Poisoning complete-linkage hierarchical clustering

Battista Biggio[1], Samuel Rota Bulò[2], Ignazio Pillai[1], Michele Mura[1]
Eyasu Zemene Mequanint[1], Marcello Pelillo[3] and Fabio Roli[1]

[1] University of Cagliari, Italy
[2] FBK-irst, Trento, Italy
[3] Ca' Foscari University, Venice, Italy

Abstract. Clustering algorithms are largely adopted in security applications as a vehicle to detect malicious activities, although few attention has been paid on preventing deliberate attacks from subverting the clustering process itself. Recent work has introduced a methodology for the security analysis of data clustering in adversarial settings, aimed to identify potential attacks against clustering algorithms and to evaluate their impact. The authors have shown that single-linkage hierarchical clustering can be severely affected by the presence of a very small fraction of carefully-crafted poisoning attacks into the input data, highlighting that the clustering algorithm may be itself the weakest link in a security system. In this paper, we extend this analysis to the case of complete-linkage hierarchical clustering by devising an ad hoc poisoning attack. We verify its effectiveness on artificial data and on application examples related to the clustering of malware and handwritten digits.

1 Introduction

Clustering algorithms play an important role in data analysis by allowing us to gain insight into large sets of unlabeled data. Recently, clustering algorithms have been adopted in the context of computer security to solve different problems, *e.g.*, spotting compromised domains in DNS traffic [1], gathering information about tools and sources of attacks against Internet websites [2], detecting malicious software (*i.e.*, malware) such as computer viruses or worms [3, 4], and even identifying repackaged Android applications and Android mobile malware [5, 6].

The collection of data for most of the aforementioned scenarios is carried out in an unsupervised way; *e.g.*, malware samples such as files infected by computer viruses are often gathered from the Internet using honeypots (*i.e.*, machines that purposely expose known vulnerabilities to be infected by malware [7]), or other ad-hoc services, like VirusTotal.[1] Accordingly, the clustering algorithms used to analyze such data are exposed to possible attacks. Indeed, carefully-crafted samples might be injected into the collected data to subvert the clustering process and prevent the system from gaining useful knowledge. Due to these intrinsically adversarial scenarios, evaluating the *security* of clustering algorithms against carefully-designed attacks and proposing suitable countermeasures has become an important issue.

[1] `http://virustotal.com`

In the literature, the problem of learning in adversarial environments has been mainly addressed in the area of supervised classification [8–11], and regression [12]. Instead, only *few* works have addressed the issue of security evaluation (and the design of countermeasures) of unsupervised learning approaches such as clustering algorithms. The pioneering work in [13, 14] has focused the attention on the problem of devising specific attacks to subvert the clustering process. They showed how points could be easily *hidden* within an existing cluster by forming a *fringe* cluster, *i.e.*, by placing such points sufficiently close the border of an existing cluster. They further designed attacks consisting in adding points in a way to *bridge* two clusters, *i.e.*, by inducing the fusion of two clusters. A step further has been taken in [15], where the authors considered several potential attack scenarios in a more systematic manner. Indeed, they introduced a model of the attacker that allows one to make specific assumptions on the adversary's goal, knowledge of the attacked system, and capability of manipulating the input data, and to subsequently formalize a corresponding *optimal* attack strategy.

In this paper, we extend the clustering security analysis proposed in [15], which was focused on single-linkage hierarchical clustering, to the case of complete linkage by devising an ad hoc poisoning attack. The reason is that single- and complete-linkage hierarchical clustering algorithms are among the most used ones for the purpose of malware detection and classification [3, 4]. To cope with the computational problem of determining the optimal attack strategy, we propose some heuristics that allow us to find good approximate solutions. We finally verify the effectiveness of our approach on artificial data and on application examples related to the clustering of malware and handwritten digits.

2 Clustering in adversarial settings

We review here the framework proposed in [15], which introduces an adversary's model that can be used to identify and devise attacks against clustering algorithms. The adversary's model comprises the definition of the adversary's goal, knowledge of the attacked system, and capability of manipulating the input data.

Adversary's goal. The adversary's goal is defined based on the attack specificity and the security violation pursued by the adversary. The attack specificity can be *targeted*, if it involves only the clustering of a given subset of samples; or *indiscriminate*, if it potentially affects the clustering of any sample. The security violations jeopardize the *integrity* of a system, its *availability*, or the *privacy* of its users. The *availability violations* are targeted at compromising the functionality of the system, thus causing a denial of service. In a supervised setting this amounts to achieving the largest possible classification error [10, 8, 16], while in the unsupervised case it entails attacks that induce a significant perturbation in the clustering result. The *integrity violations* aim at pursuing some specific malicious activities without significantly compromising the normal system operation. In the supervised learning setting [10, 11], these attacks camouflage some malicious samples (*e.g.*, spam emails) to evade detection, without affecting the classification of legitimate samples. In the unsupervised setting, instead, integrity violations are attacks aiming at deflecting the grouping for specific samples, while limiting the changes to the original clustering. As an example, an attacker might change some samples in a way to hide them in a different cluster, without excessively altering the initial clusters. Finally, the *privacy violations* try to obtain information about the system's users from the clustered data.

Adversary's knowledge. The knowledge that the adversary has about the system can be divided into: (i) *knowledge about the data*, *i.e.* the adversary knows the dataset or a surrogate set sampled from the same distribution; (ii) *knowledge of the feature space*, *i.e.* the adversary knows how features are extracted for each sample; (iii) *knowledge about the algorithm*, *i.e.* the adversary knows the clustering algorithm and thus how data is organized into clusters; (iv) *knowledge about the algorithm's parameters*, *i.e.* the adversary knows the parameters used to run the clustering algorithm. The scenario where the adversary has all the aforementioned types of knowledge is referred to as the *perfect knowledge* case.

Adversary's capability. The adversary's capability defines in which way and to what extent the attacker can influence the clustering process. In practice, it imposes some limits to the power of the attacker. In the supervised case [10, 8], an attacker can exercise an influence on the training data and test data (*a.k.a. causative influence*) or on the test data only (*a.k.a. exploratory influence*). In the unsupervised case, instead, there is no distinction between training and test set, so the adversary can exercise only a causative influence by manipulating the samples to be clustered. The capabilities of the adversary can be circumscribed by imposing a maximum number of samples that can be manipulated, *e.g.* in the case of malware collected through *honeypots* [7] the adversary might easily send few samples without having access to the rest of the data. An additional constraint consists in limiting the extent of the modifications that the attacker can do to a sample in order to preserve its malicious functionality. Indeed, malicious samples like spam emails or malware code may not be manipulated in an unconstrained manner. Such a constraint can be expressed in terms of a suitable distance measure between the original, non-manipulated attack samples and the manipulated ones, as in [8, 17, 10, 11].

3 Poisoning attacks

Once the adversary's model has been defined, one can design an *optimal* attack strategy that specifies how data should be manipulated to meet the adversary's goal, given the restrictions imposed by her knowledge and capabilities. In this section, we focus on *poisoning* attacks, *i.e.*, attacks targeted at violating the system's availability by indiscriminately corrupting the cluster assignment of any data point through the insertion of well-crafted *poisoning* samples in the input data. We additionally take the worst-case perspective in which the adversary has *perfect knowledge*. In more formal terms, following [15], the adversary's goal is to maximize a distance measure between the clustering $\mathcal{C}$ obtained from the original, unchanged dataset $\mathcal{D}$ and the clustering obtained by the application of the clustering algorithm on the contaminated dataset $\mathcal{D}'$. We assume $\mathcal{D}'$ to be the union of $\mathcal{D}$ with a set of (poisoning) *attack samples* $\mathcal{A}'$, *i.e.* $\mathcal{D}' = \mathcal{D} \cup \mathcal{A}'$. Accordingly, the objective function for the adversary can be written as

$$g(\mathcal{A}') = d_c(\mathcal{C}, f_{\mathcal{D}}(\mathcal{D} \cup \mathcal{A}')) \,, \tag{1}$$

where d_c is a distance measure between clusterings, and $f_{\mathcal{D}}(\mathcal{D}')$ denotes the output of the clustering algorithm f run on the data $\mathcal{D}'$, but restricted to the samples in $\mathcal{D}$, since we are interested in measuring the clustering corruption with respect only to the original data samples. The capability of the adversary is circumscribed by imposing a maximum number of m attack samples, *i.e.* $|\mathcal{A}'| \leq \mathsf{m}$, and by imposing a box constraint on the feature values of the attack samples, *i.e.* $\boldsymbol{x}_{\rm lb} \leq \boldsymbol{a} \leq \boldsymbol{x}_{\rm ub}$ for all $\boldsymbol{a} \in \mathcal{A}'$, where $\boldsymbol{a} \leq \boldsymbol{b}$ means that the inequality holds for all vector components. In other terms, the set of attack samples $\mathcal{A}'$ is an element of $\Omega_{\mathsf{p}} = \{\{\boldsymbol{a}'_i\}_{i=1}^{\mathsf{m}} : \boldsymbol{x}_{\rm lb} \leq \boldsymbol{a}'_i \leq \boldsymbol{x}_{\rm ub} \text{ for } i = 1, \ldots, \mathsf{m}\}$. To summarize, the optimal poisoning attack strategy with perfect knowledge under the aforementioned capabilities is the solution of the following optimization problem:

$$\operatorname{maximize} g(\mathcal{A}') \quad \text{s.t.}\, \mathcal{A}' \in \Omega_{\rm p}\,. \tag{2}$$

4 Poisoning complete-linkage hierarchical clustering

In this section, we focus on solving the optimization problem given by Eq. (2) for the *complete-linkage* hierarchical clustering algorithm.

Before delving into the details of our derivation, it is worth remarking here that hierarchical clustering algorithms do not output a given partitioning of the data into a set of clusters directly. They rather produce a *hierarchy* of clusterings by carrying out an *agglomerative* bottom-up procedure [18]: each point is initially considered as a separate cluster; then, at each iteration, the two *closest* clusters are fused according to a given distance measure between clusters (*i.e.*, the so-called *linkage* criterion), until a single cluster (containing all data points) is obtained. This procedure can be represented as a tree-like data structure called *dendrogram*. To obtain a given partitioning of the data, the dendrogram has to be *cut* at a certain height. Points that remain interconnected after the cut will be considered part of the same cluster. Depending on the linkage criterion, several variants of hierarchical clustering have been defined; in particular, for the *complete-linkage* and the *single-linkage* clustering algorithms the distance between any two clusters $\mathcal{C}_1$ and $\mathcal{C}_2$ is respectively defined as the maximum and minimum Euclidean distance between all pairs of samples in $\mathcal{C}_1 \times \mathcal{C}_2$.

We denote sample-to-cluster assignments as a binary matrix $\mathtt{Y} \in \{0,1\}^{\mathsf{n}\times\mathsf{k}}$ where $\mathtt{Y}_{ik} = 1$ if the $i^{\rm th}$ sample is assigned to the $k^{\rm th}$ cluster. We also define the distance measure d_c between clusterings used in (1) as $d_c(\mathtt{Y}, \mathtt{Y}') = \|\mathtt{Y}\mathtt{Y}^\top - \mathtt{Y}'\mathtt{Y}'^\top\|_F$, where $\|\cdot\|_F$ is the Frobenius norm. This distance counts the number of times two samples have been clustered together in one clustering and not in the other, and viceversa. It is also known as the Mirkin metric, and its relationships with other clustering indices can be found here [19]. To employ this distance measure with hierarchical clustering, we have to specify an appropriate *dendrogram cut* criterion. Similarly to [15], we take the worst-case scenario for the adversary in order to obtain the minimum performance degradation incurred by the clustering algorithm under attack. This translates into selecting the dendrogram cut minimizing the distance d_c between the uncontaminated clustering result $\mathcal{C}$ and the one induced by the cut.

The optimization problem in (2) is hard to solve due to the presence of function $f_{\mathcal{D}}$, which has in general no analytic form and a discrete output. For this reason, we propose in the following some greedy optimization strategies aimed at finding a local maximum of the objective function, by adding one attack sample at a time, *i.e.*, $|\mathcal{A}'| = \mathsf{m} = 1$.

Poisoning attack. The idea of the optimization heuristic is to generate a set $\mathcal{S}$ of $2k$ candidate attack samples that could significantly compromise the clustering result, and retain as attack sample the one yielding the highest value of the objective function. These candidate samples are determined in a way to potentially induce a cluster to be split and to possibly induce one of those parts to be merged to another cluster. To populate the set $\mathcal{S}$, we determine for each cluster $\mathcal{V} \in \mathcal{C}$ a pair of points $(\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathcal{V} \times \mathcal{V}$ with maximum distance within the cluster (*a.k.a.* cluster diameter), *i.e.* $(\boldsymbol{x}_1, \boldsymbol{x}_2) \in \arg\max_{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{V}\times\mathcal{V}} ||\boldsymbol{x} - \boldsymbol{y}||$. Now, to induce the cluster $\mathcal{V}$ to be split we aim at increasing the diameter of $\mathcal{V}$. To this end, we determine two points $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ that are collinear with $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, but outside the segment joining them, *i.e.* $\boldsymbol{z}_1 = \boldsymbol{x}_1 + \frac{1}{2}\alpha_1 \boldsymbol{l}$ and $\boldsymbol{z}_2 = \boldsymbol{x}_2 - \frac{1}{2}\alpha_2 \boldsymbol{l}$ where $\boldsymbol{l} = (\boldsymbol{x}_1 - \boldsymbol{x}_2)/\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$ and $\alpha_{1,2} > 0$. The parameters $\alpha_{1,2}$ are determined as the minimum between the cluster diameter of $\mathcal{V}$ and the distance of $\boldsymbol{x}_{1,2}$ to the closest sample not in $\mathcal{V}$, *i.e.* $\alpha_{1,2} = \min_{\boldsymbol{z} \in (\mathcal{D}\setminus\mathcal{V}) \cup \{\boldsymbol{x}_{2,1}\}} \|\boldsymbol{x}_{1,2} - \boldsymbol{z}\|$. By computing $\boldsymbol{z}_{1,2}$ for each cluster, we obtain the set of candidate attack samples $\mathcal{S}$ (see Fig. 1).

We can now select the attack sample in $\boldsymbol{z} \in \mathcal{S}$ maximizing $g(\{\boldsymbol{z}\})$ and take $\mathcal{A}' = \{\boldsymbol{z}\}$ as the optimal (in an heuristic sense) attack strategy. We call this strategy *Extend (Best)*. For the sake of computational efficiency, we also experiment a strategy that approximates the clustering matrix $\mathtt{Y}'$ directly, without explicitly running the clustering algorithm for each candidate attack point. Specifically, given a candidate attack sample $\boldsymbol{z} \in \mathcal{S}$, we split its cluster $\mathcal{V}$ in two parts, one part containing the $|\mathcal{V}|/2$ closest points to $\boldsymbol{z}$ in $\mathcal{V}$. The newly constructed cluster containing $\boldsymbol{z}$ will be merged with another cluster $\mathcal{W} \in \mathcal{C}$, if the distance between $\boldsymbol{z}$ and $\mathcal{W}$ is smaller than the diameter of $\mathcal{V}$. Given this new clustering represented with the matrix $\mathtt{Y}'$, we can evaluate the objective value of $\boldsymbol{z}$ and again retain the one with the best value among the ones in $\mathcal{S}$. We call this strategy *Extend (Hard)*. Finally, we also experiment the computation of a soft version of the clustering matrix $\mathtt{Y}'$, where the element $\mathtt{Y}_{ki}$ holds the posterior probability of class k given sample $\boldsymbol{x}_i$ computed with the Bayes rule using a likelihood estimated with a Gaussian kernel density estimator (as done in [15]). The soft matrix $\mathcal{Y}'$ is computed for each candidate attack sample $z \in \mathcal{S}$ and the objective value in (2) evaluated. Finally, the sample with the best objective is retained. We call this last strategy *Extend (Soft)*.

5 Experiments

Following the experimental setting in [15], in this section we report an empirical evaluation of the effectiveness of our heuristic poisoning attacks on an artificial two-dimensional dataset, a realistic application example on malware clustering, and a high-dimensional problem involving clustering of handwritten digits. In each experiment, the proposed attacks are compared against the following two

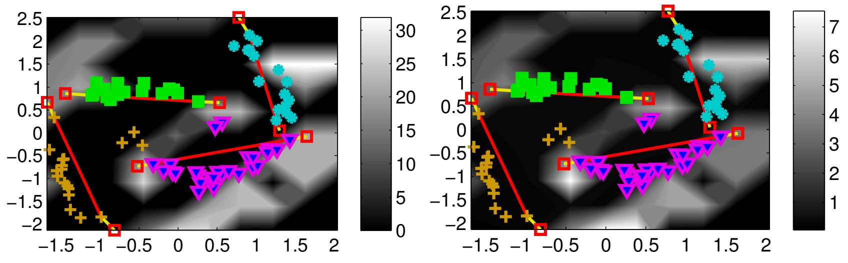

Fig. 1: Poisoning complete-linkage hierarchical clustering. In each plot, 100 samples grouped into 4 clusters are represented with different markers and colors. The red segments highlight the cluster's diameter, while the red squares are the candidate attack samples in $\mathcal{S}$. The objective function of (2), shown in the background for each greedy attack location ($|\mathcal{A}'| = 1$) is computed with hard (left plot) and soft assignments (right plot). Note how the set $\mathcal{S}$ of candidate attack points includes local maxima of the objective function.

strategies: *Random*, that chooses the attack point at random from the minimum box enclosing the data; and *Random (Best)*, that randomly selects $2k$ attack points from the same enclosing box (being k the number of clusters), and retains the one that maximizes the objective function. The rationale is to compare our methods against random-based strategies that exhibit similar computational complexities: *Extend (Best)* and *Random (Best)* require re-running the clustering algorithm $2k$ times at each iteration to evaluate the objective and select the best candidate attack sample, while *Extend (Hard)*, *Extend (Soft)* and *Random* select the attack point without re-running the clustering algorithm.

As for evaluating the attack effectiveness, we report the value of the objective function and the number of clusters obtained at each iteration, and the two measures *Split* and *Merge* defined in [15] as follows. Let $\mathcal{C}$ and $\mathcal{C}'$ be the initial and the final clustering of the samples in $\mathcal{D}$, and $\mathtt{C}$ a binary matrix whose elements $\mathtt{C}_{kk'}$ indicate the co-occurrence of at least one sample in the kth cluster of $\mathcal{C}$ and in the k'th cluster of C', then:

$$\text{Split} = \operatorname*{mean}_i \sum_j \mathtt{C}_{ij} \, , \; \text{Merge} = \operatorname*{mean}_j \sum_i \mathtt{C}_{ij} \, . \tag{3}$$

The rationale is that *Split* evaluates the extent to which the initial clusters are split across distinct final clusters, while *merge* evaluates to what extent the final clusters include points originally belonging to distinct initial clusters.

Artificial data. We consider here the standard two-dimensional banana-shaped dataset from PRTools.[2] A particular instance of this data is shown in Fig. 1. The number of clusters is set to $k = 4$, which corresponds to the ideal, untainted clustering $\mathcal{C}$ considered in this case. The experiment is repeated five times, each time by randomly sampling 80 data points, and adding up to 20 attack samples (*i.e.*, 20% of the data). As described in Sect. 4, the attack proceeds greedily

[2] `http://prtools.org`

	Banana (20%)		Malware (5%)		Digits (1%)	
	Split	*Merge*	*Split*	*Merge*	*Split*	*Merge*
Random	1.70 ± 0.27	1.56 ± 0.31	1.10 ± 0.09	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
Random (Best)	2.20 ± 0.32	1.52 ± 0.31	1.33 ± 0.12	1.31 ± 0.39	1.00 ± 0.00	1.00 ± 0.00
Extend (Hard)	2.10 ± 0.13	1.97 ± 0.41	1.33 ± 0.12	1.15 ± 0.08	1.00 ± 0.00	1.13 ± 0.18
Extend (Soft)	1.90 ± 0.38	1.81 ± 0.18	1.27 ± 0.09	1.11 ± 0.17	1.60 ± 0.15	1.07 ± 0.15
Extend (Best)	2.15 ± 0.22	2.05 ± 0.11	1.83 ± 0.00	1.36 ± 0.14	1.27 ± 0.28	1.07 ± 0.15
Optimal	2.00 ± 0.31	2.28 ± 0.41				

Table 1: Split and Merge averaged values and standard deviations for the Banana-shaped dataset (at 20% poisoning), the Malware dataset (at 5% poisoning), and the Digit dataset (at 1% poisoning).

by adding one sample at a time. After adding each attack sample, we allow the clustering algorithm to change the number of clusters from 2 to 50. The criterion used to determine the number of clusters is to minimize the distance of the current partitioning with the clustering in the absence of attack, as explained in Sect. 4. *Extend (Soft)* estimates soft clustering assignments $\mathtt{Y}'$ using a Gaussian KDE, whose kernel bandwidth h is set as the average distance between each pair of data points, yielding $h \approx 2$ in each run. On this dataset, we also compare our heuristic attacks with a computationally-expensive greedy attack that selects the best attack point at each iteration (by re-running the clustering algorithm and evaluating the objective) from an equally-spaced grid of 50×50 points in $[-2.5, 2.5] \times [-2.5, 2.5]$. This attack can be thus retained very close to the optimal greedy attack, and we refer to it as *Optimal (Grid Search)*.

Results are reported in Fig. 2 (first column). From the top plot, one may note how *Extend (Best)* is able to achieve very close values of the objective function to those attained by *Optimal (Grid Search)*, denoting the effectiveness of the considered candidate attack points. *Random (Best)* performs only slightly worse, in this case, due to the fact that the feature space is only two-dimensional and bounded, and that the local maxima of the objective function typically cover large areas (see, *e.g.*, Fig. 1). *Extend (Hard)* and *Extend (Soft)* perform similarly to *Random (Best)*, and better than *Random*, confirming to some extent that predicting the output of the complete-linkage clustering algorithm after the addition of a given data point may not always be as trivial as assumed by our heuristics. The bottom plot shows how the number of selected clusters vary as the attack progresses. The main effect is an oscillation of the number of clusters, highlighting that initial clusters are fragmented, and that the resulting fragments are then merged to form distinct clusters. This effect is also confirmed by the *Split* and *Merge* values reported in Table 1.

Malware clustering. We focus here on a real-world application example involving the behavioral malware clustering algorithm proposed in [3]. Its goal is to obtain malware clusters that can be used to automatically generate network signatures that can in turn spot botnet command-and-control (C&C) and other malware-related communications at the network perimeter. With respect to the original algorithm, we made the same simplifications done in [15], and use the complete-linkage criterion instead of the single linkage. Each malware is repre-

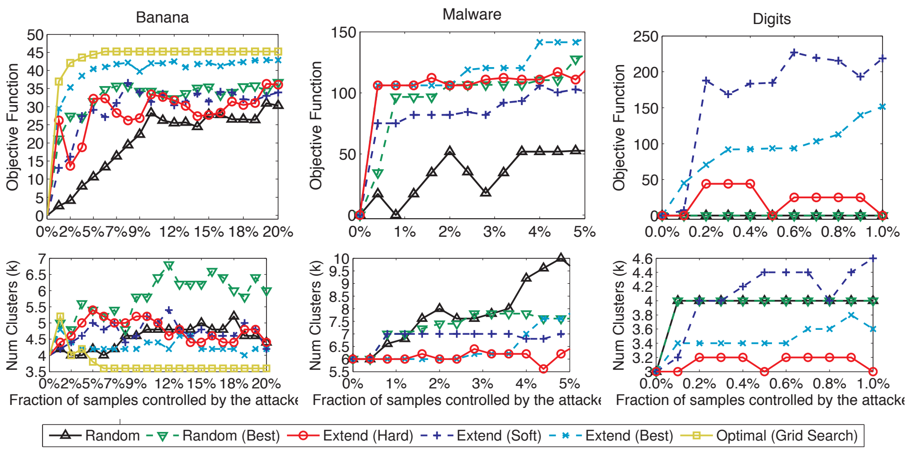

Fig. 2: Results for the Banana-shaped (first column), the Malware (second column), and the Digit (third column) datasets. Plots in the top and the bottom row respectively show how the objective function $d_c(\mathtt{Y}, \mathtt{Y}')$ and the number of clusters vary as the fraction of attack samples increases.

sented by six feature values, normalized in $[0, 1]$: (i) number of GET requests; (ii) number of POST requests; (iii) average URL length; (iv) average number of parameters in the request; (v) average amount of data sent by POST requests; and (vi) average response length. We use a dataset of 1,000 malware samples chosen from *Dataset 1* of [3], which includes malware collected from different sources, like MWCollect[3] and Malfease[4]. The experiments are repeated five times, by randomly selecting a subset of 475 malware samples in each run. As in [3, 15], the initial set of clusters $\mathcal{C}$ is selected as the partitioning that minimizes the value of the Davies-Bouldin Index [20], yielding approximately 9 clusters in each run. During the attack, a maximum of 25 attack samples are added (*i.e.*, 5% of the data), while the clustering algorithm can vary the number of clusters from 2 to 50. The value of h for the KDE used in *Extend (Soft)* is set to 0.2, as it is close to the average distance between each pair of samples in each run.

Results are shown in Fig. 2 (second column). The effect of the attack is essentially the same as for the Banana-shaped data, as witnessed by the variation in the number of clusters (bottom plot) and from the values of *Split* and *Merge* in Table 1: the initial clusters are fragmented to form different clusters. *Extend (Best)* outperforms again the other methods, as shown by the values of the objective function (top plot). *Extend (Soft)* performs slightly worse than *Random (Best)*, instead, while *Random* tends to increase the number of clusters but not the objective function. This happens since most of the attack points are clustered separately in new, additional clusters without affecting the initial clusters at all.

[3] Collaborative Malware Collection and Sensing, `https://alliance.mwcollect.org`.

[4] Project Malfease, `http://malfease.oarci.net`.

Handwritten digits. We finally consider clustering of handwritten digits from the MNIST dataset [21],[5] where each digit is represented as a grayscale image of 28×28 pixels. Each pixel is considered here as a feature value (normalized in $[0, 1]$ by dividing its value by 255). The feature space has thus 784 dimensions. For simplicity, as in [15], we restrict our analysis on a subset of the data made up of the three digits '0', '1', and '6'. Three initial clusters, each representing one of the considered digits, are obtained by first computing the average digit for each class, and then selecting 700 samples per class, by retaining the closest samples to the corresponding average digit. We run the experiments five times, each time by randomly choosing 330 samples per digit from the corresponding set of 700 pre-selected samples. While the attack proceeds, the attacker can inject up to 10 attack samples (1% of the data), while the clustering algorithm can select from 2 to 100 clusters. The value of h for the KDE used in *Extend (Soft)* is set as $h \approx 1$, based on the average distance between all pairs of samples in each run.

Results are shown in Fig. 2 (third column). Notably, in this case *Extend (Soft)* outperforms the other methods, including *Extend (Best)*. *Extend (Soft)* deals indeed with less sharp variations of the objective function, that may in turn allow it to identify a better combination of attack points in the end. The random-based methods are completely ineffective, instead. This is due to the high dimensionality of the feature space, which drastically reduces the chances of finding a good local maxima by selecting the attack points at random. In particular, it can be noted from the bottom plot in Fig. 2 (third column) that the number of clusters induced by the random-based attacks increases from 3 to 4, while the objective function (top plot) remains at zero, and *Split* and *Merge* do not vary (see Table 1). This essentially means that all the corresponding attack points are clustered together in a single cluster, separated from the initial ones. Finally, it is worth noting that *Merge* approximately equals 1 in Table 1 for all the considered attacks, highlighting that even the effective (non-random) attacks are mostly able to split the initial clusters but not to form clusters that aggregate samples initially belonging to different clusters.

6 Conclusions and future work

In this paper, we addressed the problem of evaluating the security of clustering algorithms in adversarial settings. We showed with real-world experiments that complete-linkage clustering may be significantly vulnerable to deliberate attacks. In general, finding the optimal attack strategy for an arbitrary clustering algorithm is a difficult problem. Therefore, we have to rely on heuristic algorithms in order to carry out our analysis. For the sake of efficiency, these heuristics should be heavily dependent on the targeted clustering algorithm, as in our case. Nevertheless, it would be interesting to devise more general methods that can use the clustering algorithm as a black box and find a solution by performing a stochastic search on the solution space (*e.g.*, by simulated annealing), or an educated exhaustive search (*e.g.*, by using branch-and-bound techniques).

[5] Publicly available at `http://cs.nyu.edu/~roweis/data.html`.

Acknowledgements. This work has been partly supported by the project "Security of pattern recognition systems in future internet" (CRP-18293) funded by Regione Autonoma della Sardegna, L.R. 7/2007, Bando 2009.

References

1. Perdisci, R., Corona, I., Giacinto, G.: Early detection of malicious flux networks via large-scale passive DNS traffic analysis. IEEE Trans. Dependable and Secure Comp. **9**(5) (2012) 714–726
2. Pouget, F., Dacier, M., Zimmerman, J., Clark, A., Mohay, G.: Internet attack knowledge discovery via clusters and cliques of attack traces. J. of Information Assurance and Security **1**(1) (2006)
3. Perdisci, R., Ariu, D., Giacinto, G.: Scalable fine-grained behavioral clustering of http-based malware. Computer Networks **57**(2) (2013) 487 – 500
4. Rieck, K., Trinius, P., Willems, C., Holz, T.: Automatic analysis of malware behavior using machine learning. J. Comput. Secur. **19**(4) (2011) 639–668
5. Hanna, S., Huang, L., Wu, E., Li, S., Chen, C., Song, D.: Juxtapp: a scalable system for detecting code reuse among Android applications. In: Proc. 9th Int'l Conf. Det. Intrusions and Malware, and Vulnerability Assessment. (2013) 62–81
6. Burguera, I., Zurutuza, U., Nadjm-Tehrani, S.: Crowdroid: behavior-based malware detection system for android. In: SPSM'11. (2011) 15–26
7. Spitzner, L.: Honeypots: Tracking Hackers. Addison-Wesley Professional (2002)
8. Biggio, B., Fumera, G., Roli, F.: Security evaluation of pattern classifiers under attack. IEEE Trans. Knowledge and Data Eng. **26**(4) (2014) 984–996
9. Brückner, M., Kanzow, C., Scheffer, T.: Static prediction games for adversarial learning problems. J. Mach. Learn. Res. **13** (2012) 2617–2654
10. Huang, L., Joseph, A.D., Nelson, B., Rubinstein, B., Tygar, J.D.: Adversarial machine learning. In: ACM Workshop AISec '11. (2011) 43–57
11. Barreno, M., Nelson, B., Sears, R., Joseph, A.D., Tygar, J.D.: Can machine learning be secure? In: ASIACCS '06. (2006) 16–25
12. Großhans, M., Sawade, C., Brückner, M., Scheffer, T.: Bayesian games for adversarial regression problems. In: ICML. Volume 28. (2013)
13. Dutrisac, J.G., Skillicorn, D.: Hiding clusters in adversarial settings. In: ISI'08. (2008) 185–187
14. Skillicorn, D.B.: Adversarial knowledge discovery. IEEE Intelligent Systems **24** (2009) 54–61
15. Biggio, B., Pillai, I., Rota Bulò, S., Ariu, D., Pelillo, M., Roli, F.: Is data clustering in adversarial settings secure? In: ACM Workshop AISec'13. (2013) 87–98
16. Biggio, B., Nelson, B., Laskov, P.: Poisoning attacks against support vector machines. In: ICML. (2012)
17. Kolcz, A., Teo, C.H.: Feature weighting for improved classifier robustness. In: CEAS. (2009)
18. Jain, A.K., Dubes, R.C.: Algorithms for clustering data. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1988)
19. Meilă, M.: Comparing clusterings: An axiomatic view. In: ICML. (2005) 577–584
20. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. Journal of Intelligent Information Systems **17**(2-3) (December 2001) 107–145
21. LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., Drucker, H., Guyon, I., Müller, U., Säckinger, E., Simard, P., Vapnik, V.: Comparison of learning algorithms for handwritten digit recognition. In: Int'l Conf. on Artificial Neural Networks. (1995) 53–60