A Novel Big Data Analytics Approach for Supporting Cyber Attack Detection via Non-linear Analytic Prediction of IP Addresses

Alfredo Cuzzocrea, Enzo Mumolo, Edoardo Fadda, Marco Tessarotto

A Novel Big Data Analytics Approach for Supporting Cyber Attack Detection via Non-linear Analytic Prediction of IP Addresses

Alfredo Cuzzocrea[1,2(✉)], Enzo Mumolo[3], Edoardo Fadda[4,5], and Marco Tessarotto[3]

[1] University of Calabria, Rende, Italy
alfredo.cuzzocrea@unical.it
[2] LORIA, Nancy, France
[3] University of Trieste, Trieste, Italy
mumolo@units.it, marco.tessarotto@regione.fvg.it
[4] Politecnico di Torino, Torino, Italy
edoardo.fadda@polito.it
[5] ISIRES, Torino, Italy

Abstract. Computer network systems are often subject to several types of attacks. For example the distributed Denial of Service (DDoS) attack introduces an excessive traffic load to a web server to make it unusable. A popular method for detecting attacks is to use the sequence of source IP addresses to detect possible anomalies. With the aim of predicting the next IP address, the Probability Density Function of the IP address sequence is estimated. Prediction of source IP address in the future access to the server is meant to detect anomalous requests. In other words, during an access to the server, only predicted IP addresses are permitted and all others are blocked. The approaches used to estimate the Probability Density Function of IP addresses range from the sequence of IP addresses seen previously and stored in a database to address clustering, normally used by combining the K-Means algorithm. Instead, in this paper we consider the sequence of IP addresses as a numerical sequence and develop the nonlinear analysis of the numerical sequence. We used nonlinear analysis based on Volterra's Kerners and Hammerstein's models. The experiments carried out with datasets of source IP address sequences show that the prediction errors obtained with Hammerstein models are smaller than those obtained both with the Volterra Kernels and with the sequence clustering by means of the K-Means algorithm.

Keywords: Cyber attack · Distributed Denial of Service · Hammerstein models

A. Cuzzocrea—This research has been made in the context of the Excellence Chair in Computer Engineering – Big Data Management and Analytics at LORIA, Nancy, France.

O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12249, pp. 978–991, 2020.
https://doi.org/10.1007/978-3-030-58799-4_70

1 Introduction

User modeling is an important task for web applications dealing with large traffic flows. They can be used for a variety of applications such as to predict future situations or classify current states. Furthermore, user modeling can improve detection or mitigation of Distributed Denial of Service (DDoS) attack [12,15,17], improve the quality of service (QoS) [19], individuate click fraud detection and optimize traffic management. In peer-to-peer (P2P) overlay networks, IP models can also be used for optimizing request routing [1]. Those techniques are used by severs for deciding how to manage the actual traffic. In this context, also outlier detection methods are often used if only one class is known. If, for example, an Intrusion Prevention System wants to mitigate DDoS attacks, it usually has only seen the normal traffic class before and it has to detect the outlier class by its different behavior. In this paper we deal with the management of DDos because nowadays it has become a major threat in the internet. Those attacks are done by using a large scaled networks of infected PCs (bots or zombies) that combine their bandwidth and computational power in order to overload a publicly available service and denial it for legal users. Due to the open structure of the internet, all public servers are vulnerable to DDoS attacks. The bots are usually acquired automatically by hackers who use software tools to scan through the network, detecting vulnerabilities and exploiting the target machine. Furthermore, there is also a strong need to mitigate DDoS attacks near the target, which seems to be the only solution to the problem in the current internet infrastructure. The aim of such a protection system is to limit their destabilizing effect on the server through identifying malicious requests. There are multiple strategies with dealing with DDoS attacks. The most effective ones are the near-target filtering solutions. They estimates normal user behavior based on IP packet header information. Then, during an attack the access of outliers is denied. One parameter that all methods have in common is the source IP address of the users. It is the main discriminant for DDoS traffic classification. However, the methods of storing IP addresses and estimating their density in the huge IP address space, are different. In this paper, we present a novel approach based on system identification techniques and, in particular, on the Hammerstein models. The paper is organized as follows. In Sect. 2 we present the available methods for DDoS traffic classification (a broader overview of state-of-the-art mitigation research is given by [10]). In Sects. 3 and 4 we present our proposed a technique based based on Hammerstein models and we recall some similar model. Experimental results reported in Sect. 5 confirm the effectiveness of our approach. Although DDoS mitigation is the most important practical application for IP density estimation, we do not restrict the following work on this topic. Our generic view on IP density estimation may be valuable to other applications as well. One might think of preferring regular customers in overload situations (flash crowd events), identifying non-regular users on websites during high click rates on online advertisements (click fraud detection) or optimizing routing in peer-to-peer networks. Finally, in Sect. 6 we draw conclusions and indicate future work.

2 Related Work

In this section we review existing IP density estimation approaches. Furthermore, we formulate the often implicitly used ideas in a probabilistic way using the PDF.

2.1 History-Based IP Filtering

One of the first work on the field of IP filtering is [18]. In their work, they proposed an algorithm called History based IP Filtering (HIF). Given an IP address database (IAD) containing all frequently observed IP addresses, during an attack, the algorithm allows the access to the website only to the IP addresses from the IAD. The idea behind this algorithm was introduced in [13]. In this paper, the authors observe that the IP addresses from Code Red worm attacks were different from the standard IPs accessing the website. In [18], the authors add an IP addresses to the IAD if a certain threshold (e.g. a certain number of packets) is exceeded. Later during an attack, the decision whether an IP has to be blocked or not is binary. This means, the density estimation results in a simple step function with only two values: Zero or a positive value, which is the same for all IP addresses. The PDF can be calculated as follows:

$$f(s) = \frac{\min(n_s, 1)}{\sum_{i=0}^{N-1} \min(n_{s_i}, 1)} \tag{1}$$

where n_s is the number of occurrence of an IP address s in the training set. The advantage of HIF is the low computational load required for its implementation. Nevertheless, it cannot be differentiated between users which revisit the server more often than others and is therefore a less precise density estimator.

2.2 Adaptive History-Based IP Filtering

To compensate the shortcomings of HIF, [12] presented Adaptive History-based IP Filtering (AHIF). AHIF instead of using the binary decision, uses histograms, i.e.,

$$f(s) = \frac{n_s}{\sum_{i=0}^{N-1} n_{s_i}} \tag{2}$$

where s represents a range of IP addresses (a bin of the histogram). So far, constant width networks with fixed network masks ranging from 16 up to 24 bit are used as source bins. The actual prediction C_α of the appearance of an IP address (or an IP range) is done using a threshold over $f(s)$, i.e.,

$$C_\alpha(s) := \begin{cases} \text{reject}, & \text{if } f(s) \geq \alpha \\ \text{accept}, & \text{otherwise} \end{cases} \tag{3}$$

During attack mitigation, the most appropriate network mask is chosen such that a maximum numbers of firewall rules is not exceeded and the attack traffic is reduced to be below the maximum server capacity. It is shown, that the adaptive method performs better in terms of predicting user IP addresses during an attack. However, neighbor relations (between source networks) are not taken into account.

2.3 Clustering of Source Address Prefixes

In [15], the authors introduce algorithms for mitigating DDoS attacks by filtering source address prefixes. Unlike AHIF, network masks may be at different sizes. For finding the appropriate networks, the authors are using a hierarchical agglomerative clustering algorithm with single linkage. The used distance measure is defined with respect to network boundaries. In general, the proposed method takes both into account - the amount of requests from a source network as well as neighboring density estimation as a result of using a (generalizing) clustering method. Unfortunately, hierarchical clustering methods consume a lot of memory and it was found [17], that the presented method is not applicable in practice on large source IP datasets.

3 Non-linear Analytic Prediction of IP Addresses

Data driven identification of mathematical models of physical systems (i.e. nonlinear) starts with representing the systems as a black box. In other terms, while we may have access to the inputs and outputs, the internal mechanisms are totally unknown to us. Once a model type is chosen to represent the system, its parameters are estimated through an optimization algorithm so that eventually the model mimics at a certain level of fidelity the inner mechanism of the nonlinear system or process using its inputs and outputs. These approach is, for instance, widely used in the related *big data analytics* area (e.g., [3,5,7–9]).

In this work, we consider a particular sub-class of nonlinear predictors: the Linear-in-the-parameters (LIP) predictors. LIP predictors are characterized by a linear dependence of the predictor output on the predictor coefficients. Such predictors are inherently stable, and that they can converge to a globally minimum solution (in contrast to other types of nonlinear filters whose cost function may exhibit many local minima) avoiding the undesired possibility of getting stuck in a local minimum. Let us consider a causal, time-invariant, finite-memory, continuous nonlinear predictor as described in (4).

$$\hat{s}(n) = f[s(n-1), \ldots, s(n-N)] \tag{4}$$

where $f[\cdot]$ is a continuous function, $s(n)$ is the input signal and $\hat{s}(n)$ is the predicted sample. We can expand $f[\cdot]$ with a series of basis functions $f_i(n)$, as shown in (5).

$$\hat{s}(n) = \sum_{i=1}^{\infty} h(i) f_i[s(n-i)] \tag{5}$$

where $h(i)$ a re proper coefficients. To make (5) realizable we truncate the series to the first N terms, thus we obtain

$$\hat{s}(n) = \sum_{i=1}^{N} h(i) f_i[s(n-i)] \tag{6}$$

In the general case, a linear-in-the-parameters nonlinear predictor is described by the input-output relationship reported in (7).

$$\hat{s}(n) = \boldsymbol{H}^T \boldsymbol{X}(n) \tag{7}$$

where $\boldsymbol{H}^T$ is a row vector containing predictor coefficients and $\boldsymbol{X}(n)$ is the corresponding column vector whose elements are nonlinear combinations and/or expansions of the input samples.

3.1 Linear Predictor

Linear prediction is a well known technique with a long history [14]. Given a time series $\boldsymbol{X}$, linear prediction is the optimum approximation of sample $x(n)$ with a linear combination of the N most recent samples. That means that the linear predictor is described as Eq. (8).

$$\hat{s}(n) = \sum_{i=1}^{N} h_1(i) s(n-i) \tag{8}$$

or in matrix form as

$$\hat{s}(n) = \boldsymbol{H}^T \boldsymbol{X}(n) \tag{9}$$

where the coefficient and input vectors are reported in (10) and (11).

$$\boldsymbol{H}^T = \begin{bmatrix} h_1(1)\ h_1(2) \ldots h_1(N) \end{bmatrix} \tag{10}$$

$$\boldsymbol{X}^T = \begin{bmatrix} s(n-1)\ s(n-2) \ldots s(n-N) \end{bmatrix} \tag{11}$$

3.2 Non-linear Predictor Based on Volterra Series

As well as Linear Prediction, Non Linear Prediction is the optimum approximation of sample $x(n)$ with a non linear combination of the N most recent samples. Popular nonlinear predictors are based on Volterra series [16]. A Volterra predictor based on a Volterra series truncated to the second term is reported in (12).

$$\hat{x}(n) = \sum_{i=1}^{N_1} h_1(i) x(n-i) + \sum_{i=1}^{N_2} \sum_{j=i}^{N_2} h_2(i,j) x(n-i) x(n-j) \tag{12}$$

where the symmetry of the Volterra kernel (the h coefficients) is considered. In matrix terms, the Volterra predictor is represented in (13).

$$\hat{s}(n) = \boldsymbol{H}^T \boldsymbol{X}(n) \tag{13}$$

where the coefficient and input vectors are reported in (15) and (15).

$$\boldsymbol{H}^T = \begin{bmatrix} h_1(1) & h_1(2) \ldots h_1(N) \\ h_2(1,1) & h_2(1,2) \ldots h_2(N_2, N_2) \end{bmatrix} \tag{14}$$

$$\boldsymbol{X}^T = \begin{bmatrix} s(n-1) & s(n-2) \ldots s(n-N_1) \\ s^2(n-1) & s(n-1)s(n-2) \ldots s^2(n-N_2) \end{bmatrix} \tag{15}$$

3.3 Non-linear Predictor Based on Functional Link Artificial Neural Networks (FLANN)

FLANN is a single layer neural network without hidden layer. The nonlinear relationships between input and output are captured through function expansion of the input signal exploiting suitable orthogonal polynomials. Many authors used for examples trigonometric, Legendre and Chebyshev polynomials. However, the most frequently used basis function used in FLANN for function expansion are trigonometric polynomials [20]. The FLANN predictor can be represented by Eq. (16).

$$\hat{s}(n) = \sum_{i=1}^{N} h_1(i)s(n-i) + \sum_{i=1}^{N}\sum_{j=1}^{N} h_2(i,j)\cos[i\pi s(n-j)] + \sum_{i=1}^{N}\sum_{j=1}^{N} h_2(i,j)\sin[i\pi s(n-j)] \tag{16}$$

Also in this case the Flann predictor can be represented using the matrix form reported in (13).

$$\hat{s}(n) = \boldsymbol{H}^T \boldsymbol{X}(n) \tag{17}$$

where the coefficient and input vectors of FLANN predictors are reported in (25) and (26).

$$\boldsymbol{H}^T = \begin{bmatrix} h_1(1) & h_1(2)\ldots h_1(N) \\ h_2(1,1) & h_2(1,2)\ldots h_2(N_2,N_2) \\ h_3(1,1) & h_3(1,2)\ldots h_3(N_2,N_2) \end{bmatrix} \tag{18}$$

$$\boldsymbol{X}^T = \begin{bmatrix} s(n-1) & s(n-2) & \ldots s(n-N) & \\ \cos[\pi s(n-1)] & \cos[\pi s(n-2)] & \ldots \quad \ldots & \cos[N_2\pi s(n-N_2)] \\ \sin[\pi s(n-1)] & \sin[\pi s(n-2)] & \ldots \quad \ldots & \sin[N_2\pi x(s-N_2)] \end{bmatrix} \tag{19}$$

3.4 Non-linear Predictors Based on Hammerstein Models

Previous research [4] shown that many real nonlinear systems, spanning from electromechanical systems to audio systems, can be modeled using a static nonlinearity. These terms capture the system nonlinearities, in series with a linear function, which capture the system dynamics as shown in Fig. 1.

Indeed, the front-end of the so called Hammerstein Model is formed by a nonlinear function whose input is the system input. Of course the type of nonlinearity depends on the actual physical system to be modeled. The output of the nonlinear function is hidden and is fed as input of the linear function. In the following, we assume that the non-linearity is a finite polynomial expansion, and the linear dynamic is realized with a Finite Impulse Response (FIR) filter.

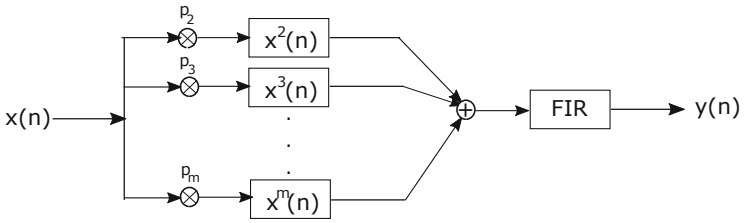

Fig. 1. Representation of the Hammerstein Models

Furthermore, in contrast with [4], we assume a mean error analysis and we postpone the analysis in the robust framework in future work. In other word,

$$z(n) = p(2)x^2(n) + p(3)x^3(n) + \ldots p(m)x^m(n) = \sum_{i=2}^{M} p(i)x^i(n) \tag{20}$$

On the other hand, the output of the FIR filter is:

$$\begin{aligned} y(n) &= h_0(1)z(n-1) + h_0(2)z(n-2) + \ldots + h_0(N)z(n-N) \\ &= \sum_{j=1}^{N} h_0(j)z(n-j) \end{aligned} \tag{21}$$

Substituting (20) in (23) we have:

$$\begin{aligned} y(n) &= \sum_{i=1}^{N} h_0(i)z(n-i) = \sum_{j=1}^{N} h_0(j) \sum_{i=2}^{M} p(i)x^i(n-j) \\ &= \sum_{i=2}^{M} \sum_{j=1}^{N} h_0(j)p(i)x^i(n-j) \end{aligned} \tag{22}$$

Setting $c(i,j) = h_0(j)p(i)$ we write

$$y(n) = \sum_{i=2}^{M} \sum_{j=1}^{N} c(i,j)x^i(n-j) \tag{23}$$

This equation can be written in matrix form as

$$\hat{s}(n) = \boldsymbol{H}^T \boldsymbol{X}(n) \tag{24}$$

where

$$\boldsymbol{H}^T = \begin{bmatrix} c(2,1) & c(2,2) \ldots c(2,N_2) \\ c(3,1) & c(3,2) \ldots c(3,N_2) \\ \ldots c(M,1) & C(M,2) \ldots C(M,N) \end{bmatrix} \tag{25}$$

$$\boldsymbol{X}^T = \begin{bmatrix} s^2(n-2) & s^2(n-3) \ldots s^2(n-N) \\ s^3(n-2) & s^3(n-3) \ldots s^3(n-N) \\ s^M(n-2) & s^M(n-3) \ldots s^M(n-1) \; s^M(n-3) \ldots s^M(n-N) \end{bmatrix} \tag{26}$$

4 Predictor Parameters Estimation

So far we saw that all the predictors can be expressed, at time instant n, as

$$\hat{s}(n) = \boldsymbol{H}^T \boldsymbol{X}(n) \tag{27}$$

with different definitions of the input, $\boldsymbol{X}(n)$, end parameters vectors $\boldsymbol{H}^T$. There are two well known possibilities for estimating the optimal parameter vector.

4.1 Block-Based Approach

The Minimum Mean Square estimation is based on the minimization of the mathematical expectation of the squared prediction error $e(n) = s(n) - \hat{s}(n)$

$$E[e^2] = E[(s(n) - \hat{s}(n))^2] = E[(s(n) - \boldsymbol{H}^T \boldsymbol{X}(n))^2] \tag{28}$$

The minimization of (28) is obtain by setting to zero the Laplacian of the mathematical expectation of the squared prediction error:

$$\nabla_H E[e^2] = E[\nabla_H e^2] = E[2e(n)\nabla_H e] = 0 \tag{29}$$

which leads to the well known unique solution

$$\boldsymbol{H}_{opt} = \boldsymbol{R}_{xx}^{-1} \boldsymbol{R}_{sx} \tag{30}$$

where

$$\boldsymbol{R}_{xx}(n) = E[\boldsymbol{X}(n)\boldsymbol{X}^T(n)] \tag{31}$$

is the statistical auto-correlation matrix of the input vector $\boldsymbol{X}(n)$ and

$$\boldsymbol{R}_{sx}(n) = E[s(n)\boldsymbol{X}(n)] \tag{32}$$

is the statistical cross-correlation vector between the signal $s(n)$ and the input vector $\boldsymbol{X}(n)$. The mathematical expectations of the auto and cross correlation are estimated using

$$\boldsymbol{R}_{xx}(n) = \frac{\sum_{k=1}^{n} \boldsymbol{X}(n)\boldsymbol{X}^T(n)}{n} \tag{33}$$

is the statistical auto-correlation matrix of the input vector $\boldsymbol{X}(n)$ and

$$\boldsymbol{R}_{sx}(n) = \frac{\sum_{k=1}^{n} s(k)(n)\boldsymbol{X}(n)}{n} \tag{34}$$

4.2 Adaptive Approach

Let us consider a general second order terms of a Volterra predictor

$$y(n) = \sum_{k=0}^{N-1} \sum_{r=0}^{N-1} h_2(k,r)x(n-k)x(n-r) \tag{35}$$

It can be generalized for higher order term as

$$\sum_{k_1=1}^{N} \cdots \sum_{k_p=1}^{N} c_{k_1} \cdots c_{k_p} H_p \left\{ x_{k_1}(n), \cdots x_{k_p}(n) \right\} \tag{36}$$

where

$$\sum_{k=1}^{N} c_k x_k(n). \tag{37}$$

For the sake of simplicity and without loss of generality, we consider a Volterra predictor based on a Volterra series truncated to the second term

$$\hat{r}(n) = \sum_{i=1}^{N_1} h_1(i) r(n-i) + \sum_{i=1}^{N_2} \sum_{j=i}^{N_2} h_2(i,j) r(n-i) r(n-j) \tag{38}$$

By defining

$$H^T(n) = |h_1(1), \cdots, h_1(N_1), h_2(1,1), \cdots, h_2(N_2, N_2)| \tag{39}$$

and

$$\begin{array}{c} X^T(n) = \big| r(n-1), \cdots, r(n-N_1), r^2(n-1) \\ r(n-1)r(n-2), \cdots, r^2(n-N_2) \big| \end{array} \tag{40}$$

Equation (38) can be rewritten as follows

$$\hat{r}(n) = H^T(n) X(n). \tag{41}$$

In order to estimate the best parameters H, we consider the following loss function

$$J_n(H) = \sum_{k=0}^{n} \lambda^{n-k} \left[\hat{r}(k) - H^T(n) X(k) \right]^2 \tag{42}$$

where λ^{n-k} weights the relative importance of each squared error. In order to find the H that minimizes the convex function (42) it is enough to impose its gradient to zero, i.e.,

$$\nabla_H J_n(H) = 0 \tag{43}$$

That is equivalent to

$$R_{XX}(n) H(n) = R_{rX}(n) \tag{44}$$

where

$$\begin{array}{c} R_{XX}(n) = \sum_{k=0}^{n} \lambda^{n-k} X(k) X^T(k) \\ R_{rX}(n) = \sum_{k=0}^{n} \lambda^{n-k} r(k) X(k) \end{array} \tag{45}$$

It follows that the best H can be computed by

$$H(n) = R_{XX}^{-1}(n) R_{rX}(n) \tag{46}$$

Since

$$R_{XX}(n) = \lambda R_{XX}(n-1) + X(n) X^T(n) \tag{47}$$

it follows that

$$R_{XX}^{-1}(n) = \frac{1}{\lambda}\left[R_{XX}^{-1}(n-1) - k(n)X^T(n)R_{XX}^{-1}(n-1)\right] \tag{48}$$

where $k(n)$ is equal to

$$k(n) = \frac{R_{XX}^{-1}(n-1)X(n)}{\lambda + X^T(n)R_{XX}^{-1}(n-1)X(n)} \tag{49}$$

Instead, matrix $R_{rX}(n)$ in (46) can be written as

$$R_{rX}(n) = \lambda R_{rX}(n-1) + r(n)X(n) \tag{50}$$

Thus, inserting Eq. (50) and Eq. (48) in Eq. (46) and rearranging the terms, we obtain

$$H(n) = H(n-1) + R_{XX}^{-1}(n)X(n)\epsilon(n) \tag{51}$$

where

$$\epsilon = \hat{r}(n) - H^T(n-1)X(n) \tag{52}$$

By recalling Eq. (49), we can write Eq. (51) as

$$H(n) = H(n-1) + \epsilon(n)k(n) \tag{53}$$

By introducing, the new notation,

$$F(n) = S^T(n-1)X(n) \tag{54}$$

The previous equations can be resumed by the following system

$$\begin{cases} L(n) = S(n-1)F(n) \\ \beta(n) = \lambda + F^T(n)F(n) \\ \alpha(n) = \frac{1}{\beta(n)+\sqrt{\lambda\beta(n)}} \\ S(n) = \frac{1}{\sqrt{\lambda}}\left[S(n-1) - \alpha(n)L(n)F^T(n)\right] \\ \epsilon(n) = \hat{r}(n-1) - \alpha(n)L(n)F^T(n) \\ \epsilon(n) = H(n-1) + L(n)\frac{\epsilon(n)}{\beta(n)} \end{cases} \tag{55}$$

It should be noted that by using Eq. (55) the estimation adapts in each step in order to decrease the error. Thus, the system structure is somehow similar to the Kalman filter.

Finally, we define the estimation error as

$$e(n) = r(n) - H^T(n)X(n) \tag{56}$$

It is worth noting that the computation of the predicted value from Eq. (41) requires $6N_{\text{tot}} + 2N_{\text{tot}}^2$ operations, where $N_{\text{tot}} = N_1 + N_2\left(N_2 + 1\right)/2$.

5 Experiments

In order to prove the effectiveness of the proposed approach, in this section we present our experimental results for a real dataset. Specifically, we consider the requests made to the 1998 World Cup Web site between April 30, 1998 and July 26, 1998[1]. During this period of time the site received 1,352,804,107 requests. The fields of the request structure contain the following information:

- *timestamp* the time of the request, stored as the number of seconds since the Epoch. The timestamp has been converted to GMT to allow for portability. During the World Cup the local time was 2 h ahead of GMT (+0200). In order to determine the local time, each timestamp must be adjusted by this amount.
- *clientID* a unique integer identifier for the client that issued the request; due to privacy concerns these mappings cannot be released; note that each *clientID* maps to exactly one IP address, and the mappings are preserved across the entire data set - that is if IP address 0.0.0.0 mapped to *clientID* X on day Y then any request in any of the data sets containing *clientID* X also came from IP address 0.0.0.0
- *objectID* a unique integer identifier for the requested URL; these mappings are also 1-to-1 and are preserved across the entire data set
- *size* the number of bytes in the response
- *method* the method contained in the client's request (e.g., GET).
- *status* this field contains two pieces of information; the 2 highest order bits contain the HTTP version indicated in the client's request (e.g., HTTP/1.0); the remaining 6 bits indicate the response status code (e.g., 200 OK).
- *type* the type of file requested, generally based on the file extension (.html), or the presence of a parameter list.
- *server* indicates which server handled the request. The upper 3 bits indicate which region the server was at; the remaining bits indicate which server at the site handled the request.

In the dataset, 87 days are reported. We use the first one in order to initialise the estimator in (38) and we use the others as test set by using a rolling horizon method (as in [4]). Particularly, for each day t we compute the estimation by using all the IP observations in the previous days $[0, t-1]$. The results are reported in Fig. 2. The increase of errors in June is due to the sudden increment of different IP accessing the website due to the start of the competition (see [2]). It should be noted that the estimation error decrease exponentially despite dealing with several millions of IPs.

Since the computation of the optimal coefficient $H(n)$ may require some time, we measure the percentage of available data that our approach needs in order to provide good results. Particularly, in this experiment we consider the average estimation error done by the model at time t by considering a subset of the IPs observed in interval $[0, t-1]$. The experimental results on real data cubes are depicted in Fig. 3.

[1] ftp://ita.ee.lbl.gov/html/contrib/WorldCup.html.

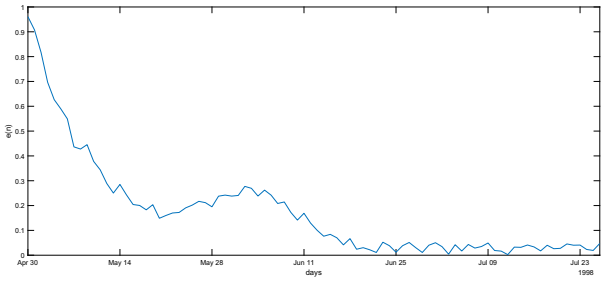

Fig. 2. Estimation error for each day of activity of the website.

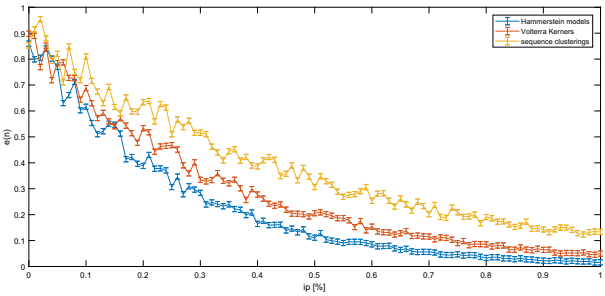

Fig. 3. Estimation error vs. percentage size of the training set

It should be noted that the Hammerstain model outperform the results by the Volterra's kernel as well as the clustering techniques. In more detail, the clustering techniques are the one less performing. This is due to the nature of the clustering techniques that exploit the geometric information of the data more than their time dependency. We highlight that despite the calculation of $H(n)$ is computational intensive, this does not effect the real time applicability of the method. In fact, the access decision is taken by considering the estimator $\hat{r}(n)$ that is computed once per day. Thus the computation of $H(n)$ does not need to be fast.

6 Conclusions and Future Work

In this paper, we presented a new way to deal with cyber attack by using Hammerstein models. Experimental results clearly confirm the effectiveness of the proposed techniques for a real data set, outperforming other well-known techniques. Future work will have two objectives. First, we want to consider the problem in a stochastic optimization settings, as for example in [11]. Second, we want to test the approach on other case studies, by also exploiting *knowledge management methodologies* (e.g., [6]).

Acknowledgements. This research has been partially supported by the French PIA project "Lorraine Université d'Excellence", reference ANR-15-IDEX-04-LUE.

References

1. Agrawal, A., Casanova, H.: Clustering hosts in P2P and global computing platforms, pp. 367–373, June 2003. https://doi.org/10.1109/CCGRID.2003.1199389
2. Arlitt, M., Jin, T., Hewlett-Packard Laboratories: A workload characterization study of the 7998 world cup web site
3. Bonifati, A., Cuzzocrea, A.: Storing and retrieving XPath fragments in structured P2P networks. Data Knowl. Eng. **59**(2), 247–269 (2006)
4. Cerone, V., Fadda, E., Regruto, D.: A robust optimization approach to kernel-based nonparametric error-in-variables identification in the presence of bounded noise. In: 2017 American Control Conference (ACC). IEEE, May 2017. https://doi.org/10.23919/acc.2017.7963056
5. Chatzimilioudis, G., Cuzzocrea, A., Gunopulos, D., Mamoulis, N.: A novel distributed framework for optimizing query routing trees in wireless sensor networks via optimal operator placement. J. Comput. Syst. Sci. **79**(3), 349–368 (2013)
6. Cuzzocrea, A.: Combining multidimensional user models and knowledge representation and management techniques for making web services knowledge-aware. Web Intell. Agent Syst. **4**(3), 289–312 (2006)
7. Cuzzocrea, A., Bertino, E.: Privacy preserving OLAP over distributed XML data: a theoretically-sound secure-multiparty-computation approach. J. Comput. Syst. Sci. **77**(6), 965–987 (2011)
8. Cuzzocrea, A., Moussa, R., Xu, G.: OLAP*: effectively and efficiently supporting parallel OLAP over big data. In: Cuzzocrea, A., Maabout, S. (eds.) MEDI 2013. LNCS, vol. 8216, pp. 38–49. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41366-7_4
9. Cuzzocrea, A., Russo, V.: Privacy preserving OLAP and OLAP security. In: Encyclopedia of Data Warehousing and Mining, Second Edition (4 Volumes), pp. 1575–1581 (2009)
10. Dietrich, S., Long, N., Dittrich, D.: Analyzing distributed denial of service tools: the shaft case, pp. 329–339, December 2000
11. Fadda, E., Perboli, G., Tadei, R.: Customized multi-period stochastic assignment problem for social engagement and opportunistic IoT. Comput. Oper. Res. **93**, 41–50 (2018)
12. Goldstein, M., Lampert, C., Reif, M., Stahl, A., Breuel, T.: Bayes optimal DDoS mitigation by adaptive history-based IP filtering. In: Seventh International Conference on Networking (ICN 2008), pp. 174–179 (2008)
13. Jung, J., Krishnamurthy, B., Rabinovich, M.: Flash crowds and denial of service attacks: characterization and implications for CDNs and web sites, pp. 293–304, January 2002. https://doi.org/10.1145/511446.511485
14. Makhoul, J.: Linear prediction: a tutorial review. Proc. IEEE **63**(4), 561–580 (1975)
15. Pack, G., Yoon, J., Collins, E., Estan, C.: On filtering of DDoS attacks based on source address prefixes, pp. 1–12, August 2006. https://doi.org/10.1109/SECCOMW.2006.359537
16. Peng, Z., Changming, C.: Volterra series theory: a state-of-the-art review. Chin. Sci. Bull. (Chin. Version) **60**, 1874 (2015). https://doi.org/10.1360/N972014-01056
17. Tan, H.X., Seah, W.: Framework for statistical filtering against DDoS attacks in MANETs, p. 8, January 2006. https://doi.org/10.1109/ICESS.2005.57

18. Peng, T., Leckie, C., Ramamohanarao, K.: Protection from distributed denial of service attacks using history-based IP filtering. In: IEEE International Conference on Communications 2003, ICC 2003, vol. 1, pp. 482–486 (2003)
19. Yang, Y., Lung, C.H.: The role of traffic forecasting in QoS routing - a case study of time-dependent routing, vol. 1, pp. 224–228, June 2005. https://doi.org/10.1109/ICC.2005.1494351
20. Zhao, H., Zhang, J.: Adaptively combined FIR and functional link artificial neural network equalizer for nonlinear communication channel. IEEE Trans. Neural Netw. **20**(4), 665–674 (2009)