A Dynamic Ensemble Learning Framework for Data Stream Analysis and Real-Time Threat Detection

Konstantinos Demertzis[1], Lazaros Iliadis[1], and Vardis-Dimitris Anezakis[2](✉)

[1] School of Engineering, Department of Civil Engineering, Democritus University of Thrace, University Campus, Kimmeria, Xanthi, Greece
kdemertz@fmenr.duth.gr, liliadis@civil.duth.gr
[2] Department of Forestry and Management of the Environment and Natural Recourses, Democritus University of Thrace, 193 Pandazidou St., 68200 N Orestiada, Greece
danezaki@fmenr.duth.gr

Abstract. Security incident tracking systems receive a continuous, unlimited inflow of observations, where in the typical case the most recent ones are the most important. These data flows and characterized by high volatility. Their characteristics can change drastically over time in an unpredictable way, differentiating their typical normal behavior. In most cases it is not possible to store all of the historical samples, since their volume is unlimited. This fact requires the extraction of real-time knowledge over a subset of the flow, which contains a small but recent percentage of all observations. This creates serious objections to the accuracy and reliability of the employed classifiers. The research described herein, uses a Dynamic Ensemble Learning (DYENL) approach for Data Stream Analysis (DELDaStrA) which is employed in RealTime Threat Detection systems. More specifically, it proposes a DYENL model that uses the "Kappa" architecture to perform analysis of data flows. The DELDaStrA is based on the hybrid combination of k Nearest Neighbor (kNN) Classifiers, with Adaptive Random Forest (ARF) and Primal Estimated SubGradient Solver for Support Vector Machines (SVM) (SPegasos). In fact, it performs a dynamic extraction of the weighted average of the three results, to maximize the classification accuracy.

Keywords: Dynamic ensemble learning · Big data · Data streams analysis "Kappa" architecture · Critical infrastructure · Real-time threat detection

1 Introduction

The data created by SCADA [31] and more generally by Industrial Control Systems (ICS) [20], has caused an exponential increase of the obtained information. This fact has led to the adoption of architectures which incorporate proper algorithms for real-time data stream processing. These algorithms are dynamically adjusted by new models or when the data are produced as a function of time [5]. The "Kappa" architecture uses a real-time engine and it is the most suitable approach for the analysis of data flows [25].

For each new sample, a small gradual update of the model takes place, which gradually improves as more data arrive. The error in the real-time engine is calculated at each iteration as data characteristics can change drastically and in an unpredictable way. This changes the typical, normal behavior, and an object that may have been considered extreme, can be included in the normal observations, due to rapid developments in the data stream (Fig. 1).

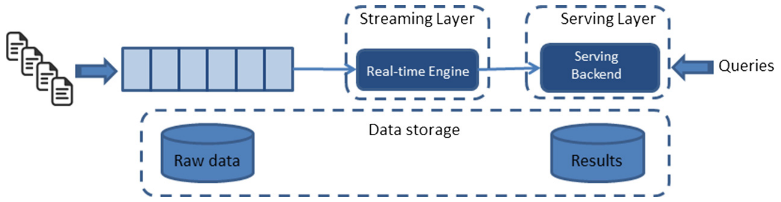

Fig. 1. Kappa architecture (https://www.oreilly.com/ideas/applying-the-kappa-architecture-in-the-telco-industry)

Due to the unlimited volume of data, data mining is performed on a subset of the flow, which is called a sliding window (SLWI). Clearly the SLIWI contains a small but recent percentage of the observations included in the global set. The goal of these data processing algorithms is to minimize the cumulative error for all iterations, which can be calculated by the following function (1) [2]:

$$I_n[w] = \sum_{j=1}^{n} V\left(\left\langle w, x_j \right\rangle, y_j\right) = \sum_{j=1}^{n} \left(x_j^T w - y_j\right)^2 \tag{1}$$

where $x_j \in R^d$, $w \in R^d$ and $y_j \in R$ supposing that X$i \times d$ is a data matrix and Y$i \times 1$ is a target values vector, obtained after the arrival of the first i data points. If we accept that the covariance matrix $\Sigma i = X^T X$ is reversable, the optimal solution $f^*(x) = \langle w*, x \rangle$ is given by the following function (2):

$$w^* = \left(X^T X\right)^{-1} X^T \Upsilon = \Sigma_i^{-1} \sum_{j=1}^{i} x_j y_j \tag{2}$$

If we estimate the covariance matrix $\Sigma_i = \sum_{j=1}^{i} x_j x_j^T$ the time complexity $(TC)\, changes from\, O\left(id^2\right)(d \times d)$ and it becomes $O(d^3)$, whereas the rest of the multiplication requires TC equal to $O(d^2)$. Thus, the TC finally becomes equal to $O(id^2 + d^3)$. If n is the number of points in the dataset, it is necessary to recalculate the solution after the arrival of each new data point $i = 1, 2, \ldots, n$. So, the final time complexity is of the order $O(n^2 d^2 + nd^3)$ which would make the algorithm unsuitable for application in demanding fast changing environments such as the one under consideration [2, 24]. It is therefore important to note that in-stream processing is subject to time constraints, as applications require explanatory results in real time, and there are also significant memory requirements.

It is clear from the above, that a secure approach for data flow mining problems, requires robust systems characterized by reliability and high accuracy rates, without a demand for high resources availability. Good preparation and methodological determination of their operating parameters is needed, to avoid long-term convergence, or undesirable fluctuations in accuracy, which may be associated with frequent model updates and instability or loss of generalization, which may be due to corrupted and noisy data.

1.1 Literature Review

Soft computing techniques are capable to model and detect cyber security threats [6–14] and they also offer optimization mechanisms in order to produce reliable results. In many applications, learning algorithms have to act in dynamic environments where data are collected in the form of transient data streams. Krawczyk et al. [21] investigated 3 data stream classification as well as regression tasks. Besides presenting a comprehensive spectrum of ensemble approaches for data streams, authors also discussed advanced learning concepts such as imbalanced data streams. According to Liu et al. [26] a weight computation policy based on confidence was presented to deal with the problem in the sub-classifier's weight in dynamic data stream ensemble classification. The policy fully considers influence of the sample on the weight of the sub-classifier. Krawczyk and Cano [22] introduced a dynamic and self-adapting threshold that was able to adapt to changes in the data stream, by monitoring outputs of the ensemble to exploit underlying diversity in order to efficiently anticipate drifts.

Nowadays, the intrusion detection systems (IDS) have become one of the most important weapons against cyber-attacks. Chand et al. [4] performed a comparative analysis of SVM classifier's performance when it was stacked with other classifiers like BayesNet, AdaBoost and Random Forest. Ahmin and Ghoualmi-Zine [1] used two different classifiers iteratively, where each-iteration represented one level in the built model. To ensure the adaptation of their model, authors added a new level whenever the sum of new attacks and the rest of the training dataset reached the threshold.

Data mining in non-stationary data streams is gaining more attention recently, especially in the context of Internet of Things and Big Data. Losing et al. [27] proposed the Self Adjusting Memory (SAM) model for the k-Nearest Neighbor (k-NN) algorithm since k-NN constitutes a proven classifier within the streaming setting. SAM-kNN could deal with heterogeneous concept drift, i.e. different drift types and rates, using biologically inspired memory models and their coordination. Rani and Sumathy [28] used k-NN algorithm to determine the best optimal subset.

There are a few researches about Primal Estimated sub-Gradient Solver for SVM (Pegasos) algorithm. Shalev-Shwartz et al. [29] described and analyzed a simple and effective stochastic sub-gradient descent algorithm for solving the optimization problem cast by SVM. Their algorithm was particularly well suited for large text classification problems, where authors demonstrated an order-of-magnitude speedup over previous SVM methods. Farda [18] explored machine learning in Google Earth Engine and its accuracy for multi-temporal land used mapping of coastal wetland area.

1.2 Datasets

Appropriate datasets were chosen that closely simulate ICS communication and transaction data. They were used in the development and evaluation of the proposed model. The following preprocessed network transaction data, and preprocessed to strip lower layer transmission data, were used in this research (e.g. TCP, MAC) [15]:

- The water_tower_dataset includes 23 independent parameters and 236,179 instances, from which 172,415 are normal and 63,764 outliers. Totally 86,315 normal instances were used in the training phase (water_train_dataset) whereas the water_test_dataset comprised of 86,100 normal instances and 63,764 outliers.
- The gas_dataset includes 26 independent features and 97,019 instances, from which 61,156 normal and 35,863 outliers. Training of the algorithm was done with the gas_train_dataset that contains 30,499 normal instances, whereas the gas_test_-dataset comprises of 30,657 normal instances and 35,863 outliers.
- Finally, the electric_dataset includes 128 independent variables with 146,519 instances, from which 90,856 normal and 55,663 outliers. The training was performed 4 based on the electric_train_dataset comprising of 45,402 normal instances, whereas the rest 45,454 normal and the 55,663 outliers, belong to the electric_test_dataset.

More details regarding the dataset and their choice can be found in [15].

2 Proposed Dynamic Weighted Average Methodology

This research proposes an intelligent and dynamic Ensemble Machine Learning system (EMLS) [32] aiming to develop a stable and accurate framework, which will have the ability to generalize. The EMLS employs an innovative version of the “Kappa” architecture that combines the ARF, SPegasos and k-NN SAM algorithms. DEL-DaStrA performs real time analysis and assessment of critical infrastructure data, in order to classify and identify undesirable digital security situations, related to cyber-attacks. The reason for using the ensemble approach, is the multivariance that usually appears in such multifactorial problems of high complexity, due to the heterogeneity of the data flows. This is a typical case of digital security and critical infrastructures.

The two most important advantages of the Ensemble Techniques focus on the fact that they offer better prediction and more stable models, as the overall behavior of a multiple model is less noisy than a corresponding single [23]. Also, an Ensemble method can lead to very stable prediction models, while offering generalization. Finally, these models can reduce the bias, the variance, and they can avoid overfitting [17] producing robust learning models.

Three classifiers were employed in the development of this model (Ensemble Size). The number of the classifiers was determined after considering the law of *diminishing returns in ensemble construction* in a trial and error approach. The applied algorithms were chosen based on their different decision-making philosophy and methodology to address the problem, in order to cover the number of possible cases associated with the tactic of attacks against critical infrastructure. In general, the choice was based on both static tests combined with the trial and error method, but also on the basic properties of these algorithms regarding the way they handle each situation.

More specifically, the following approaches were used: SVM non-parametric models due to the way they handle outliers. Random Forests which are using subsets of the training sets with bagging, and subsets of features that favor the reduction of the outliers' or extreme values' effect. The k-NN classifier is automatically non-linear, it can detect linear or non-linear distributed data and it tends to perform very well with a lot of data points. Also, the choice of the algorithms was based on the diversity of their operation and parameterization (Reliability of Ensemble) which is achieved with different architectures, hyper-parameter settings and training techniques. The weights' determination of the different models of the Ensemble, was based exclusively on static trial and error tests [16].

The DELDaStrA operation mode, includes the parallel analysis of the data flow by all three algorithms and the dynamic extraction of the weighted average of the three results. More specifically, each data flow is checked by each algorithm and the classification accuracy is obtained. Then the maximum accuracy isincreased by a weight equal to 0.6 whereas in the rest of the forecasts this weight is equal to 0.2 and the weighted average is calculated. This process is presented in the pseudocode of the following Algorithm 1.

```
Algorithm 1. Dynamic Weighted Average
Input: $x_1$, $x_2$, $x_3$ /* classifier accuracy
Step 1: if (($x_1 > x_2$) && ($x_1 > x_3$))
        max = $x_1$; else if($x_2 > x_3$)
        max = $x_2$; else max = $x_3$;
Step 2: Set $w_{max}$=0.6, $w_1$=0.2 and $w_2$=0.2
Step 3: Calculate $\bar{x} = \frac{x_1 w_1 + x_2 w_2 + x_{max} w_{max}}{w_1 + w_2 + w_{max}}$
Output: The dynamic weighted average of classification accuracy
```

The use of the weighted average potential significantly enhances the visualization of the trends in the estimated state, as it eliminates or at least minimizes the statistical noise of the data streams. This is one of the best ways to assess the strength of a trend and the likelihood of its reversal, as it places more weight on the classification with the highest accuracy. It provides real indications before the start of a new situation or event, thus allowing for a quick and optimal decision.

It is also important to note that this dynamic process ensures the adaptation of the system to new situations, by offering generalization which is one of the key issues in the field of machine learning. In this way we are implementing a robust framework capable of responding to high complexity problems. Also, this architecture greatly accelerates the process of making an optimal decision with the rapid convergence of the multiple model, which is less noisy and much more reliable than a single learning algorithm [23].

3 Ensemble Algorithms

3.1 Adaptive Random Forests

It is clear that data flow management and especially knowledge extraction procedures with Machine Learning algorithms applied on the flows, are unlikely to be performed with iterations over input data. Accordingly, the adaptation of the Random Forest algorithm depends on a suitable accumulation process that is partly achieved by bootstrap, and partly by limiting any decision to divide the sheets into a subset of attributes. This is achieved by modifying the base tree algorithm, by effectively reducing the set of features examined for further separation into random subsets of size m, όπου $m < M$ (M corresponds to the total number of characteristics examined per case) [19].

In non-streaming bagging, each of the n-base models is trained in a Z-size bootstrap sample, created by random samples being substituted by the original training kit. Each bootstrapped sample contains a prototype training snapshot K, where P (K = k) follows a binomial distribution. For large values of Z this binomial distribution adheres to a Poisson distribution with λ = 1. In contrast to the ARF method for streaming data, Poisson is used with λ = 6 instead of Poisson λ = 1. This “feedback” has the practical effect of increasing the possibility of assigning higher weights to instances during the training of the basic models.

ARF is an adaptation of the original Random Forest algorithm, which has been successfully applied to a multitude of machine learning tasks. In layman’s terms the original Random Forest algorithm is an ensemble of decision trees, which are trained using bagging and where the node splits are limited to a random subset of the original set of features. The “Adaptive” part of ARF comes from its mechanisms to adapt to different kinds of concept drifts, given the same hyper-parameters.

The overall ARF pseudo-code is presented below [19].

Algorithm 2. Adaptive Random Forests

```
function ARF ($m, n, \delta_w, \delta_d$)
 $T \leftarrow CreateTrees(n)$
 $W \leftarrow InitWeits(n)$
 $B \leftarrow \emptyset$
 while HasNext(S) do
   $(x, y) \leftarrow next(S)$
   for all $t \in T$ do
       $\check{y} \leftarrow predict(t, x)$
       $W_{(t)} \leftarrow P(W_{(t)}, \check{y}, y)$
       RFTreeTrain ($m, t, x, y$)
       if C ($\delta_w, t, x, y$) then
           $b \leftarrow CreateTrees()$
           $B(t) \leftarrow b$
       end if
   end for
   for all $b \in B$ do
       RFTreeTrain ($m, b, x, y$)
   end for
 end while
end function
```

Where m: the maximum features evaluated per split; n: the total number of trees (n = |T|); δ_w: the warning threshold; δ_d: the drift threshold; c(·): the change detection method; S: the data stream; B: the Set of background trees; W(t): the Tree t weight; P (·): the learning performance estimation function.

3.2 K-NN Classifier with Self Adjusting

The k-NN SAM algorithm is inspired by the Short-Term and Long-Term memory (STM & LTM) model [27]. The information arriving in STM, are accompanied by relevant knowledge from the LTM. The information that receives enough attention is transferred in the LTM in the form of the Synaptic Consolidation. The memories are assigned the following sets M_{ST}, M_{LT}, M_C which are subsets of the $R^n \times \{1,\ldots,c\}$. The STM is a dynamic sliding window that contains the most *recent m* examples of the data flow [27]:

$$M_{ST} = \{(x_i, y_i) \in R^n \times \{1,\ldots,c\} | i = t - m + 1,\ldots,t\} \tag{3}$$

The LTM retains all of the initial information and unlike the STM, it is not a continuous part of the data flow. It is a set of points p:

$$M_{LT} = \{(x_i, y_i) \in R^n \times \{1,\ldots,c\} | i = 1,\ldots,p\} \tag{4}$$

The combined memory C_M is the union of both memories with size m + p:

$$M_C = M_{ST} \cup M_{LT} \tag{5}$$

Each set includes the weighted k-NN classifier:

$$R^n \times \{1,\ldots,c\}, k - NN_{M_{ST}}, k - NN_{M_{LT}}, k - NN_{M_C} \tag{6}$$

The k-NN approach assigns a label to each data point x based on a set $Z = \{(x_i, y_i) \in R^n \times \{1,\ldots,c\} | i = 1,\ldots,n\}$:

$$k - NN_Z(x) = argmax\left\{ \sum_{x_i \in N_k(x,Z) | y_i = \hat{c}} \frac{1}{d(x_i, x)} |\hat{c} = 1,..,c \right\} \tag{7}$$

where $d(x_i, x)$ is the Euclidean distance between two points and $N_k(x, Z)$ returns the set comprising of the k nearest neighbors x in Z [27].

3.3 Primal Estimated Sub-Gradient Solver for SVM

The SPegasos is a simple and effective stochastic sub-gradient descent algorithm for solving the optimization problem by using SVM [29]. Initially, w_1 is defined. In the t iteration of the algorithm, we use a random training example (x_{i_t}, y_{i_t}) by choosing an index $i_t \in \{1,\ldots,m\}$. Then we use the following Eq. (8):

$$\min_{w} \frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum_{x,y\in S} l(w;(x,y)) \tag{8}$$

where $l(w;(x,y)) = \max\{0, 1 - y\langle w, x\rangle\}$, with a sample (x_{i_t}, y_{i_t}), giving input to the following function:

$$f(w, i_t) = \frac{\lambda}{2}\|w\|^2 + l(w;(x_{i_t}, y_{i_t})) \tag{9}$$

where

$$\nabla_t = \lambda w_t - \mathbb{1}[y_{i_t}\langle w_t, x_{i_t}\rangle < 1] y_{i_t} x_{i_t} \tag{10}$$

and $\mathbb{1}[y_{i_t}\langle w_t, x_{i_t}\rangle < 1]$ is the index function, which takes the value 1 if the argument is true and it becomes equal to 0 in any other case. Then, we update the relation $w_{t+1} \leftarrow w_t - \eta_t \nabla_t$ by using weight step $\eta_t = \frac{1}{\lambda t}$. After T iterations, the last value of the weight is the $\mathrm{w}_{\mathrm{T}\ +1}$ [29].

4 Results and Discussion

We have evaluated the performance of the proposed methods by measuring the average values for Kappa Statistic and Kappa Temporal Statistic. The results of all experiments are shown in the following Tables 1, 2 and 3.

The learning evaluation used 10,000 instances and the validation of the results was done by employing the Prequential Evaluation method [3]. The training window used 5,000 instances. Window based approaches were allowed to store 5,000 samples (for the sake of completeness, we also report the error rates of all window-based approaches with a window size of 1,000 samples) but never more than 10% of the whole dataset. This large amount gives the approaches a high degree of freedom and prevents the concealment of their qualities with a too restricted window.

Table 1. Results for the *water_tower_dataset*

Network traffic analysis				
Performance metrics				
Classifier	Window size 5000		Window size 1000	
	Kappa statistic	Kappa temporal statistic	Kappa statistic	Kappa temporal statistic
k-NN SAM	**74.56%**	**75.29%**	**79.22%**	**79.96%**
SPegasos	72.07%	72.94%	74.65%	76.51%
ARF	71.86%	72.47%	75.24%	77.72%
Ensemble averaging	**73.52%**	**74.26%**	**77.51%**	**78.82%**

Table 2. Results for the *gas_dataset*

Network traffic analysis				
Performance metrics				
Classifier	Window size 5000		Window size 1000	
	Kappa statistic	Kappa temporal statistic	Kappa statistic	Kappa temporal Statistic
k-NN SAM	**72.03%**	**72.73%**	**74.18%**	**75.41%**
SPegasos	72.02%	72.69%	73.94%	75.01%
ARF	71.83%	72.41%	73.87%	74.89%
Ensemble averaging	**71.99%**	**72.66%**	**74.07%**	**75.23%**

Table 3. Results for the *electric_dataset*

Network traffic analysis				
Performance metrics				
Classifier	Window size 5000		Window size 1000	
	Kappa statistic	Kappa temporal statistic	Kappa statistic	Kappa temporal statistic
k-NN SAM	**75.72%**	**76.33%**	**78.93%**	**79.56%**
SPegasos	75.63%	76.12%	77.95%	78.93%
ARF	74.47%	75.16%	76.18%	77.97%
Ensemble averaging	**75.45%**	**76.05%**	**78.19%**	**79.17%**

The assessment of the actual error of the data flow classifiers, is done in terms of the Accuracy Kappa statistic and the Kappa-Temporal statistic. The "true" label is presented right after the instance has been used for testing, where there is a delay between the time an instance is presented and the moment in which its "true" label becomes available [30]. The use of the dynamically estimated weighted average is the optimal approach, considering that is solves a real problem of information systems security, where it is rare for all data flows to have the same importance. The algorithm, which has achieved the highest accuracy for each data stream, is multiplied by the corresponding weighting factor of 0.6, reflecting its transient superiority and hence the relative importance of the model to the particular algorithm at that time.

Based on this technique, the model is led to a relatively smooth but high learning rate, which determines how quickly learning is converging. A high rate of learning can lead to faster convergence and oscillation around optimal weight values, while the low rate of learning results in slower convergence and can lead to trapping at local extremes. The high learning rate is confirmed by the high accuracy rates of the model, since very small size data flows are considered compared to the evaluation of a batch data set. According to this technique, the quality of the model's adaptation is interpreted as a "better forecasting" rate, due to the increased percentage of classification precision. More specifically, the temporal bias created to the dynamics of a model at a specific time, is reflected in the high precision percentages of Table 1.

An additional important interpretation, resulting from the high accuracy of the 9 learning algorithms and the mild "mutation", attributable to the dynamically determined weighted average, is to assist in discovering the local extremes that may be included in a data flow or in a learning window. This is expected, since new areas of the multidimensional solution space are examined.

On the contrary, if the "mutation" rate was too high, it could lead to a reduction in the exploitation of highly suitable areas of the solution space, and it could trap the system into solutions that do not generalize [30, 33]. An important comment also refers to the Kappa coefficient that links the level of observed agreement to the level of the random agreement. It estimates the variability in each observer rater variation that occurs when the same observer - evaluates differently in repeated evaluations of the same size. The maximum value of the Kappa index represents the full agreement between observers - markers, while the minimum value 0 is interpreted as there is only random agreement and thus no reliability between observers - markers.

As we can see, there is considerable reliability in all cases tested, which also strengthens the overall reliability and usability of the proposed model. Similarly, by attempting a comparison of the results between the algorithms, we see that the ARF method generally needs a larger number of cases to yield new data. In addition, ARF works by combining some loose linear boundaries on the decision surface, as opposed to SPegasos which can achieve max margin in non-linear boundaries. Therefore, given that sliding windows are characterized by a small amount of data, SPegasos yielded higher success rates than ARF. Regarding the comparison between SPegasos and k-NN SAM, an clear reason that k-NN SAM performed better, is because a particular problem is located in a high-dimensional space where this algorithm is more efficient. Also, the optimal combination of the two levels of memory, the different retention intervals between the memories and the transfer of knowledge, has been shown to minimize errors and to increase classification accuracy.

5 Conclusions

An innovative, reliable and highly effective cyberattack detection system, based on sophisticated computational intelligence, was presented in this paper. The DELDaStrA, is an innovative effort to analyze large-scale, reliable and accurate data flows in order to detect cyber-attacks in critical infrastructure networks. The implementation of DELDaStrA was based on the philosophy of the dynamic ensemble learning method, which ensures the adaptation of the system to new situations offering impartiality and generalization. It is a robust framework capable of responding to high complexity problems. The performance of the proposed system has been tested by using three multidimensional datasets of high complexity. These datasets were obtained after extensive research in the operation of ICS (SCADA, DCS, PLC). They realistically state the operating states of these devices under normal conditions and under situations of cyberattacks. The very high precision results that have emerged, reinforce the general methodology followed. Proposals for the development and future improvements of this system, should focus on further optimizing the algorithms used to achieve an even more efficient, accurate and faster classification process. Also, new approaches for further optimization should be considered, by employing self-improvement and adaptive learning, which will fully automate the cyber-detection process.

References

1. Ahmim, A., Ghoualmi-Zine, N.: A new adaptive intrusion detection system based on the intersection of two different classifiers. Int. J. Secur. Netw. **9**(3), 125–132 (2014)
2. Aretz, K., Bartram, S.M., Pope, P.F.: Asymmetric loss functions and the rationality of expected stock returns. Int. J. Forecast. **27**(2), 413–437 (2011)
3. Brzezinski, D., Stefanowski, J.: Prequential AUC for classifier evaluation and drift detection in evolving data streams. In: Appice, A., Ceci, M., Loglisci, C., Manco, G., Masciari, E., Ras, Z.W. (eds.) NFMCP 2014. LNCS (LNAI), vol. 8983, pp. 87–101. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17876-9_6
4. Chand, N., Mishra, P., Krishna, C.R., Pilli, E.S., Govil, M.C.: A comparative analysis of SVM and its stacking with other classification algorithm for intrusion detection. In: Proceedings - 2016 International Conference on Advances in Computing, Communication and Automation, ICACCA 2016, pp. 1–6 (2016)
5. Dedić, N., Stanier, C.: Towards differentiating business intelligence, big data, data analytics and knowledge discovery. In: Piazolo, F., Geist, V., Brehm, L., Schmidt, R. (eds.) ERP Future 2016. LNBIP, vol. 285, pp. 114–122. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58801-8_10
6. Demertzis, K., Iliadis, L.: A hybrid network anomaly and intrusion detection approach based on evolving spiking neural network classification. In: Sideridis, A.B., Kardasiadou, Z., Yialouris, C.P., Zorkadis, V. (eds.) E-Democracy 2013. CCIS, vol. 441, pp. 11–23. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11710-2_2
7. Demertzis, K., Iliadis, L.: Evolving computational intelligence system for malware detection. In: Iliadis, L., Papazoglou, M., Pohl, K. (eds.) CAiSE 2014. LNBIP, vol. 178, pp. 322–334. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07869-4_30
8. Demertzis, K., Iliadis, L.: Evolving smart URL filter in a zone-based policy firewall for detecting algorithmically generated malicious domains. In: Gammerman, A., Vovk, V., Papadopoulos, H. (eds.) SLDS 2015. LNCS (LNAI), vol. 9047, pp. 223–233. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17091-6_17
9. Demertzis, K., Iliadis, L.: A bio-inspired hybrid artificial intelligence framework for cyber security. In: Daras, N.J., Rassias, M.T. (eds.) Computation, Cryptography, and Network Security, pp. 161–193. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18275-9_7
10. Demertzis, K., Iliadis, L.: SAME: an intelligent anti-malware extension for android ART virtual machine. In: Núñez, M., Nguyen, N.T., Camacho, D., Trawiński, B. (eds.) ICCCI 2015. LNCS (LNAI), vol. 9330, pp. 235–245. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24306-1_23
11. Demertzis, K., Iliadis, L.: Bio-inspired hybrid intelligent method for detecting android malware. In: Kunifuji, S., Papadopoulos, G.A., Skulimowski, A.M.J., Kacprzyk, J. (eds.) Knowledge, Information and Creativity Support Systems. AISC, vol. 416, pp. 289–304. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-27478-2_20
12. Demertzis, K., Iliadis, L.: Ladon: a cyber-threat bio-inspired intelligence management system. J. Appl. Math. Bioinf. **6**(3), 45–64 (2016)

13. Demertzis, K., Iliadis, L., Spartalis, S.: A spiking one-class anomaly detection framework for cyber-security on industrial control systems. In: Boracchi, G., Iliadis, L., Jayne, C., Likas, A. (eds.) EANN 2017. CCIS, vol. 744, pp. 122–134. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65172-9_11
14. Demertzis, K., Iliadis, L., Anezakis, V.-D.: An innovative soft computing system for smart energy grids cybersecurity. Adv. Build. Energy Res. **12**(1), 3–24 (2018)
15. Demertzis, K., Iliadis, L., Anezakis, V.D.: A deep spiking machine-hearing system for the case of invasive fish species. In: 2017 IEEE International Conference on Innovations in Intelligent Systems and Applications, pp. 23–28. IEEE (2017)
16. Demertzis, K., Iliadis, L., Anezakis, V.-D.: Commentary: Aedes albopictus and Aedes japonicus—two invasive mosquito species with different temperature niches in Europe. Front. Environ. Sci. **5**(DEC), 85 (2017)
17. Dietterich, Thomas G.: Ensemble methods in machine learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 1–15. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45014-9_1
18. Farda, N.M.: Multi-temporal land use mapping of coastal wetlands area using machine learning in Google earth engine. In: 5th Geoinformation Science Symposium 2017, vol. 98, no. 1, pp. 1–12 (2017)
19. Gomes, H.M., et al.: Adaptive random forests for evolving data stream classification. Mach. Learn. **106**(9–10), 1469–1495 (2017). https://doi.org/10.1007/s10994-017-5642-8
20. Hurst, W., Merabti, M., Fergus, P.: A survey of critical infrastructure security. In: Butts, J., Shenoi, S. (eds.) ICCIP 2014. IAICT, vol. 441, pp. 127–138. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45355-1_9
21. Krawczyk, B., Minku, L.L., Gama, J., Stefanowski, J., Woźniak, M.: Ensemble learning for data stream analysis: a survey. Inf. Fus. **37**, 132–156 (2017)
22. Krawczyk, B., Cano, A.: Online ensemble learning with abstaining classifiers for drifting and noisy data streams. Appl. Soft Comput. **68**, 677–692 (2018)
23. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms, 1st edn. Wiley, Hoboken (2004). ISBN 0-471-21078-1
24. Kushner, H.J., Yin, G.G.: Stochastic Approximation and Recursive Algorithms and Applications. Stochastic Modeling and Applied Probability, vol. 35, 2nd edn. Springer, Heidelberg (2003). https://doi.org/10.1007/b97441
25. Lin, J.: The Lambda and the Kappa. IEEE Internet Comput. **21**(5), 60–66 (2017)
26. Liu, S.M., Liu, T., Wang, Z.Q., Xiu, Y., Liu, Y.X., Meng, C.: data stream ensemble classification based on classifier confidence. J. Appl. Sci. **35**(2), 226–232 (2017)
27. Losing, V., Hammer, B., Wersing, H.: KNN classifier with self-adjusting memory for heterogeneous concept drift. In: 16th IEEE International Conference on Data Mining, vol. 7837853, pp. 291–300. IEEE (2017)
28. Rani, M.S., Sumathy, S.: Analysis of KNN, C5.0 and one class SVM for intrusion detection system. Int. J. Pharm. Technol. **8**(4), 26251–26259 (2016)
29. Shalev-Shwartz, S., Singer, Y., Srebro, N., Cotter, A.: Pegasos: primal estimated sub-gradient solver for SVM. Math. Program. **127**(1), 3–30 (2011)
30. Vinagre, J., Jorge, A.M., Gama, J.: Evaluation of recommender systems in streaming environments. In: Workshop on Recommender Systems Evaluation: Dimensions and Design, SV, US, pp. 1–6 (2014)
31. Wang, C., Fang, L., Dai, Y.: A simulation environment for SCADA security analysis and assessment. In: Conference on Measuring Technology and Mechatronics Automation, vol. 1, pp. 342–347. IEEE (2010)

32. Zhou, Z.H.: Ensemble Methods: Foundations and Algorithms. Chapman & Hall/CRC Machine Learning & Pattern Recognition Series, 1st edn. CRC Press, T&F, New York (2012)
33. Žliobaitė, I., Bifet, A., Read, J., Pfahringer, B., Holmes, G.: Evaluation methods and decision theory for classification of streaming data with temporal dependence. Mach. Learn. **98**(3), 455–482 (2014)