.

CMLHL Projection

When applying CMLHL to the analysed dataset, the projection (two main components) shown in Fig. 3 has been obtained. As expected, CMLHL obtained a sparser projection than MLHL and PCA, revealing the structure of the dataset in a clearer way.

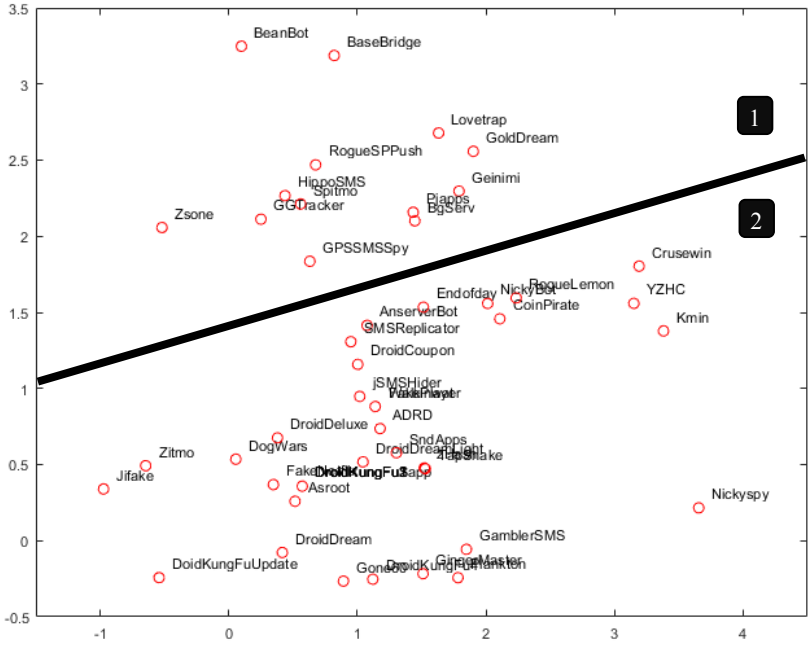

Fig. 3**.** CMLHL projection of Malgenome families.

The parameter values of the CMLHL model for the projections shown in Fig. 3 are; Number of output dimensions: 3. Number of iterations: 100, learning rate: 0.0406, p: 1.92, τ: 0.44056.

In Fig 3 it is easy to visually identify at least two main groups of data, labelled as 1 and 2. It has been checked that families in each one of these groups are similar in a certain way; group 1 gathers all the families

with dangerous SMS activity, as "SMS activation" and "SMS financial charges" are present in all the families in this group. On the other hand, none of the families in this group present any of the following features: USB or "PKG activation", and "user-account personal information stealing". This group is also characterized by the almost complete absence of privilege escalation, as only one of those features (RATC/Zimperlich) is present in only one of the families (*BaseBridge*). Regarding group 2, none of the families in Group 2 present the feature "phone-call financial charges".

From a deeper analysis of such groups, some subgroups can be distinguished and are identified in Fig. 4. Additionally, the families located in each one of these groups are listed in Table 2.

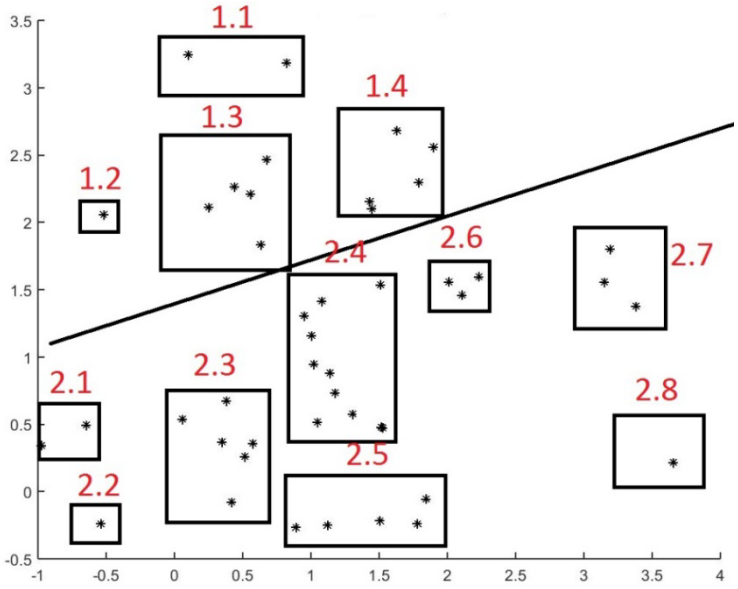

Fig. 4. CMLHL projection of Malgenome families with identified subgroups.

Table 2. Families allocation to subgroups defined in CMLHL projection.

Group	Subgroup	Families
1	1.1	*BaseBridge, BeanBot*
	1.2	*Zsone*
	1.3	*GGTracker, GPSSMSSpy, HippoSMS, RogueSPPush, Spitmo*
	1.4	*BgServ, Geinimi, GoldDream, Lovetrap, Pjapps*
2	2.1	*Jifake, Zitmo*
	2.2	*DroidKungFuUpdate*
	2.3	*Asroot, DogWars, DroidDeluxe, DroidDream, DroidKungFu1, DroidKungFu2, DroidKungFu3, DroidKungFuSapp, FakeNetflix*
	2.4	*ADRD, AnserverBot, DroidCoupon, DroidDreamLight, Endofday, FakePlayer, jSMSHider, SMSReplicator, SndApps, TapSnake, Walkinwat, zHash*
	2.5	*DroidKungFu4, GamblerSMS, GingerMaster, Gone60, Plankton*
	2.6	*CoinPirate, NickyBot, RogueLemon*
	2.7	*Crusewin, Kmin, YZHC*
	2.8	*Nickyspy*

All the variants of *DroidKungFu* malware are located in the bottom-left side of the projection (groups 2.2, 2.3, and 2.5). *Jifake* and *Zitmo* are gathered in the same subgroup (2.1) as they are the only two families in group 2 presenting the drive-by download installation feature.

ISOMAP Projection

In Fig. 5 it is shown the projections obtained by ISOMAP algorithm where each sample is labelled with the name of the family it belongs.

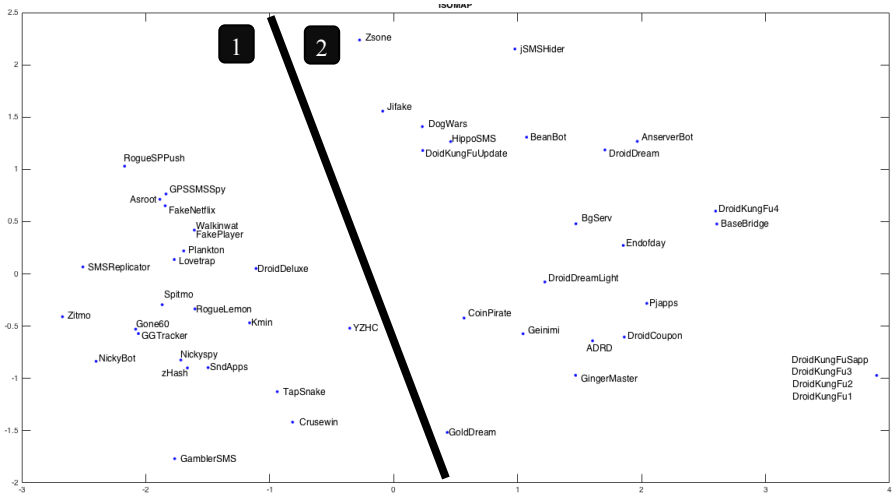

Fig. 5**.** ISOMAP projection of Malgenome families.

The parameter values of the ISOMAP model for the projection shown in Fig. 5 are: number of neighbours: 10.

ISOMAP clearly visualize the internal dataset structure, with a main division into 2 groups (1 and 2 in Fig. 5). For a deeper analysis, these two main groups are split in different subgroups as shown in Fig. 6.

From groups in Fig 5, it can be highlighted that group 1 contains Malgenome families that present "standalone" but not "repackaging" installation features (see Table 3). However, in case of group 2, none of its samples present "standalone installation" feature and all of them present the "repackaging installation" feature.

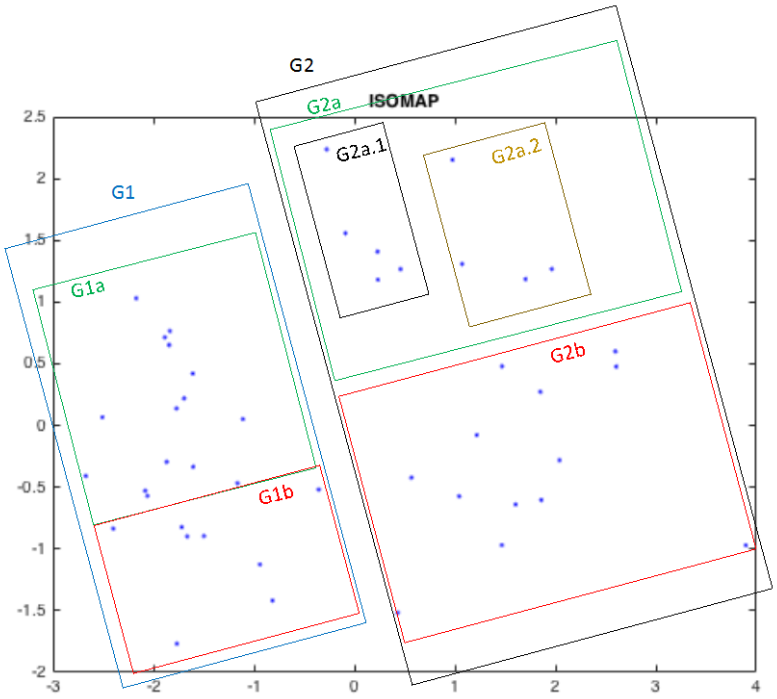

Fig. 6**.** ISOMAP projections of Malgenome families with identified subgroups.

As shown in Fig. 6, group 1 is divided in 2 subgroups (G1a and G1b), where G1a gathers families that do not present the "BOOT activation" feature as opposed to G1b, where its samples present this "BOOT activation" feature. Similarly, G2 is clearly divided into 2 subgroups (G2a and G2b), with analogous characteristics to samples in G1a and G2a respectively ("BOOT activation" for G2a and the opposite for

G2b). Finally, G2a presents samples with dangerous activity of "NET remote control" (group G2a.1), and samples without such dangerous feature (group G2a.2).

Table 3 shows the Malgenome families contained in each one of the identified groups, and the features characterizing all the families in that group.

Table 3. Families allocation to subgroups defined in ISOMAP projection.

Group	Subgroup		Families and features
G1	G1a		*RogueSPPush, GPSSMSSpy, Asroot, FakeNetflix, Walkinwat, FakePlayer, Plankton, SMSReplicator, Lovetrap, DroidDeluxe, Spitmo, Zitmo, Spitmo, RogueLemon, Gone60, GGTracker, Kmin* **Present features: 4 (standalone installation) and 5 (BOOT activation)** **Not-present feature: 1(repackaging installation)**
	G1b		*NickyBot, Nickyspy, zHash, SndApps, YZHC, TapSnake, Crusewin, GamblerSMS* **Present features: 4 (standalone installation)** **Not-present feature: 1(repackaging installation) and 5 (BOOT activation)**
G2	G2a	G2a.1	*Zsone, Jifake, DogWars, HippoSMS, DoidKungFuUpdate* **Present features: 1 (repackaging installation) and 5 (BOOT activation)** **Not-present feature: 4 (standalone installation) and 19 (NET remote control)**
		G2a.2	*jSMSHider, BeanBot, AnserverBot, DroidDream* **Present features: 1 (repackaging installation), 5 (BOOT activation) and 19 (NET remote control)** **Not-present feature: 4 (standalone installation)**
	G2b		*DroidKungFu4, BaseBridge, BgServ, Endofday, DroidDreamLight, Pjapps, DroidCoupon, CoinPirate, Geinimi, ADRD, GingerMaster, DroidKungFuSapp, DroidKungFu3, DroidKungFu2, DroidKungFu1* **Present features: 1 (repackaging installation) and 19 (NET remote control)** **Not-present feature: 4 (standalone installation), and 5 (BOOT activation)**

CCA Projection

Fig. 7 presents the projection obtained by CCA of Malgenome families, where it can be seen that a clear internal structure of the dataset can not be identified, and malware families can not be clearly gathered in groups, as it happened in previous results.

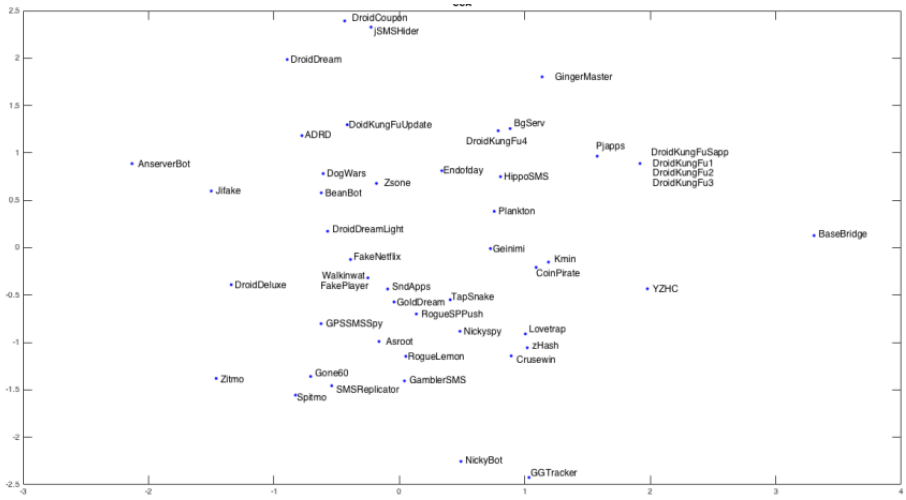

Fig. 7. CCA projection of Malgenome families.

Different combinations of values were tested for the parameters of the CCA model. The best projection obtained is the one shown in Fig. 7, that was generated with 1,000 epochs, alpha=0.5 and lambda=1.5152.

SOM results

Finally, SOM has been also applied to the Malgenome dataset and the obtained U-matrix is shown in Fig. 8. Each one of the neurons in the map has been labelled with the names of the malware families to which the neuron responds. From this figure, and according to the inter-neuron distances, neurons in the map could be easily split in two main groups (G1 and G2). At the same time, G1 could be divided in two subgroups (G1a and G1b), and G1a could also be divided into three subgroups as neuron distances are high between them (blue color means high distance in Fig. 8). In the case of G2, authors believe that it can not be divided into subgroups, as neuron distances within G2 are quite small, so it can not be said that families in this group (G2) are very different.

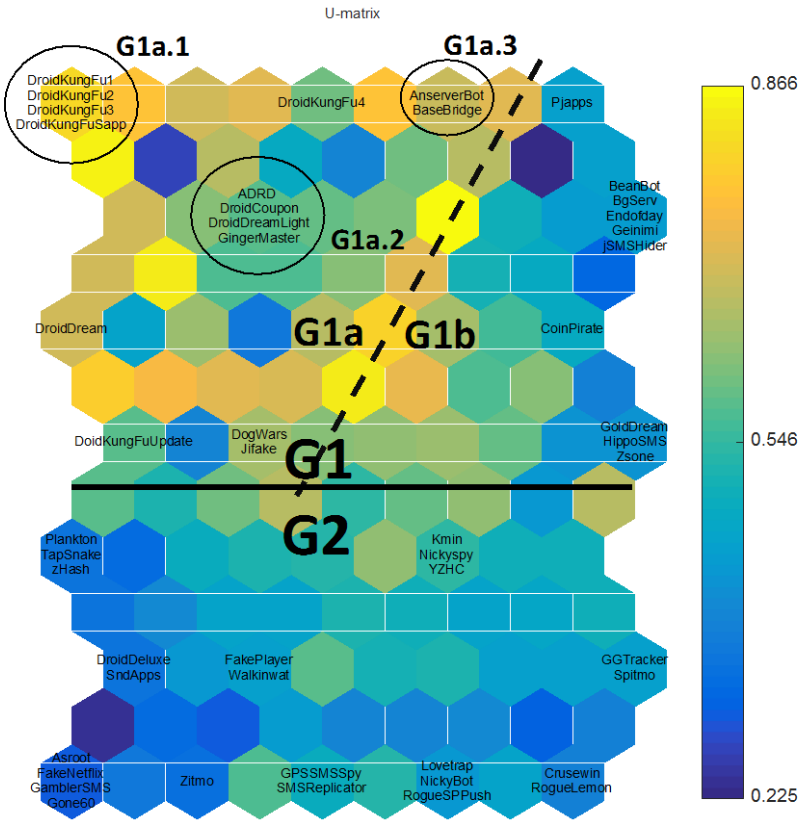

Fig. 8**.** SOM U-matrix for Malgenome families with identified groups.

The parameter values of the SOM model for the mapping shown in Fig. 8 are; map size: [7, 5], lattice: hexagonal, neighbourhood function: Gaussian. On the other hand, some metrics about the obtained mapping are: quantization error = 1.1, and topographic error = 0.0. In general terms, it can be said that the "repackaging installation" and "standalone installation" features are the only ones that let distributing samples in groups G1 and G2. In the case of subgroups G1a and G1b, it is the presence of the "SMS financial charges" feature what characterize malware families in each one of them.

Table 4 shows the Malgenome families contained in each one of the identified groups by the SOM network, and the features characterizing all the families in that group. General information (present and not-present features) of a group (i.e. group 1) is applicable to all malware families contained in its subgroups (i.e. subgroups G1a1, G1b, etc.).

Table 4. Families allocation to subgroups defined in SOM u-matrix.

<table>
<tr><th colspan="2">G1
Present feature: 1 (repackaging installation)
Not-present feature: 4 (standalone installation)</th></tr>
<tr><th colspan="2">G1a
Present feature: 22 (SMS financial charges)</th></tr>
<tr><td>G1a1</td><td>DroidKungFu1, DroidKungFu2, DroidKungFu3, DroidKungFuSapp
Present features: 11 (BATT activation), 14 (exploid privilege escalation), 18 (encrypted privilege escalation)
Not-present feature: 2 (update installation), 24 (SMS personal information stealing)</td></tr>
<tr><td>G1a2</td><td>DroidCoupon, DroidDreamLight, GingerMaster, ADRD
Not-present feature: 11 (BATT activation), 14 (exploid privilege escalation), 18 (encrypted privilege escalation), 2 (update installation), 24 (SMS personal information stealing)</td></tr>
<tr><td>G1a3</td><td>AnserverBot, BaseBridge
Present features: 11 (BATT activation), 2 (update installation), 24 (SMS personal information stealing)
Not-present feature: 14 (exploid privilege escalation), 18 (encrypted privilege escalation)</td></tr>
<tr><td>Other samples</td><td>DogWars, DoidKungFuUpdate, DroidDream, DroidKungFu4, Jifake</td></tr>
<tr><th colspan="2">G1b
Not present features: 22 (SMS financial charges)</th></tr>
<tr><td colspan="2">BeanBot, BgServ, CoinPirate, Endofday, Geinimi, GoldDream, HippoSMS, jSMSHider, Pjapps, Zsone</td></tr>
<tr><th colspan="2">G2
Present features: 4 (standalone installation)
Not-present feature: 1 (repackaging installation)</th></tr>
<tr><td colspan="2">Asroot, Crusewin, DroidDeluxe, FakeNetflix, FakePlayer, GamblerSMS, GGTracker, Gone60, GPSSMSSpy, Kmin, Lovetrap, NickyBot, Nickyspy, Plankton, RogueLemon, RogueSPPush, SMSReplicator, SndApps, Spitmo, TapSnake, Walkinwat, YZHC, zHash, Zitmo</td></tr>
</table>

4 Conclusions

From the results shown in section 3, it can be concluded that dimensionality reduction techniques are an interesting proposal to visually analyse the structure of a high-dimensionality dataset in general terms. More specifically, when studying Android malware families, this kind of techniques let us gain deep knowledge about the nature of such app families. Thanks to the obtained projections, similarities and differences of the studied families are identified

From the extensive set of applied DRTs, PCA, MLHL and CCA failed in generating an informative visualization of samples by reducing the dimensionality of them to 2D. On the other hand and generally speaking, it can be said that the DRTs that group malware families, are able to do that in a way consistent with the seminal characterization of Malgenome dataset [7]. More precisely, it is worth mentioning that installation features (repackaging and standalone) have been identified by ISOMAP and SOM as the most important ones for a general characterization of Android malware families (see section 3 for further details). It is an important result as repackaging is one of the most common techniques applied to hide malware (86% of the malware apps in the original dataset were repackaged versions of different legitimate apps including paid apps, popular game apps, powerful utility apps, etc. [6]).

In a complementary way, CMLHL identified some activation (SMS, USB, and PKG) and financial charges (SMS, and phone call) as the most important ones for such a task. Knowledge generated by the application of DRTs could be applied to improve the detection rate of Android Malware at different stages (markets, devices, etc.) thanks to the characterization of the different families.

As a final conclusion, it can be said that the identification and characterization of Android malware is still and open challenge that requires great efforts to be devoted in coming years.

5 Future Work

As future work it is planned to apply new DTRs models and compare them with other supervised algorithms such as decision threes in order to gain deep knowledge of the dataset. It will also be analysed other datasets related to cybersecurity applying the same approach followed in this research in order to generalize to other datasets the proposed method.

References

[1] 'Smartphone sales by OS worldwide 2009-2017 | Statistic', *Statista*. [Online]. Available: https://www.statista.com/statistics/266219/global-smartphone-sales-since-1st-quarter-2009-by-operating-system/. [Accessed: 21-Nov-2017].

[2] 'Android Operating System Statistics - AppBrain'. [Online]. Available: https://www.appbrain.com/stats/stats-index. [Accessed: 21-Nov-2017].

[3] 'SophosLabs 2018 Malware Forecast', SophosLabs., 2017.

[4] 'Malwarebytes Labs Report: Cybercrime Tactics and Techniques Q3 2017', Malwarebytes.

[5] S. Arshad, M. A. Shah, A. Khan, and M. Ahmed, 'Android Malware Detection & Protection: A Survey', *Int. J. Adv. Comput. Sci. Appl. Ijacsa*, 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*, Washington, DC, USA, 2012, pp. 95–109.

[7] 'Android Malware Genome Project'. [Online]. Available: http://www.malgenomeproject.org/. [Accessed: 21-Nov-2017].

[8] M. Paliwal and U. A. Kumar, 'Neural networks and statistical techniques: A review of applications', *Expert Syst. Appl.*, vol. 36, no. 1, pp. 2–17, Jan. 2009.

[9] R. F. Garcia, J. L. C. Rolle, M. R. Gomez, and A. D. Catoira, 'Expert condition monitoring on hydrostatic self-levitating bearings', *Expert Syst. Appl.*, vol. 40, no. 8, pp. 2975–2984, Jun. 2013.

[10] L. C. S. Bárcena, A. Pérez, Á. Herrero, and E. Corchado, 'Analyzing Key Factors of Human Resources Management.', in *Intelligent Data Engineering and Automated Learning - IDEAL 2011 - 12th International Conference, Norwich, UK, September 7-9, 2011. Proceedings*, 2011, pp. 463–473.

[11] C. Crespo-Turrado, M. del C. Meizoso-López, F. S. Lasheras, B. A. Rodríguez-Gómez, J. L. Calvo-Rolle, and F. J. de C. Juez, 'Missing Data Imputation of Solar Radiation Data under Different Atmospheric Conditions.', *Sensors*, vol. 14, no. 11, pp. 20382–20399, 2014.

[12] J. L. Calvo-Rolle, I. Machón González, and H. López García, 'Neuro-robust controller for non-linear systems', *Dyna*, vol. 86 (3), pp. 308–317.

[13] E. Corchado, Á. Herrero, and J. M. Sáiz, 'Testing CAB-IDS Through Mutations: On the Identification of Network Scans.', in *Knowledge-Based Intelligent Information and Engineering Systems, 10th International Conference, KES 2006, Bournemouth, UK, October 9-11, 2006, Proceedings, Part II*, 2006, pp. 433–441.

[14] R. M. Sánchez, Á. Herrero, and E. Corchado, 'Visualization and Clustering for SNMP Intrusion Detection', *Cybern. Syst.*, vol. 44, no. 6–7, pp. 505–532, 2013.

[15] C. Pinzón, J. F. de Paz, Á. Herrero, E. Corchado, J. Bajo, and J. M. Corchado, 'idMAS-SQL: Intrusion Detection Based on MAS to Detect and Block SQL injection through data mining.', *Inf Sci*, vol. 231, pp. 15–31, 2013.

[16] 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 *10th International Conference on Hybrid Intelligent Systems (HIS 2010), Atlanta, GA, USA, August 23-25, 2010*, 2010, pp. 73–78.

[17] Á. Herrero, U. Zurutuza, and E. Corchado, 'A Neural-Visualization IDS for Honeynet Data', *Int J Neural Syst*, vol. 22, no. 2, 2012.

[18] D. Atienza, Á. Herrero, and E. Corchado, 'Neural Analysis of HTTP Traffic for Web Attack Detection.', in *International Joint Conference - CISIS'15 and ICEUTE'15, 8th International Conference on Computational Intelligence in Security for Information Systems / 6th International Conference on EUropean Transnational Education, Burgos, Spain, 15-17 June, 2015*, 2015, pp. 201–212.

[19] L. Cen, C. S. Gates, L. Si, and N. Li, 'A Probabilistic Discriminative Model for Android Malware Detection with Decompiled Source Code', *IEEE Trans. Dependable Secure Comput.*, vol. 12, no. 4, pp. 400–412, 2015.

[20] 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, no. Supplement C, pp. 638–646, Jan. 2018.

[21] B. Sanz *et al.*, 'Mama: Manifest Analysis for Malware Detection in Android', *Cybern. Syst.*, vol. 44, no. 6–7, pp. 469–488, Oct. 2013.

[22] P. Teufl, M. Ferk, A. Fitzek, D. Hein, S. Kraxberger, and C. Orthacker, 'Malware Detection by Applying Knowledge Discovery Processes to Application Metadata on the Android Market Google Play', *Sec Commun Netw*, vol. 9, no. 5, pp. 389–419, Mar. 2016.

[23] J.-W. Jang, J. Yun, A. Mohaisen, J. Woo, and H. K. Kim, 'Detecting and classifying method based on similarity matching of Android malware behavior with profile', *SpringerPlus*, vol. 5, p. 273, 2016.

[24] A. Altaher, 'An improved Android malware detection scheme based on an evolving hybrid neuro-fuzzy classifier (EHNFC) and permission-based features', *Neural Comput. Appl.*, vol. 28, no. 12, pp. 4147–4157, Dec. 2017.

[25] M. Wagner *et al.*, 'A Survey of Visualization Systems for Malware Analysis', 2015.

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

[27] W. Park, K. H. Lee, K. S. Cho, and W. Ryu, 'Analyzing and detecting method of Android malware via disassembling and visualization', in *2014 International Conference on Information and Communication Technology Convergence (ICTC)*, 2014, pp. 817–818.

[28] V. Moonsamy, J. Rong, and S. Liu, 'Mining permission patterns for contrasting clean and malicious android applications', *Spec. Sect. Intell. Big Data Process.*, vol. 36, no. Supplement C, pp. 122–132, Jul. 2014.

[29] O. Somarriba, U. Zurutuza, R. Uribeetxeberria, L. Delosières, and S. Nadjm-Tehrani, 'Detection and Visualization of Android Malware Behavior', *JECE*, vol. 2016, Apr. 2016.

[30] E. Corchado, D. MacDonald, and C. Fyfe, 'Maximum and minimum likelihood Hebbian learning for exploratory projection pursuit', *Data Min. Knowl. Discov.*, vol. 8, no. 3, pp. 203–225, 2004.

[31] E. Corchado and Á. Herrero, 'Neural visualization of network traffic data for intrusion detection', *Appl Soft Comput*, vol. 11, no. 2, pp. 2042–2056, 2011.

[32] H. Quintián and E. Corchado, 'Beta Hebbian Learning as a New Method for Exploratory Projection Pursuit', *Int J Neural Syst*, vol. 27, no. 6, pp. 1–16, 2017.

[33] K. Pearson, 'LIII. On lines and planes of closest fit to systems of points in space', *Lond. Edinb. Dublin Philos. Mag. J. Sci.*, vol. 2, no. 11, pp. 559–572, Nov. 1901.

[34] E. Oja, 'Principal components, minor components, and linear neural networks', *Neural Netw.*, vol. 5, no. 6, pp. 927–935, 1992.

[35] C. Fyfe, 'A Neural Network for PCA and Beyond', *Neural Process. Lett.*, vol. 6, no. 1–2, pp. 33–41, 1997.

[36] S. Hou and P. D. Wentzell, 'Re-centered kurtosis as a projection pursuit index for multivariate data analysis', *J. Chemom.*, vol. 28, no. 5, pp. 370–384, 2014.

[37] C. Fyfe, D. R. McGregor, and R. Baddeley, *Exploratory projection pursuit: an artificial neural network approach*. Department of Computer Science, University of Strathclyde, 1994.

[38] E. Corchado and C. Fyfe, 'Connectionist techniques for the identification and suppression of interfering underlying factors', *Int. J. Pattern Recognit. Artif. Intell.*, vol. 17, no. 08, pp. 1447–1466, 2003.

[39] H. Chang, D.-Y. Yeung, and Y. Xiong, 'Super-resolution through neighbor embedding', in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, 2004, vol. 1, pp. I–I.

[40] P. Demartines and J. Herault, 'Curvilinear component analysis: a self-organizing neural network for nonlinear mapping of data sets', *IEEE Trans. Neural Netw.*, vol. 8, no. 1, pp. 148–154, 1997.

[41] G. Cirrincione, J. Hérault, and V. Randazzo, 'The on-line curvilinear component analysis (onCCA) for real-time data reduction', in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1–8.

[42] J. W. Sammon, 'A Nonlinear Mapping for Data Structure Analysis', *IEEE Trans Comput*, vol. 18, no. 5, pp. 401–409, May 1969.

[43] N. Chen, B. Ribeiro, A. Vieira, and A. Chen, 'Clustering and visualization of bankruptcy trajectory using self-organizing map', *Expert Syst. Appl.*, vol. 40, no. 1, pp. 385–393, Jan. 2013.

[44] J. J. Fuertes, M. Domínguez, P. Reguera, M. A. Prada, I. Díaz, and A. A. Cuadrado, 'Visual dynamic model based on self-organizing maps for supervision and fault detection in industrial processes', *Eng. Appl. Artif. Intell.*, vol. 23, no. 1, pp. 8–17, 2010.

[45] T. Kohonen, 'The self-organizing map', *Neurocomputing*, vol. 21, no. 1–3, pp. 1–6, 1998.

[46] E. Mohebi and A. Bagirov, 'Constrained Self Organizing Maps for Data Clusters Visualization',

Neural Process. Lett., vol. 43, no. 3, pp. 849–869, Jun. 2016.

[47] Y. Wu, T. K. Doyle, and C. Fyfe, 'Multi-layer Topology Preserving Mapping for K-Means Clustering', in *Intelligent Data Engineering and Automated Learning - Ideal 2011*, vol. 6936, H. Yin, W. Wang, and V. RaywardSmith, Eds. Berlin: Springer-Verlag Berlin, 2011, pp. 84–91.

[48] H. Quintián and E. Corchado, 'Beta Scale Invariant Map', *Eng. Appl. Artif. Intell.*, vol. 59, pp. 218–235, 2017.

[49] E. K. Haimoudi, H. Fakhouri, L. Cherrat, and M. Ezziyyani, 'Towards a New Approach to Improve the Classification Accuracy of the Kohonen's Self-Organizing Map During Learning Process', *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 3, pp. 230–236, Mar. 2016.

[50] T. Kohonen, 'Essentials of the self-organizing map', *Neural Netw.*, vol. 37, pp. 52–65, 2013.