Conformal Clustering and Its Application to Botnet Traffic

Giovanni Cherubin[1,2](✉), Ilia Nouretdinov[1], Alexander Gammerman[1], Roberto Jordaney[2], Zhi Wang[2], Davide Papini[2], and Lorenzo Cavallaro[2]

[1] Computer Learning Research Centre and Computer Science Department, Royal Holloway University of London, Egham Hill, Egham, Surrey TW20 OEX, UK
`Giovanni.Cherubin.2013@live.rhul.ac.uk`

[2] Systems Security Research Lab and Information Security Group, Royal Holloway University of London, Egham Hill, Egham, Surrey TW20 OEX, UK

Abstract. The paper describes an application of a novel clustering technique based on Conformal Predictors. Unlike traditional clustering methods, this technique allows to control the number of objects that are left outside of any cluster by setting up a required confidence level. This paper considers a multi-class unsupervised learning problem, and the developed technique is applied to bot-generated network traffic. An extended set of features describing the bot traffic is presented and the results are discussed.

Keywords: Information security · Botnet · Confident prediction · Conformal prediction · Clustering

1 Introduction

Within the past decade, security research begun to rely heavily on machine learning to develop new techniques to help in the identification and classification of cyber threats. Specifically, in the area of network intrusion detection, botnets are of particular interest as these often hide within legitimate applications traffic. A botnet is a network of infected computers controlled by an attacker, the *botmaster*, via the Command and Control server (*C&C*). Botnets are a widespread malicious activity among the Internet, and they are used to perform attacks such as phishing, information theft, click-jacking, and Distributed Denial of Service (DDoS). Bots detection is a branch of network intrusion detection which aims at identifying botnet infected computers (*bots*). Recent studies, such as [9], rely on clustering and focus their analysis on high level characteristics of network traffic (*network traces*) to distinguish between different botnet threats. We take an inspiration from this approach, and apply Conformal Clustering, a technique based on Conformal Predictors (CP) [10], with an extended set of features. We produce clusters from unlabelled training examples; then on a test set we associate a new object with one of the clusters. Our aim is to achieve a high intra-cluster similarity in terms of application layer protocols (**http**, **irc** and **p2p**).

A. Gammerman et al. (Eds.): SLDS 2015, LNAI 9047, pp. 313–322, 2015.
DOI: 10.1007/978-3-319-17091-6_26

In previous work [4,5,8] the conformal technique was applied to the problem of anomaly detection. It also demonstrated how to create clusters: a prediction set produced by CP was interpreted as a set of possible objects which conform to the dataset and therefore are not anomalies; however the prediction set may consist of several parts that are interpreted as clusters, where the significance level is a "trim" to regulate the depth of the clusters' hierarchy. The work in [8] was focused on a binary (anomaly/not anomaly) unsupervised problem. This paper generalizes [8] for a multi-class unsupervised learning problem. This includes the problem of clusters creation, solved here by using a *neighbouring rule*, and the problem of evaluating clusters accuracy, solved by using Purity criterion. For evaluating efficiency we use Average P-Value criterion, earlier presented in [8].

In our approach we extract features from network traces generated by a bot. Then we apply preprocessing and dimensionality reduction with t-SNE; the use of t-SNE, previously used in the context of Conformal Clustering in [8], is here needed for computational efficiency, since the way we here apply Conformal Clustering has a time complexity increasing as $\Delta \times \ell^d$, where Δ is the complexity to calculate a P-Value and varies respect to the chosen non-conformity measure, ℓ is the number of points per side of the grid, and d is the number of dimensions. This complexity can be reduced further for some underlying algorithms. The dataset is separated into *training* and *test* sets, and clustering is applied to *training* set. After this the testing objects are associated with the clusters.

An additional contribution made by this paper is related to feature collection: an algorithm based on *Partial Autocorrelation Function* (PACF) to detect a periodic symbol in binary time series is proposed.

2 Data Overview

The developed system is run on *network traces* produced by different families of botnets. A network trace is a collection of network packets captured in a certain window of time. A network trace can be split into *network flows* (netflows), a collection of packets belonging to the same communication. A netflow contains high level information of the packet exchange, such as the communicating IP addresses and ports, a timestamp, the duration, and the number and size of exchanged packets (transmitted, received and total). As [9] suggested, a system using only netflows, thus not modelling the content of the packets, is reliable even when the traffic between *bot* and *C&C* is encrypted.

In the dataset creation phase we extract a feature vector from every network trace. A feature vector is composed of the following 18 features: *median* and *MAD* (Mean Absolute Deviation) of *netflows duration*, *median* and *MAD* of *exchanged bytes*, communication *frequency*, use *percentage* of *TCP* and *UDP* protocols, use *percentage* of *ports* respectively in three ranges[1], *median* and *MAD* of *transmitted* and *received bytes* considering the *bot* as source, *median* and *MAD* of *transmitted* and *received bytes* considering the connection initiator

[1] We base these ranges on the standard given by *IANA*: System Ports (0–1023), User Ports (1024–49151), and Dynamic and/or Private Ports (49152–65535).

as source. Duration of netflows, transmitted received and total exchanged bytes have been used in past research for bots detection [9]; these quantities were usually modelled by their mean value. We model them here by using median and MAD, since normal distribution is not assumed; furthermore, median is invariant in non-linear rescaling.

Since, as others observed [9], in most botnet families *bots* communicate periodically, we introduce the feature *frequency* which takes into account the period of communication, if any. We can detect the period of communication by looking at the netflows timestamps, and constructing a binary time series y_t as:

$$y_t = \begin{cases} 1, \text{if flow occurred at time } t \\ 0, \text{if no flow occurred at time } t \end{cases} \qquad \text{for } t \text{ in } \{0,1,...\}, \tag{1}$$

where time t is measured in seconds.

For a period T we then define the feature frequency to be:

$$\text{frequency} = \begin{cases} 0, & \text{if } T = \infty (\text{no period}) \\ 1/T, & \text{otherwise} \end{cases} \qquad \text{for } T > 0.$$

which is consistent whenever a period is found or not. Later in this section we introduce a novel approach for detecting periodicity within a binary time series defined as in Eq. 1.

The dataset we use contains traffic from 9 families of botnets and we will group them with respect to the application layer protocol they are based on. We hence define three classes of botnets: **http**, **irc** and **p2p** based. Our goal is to produce clusters containing objects from the same class. In this paper *objects* and *feature vectors* refer to the same concept; *example* refers to a labelled object.

2.1 Periodicity Detection Based on PACF

Equation (1) defines a binary time series y_t, such that $y_t = 1$ when some event has happened at time t, $y_t = 0$ otherwise. Our goal is to check whether the time series contains a periodic event, and if so we want to determine the period T of this event. The study [1] calls this task 'Symbol Periodicity' detection. Past studies in *bots* detection, such as [9], approached the problem by using Power Spectral Density (PSD) of the Fast Fourier Transform of the series.

We propose an algorithm based on Partial Autocorrelation Function (PACF) for achieving this goal, which is simple to implement, well performing under noisy conditions[2], and which may be extended to capture more than one periodic event.

Given a generic time series u_t, the PACF of lag k is the autocorrelation between u_t and u_{t-k}, removing the contribution of the lags in between, $t-1$ to $t-k+1$. The PACF coefficients ϕ_{kk} between u_t and u_{t-k}, are defined as [2]:

[2] By noise we mean events which can happen at any time t; let W=Integer($L * \nu$), where L is the length of y_t and $\nu \in [0,1]$ the percentage of noise (noise level), we simulate noise in our artificial time series by setting $y_t = 1$ for a number W of positions t uniformly sampled in $[1, L]$.

$$\phi_{11} = \rho_1$$
$$\phi_{22} = (\rho_2 - \rho_1^2)/(1 - \rho_1^2)$$
$$\phi_{kk} = \frac{\rho_k - \sum_{j=1}^{k-1} \phi_{k-1,j}\rho_{k-j}}{1 - \sum_{j=1}^{k-1} \phi_{k-1,j}\rho_j} \quad , k = 3, 4, 5, ...$$

where ρ_i is the autocorrelation for lag i, and $\phi_{kj} = \phi_{k-1,j} - \phi_{kk}\phi_{k-1,k-j}$ for $j = 1, 2, ..., k-1$.

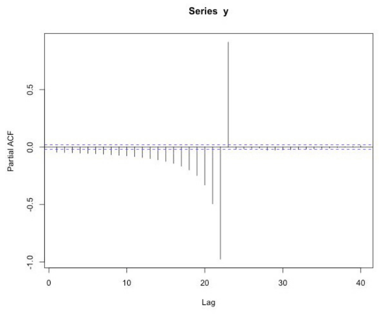

Fig. 1. PACF over a binary time series with one periodic event of periodicity $T = 23$ exhibits a peak at lag $k = 23$

From our experiments on artificial binary time series with one periodic event we noticed that PACF on them presents a high peak at the lag corresponding to the period T. For instance, Fig. 1 is the PACF over an artificial binary time series of 10^4 elements, having a periodic event with period $T = 23$. This fact holds true even under noisy conditions. We run our experiments on artificial binary time series of length 10^4, testing all the periods in $\{3, 4, ..., 100\}$, inserting different percentages of white noise and computing PACF over 150 lags. The period is estimated as the lag at which PACF is maximum, and checked if equal to the true period. The experiments show that for a time series of length $L = 10^4$ the accuracy remains 1 for the noise level $\nu = 0.05$ and becomes 0.83 for $\nu = 0.2$. If $L = 10^5$, then it is pure for up to $\nu = 0.1$, while for noise level $\nu = 0.2$ it is 0.96.

So far we assumed to know that a periodicity existed in y_t. In case, as for our dataset, we do not assume *a priori* a periodicity exists, we can use a threshold for performing detection. We noticed that relevant peaks in PACF are larger than 0.65. Hence we compute PACF, get the maximum of it, and consider its lag to be a valid period only if its value is larger than a threshold $\vartheta = 0.65$; otherwise, we consider the time series to be aperiodic.

3 Conformal Clustering Approach

We perform the following steps on a dataset X created as in Sect. 2:

1. Preprocessing of X;
2. Dimensionality reduction with t-SNE (produces Z);
3. Z is split into *training* (100) and *test* (34) set;
4. Conformal Clustering on *training* set;
5. Test objects are associated to the clusters following a neighbouring rule.

After this, evaluation criteria *Average P-Value* (APV) and *Purity* are computed.

3.1 Preprocessing and Dimensionality Reduction

We apply log-transformation to bytes-related features of the dataset because they take values in a large range. t-SNE requires the dataset to be consistently normalized before application. We apply normalization in $[0, 1]$ to all features u:

$$u_{01} = \frac{u - min(u)}{max(u) - min(u)},$$

Our application of Conformal Clustering requires to compute ℓ^d P-Values, where ℓ is the number of points per grid side and d is the number of features. We use t-SNE, as [8] did before, as a dimensionality reduction algorithm before clustering.

T-SNE [6] is originally developed as a visualization algorithm for high dimensional data. However, thanks to its ability of keeping far in the low dimensional projection dissimilar objects of the high dimensional data, and *vice versa* for similar objects, it reveals to be a good dimensionality reduction algorithm. We can trim a few parameters of it, such as *perplexity* and *distance metric*. Perplexity may be viewed as the number of effective neighbours each object has in high dimensional space; distance metric is a similarity measure between objects in high and low dimensional space. For our experiments we used *Euclidean Distance* as a distance metric, and trimmed the value of perplexity. By applying t-SNE to our preprocessed dataset we obtain $Z = \{z_1, z_2, ..., z_N\}$, a 2-D projection of it.

3.2 Conformal Clustering

CP allows to have a confidence measure on predictions. Given a bag of observations $D = \wr z_1, .., z_{n-1} \wr$, $z_i \in \mathbf{Z}$, a new object z and a significance level ε, CP allows to determine if z comes from D with an error on the long run of at most ε. The only property required by CP is that $\wr z_1, .., z_{n-1} \wr$ are exchangeable. Note that exchangeability property is weaker than *iid*, since:

$$iid \implies exchangeable.$$

We define a non-conformity measure $A : \mathbf{Z}^{(*)} \times \mathbf{Z} \longmapsto \mathbb{R}$, to be a function which accepts a bag of objects and an object z_i, and returns a scalar representing how much z_i is conform to the other objects. The result of CP is a P-Value p_n and a boolean answer indicating if the new object is conform to D. Follows a description of the algorithm.

```
Data: Bag of objects $D = \wr z_1, .., z_{n-1} \wr$, non-conformity measure $A$,
      significance level $\varepsilon$, a new object $z$
Result: P-Value $p_n$, True if $z$ is conform to training objects

Set provisionally $z_n = z$ and $D = \wr z_1, .., z_n \wr$
for $i \leftarrow 1$ to $n$ do
    $\alpha_i \leftarrow A(D \setminus z_i, z_i)$
end
$\tau = U(0,1)$
$p_n = \frac{\#\{i:\alpha_i > \alpha_n\} + \#\{i:\alpha_i = \alpha_n\}\tau}{n}$
if $p_n > \varepsilon$ then
    Output True
else
    Output False
```

Algorithm 1. Conformal Prediction using new examples alone

In the algorithm, τ is sampled in *Uni(0,1)* to obtain a *smoothed conformal predictor*, as suggested by [3]. Smoothed conformal predictor is *exactly valid*, which means that the probability of error equals ε on the long run. Confidence level is defined as $1 - \varepsilon$, which is the probability for a new example generated by the same distribution to be covered by the prediction set.

Conformal Clustering uses CP to create a set of predictions Γ^{ε}, which contains the new objects which are conform to old objects in D. These objects are then clustered by using a *neighbouring rule*. Follows a description of the algorithm. A d-dimensional grid of ℓ equally spaced points per side is created within the feature values range; d is the number of features. P-Values are computed using CP for each point of the grid considering them as new objects z respect to the bag of *training* observations Z. Conform points are predicted respect to a significance value ε. These points are then clustered by using a neighbouring rule: two points z_i, z_j are in the same cluster if they are neighbours on the grid. In this context, significance level ε can be used as a trim to regulate the depth of the clusters (see Fig. 2). In fact, ε is responsible for the percentage of instances left outside any clusters; the more of them there are, the less connected to each other are the remaining ones, which leads to a deeper level of hierarchical clustering.

Once created the clusters, *test* objects are associated to them by using the rule: a *test* object is from a cluster if its distance from one of the cluster point is smaller or equal to the grid unit. If a point is associated to more than one clusters, these clusters are merged.

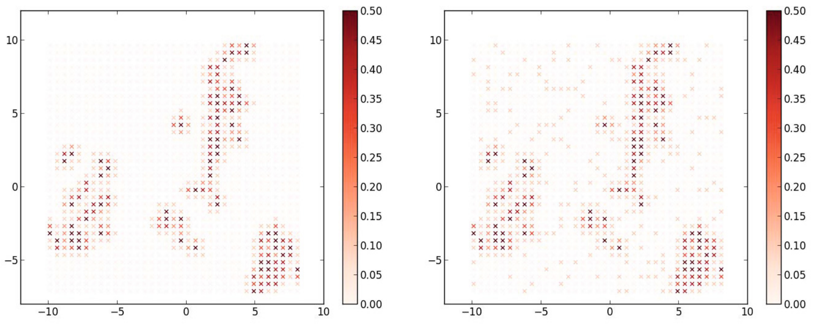

Fig. 2. Trimming significance level ε on P-Values grid. Non-conformity measure *k-NN* with $k = 1$ (*left*) and *KDE* with $h = 0.1$ (*right*) are used. Coloured points are the predicted ones with respect to a significance level.

3.3 Non-conformity Measures

A non-conformity measure is a function $A : \mathbf{Z}^{(*)} \times \mathbf{Z} \to \mathbb{R}$, where $\mathbf{Z}^{(*)}$ is the set of possible bags over $\mathbf{Z}$. CP is valid for every non-conformity measure, but some of them are more efficient; *efficiency* is later presented in this document as a performance criterion. In our experiments we used non-conformity measures *k-Nearest Neighbours* (k-NN) and *Kernel Density Estimation* (KDE).

A_i, k-NN non-conformity measure for object z_i, given δ_{ij} to be the j-th smallest distance between z_i and the objects in $\wr z_1, ..., z_n \wr \setminus z_i$, is:

$$A_i = \sum_{j=1}^{k} \delta_{ij},$$

where k is the chosen number of neighbours.

A_i, KDE non-conformity measure for a kernel function $K : \mathbb{R}^d \to \mathbb{R}$, where d is the number of features, is:

$$A_i = -\left(\frac{1}{nh^d} \sum_{j=1}^{n} K\left(\frac{z_i - z_j}{h}\right)\right),$$

where h is the kernel bandwidth. We here use a Gaussian kernel:

$$K(u) = \frac{1}{2\pi} e^{-\frac{1}{2}u^2}.$$

4 Results

We measure the performances of our system by using two criteria: *Purity* and *Average P-Value* (APV). Purity is an accuracy criterion which measures the homogeneity of clusters. It is the weighted sum, for all the clusters, of the percentage of objects from the most frequent class respect to the cluster cardinality. Purity [7] is formally defined as:

$$Purity(\Omega, \mathcal{C}) = \frac{1}{n} \sum_k \max_j \#\{\omega_k \cap c_j\},$$

where $\Omega = \{\omega_1, ..., \omega_K\}$ is the set of clusters, and $\mathcal{C} = \{c_1, ..., c_J\}$ is the set of classes. For a parameter set we compare Purity for a fixed ε. The use of Purity is usually avoided for evaluating clustering algorithms such as *K-Means* or *Hierarchical Clustering*, because it is highly conditioned by the number of clusters. We although employ this criterion here because the way we trim Conformal Clustering parameters does not essentially influence the number of clusters for the same ε. APV is an efficiency criterion introduced by [8]; efficiency in CP measures the size of the prediction set Γ^ε. APV is defined to be the mean of P-Values of the P-Values grid. We want this parameter to be as small as possible.

Table 1. Conformal Clustering respectively with k-NN trimming k (*above*) and KDE gaussian kernel trimming bandwidth h (*below*)

k	1	2	3	4	5	6	7	8	9	10
APV	**0.129**	0.139	0.141	0.147	0.160	0.167	0.176	0.183	0.189	0.193
Purity ($\varepsilon = 0.2$)	**0.990**	0.970	0.970	0.960	0.960	0.960	0.960	0.940	0.920	0.920

h	0.001	0.005	0.01	0.05	0.1	0.2	0.3	0.4	0.5	1.0
APV	0.404	0.332	0.299	0.165	**0.130**	0.138	0.146	0.155	0.165	0.211
Purity ($\varepsilon = 0.2$)	**1.000**	0.980	**1.000**	0.990	0.990	0.990	0.970	0.970	0.950	0.920

In our first experiment we apply Conformal Clustering to a t-SNE projection with perplexity 50. Table 1 shows the results of using non-conformity measures k-NN and KDE trimming their parameters. The significance level was set to $\varepsilon = 0.2$. We notice that KDE manages to achieve perfect Purity, with the penalty of a larger APV. Non-conformity measure k-NN obtains its best Purity and APV for the value $k = 1$. K-NN also gets for this value the best APV overall. Further we will see from Fig. 3 that it is more robust on perplexity than KDE. KDE efficiency quickly fails for small bandwidth ($h < 0.05$), which may be due to high computational precision required by this method. Figure 2 shows k-NN and KDE predictions on P-Values grid respect to the significance level.

In the second experiment we inspect how t-SNE perplexity influences the results. We experiment with many values of perplexity, apply Conformal Clustering with k-NN ($k = 1$) and KDE (bandwidth $h = 0.1$) non-conformity measures, and measure APV and Purity. Figure 3 shows the results for k-NN and KDE non-conformity measures. The trends for Purity are both ascendant as perplexity increases, and after approximately 20 they get on average very close to perfect Purity. As for the efficiency, while for k-NN APV does not look strictly

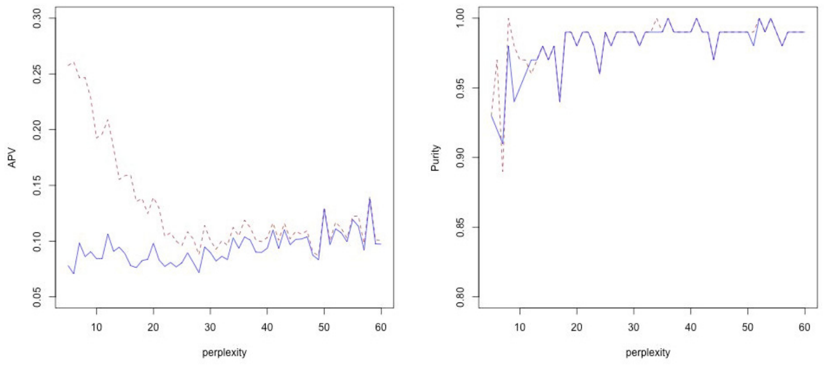

Fig. 3. Trimming t-SNE perplexity and evaluating Conformal Clustering for non-conformity measures KDE (gaussian kernel, bandwidth $h = 0.1$) in dashed brown line, and k-NN ($k = 1$) in blue line. We evaluate criteria APV (*left*) and Purity (*right*).

correlated to perplexity (despite a slow increasing trend as perplexity grows), for KDE APV decreases, which indicates that for KDE perplexity value can be set larger to get better performances. These results indicate k-NN to be a better candidate than KDE as a non-conformity measure for Conformal Clustering, but further experiments on different training sets should investigate this.

5 Conclusions and Future Work

We described here a novel clustering technique called Conformal Clustering, extending previous research for a multi-class unsupervised learning setting. We presented its application for clustering network traffic generated by *bots*. A neighbouring rule was introduced for creating clusters from a prediction set of objects. Purity and APV criteria were used for evaluating clustering performances. We also proposed a novel algorithm based on PACF for detecting a single periodicity in a binary time series.

For future research we plan to develop various criteria of accuracy and efficiency. Our aim is also to reduce the computational complexity of Conformal Clustering. If the problem does not require reduction of dimensionality then an application of t-SNE can be avoided. In high dimension a potential alternative to t-SNE is using an irregular (random) grid instead of the current one, but this would need some revision of the clustering definition. The methodology was shown on the example of two non-conformity measures, each having two parameters (including perplexity), and we compared their quality. Using more data, this can be extended further to find the best approach. We also plan to develop new non-conformity measures related to underlying algorithms like BotFinder [9]. The experiments for the proposed periodicity detection algorithm computed PACF for a number of lags close to the period to detect; future studies may investigate its performance when using a different number of lags. PACF-based periodicity detection can be also extended for detecting more than one period in an instance.

From a security perspective, our dataset contained only network traces from infected computers. One more direction is to consider extending this approach to model family-based clusters of botnets and on assessing its ability to detect bot-infected machines by monitoring real-world unknown network traffic.

Acknowledgments. This work is supported by EPSRC grants EP/K033344/1 ("Mining the Network Behaviour of Bots"), EP/K006266/1 and by the National Natural Science Foundation of China (No.61128003) grant. We are grateful to Vladimir Vovk, Zhiyuan Luo and Hugh Shanahan for useful discussions.

References

1. Elfeky, M.G., Aref, W.G., Elmagarmid, A.K.: Periodicity detection in time series databases. IEEE Transactions on Knowledge and Data Engineering **17**(7), 875–887 (2005)
2. Enders, W.: Applied econometric time series (1995)
3. Gammerman, A., Vovk, V.: Hedging predictions in machine learning. The Computer Journal **50**(2), 151–163 (2007)
4. Laxhammar, R., Falkman, G.: Sequential conformal anomaly detection in trajectories based on hausdorff distance. In: 2011 Proceedings of the 14th International Conference on Information Fusion (FUSION), pp. 1–8. IEEE (2011)
5. Lei, J., Rinaldo, A., Wasserman, L.: A conformal prediction approach to explore functional data. Annals of Mathematics and Artificial Intelligence, pp. 1–15 (2013)
6. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of Machine Learning Research **9**(2579–2605), 85 (2008)
7. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to information retrieval, vol. 1. Cambridge University Press, Cambridge (2008)
8. Smith, J., Nouretdinov, I., Craddock, R., Offer, C., Gammerman, A.: Anomaly Detection of Trajectories with Kernel Density Estimation by Conformal Prediction. In: Iliadis, L., Maglogiannis, I., Papadopoulos, H., Sioutas, S., Makris, C. (eds.) Artificial Intelligence Applications and Innovations. IFIP AICT, vol. 437, pp. 271–280. Springer, Heidelberg (2014)
9. Tegeler, F., Fu, X., Vigna, G., Kruegel, C.: Botfinder: Finding bots in network traffic without deep packet inspection. In: Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, pp. 349–360. ACM (2012)
10. Vovk, V., Gammerman, A., Shafer, G.: Algorithmic learning in a random world. Springer (2005)