On Partitional Clustering of Malware

Renato Cordeiro de Amorim
Department of Computer Science and Information Systems
Birkbeck University of London
Malet Street, London WC1E 7HX
Email: renato@dcs.bbk.ac.uk

Peter Komisarczuk
School of Computing & Technology
University of West London
St Mary's Road, London W5 5RF
Email: peter.komisarczuk@uwl.ac.uk

***Abstract*—In this paper we fully describe a novel clustering method for malware, from the transformation of data into a manipulable standardised data matrix, finding the number of clusters until the clustering itself including visualisation of the high-dimensional data. Our clustering method deals well with categorical data and clusters the behavioural data of 17,000 websites, acquired with Capture-HPC, in less than 2 minutes.**

Keywords: Malware; Intelligent K-Means; Clustering.

I. Introduction

Malware is a popular term used to describe software designed to perform undesirable actions, such as gain unauthorised access to computers, exfiltrate sensitive data or simply disrupt the normal operation of computers.

Large amounts of malware are discovered daily, already approaching 10,000 in 2010 [18]. Such quantity of malware coupled with the possibility of financial gain have created a rather wide variety of attack/deployment strategies, giving birth to names such as viruses, worms, spyware, adware, etc.

Malware has created a whole new industry focused in its development or software tools for detection. As an example of the latter we have Capture-HPC [9], a high-interaction honeypot client. Honeypot clients work by monitoring the state of an unprotected network client normally using kernel callback mechanisms. They look for changes in the file system, registry, network activity and so on, filtering out common state changes through an exclusion list. When a state change is detected, suggesting a possible malware activity, Capture-HPC updates a log file containing comprehensive information regarding the state change, including a list of files and registry keys changed as well as launched processes and attempted tcp connections that may have happened.

The nature of Capture-HPC makes sure there are no false negatives except when Capture-HPC itself is corrupted. In order to increase the detection rate of Capture-HPC for our experiments, we have given it the additional capability of emulating the presence of ActiveX components. This way a malware will always detect as installed any ActiveX component it is trying to exploit.

We believe that organising malware into homogeneous clusters may be helpful to generate a faster response to new threats, and better understanding of malware activities. Clearly one can generate different clusters by taking different points of view, such as the malware binaries or its behaviour. We have opted for the latter and Capture-HPC has proved to be an excellent tool to gather the behaviour of malware [11, 24, 23], being able to detect a high amount of different malware related activities.

The idea of homogeneity is directly linked to similarity and by consequence to a distance measure. In clustering squared Euclidean metric is the most popular distance measure, but it is not the most appropriate when an entity, in our case websites, can have a high amount of features, the activities. Such scenarios define a high-dimensional space and these can be difficult to organise due to the so called curse of dimensionality [7].

We present here a fast method for malware clustering. Our method generates a data matrix from the behavioural data of malware, standardise this data matrix and find the number of clusters in a dataset by using intelligent K-Means [20]. We show that the number of clusters obtained by our method is correct by ratify this number by analysing the Hartigan index [14] and perform visual analysis of the clusters by using the 2 first principal components of the data matrix.

Methods based on the presence or absence of a particular malware activity can be quite difficult to cluster with a fast partitional algorithm such as K-Means. The reason for this is that the data is most, if not all, categorical. Nevertheless, our method clusters 17,000 websites in less than 2 minutes.

II. Background and Related Work

Cova, Kruegel, and Vigna [11] empirically showed that most cases Capture-HPC could not detect a malware, it was because the malware was targeting a plugin not present in the system. Taking this into account we have updated Capture-HPC to emulate the presence of any requested ActiveX component by using AxMock [1].

The organisation of malware could follow a supervised, semisupervised or unsupervised learning approach. While supervised and semisupervised require labelled data stating that a particular behaviour should belong to a particular group, unsupervised which is also known as clustering, is a data-driven approach that does not require labelled data at all. Clustering attempts to learn the patterns in a dataset using solely the data itself and a distance measure.

[1]Our new version of Capture-HPC will be available at https://projects.honeynet.org/capture-hpc by summer 2012. AxMock can be downloaded at http://code.google.com/p/axmock/.

Although supervised and arguably semisupervised algorithms tend to have better accuracy than unsupervised, their requirement of labelled data can be difficult to meet in certain scenarios. Labelling a statistically significant amount of data could require an impractical effort because of the high quantity of malware released everyday. Another issue is that this labelled data would have to be highly accurate, otherwise the algorithm could learn incorrect patterns and classify malware under the wrong groups. These facts made us opt for unsupervised learning.

In general, clustering algorithms can be divided into hierarchical and partitional. The former, in its more popular agglomerative form, generates clusters by merging entities or clusters and can be visualised through a dendrogram. Partitional algorithms generate a single set of labels for the entities and can be considerably faster than its counterpart. In both cases the granularity of the clustering is defined by the number of clusters in the dataset.

The use of clustering algorithms in datasets related to malware was introduced, to the authors knowledge, by Bailey et al. [4] using, as most of the literature, hierarchical clustering. Because of the large amount of data and the so called zero-day attacks, we consider that the speed of clustering is crucial and have chosen to use partitional clustering in this research.

K-Means [5, 19] is arguably the most popular clustering algorithm there is. K-Means partitions each entity $y \in Y$, in our case websites, into K clusters around centroids $C = \{c_1, c_2, ..., c_K\}$:

1) Assign values to K centroids $c_1, c_2, ..., c_K$, normally K random entities; $S \leftarrow \{\}$
2) Assign each entity y_i in the dataset to its closest centroid c_k, generating the clustering $S' = \{S'_1, S'_2, ..., S'_K\}$.
3) Update all centroids to the centre of their respective clusters.
4) If $S \neq S'$ then $S \leftarrow S'$ and go to step 2.
5) Output the clustering $S = \{S_1, S_2, ..., S_K\}$ and centroids $C = \{c_1, c_2, ..., c_K\}$

The above algorithm iteratively minimises the sum of the squared error over K clusters, we show the K-Means criterion in Equation (1).

$$W(S, C) = \sum_{k=1}^{K} \sum_{i \in S_k} d(y_i, c_k) \qquad (1)$$

where $d(y_i, c_k)$ is a function calculating the distance between y_i and c_k. K-Means is a rather successful algorithm, its popularity is mainly due to its easy implementation, simplicity, efficiency, and empirical success [16]. One can easily find implementation of K-Means in popular data analysis software packages such as R, MATLAB and SPSS.

Due to its constant use, K-Means weaknesses are well known, among them: (i) it is a greedy algorithm. There is no guarantee its criterion will reach a global minimum, meaning that the final clustering may not be optimal. Although there have been attempts to deal with this issue, most notably the classical solution of swapping entities between clusters given by Hartigan and Wong [15], this is a very difficult problem as the minimisation of Equation (1) is a NP-Hard problem, we will leave this for future research; (ii) it requires the number of clusters to be known beforehand; (iii) the final clustering depends highly on the initial centroids given to the algorithm, these are normally found at random.

In a number of scenarios, including ours, the exact number of clusters K may not be known. The literature of clustering malware tends to use hierarchical clustering algorithms [4, 6, 26] seemingly because it is possible to run such an algorithm without knowing K. However it can be difficult to interpret results when no granularity is set via K, possibly generating clusters with no significance. Another issue is that hierarchical algorithms are known not to scale well. For instance, it may take 3 hours to cluster 75,000 [6] while our method clusters 17,000 in less than 2 minutes (see Section IV)). We find that there is a considerable amount of research effort in attempting to find K that could be used in malware datasets [10, 14, 17, 20, 21].

Regarding the weakness (iii), K-Means is a non-deterministic algorithm. It may provide different clusterings if run more than once, this characteristic raises the question of which clustering to use. A common solution is to run K-Means a number of times, generating a number of clusterings, and pick the clustering S^* which is the closest to the K-Means data model. S^* will be the clustering with the smallest $W(S, C)$ given by the K-Means criterion, Equation (1). This approach does seem to work in a number of scenarios, but it can be very lengthy when dealing with high amounts of data.

In order to deal with weaknesses (ii) and (iii) at once, we have decided to use Intelligent K-Means (iK-Means) [20] due to its considerable success in different scenarios [2, 3, 10]. The iK-Means algorithm provides an heuristic initialization for K-Means based on the concept of anomalous clusters. An anomalous cluster is a cluster of entities that are far from the centre of gravity of the dataset. IK-Means iteratively finds each of these clusters and uses their centroids and number of clusters as the parameters for K-Means. The algorithm is formalised below.

1) Assign a value to θ; set c_c as the centre of gravity of the dataset; $C_t \leftarrow \{\}$
2) Set a tentative centroid c_t as the entity farthest away from c_c.
3) Apply K-Means using two centroids, c_t and c_c generating the clustering $S = \{S_t, S_c\}$.
4) If the cardinality of $S_t \geq \theta$ then $C_t \leftarrow c_t$, otherwise discard c_t. In any case, remove S_t from the dataset.
5) If there are still entities to be clustered go to step 2.
6) Run K-Means with the centroids in C_t.

To demonstrate the method works with malware data we have chosen to ratify the number of clusters it finds with visual inspection and the Hartigan index [14] mainly because of its easy of use and popularity. This index is based on the error

W, the output of Equation (1).

$$H(k) = (N - k - 1)(\frac{W_k - W_{k+1}}{W_{k+1}}) \quad (2)$$

This index requires K-Means to be run with different values for K and may take time. We find intuitive that the centroids found by iK-Means can be used here making K-Means a deterministic algorithm. Visibly W is inversely proportional to K, the more clusters the less variance within them, the index is based on abnormal variances in $H(k)$.

Unfortunately the malware datasets are very likely to be large in terms of websites and measurements, visibly high-dimensional datasets. The curse of dimensionality, a term coined by Bellman [7] states that as the number of dimensions increases so does the sparseness of data making entities to appear dissimilar, a very problematic fact to distance-based algorithms such as K-Means. This is further supported by research suggesting that the concept of nearest neighbours calculated using Euclidean distance becomes meaningless as the dimensionality of the data increases [13, 1, 8].

In order to cluster malware we need a method that supports high-dimensional spaces. This can be accomplished by selecting an appropriate distance measure for K-Means. Although the literature tends to use the Euclidean distance this is not the most appropriate in high-dimensional spaces. Empirical experiments [12, 25] show that distances such as the cosine are more appropriated than the Euclidean distance. The cosine distance for the N-dimensional x and y is defined as:

$$d(x, y) = 1 - \frac{\sum_{n=1}^{N} x_n \cdot y_n}{\sum_{n=1}^{N} (x_n)^2 \cdot \sum_{n=1}^{N} (y_n)^2} \quad (3)$$

A somewhat easier way to apply the cosine distance, is to perform an extra step in the pre-processing of data by dividing each row vector y_i representing a website by the vector's norm $\sqrt{\sum_{n=1}^{N} y_{in}^2}$ [13, 22, 27]. We find this particularly helpful to calculate centroids of each cluster.

As final consideration for this section, a clustering algorithm regardless of being partitional or hierarchical, will not yield that a given cluster is composed of malware. Clustering algorithms simply find that 2 or more clusters are dissimilar according to a given distance measure and cannot state what they are actually composed of. In order to define what malware family a cluster contains one would need the analysis of a field expert. Clearly this expert would not need to analyse the whole cluster but solely the malware that is the closest to the cluster centroid.

III. Method

In order to apply any clustering method we need to create and standardise a data matrix representing the whole dataset Y. In this data matrix each instance of $Y = \{y_1, y_2, ..., y_N\}$ represents a website and each column $v = \{1, 2, ..., M\}$ a feature. In our method y_{iv} may be assigned 0 or 1 representing the absence or presence of a particular malware behaviour, a feature, in the website y_i.

The first step of our method is to apply Capture-HPC on each website, recording its activities in a log file. The amount of time used for recording stays as a parameter, but we suggest it should not be less than a minute as some websites may take time to load. The full list of activities these websites performed on an unsecured network client becomes our initial list of features.

The second step is to filter the list of features. Although Capture-HPC uses a exclusion list to filter out expected state changes, it records data such as time and process ID generating groups of features that represent in fact the same malware activity. We disregard any part of the log file that is not directly linked to the activities, but to management, such as timestamps, process IDs, paths to files and IP numbers in tcp-connections. This filtering ensures each feature is unique in terms of what states the malware is changing, effectively reducing the number of features.

The third step is to create the data matrix, by assigning a value to each y_{iv}, and standardise it. When creating the entry y_i in the data matrix we read the log file for this particular website and search for of the M features we have listed. If a feature v is found to be in the log file then y_{iv} is set to 1, otherwise 0. Each value in our data matrix is categorical, making its standardization less obvious. We have opted to use a method presented by Mirkin [20]. In this each feature is transformed into 2 new features (since we have only 2 possible categories). Only one of the new features is assigned 1, the new feature corresponding to the category in the original feature, the other is assigned 0. We then standardise the data numerically by subtracting each of the values y_{iv} by the new feature average $\bar{y_v}$, linking the final value of y_{iv} to the frequency of v over Y.

In the forth step we apply the Intelligent K-Means algorithm with $\theta = 1$. We then sort the clusters in descending order by the number of websites found in each of them, in the anomalous cluster part of iK-Means. We choose the number of clusters by analysing when the cardinality of clusters stabilizes in a relatively small value.

Optionally, one can also ratify the number of clusters by using for instance the Hartigan index and the less reliable visual inspection. The former analyses the differences of the K-Means criterion shown in Equation (1) under different values for K. Regarding visual inspection, we have chosen to plot the whole dataset over its 2 first principal components. Because the data is categorical in nature the clusters structure can be rather difficult to see. One may wish to add a small mount of uniformly random noise to help the visualization of cluster cardinalities.

IV. Experiments

We acquired a list with 17,000 possibly infected IP addresses[2] . In the first step of our method we applied Capture-HPC for 2.5 minutes to each of these websites, generating a

[2]3,273 from http://www.malwaredomainlist.com/update.php plus 13,763 from http://www.malware.com.br/lists.shtml.

total of 133,327 features. By using our second step we were able to reduce the number of features to 231, making each feature truly representative.

We then created the data matrix of initially 17,000 websites over 231 by checking the logs of each website against the features we found. We finalised the data matrix by standardising it using Mirkin's method, effectively doubling the number of features.

In the next step we applied intelligent K-Means with $\theta = 1$, finding a total of 15 cluster with more than 1 feature. We then sorted the clusters by the number of entities found in the anomalous part of the algorithm and found the cardinalities 8439, 7851, 381, 129, 121 and very small numbers after these, suggesting the dataset had 5 clusters.

The popular Hartigan index helped us to ratify 5 as the number of clusters. After the 5^{th} cluster the error given by Equation (1) ceased to be significant. Although less reliable, we also found 5 clusters in this dataset by analysing visually the 2 first components of the dataset, extracted by using principal component analysis (PCA). The image can be seen in figure (1), to which we had to add random noise between 0 and 0.25 to increase clarity. It seems to us that figure (1) presents 2 clusters at the top, the one at the right being a bit more elongated, and 3 at the bottom, totally 5 clusters.

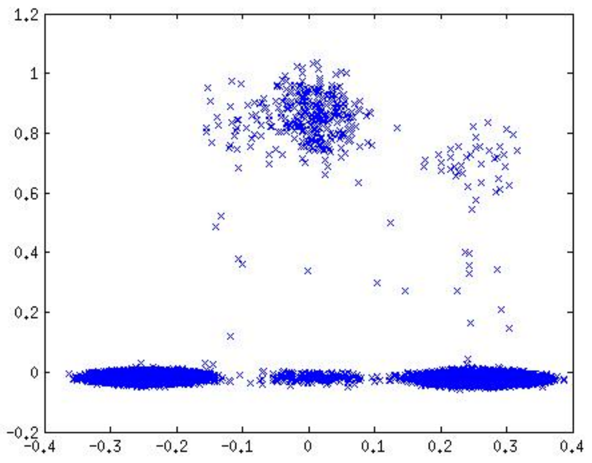

Fig. 1. A Dataset containing the behavioural data of 17,000 possibly infected websites over its 2 first principal components.

V. Conclusion

By using our method we were able to cluster a dataset of 17,000 malwares in less than 2 minutes. This was a particularly interesting database to work for 3 reasons: (i) the features were solely categorical; (ii) the cardinality of the clusters was uneven and (iii) it was a high-dimensional data matrix. Because of these three reasons the ordinary K-Means algorithm would constantly fail to cluster the dataset, generating empty clusters.

We solved the above issues by standardizing the data by transforming each feature into 2 new features allowing us to standardise them numerically through their frequencies in the dataset. We have also used the intelligent K-Means method to find the number of clusters in the dataset as well as the initial centroids for each of them. Finally, the issue of high-dimensionality was dealt by using the cosine distance rather than the more popular squared Euclidean distance.

This research assumes that the features obtained with Capture-HPC are all relevant. This sounds realistic since Capture-HPC uses an exclusion list for expected state changes, not leaving any reason why a client should suffer a change in state while not being used. Of course relevant features may have different degrees of importance for clustering particularly at different clusters. We intend to further developed a previous method used to find features weights called intelligent Minkowski K-means [2] so we can apply it to the clustering of malware.

VI. Acknowledgements

The authors would like to thanks Tiffany Youzhi Bao for her instrumental work developing AxMock ultimately allowing us to upgrade Capture-HPC to mock ActiveX components.

References

[1] C.C. Aggarwal, A. Hinneburg, and D.A. Keim. *On the surprising behavior of distance metrics in high dimensional space*. Citeseer, 2000.

[2] R. Cordeiro de Amorim and B. Mirkin. “Minkowski metric, feature weighting and anomalous cluster initializing in K-Means clustering”. In: *Pattern Recognition* (2011).

[3] R.C. de Amorim. “Constrained Intelligent K-Means: Improving Results with Limited Previous Knowledge.” In: *Advanced Engineering Computing and Applications in Sciences, 2008. ADVCOMP'08. The Second International Conference on*. IEEE. 2008, pp. 176–180.

[4] M. Bailey et al. “Automated classification and analysis of internet malware”. In: *Proceedings of the 10th international conference on Recent advances in intrusion detection*. Springer-Verlag. 2007, pp. 178–197.

[5] G.H. Ball and D.J. Hall. “A clustering technique for summarizing multivariate data”. In: *Behavioral Science* 12.2 (1967), pp. 153–155.

[6] U. Bayer et al. “Scalable, behavior-based malware clustering”. In: *Network and Distributed System Security Symposium (NDSS)*. Citeseer. 2009.

[7] R. Bellman. “Dynamic programming and Lagrange multipliers”. In: *Proceedings of the National Academy of Sciences of the United States of America* 42.10 (1956), p. 767.

[8] K. Beyer et al. “When is nearest neighbor meaningful?” In: *Database TheoryICDT99* (1999), pp. 217–235.

[9] *Capture-HPC*. Aug. 2011. URL: https://projects.honeynet.org/capture-hpc.

[10] M.M.T. Chiang and B. Mirkin. "Intelligent choice of the number of clusters in K-Means clustering: an experimental study with different cluster spreads". In: *Journal of classification* 27.1 (2010), pp. 3–40.

[11] M. Cova, C. Kruegel, and G. Vigna. "Detection and analysis of drive-by-download attacks and malicious JavaScript code". In: *Proceedings of the 19th international conference on World wide web*. ACM. 2010, pp. 281–290.

[12] S. France and D. Carroll. "Is the distance compression effect overstated? Some theory and experimentation". In: *Machine Learning and Data Mining in Pattern Recognition* (2009), pp. 280–294.

[13] S. France, D. Carroll, and H. Xiong. "Distance metrics for high dimensional nearest neighborhood recovery: Compression and normalization". In: *Information Sciences* (2011).

[14] JA Hartigan. "Clustering algorithms. 1975". In: *John Willey & Sons* ().

[15] J.A. Hartigan and M.A. Wong. "Algorithm AS 136: A k-means clustering algorithm". In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28.1 (1979), pp. 100–108.

[16] A.K. Jain. "Data clustering: 50 years beyond K-means". In: *Pattern Recognition Letters* 31.8 (2010), pp. 651–666.

[17] L. Kaufman, P.J. Rousseeuw, et al. *Finding groups in data: an introduction to cluster analysis*. Vol. 39. Wiley Online Library, 1990.

[18] B. Lau and V. Svajcer. "Measuring virtual machine detection in malware using DSD tracer". In: *Journal in Computer Virology* 6.3 (2010), pp. 181–195.

[19] J. MacQueen et al. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 281-297. California, USA. 1967, p. 14.

[20] B.G. Mirkin. *Clustering for data mining: a data recovery approach*. Vol. 3. CRC Press, 2005.

[21] D. Pelleg and A. Moore. "X-means: Extending K-means with efficient estimation of the number of clusters". In: *Proceedings of the Seventeenth International Conference on Machine Learning*. San Francisco. 2000, pp. 727–734.

[22] G. Qian et al. "Similarity between Euclidean and cosine angle distance for nearest neighbor queries". In: *Proceedings of the 2004 ACM symposium on Applied computing*. ACM. 2004, pp. 1232–1237.

[23] C. Seifert, P. Komisarczuk, and I. Welch. "True Positive Cost Curve: A Cost-Based Evaluation Method for High-Interaction Client Honeypots". In: *Emerging Security Information, Systems and Technologies, 2009. SECURWARE'09. Third International Conference on*. IEEE. 2009, pp. 63–69.

[24] C. Seifert et al. "Measurement study on malicious web servers in the. nz domain". In: *Information Security and Privacy*. Springer. 2009, pp. 8–25.

[25] A. Strehl, J. Ghosh, and R. Mooney. "Impact of similarity measures on web-page clustering". In: *Workshop on Artificial Intelligence for Web Search (AAAI 2000)*. 2000, pp. 58–64.

[26] T.F. Yen and M. Reiter. "Traffic aggregation for malware detection". In: *Detection of Intrusions and Malware, and Vulnerability Assessment* (2008), pp. 207–227.

[27] Y. Zhao and G. Karypis. "Empirical and theoretical comparisons of selected criterion functions for document clustering". In: *Machine Learning* 55.3 (2004), pp. 311–331.