Applied Soft Computing 62 (2018) 592–601

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Large-scale cyber attacks monitoring using Evolving Cauchy Possibilistic Clustering

Igor Škrjanc[a,*], Seiichi Ozawa[b], Tao Ban[c], Dejan Dovžan[a]

[a] Faculty of Electrical Engineering, University of Ljubljana, Slovenia
[b] Kobe University, Kobe, Japan
[c] National Institute of Information and Communications Technology, Tokyo, Japan

ARTICLE INFO

Article history:
Received 25 November 2016
Received in revised form 30 October 2017
Accepted 4 November 2017
Available online 16 November 2017

Keywords:
Big-data
Data stream
Evolving clustering
Cauchy density
Cyber security

ABSTRACT

We are living in an information age where all our personal data and systems are connected to the Internet and accessible from more or less anywhere in the world. Such systems can be prone to cyber-attacks; therefore the monitoring and identification of cyber-attacks play a significant role in preventing the abuse of our data and systems. The majority of such systems proposed in the literature are based on a model/classifiers built with the help of classical/off-line learning methods on a learning data set. Since cyber-attacks evolve over time such models or classifiers sooner or later become outdated. To keep a proper system functioning the models need to be updated over a period of time. When dealing with models/classifiers learned by classical off-line methods, this is an expensive and time-consuming task. One way to keep the models updated is to use evolving methodologies to learn and adapt the models in an on-line manner. Such methods have been developed, extensively studied and implemented for regression problems. The presented paper introduces a novel evolving possibilistic Cauchy clustering (eCauchy) method for classification problems. The given method is used as a basis for large-scale monitoring of cyber-attacks. By using the presented method a more flexible system for detection of attacks is obtained. The approach was tested on a database from 1999 KDD intrusion detection competition. The obtained results are promising. The presented method gives a comparable degree of accuracy on raw data to other methods found in the literature; however, it has the advantage of being able to adapt the classifier in an on-line manner. The presented method also uses less labeled data to learn the classifier than classical methods presented in the literature decreasing the costs of data labeling. The study is opening a new possible application area for evolving methodologies. In future research, the focus will be on implementing additional data filtering and new algorithms to optimize the classifier for detection of cyber-attacks.

1. Introduction

Recent developments in the Information and Computer Technology (ICT) smart technologies such as data processing, pattern or feature extraction and their classification are an important part of many automated operations in the real world. By sensing and processing various kinds of information from the physical world, many applications and tasks can be improved. By understanding the big data, an improvement in decision making can be utilized. In general analysis of data can result in improving the decisions, detection of faults and monitoring of the system operation. It can also be used for detection of frauds and monitoring, augmenting and enhancing the cyber security in real-time. The big data technologies enable the analysis of new types of social behavior which can significantly improve intelligence, comfort, security, etc. Analysis of big data can be efficiently performed by the use of clustering and classification techniques. These techniques are already very well established. Usually, these methods are used for preprocessing data in a batch form. In a modified form they can be efficiently used also for problems where the data are coming continuously as a data stream. In this case, the analytics should be done in real time based on obtained models. On-line analytics can be done using evolving identification methods.

The concept of evolving identification methods allows the adaptation of the parameters and the structure simultaneously. Since they are one-sweep algorithms, they are also usefully for big data

* Corresponding author.
E-mail address: igor.skrjanc@fe.uni-lj.si (I. Škrjanc).

https://doi.org/10.1016/j.asoc.2017.11.008
1568-4946/© 2017 Elsevier B.V. All rights reserved.

analytics. The advantages of evolving methods over the batch are that they are faster and can simply include new information into the model without retraining the whole model [1]. On the other hand, they usually generate a bit less accurate models. The evolving methods were first based on a neuro-fuzzy principle. The adaptation algorithms were based on gradient decent and chain rule approach [2]. Further the development focused on evolving fuzzy models. Where on-line clustering methods derived from the off-line versions are used for space partitioning and recursive least squares are used for local parameter identification. This recently transitioned in the idea of cloud based identification where instead of membership degrees the densities, and instead of clusters clouds are used. The presented method can be classified as a cloud based identification method.

In the digitalized world majority of critical data and systems are connected to the Internet and can be accessed from practically anywhere in the world. To protect the data and systems from unauthorized access and attacks systems for monitoring the network connections are used. Such systems should be able to distinguish between a normal connection and a cyber-attack. Since the cyber-attacks evolve over time [3] and new attacks emerge, the use of an evolving algorithm seems a natural solution for monitoring cyber-attacks. The approaches found in the literature are usually based on off-line learning methods. The classifier structure is learned beforehand based on training data. The classifier is then used for detection of intrusion. The classifier is fixed and does not take into account possible new information about the novel attacks. To include the new information the whole classifier must be retrained. Most of the currently available evolving methods are proposed for regression problems. In this paper, a novel on-line learning algorithm is presented that can adapt the classifier in an on-line manner. Further, the paper introduces the idea of using evolving methodologies for designing a cyber-attack detection system. As it will be seen from the obtained results, such system could in practice decrease the cost of labeling the learning data. Since it is an on-line approach, the inclusion of definitions for new attacks is much faster and simpler than with the batch learning algorithms. The cloud-based implementation of such system would decrease the reaction time of the detection system to new attacks.

The paper is organized as follows: after the introduction and related work review, the problem of intrusion detection is shortly described. Follows the methodology description where the Cauchy density for the data stream is presented. The general density with inner matrix norm is presented and extension to the recursive computation is given. At the end results of network intrusion detection are presented using the proposed methodology. The results are discussed and compared to other results found in the literature.

1.1. Related work

The evolving methods can be divided into two groups based on the mechanisms that they implement: methods based on fuzzy logic, which employ unsupervised space partitioning with on-line clustering (eTS [1], exTS [4,5], simpl_eTS [6], +eTS [7], FLEXFIS [8], FLEXFIS+ [9]); and methods based on neural networks that exploit the functional equivalence of neural network and fuzzy reasoning [10] (DENFIS [11], SAFIS [12], NeuroFAST [13], ENFM [14]). Identification based on data clouds was introduced recently [15,16], which is very similar to the fuzzy logic concept.

Depending on the learning abilities, the evolving methods can be divided into: *Adaptive methods* (e.g., ANFIS [17], rFCM [18], rGK [19], rPFM [20], AFPC [21]), where the initial structure of the fuzzy model must be given. The number of space partitions/clusters does not change over time, only the parameters of the membership functions and local models are adapted; *Incremental methods* (e.g., NeuroFAST [13], DENFIS [11], eTS [1], FLEXFIS [8], PANFIS [22]), Incremental Granual Fuzz Model [23], FCRM [24], eFT [25] where only adding mechanisms are implemented; *Evolving methods* (e.g., SAFIS [12], SOFNN [26], ENFM [14], eTS+ [7], ENFM [14], FLEXFIS++ [27], AHLTNM [28], SOFMLS [29], aFCR [30], ePL [31], ENF [32]) which, besides an adding mechanism, implement removing and some of them also merging and splitting mechanisms. The above-mentioned methods were mainly developed for on-line learning of regression models. They use recursive least squares method for model parameters estimation and recursive/on-line clustering for space partitioning. On-line clustering can also be used for solving the classification problems as in our case.

When dealing with the on-line clustering algorithms, the clustering of the input-output space is updated using a new data sample from the stream. The estimation of the clusters' parameters is calculated by the recursive clustering algorithm. The on-line clustering algorithms are usually derived from off-line clustering algorithms. Recursive version of Gustafson-Kessel clustering is presented in [33,19]. In [1] an on-line version of subtractive clustering is presented. In [14] a recursive method based on Gath-Geva clustering algorithm is introduced, and in [34], a recursive possibilistic fuzzy modeling approach is given. The basic on-line clustering algorithms are based on the Euclidian distance, which results in hyperspherical shapes of clusters [35]. By introducing a fuzzy covariance matrix and its inverse, by using Mahalanobis distance, also the clusters of hyper-ellipsoid shapes can be detected as shown in [19,33,36]. More on evolving approaches and on-line clustering can be found in [37].

The presented idea of classification algorithm merges the concepts of possibilistic clustering (PCM) [38] and possibilistic fuzzy clustering (PFCM) [39–41] algorithms together with the idea of density and clouds given in [42,43]. With this, an on-line clustering approach was obtained, which does not assume the normalized membership values of the data sample, as fuzzy c-means based clustering algorithms. A data sample might have a zero membership degree to clusters when it is far away from it. This fact can be utilized for detecting new patterns in the data or detecting outliers. The space partitioning is made by using an evolving methodology which classifies the data into clusters and adds new clusters when necessary. The density in [42] is based on Euclidean distance among the data samples which belong to the cluster. In this paper, the density is generalized by using a general inner matrix norm. The proposed method is a continuation of previously published work in the area of evolving systems, such as in [7], where evolving Takagi–Sugeno fuzzy system for streaming data (eTS+) is presented, or in [15,16] where cloud-based identification is presented and basic problems of such identification are given. The evolving methodology was used in [44] for LRF data mapping and in [36] for process monitoring.

The presented method Evolving Cauchy Possibilistic Clustering was implemented as a basis for large-scale monitoring of cyber-attacks. For this purpose a classifier is built which is capable of distinguishing between a normal network connection and attack. In this paper, the data set generated and managed by MIT Lincoln Labs for the purpose of 1998 DARPA Intrusion Detection Evaluation Program was used [45]. The intrusion detection systems play a fundamental role in the prevention of security breaches. Usually, the intrusion detection systems (IDSs) are rule-based. This limits the detection of novel intrusions. Moreover, encoding rules is time-consuming and highly depends on the knowledge of known intrusions [46]. Therefore, several approaches were proposed over the past decade, that are based on machine learning and data mining principles [47]. In [46] a random forest approach is proposed to build the classifier. The classifier is tested on a 10 percent KDD data set and achieves accuracy around 98 percent. In [48] a classifier is designed based on the principal component analysis. The reported attack detection rate of this classifier was 98 percent while keeping false alarm rate under 1 percent. In [49] various combinations of discretization and filtering methods are tested to improve the classification performance. The algorithms are compared on subset data that is comprised of only 10 percent of the original training data set. They tested the Entropy Minimization Discretization method, Proportional k-Interval Discretization method, Equal Width Discretization, and Equal Frequency Discretization method in combination with naive Bayes, C4.5 classifier and different feature selection methods. The best-reported error was 3 percent with 99.98 percent of attacks detected and with 6 percent of false positive detections. The results were obtained using 10-fold cross-validation. However, the results differ a substantially depending on the selection of data set subset. Based on their comparison to other methods and different subsets of the training and testing data, the accuracy of classification from about 91 percent to 95 percent is to be expected. Similar results are presented in [50] where the analysis is also extended to different types of attacks. In [51] a fast feature reduction method is presented and compared with other feature reduction methods in combination with Bayes and KNN classifier. The performance is tested on a 10 percent data set, reporting the success rate at around 98 percent. In [52] a neural network approach is used to classify the attacks. They divide the attacks into four groups. For each group, they design the neural network that can decide if the current sample belongs to a certain attack group. For each attack type, different input variables were selected. They tested the approach on the whole data set. The reported precision of the neural network was around 97 percent with around 3 percent of false alarms. The accuracy of support vector machine and k-means classifier was reported to be around 90 percent with 3 percent and 6 percent false alarm rate, respectively. Similar results for clustering classifiers are reported in [53] where four most representative clustering techniques are tested: k-means, c-means, Mountain and Subtractive clustering. The algorithms were tested on a 10 percent subset of the KDD data set. The accuracy of the k-means and c-means clustering techniques was around 91 percent, while Mountain and Subtractive clustering achieved the accuracy at around 75–78 percent. The support vector machine is studied in [54]. In [55] principal component neural networks are proposed for intrusion detection system. The semi-supervised learning of the neural network is proposed in [56]. More on intrusion detection systems can be found in [57,58].

The contribution of the presented work lies in the introduction of evolving methodology for cyber-attack monitoring which can keep the classifier up to date. The paper also extends the literature on evolving methods for classification problems.

2. Problem description

The KDD data is comprised of three different sets named learning set, testing set, and 10 percent set. The learning set consists of 4898431 connection vectors, the testing set of 311029 connection vectors and 10 percent set consist of 494021 connection vectors. The connection vectors consist of 41 input variables explained in detail in [59]. The goal is to design the IDS that is able to label the attacks and normal behavior based on the input variables.

For each connection vector, a label is given (*Action*) that labels type of the connection. A connection is labeled either as normal or as an attack. In learning and testing data set there are 22 and 37 different attack types, respectively. The attacks can be grouped into four different groups [60]:

Denial of Service (DoS): attack where attackers overload the memory or CPU resources so that the server is not able to handle legitimate users,

User-to-Root (U2R): attacker access the server with normal account and then tries to exploit vulnerability of the system to get root access to the system,

Remote-to-Local (R2L): unauthorized access from a remote machine and

Probing: an attempt to gather information about the network of computers to circumvent its security controls.

The grouping of the attack types into these four groups is shown in Table 1.

Table 1
Different groups of cyber-attacks.

DoS	R2L	U2R	Probe
back	ftp_write	buffer_overflow	ipsweep
land	guess_passwd	loadmodule	nmap
neptune	imap	perl	portsweep
pod	multihop	rootkit	satan
smurf	phf	sqlattack	mscan
teardrop	spy	xterm	saint
apache2	warezclient	ps	
processtable	warezmaster		
worm	xlock		
udpstorm	xsnoop		
mailbomb	snmpgetattack		
	snmpguess		
	httptunnel		
	sendmail		
	named		

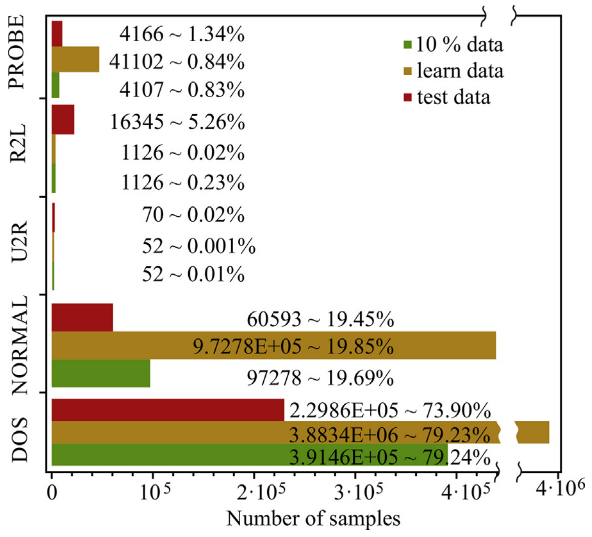


Fig. 1. Network connection distribution of the data sets.

Note that some of the attacks are only present in the learning data set and not in the testing data set (warez client and spy) and vice versa (snmpgetattack, sqlattack, named, xlock, xsnoop, sendmail, saint, apache2, udpstorm, xterm, mscan, processtable, ps, httptunnel, worm, snmpguess, and mailbomb) [61].

The distribution of the connection types for data sets is shown in Fig. 1. It can be seen that the data sets are very unbalanced. The majority of connections are labeled as DoS attacks and normal connections. The data sets include very little samples of other attack types (probe, R2L and U2R) which makes them very hard to classify [50].

3. Methodology

Since the cyber-attacks evolve [3] and new attacks emerge, the evolving methodology was tested in this paper as a basis for IDS.

The presented approach does not assume normalized membership (for example fuzzy c-means). This property is utilized for detecting new clusters. The membership of the data sample to the cluster is calculated based on Cauchy density. The algorithm is explained in the next subsections.

3.1. Cauchy density for data stream

The Cauchy density of sample k for the cluster j is defined as γ_k^j. The density is in general defined for batch data as a sum of distances between the current sample $\boldsymbol{z}(k)$ and all the previous samples belonging to the particular cluster [43,15]:

$$\gamma_k^j = \frac{1}{1+\frac{1}{\sigma_l^2}\frac{\sum_{i=1}^{M^j} d_{ki}^j}{M^j}} \quad j = 1, \ldots, m, \tag{1}$$

where σ_l is normalization constant that spreads or narrows the membership functions, d_{ki}^j denotes the square of the Euclidean distance between the current data sample $\boldsymbol{z}(k)$ and the ith sample from the jth cluster $\boldsymbol{z}_i^j$ as follows:

$$d_{ki}^j = (\boldsymbol{z}(k) - \boldsymbol{z}_i^j)^T (\boldsymbol{z}(k) - \boldsymbol{z}_i^j) \tag{2}$$

The M^j denotes the number of the samples in the cluster j. The generalization of the basic density measure is realized by introducing a positive definite inner matrix norm $\boldsymbol{A}^j$ into the distance measure:

$$d_{ki}^j = (\boldsymbol{z}(k) - \boldsymbol{z}_i^j)^T \boldsymbol{A}^j (\boldsymbol{z}(k) - \boldsymbol{z}_i^j) \tag{3}$$

The Cauchy density in that sense becomes relative because the distances are weighted in different directions with different weights. The equation for Cauchy density calculation then equals to [62]:

$$\gamma_k^j = \frac{1}{1+\frac{1}{\sigma_l^2}\frac{\sum_{i=1}^{M^j} (\boldsymbol{z}(k)-\boldsymbol{z}_i^j)^T \boldsymbol{A}^j (\boldsymbol{z}(k)-\boldsymbol{z}_i^j)}{M^j}} \tag{4}$$

In order to use the Cauchy density for on-line identification the density equation (Eq. (4)) must be transformed into a recursive form [62]:

$$\gamma_k^j = \frac{1}{1 + 1/\sigma_l^2 (\boldsymbol{z}(k) - \boldsymbol{\mu}^j)^T \boldsymbol{A}^j (\boldsymbol{z}(k) - \boldsymbol{\mu}^j) + 1/\sigma_l^2 T_A^j}, \tag{5}$$

where $T_A^j = (M^j - 1)/M^j$ trace $(\boldsymbol{A}^j \boldsymbol{\Sigma}^j)$ and where

$$\boldsymbol{\mu}^j = \frac{1}{M^j}\sum_{i=1}^{M^j} \boldsymbol{z}_i^j, \tag{6}$$

where $\boldsymbol{\mu}^j$ defines the center of the jth cluster and the lower and the upper index at $\boldsymbol{z}_i^j$ define the ith sample in the jth cluster. The notation of cluster center can also be written as $\boldsymbol{\mu}_{M_j}^j$ to explicitly express that the jth cluster consists of M^j samples. The $\boldsymbol{\Sigma}^j$ denotes the covariance matrix of the jth cluster. The covariance matrix is calculated as:

$$\boldsymbol{\Sigma}_{M^j}^j = \frac{1}{M^j - 1}\sum_{i=1}^{M^j} (\boldsymbol{z}_i^j - \boldsymbol{\mu}_{M^j}^j)(\boldsymbol{z}_i^j - \boldsymbol{\mu}_{M^j}^j)^T \tag{7}$$

The density given by Eq. (5) is in its absolute form. The value of the density can be higher than one. To limit the upper-density value to one, Eq. (5) is re-formulated into the following relative form:

$$\gamma_k^j = \frac{1 + 1/\sigma_l^2 T_A^j}{1 + 1/\sigma_l^2 (\boldsymbol{z}(k) - \boldsymbol{\mu}^j)^T \boldsymbol{A}^j (\boldsymbol{z}(k) - \boldsymbol{\mu}^j) + 1/\sigma_l^2 T_A^j}, \tag{8}$$

The relative density is important because it is much easier to set various thresholds for adding new clusters or detecting the outliers.

3.2. Different inner matrix norms

For solving different clustering, classification or regression problems, different inner matrix norms can be used. The most common are identity matrix norm and inverse covariance matrix norm.

3.2.1. Identity inner norm

The most common and basic inner matrix norm equals to identity matrix $\boldsymbol{I}$. In this case, Euclid distance is obtained, and the density becomes [43,15]:

$$\gamma_k^j = \frac{1}{1 + 1/\sigma_l^2 (\boldsymbol{z}(k) - \boldsymbol{\mu}^j)^T (\boldsymbol{z}(k) - \boldsymbol{\mu}^j) + 1/\sigma_l^2 T_I^j} \tag{9}$$

where $T_I^j = (M^j - 1)/M^j$ trace $(\boldsymbol{\Sigma}^j)$. This is a very general norm and can be used in many different regression and classification problems.

3.2.2. Inverse covariance matrix as inner matrix norm

When the inner matrix norm equals to the inverse of covariance matrix $(\boldsymbol{\Sigma}^j)^{-1}$ of the corresponding data set with M^j data samples, then the distance is called Mahalanobis distance. Introducing Eq. (3) with inner norm as inverse covariance matrix into Eq. (1), the formula for Cauchy density based on Mahalanobis distance is obtained [15]:

$$\gamma_k^j = \frac{1}{1 + 1/\sigma_l^2 (\boldsymbol{z}(k) - \boldsymbol{\mu}^j)^T (\boldsymbol{\Sigma}^j)^{-1} (\boldsymbol{z}(k) - \boldsymbol{\mu}^j) + 1/\sigma_l^2 (M^j - 1)/M^j q}, \tag{10}$$

where q is rank of covariance matrix. As the covariance matrix can be also written in its singular value decomposed form as

$$\boldsymbol{\Sigma}^j = \boldsymbol{P}^j \boldsymbol{D}^j (\boldsymbol{P}^j)^T, \tag{11}$$

the Mahalanobis distance can be also described in the following form

$$d_k^j = \sum_{r=1}^{q} \frac{\left((\boldsymbol{z}(k) - \boldsymbol{\mu}^j)^T \boldsymbol{p}_r^j\right)((\boldsymbol{z}(k) - \boldsymbol{\mu}^j)\boldsymbol{p}_r^j)}{\lambda_r} \tag{12}$$

where $\boldsymbol{p}_r^j$ stands for rth eigenvector and λ_r for rth eignevalue of the covariance matrix $\boldsymbol{\Sigma}^j$ which represent the shape of jth cluster. This shows that Mahalanobis distance is the sum of normalized distances of the current data sample from the center of the cluster in the direction of eigenvector and normalized with appropriate eigenvalue. The benefit of using Mahalanobis distance is to describe the hyper-ellipsoidally shaped cluster. The size and the shape of the hyper-ellipsoid depends on the covariance matrix of the data in the cluster.

3.3. Recursive computation of Cauchy density

In the evolving algorithm, the density should be calculated recursively. The proposed approach completely assigns the observed sample to the cluster with the maximal density if it overcomes the predefined minimal value. If the maximal density of the sample is lower than the defined threshold then the sample is either treated as an outlier [63] or as an initial sample of a new cluster (depending on the application).

With a new sample in the cluster, which is denoted as $\boldsymbol{z}(k)$, the number of samples is incremented, and the mean and the cluster covariance matrix are updated. Both, the mean and the covariance matrix of the clusters are updated recursively. The update is done in the following steps. First, the difference between the current sample and the current mean value is calculated:

$$\boldsymbol{e}^j_{M^j}(k) = \boldsymbol{z}(k) - \boldsymbol{\mu}^j_{M^j}. \tag{13}$$

Next, the mean is updated

$$\boldsymbol{\mu}^j_{M^j+1} = \boldsymbol{\mu}^j_{M^j} + \frac{1}{M^j+1}\boldsymbol{e}^j_{M^j}(k). \tag{14}$$

After that the states of the un-normalized covariance matrix are computed as:

$$\boldsymbol{S}^j_{M^j+1} = \boldsymbol{S}^j_{M^j} + \boldsymbol{e}^j_{M^j}(k)(\boldsymbol{z}(k) - \boldsymbol{\mu}^j_{M^j+1})^T \tag{15}$$

and the covariance matrix is than obtained as

$$\boldsymbol{\Sigma}^j_{M^j+1} = \frac{1}{M^j}\boldsymbol{S}^j_{M^j+1} \tag{16}$$

3.4. Evolving Cauchy Possibilistic Clustering

The evolving strategies strongly depend on the nature of the data stream and the investigated problem. In the case of cyber-attacks, the samples come randomly from different classes. Dealing with such problems all evolving mechanisms of adding, merging and deleting the clusters should be implemented. The flow chart of presented evolving methodology is shown in Fig. 2. Note that in the presented case the splitting, merging and removing were not implemented.

Next, a brief description of the core evolving mechanisms such as adding and deleting the clusters are given.

3.4.1. Adding clusters

In the classification problems, use of the direct adding method is the most common. Each sample is either added to one of the existing clusters or a new cluster in initialized. In our case, a new cluster is added if

$$\max_j \gamma^j_k < \Gamma_{\max} \tag{17}$$

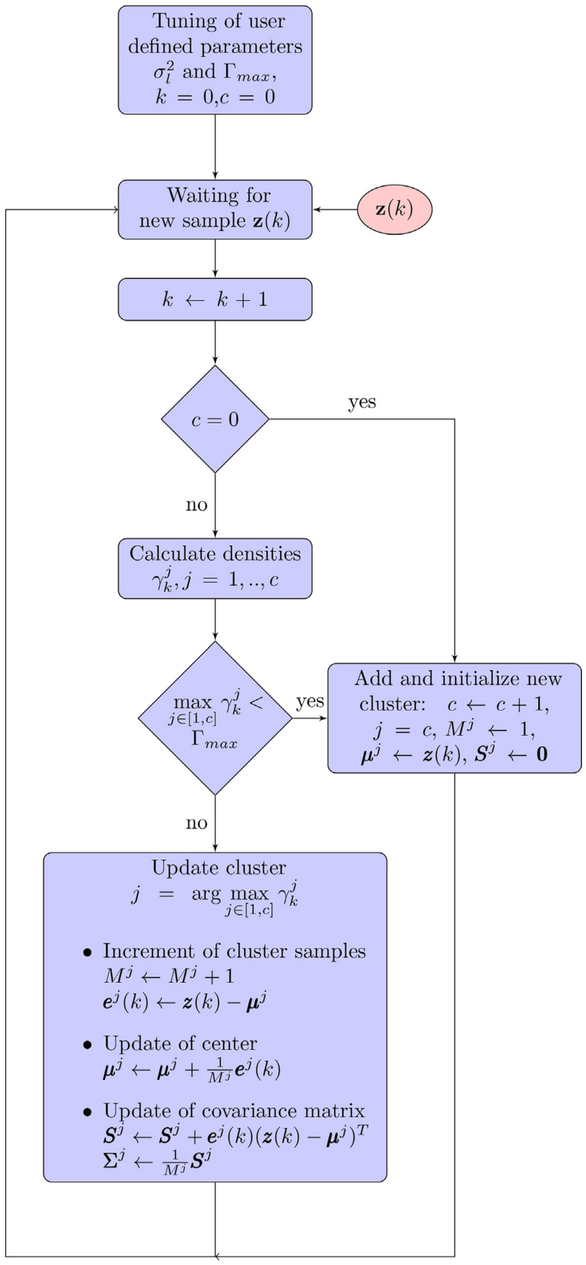
where Γ_{max} stands for the user defined maximal typicality threshold. By adding a new cluster the number of clusters is incremented $c = c + 1$, the number of elements in cluster is initialized to $M^j = 1$, the center and the covariance matrix of the cluster are initialized to $\boldsymbol{\mu}^j = \boldsymbol{z}(k)$ and $\boldsymbol{\Sigma}^j = \boldsymbol{0}$, respectively.

3.4.2. Removing clusters

The purpose of cluster removing mechanism is to detect the clusters that were added based on outliers. Namely, outliers often satisfy the adding condition. Such clusters are not valid. Since they were added based on outliers, they will probably not gather a notable number of samples M^j in a certain time frame. So if the cluster's number of samples is below a defined threshold M_{min}, the cluster is deleted [36,7].

$$M^j < M_{\min} \tag{18}$$

The deleting mechanisms work well if the data is distributed and comes into learning algorithm more or less evenly from different classes or clusters. When the data is unbalanced, as in our case, the deleting mechanism might delete a real cluster, which would decrease the performance of the classification system. In the case of the *KDD* data, deleting mechanism would treat samples from classes with a low number of samples (e.g. *R2L*, *U2R*) as outliers and would consequently delete classes created based on those samples. Therefore the deleting mechanism was not implemented in the presented case of IDS.



Tuning of user defined parameters σ_l^2 and Γ_{max}, $k = 0, c = 0$
Waiting for new sample $\mathbf{z}(k)$
$\mathbf{z}(k)$
$k \leftarrow k+1$
$c = 0$
yes
no
Calculate densities $\gamma^j_k, j = 1,..,c$
$\max_{j\in[1,c]} \gamma^j_k < \Gamma_{max}$
yes
Add and initialize new cluster: $c \leftarrow c+1$, $j = c$, $M^j \leftarrow 1$, $\boldsymbol{\mu}^j \leftarrow \boldsymbol{z}(k)$, $\boldsymbol{S}^j \leftarrow \mathbf{0}$
no
Update cluster $j = \arg\max_{j\in[1,c]} \gamma^j_k$
Increment of cluster samples $M^j \leftarrow M^j + 1$ $\boldsymbol{e}^j(k) \leftarrow \boldsymbol{z}(k) - \boldsymbol{\mu}^j$
Update of center $\boldsymbol{\mu}^j \leftarrow \boldsymbol{\mu}^j + \frac{1}{M^j}\boldsymbol{e}^j(k)$
Update of covariance matrix $\boldsymbol{S}^j \leftarrow \boldsymbol{S}^j + \boldsymbol{e}^j(k)(\boldsymbol{z}(k) - \boldsymbol{\mu}^j)^T$ $\Sigma^j \leftarrow \frac{1}{M^j}\boldsymbol{S}^j$

Fig. 2. Algorithm of Evolving Cauchy Possibilistic Clustering.

3.4.3. Merging clusters

Adding clusters is based on the distance between the current sample and the existing clusters. The distance or the adding threshold is usually dependent on the covariance matrix of the data around the cluster center. In some cases, especially when the samples come into learning algorithm randomly from different classes and are very dispersed, the algorithm usually creates more clusters than needed. To decrease the number of such clusters the merging mechanism can be implemented. The cluster merging is a mechanism that merges similar clusters together. In this way, it decreases the number of clusters and simplifies the model structure. The similarity of clusters is usually determined based on the Mahalanobis distance between two clusters, based on samples membership degrees, the correlation between clusters firing levels or in a case of regression models based on local models [36].

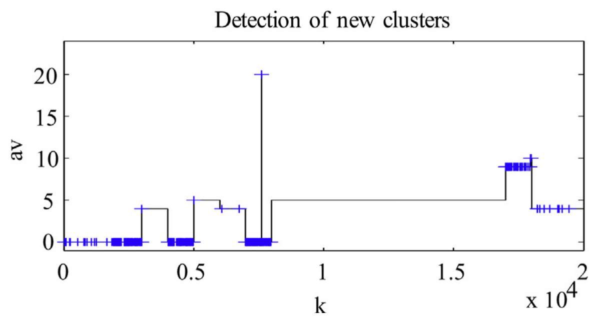


Fig. 3. Detection and initialization of a new cluster.

3.4.4. *Splitting clusters*

The mechanism for splitting clusters is meant for fine partitioning of the problem space. In the case of prediction problems, the algorithm splits clusters with high prediction errors. In the case of the classification problem, the algorithm splits the clusters that contain a certain number of miss-classified samples. In this way, finer partitioning of the problem space is obtained, and usually, the accuracy of the model is increased [36].

4. Results of network traffic monitoring using eCauchy clustering

4.1. *Experiment setup*

The presented method was tested on all three data sets (learning, testing and 10 percent data set) in an evolving mode meaning that the classifier is learned on-line as presented in Fig. 2. When a new cluster is detected the algorithm would, in practice, ask an expert for the current sample's label. In the presented case the cluster is generated automatically since labels for all data are available. The parameters of the algorithm σ_l^2 and Γ_{max} were set to 0.1 and 0.4, respectively. For classification and learning detailed class labels were used for representing attacks and normal network traffic (40 classes). The final analysis was done for five classes (*DoS*, *R2L*, *U2R*, *Probe*, *Normal*).

4.2. *Partial results*

In this subsection, partial results are presented on a block of KDD learning data set (20,000 samples) that was randomly selected to demonstrate the functioning of the method. The selected data set consisted of classes: 0-*normal*; 4-*neptune*; 5-*smurf*; 9-*portsweep*; 10-*ipsweep* and 20-*warezclient*. The selected data set was granulated into 491 clusters. The creation of new clusters is shown in Fig. 3 (crosses).

The selected data was granulated into 491 clusters. The algorithm did classify samples with activation value of 0 (normal class) into 424 clusters or granules. Class with activation value 4 (neptune attack) was described with 12 clusters, class 5 (smurf attack) with 1 cluster, class 9 (portsweep attack) with 50 clusters, class 10 (ipsweep attack) with 3 clusters and class 20 (warezclient attack) with 1 cluster. The activation value of each cluster is shown in Fig. 4 in the upper part and in the lower part the number of samples in each cluster is shown in logarithmic scale.

It is evident that some of the clusters have only one sample (the logarithmic value which is equal 0). Such clusters could also be avoided if the merging mechanism would be implemented. It can be seen that the activation value 0, which represents a normal action has relatively high number of clusters. This is due to the fact that the normal action can be very diverse. The number of clusters depends highly on the tuning parameters σ_l^2 and the maximal density threshold Γ_{max}. The tuning of these two parameters also

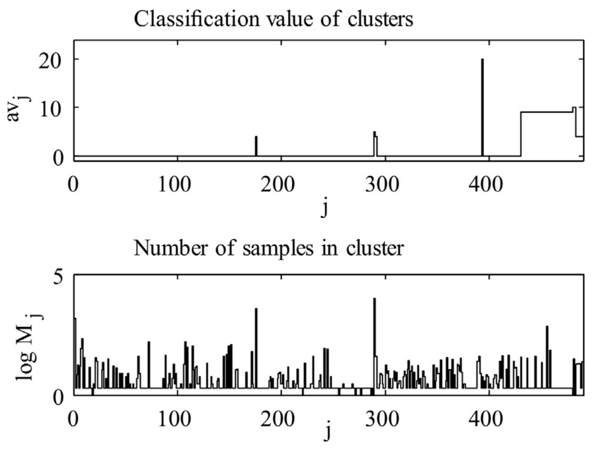


Fig. 4. Classification activation value of clusters and the number of samples in clusters.

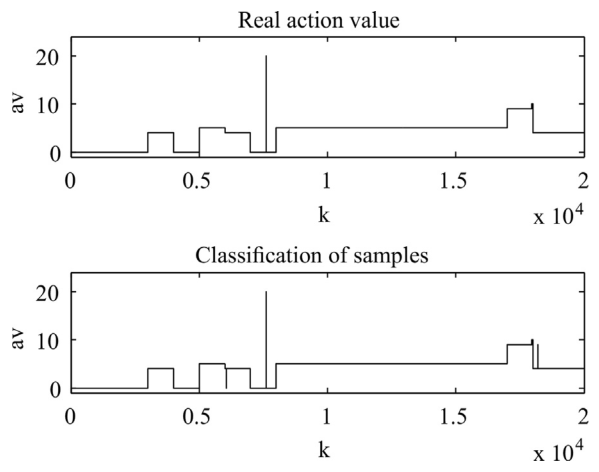


Fig. 5. Real and classified activation value for samples of data set.

influences on the number of clusters with only one representative sample.

The classification of data is presented in Fig. 5. In the upper part of Fig. 5 the real activation value (class label) is shown and in the lower part the classified activation value of different samples (class label assigned by IDS). Classification using eCauchy clustering on this data results in 2 wrong classifications.

The results of clustering are also presented with Stochastic Neighbor Embedding (tSNE) transformation technique that visualizes high-dimensional data giving to each sample a position in a two or three-dimensional map. t-SNE creates a single map that reveals the structure of the original data at many different scales. In Fig. 6 the centers of detected attack clusters are shown with colored circles the connection vectors are plotted with dots. It can be seen that the data is quite dispersed.

4.3. *Results of cyber-attacks monitoring in evolving mode*

In Tables 2–4 results are given for a full evolving mode functioning on KDD learning, test and 10% data set, respectively. The tables provide common goodness measures for classification problems: recall (TPR), precision (PPV), false positive rate (FPR), false negative rate (FNR), false discovery rate (FDR), accuracy (ACC) and number of generated clusters (Granulation). All values are in percentages.

The evolving functioning means that the classifier is constantly adapting itself. When a sample cannot be classified the algorithm asks the user to provide the label for the sample in question. The

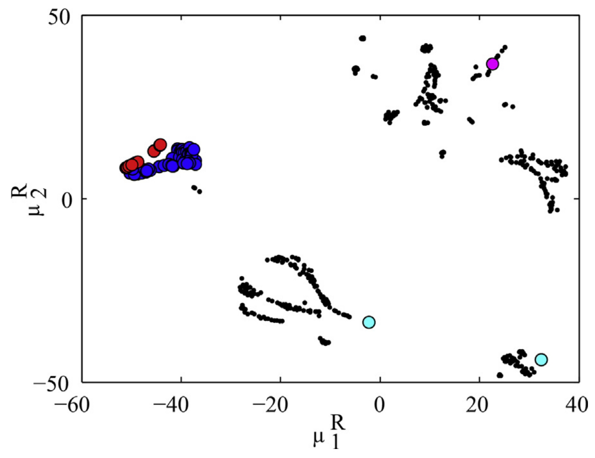


Fig. 6. Cluster centers in reduced space – circles denote the centers with different attacks.

Table 2
Results for learn KDD data set (number of generated clusters 3022, miss-classified samples 0.22%).

Class	Normal	DoS	U2R	R2L	Probe
TPR	99.49	99.93	50	23.62	94.1
PPV	99.39	99.96	74.29	59.24	92.05
FPR	0.15	0.14	0.00018373	0.0037367	0.07
FNR	0.51	0.07	50	76.38	5.9
FDR	0.61	0.04	25.71	40.76	7.95
ACC	99.78	99.92	100	99.98	99.88
Granulation	2466	215	13	30	298

Table 3
Results for test KDD data set (number of generated clusters 1212, miss-classified samples 6.08%).

Class	Normal	DoS	U2R	R2L	Probe
TPR	98.12	97.58	70	26.96	94.26
PPV	82	99.91	64.47	44.41	97.13
FPR	5.21	0.24	0.0086828	1.87	0.04
FNR	1.88	2.42	30	73.04	5.74
FDR	18	0.09	35.53	55.59	2.87
ACC	95.44	98.15	99.98	94.39	99.89
Granulation	488	214	26	115	369

Table 4
Results for 10% KDD data set (number of generated clusters 1841, miss-classified samples 0.55%).

Class	Normal	DoS	U2R	R2L	Probe
TPR	99.14	99.57	55.77	92.54	97.59
PPV	98.09	99.95	82.86	94.38	87.05
FPR	0.47	0.18	0.0012147	0.01	0.12
FNR	0.86	0.43	44.23	7.46	2.41
FDR	1.91	0.05	17.14	5.62	12.95
ACC	99.45	99.62	99.99	99.97	99.86
Granulation	1405	215	17	43	161

number of user intervention is seen directly from the number of created clusters. In the case of learning KDD data set the user had to specify 3022 labels (0.06 percent of all data) in the case of testing KDD data set the user had to specify 1212 labels (0.4 percent of all data). For 10% KDD data set the user had to label 1841 data samples (0.3 percent of all data).

The results presented in Tables 2–4 show reasonably good classification performance for normal connections, DoS and Probe attack. The classifier performance for U2R and R2L attack is rather poor. It can be seen from the results of the test data set (especially in “freez mode” Table 6) that for the R2L attack the classifier performs only a bit better than a random guess. On the contrary, the performance of R2L classification on 10% data set was reasonably good (Table 4).

Table 5
Results for test KDD set where 10% KDD data set was used as learning set (number of generated clusters 1841, miss-classified samples 8.01%).

Class	Normal	DoS	U2R	R2L	Probe
TPR	99.32	96.72	31.43	1.58	79.96
PPV	71.79	99.86	48.89	82.96	78.82
FPR	9.44	0.37	0.0073964	0.02	0.29
FNR	0.68	3.28	68.57	98.42	20.04
FDR	28.21	0.14	51.11	17.04	21.18
ACC	92.27	97.48	99.98	94.81	99.44
Granulation	1405	215	17	43	161

Table 6
Results for test KDD set where learn KDD data set was used as learning set (number of generated clusters 3022, miss-classified samples 8.33%).

Class	Normal	DoS	U2R	R2L	Probe
TPR	99.46	96.45	10	1.13	71.32
PPV	70.86	99.86	70	76.35	80.06
FPR	9.9	0.39	0.00096475	0.02	0.24
FNR	0.54	3.55	90	98.87	28.68
FDR	29.14	0.14	30	23.65	19.94
ACC	91.93	97.27	99.98	94.79	99.38
Granulation	2466	215	13	30	298

On the KDD test data set 74% of all R2L attacks were classified as a normal connection. A similar percentage was for KDD learn data set (76%), whereas with 10% KDD data set only 7% of all R2L attacks were classified as normal. On the test KDD data set 14% of all U2R attacks were classified as a normal connection and 15% as an R2L attack. In 10% KDD data set 25% of U2R attacks were classified as normal and 20% as R2L. In KDD learning set 50% of all U2R attacks were classified as normal.

4.4. Results of cyber-attacks monitoring in “freeze mode”

In the above section, the results in an evolving mode were presented. In this section, the setup of the experiment was similar as with off-line methods. First, the learning data was used to learn the classifier, the learning was then turned off, and classifier was tested on test KDD data set. Table 5 presents the results where 10% KDD data set was used for learning and Table 6 represent the results where learning KDD data set was used for learning.

Although the proposed system is not designed to function in the “freeze mode” it can be seen that the classifier still retains reasonably good performance, excluding the class R2L and U2R where the performance drops very near to the performance of a random guess. The presented results are obtained without prior variable selection or other data transformation; therefore there is still much potential for improvement of the obtained results.

4.5. Influence of parameters $\Gamma_{\max}$ and σ_l

There are two tuning parameters of the presented method Γ_{max} and σ_l. The parameter *sigma*$_l$ is one of the tuning parameters and defines the size of the clusters. This normalization constant spreads or narrows the density (membership) functions. This affects the size of the clusters (granules). The parameter Γ_{max} stands for maximal typicality threshold. This defines the threshold when a new granule (cluster) is added to the model. Both parameters have significant influence on classification results and number of generated granules (clusters).

Typical values that give a satisfactory results are $\sigma_l = 1$ and $\Gamma_{max} = 0.2$. However they should be tuned for each problem. The

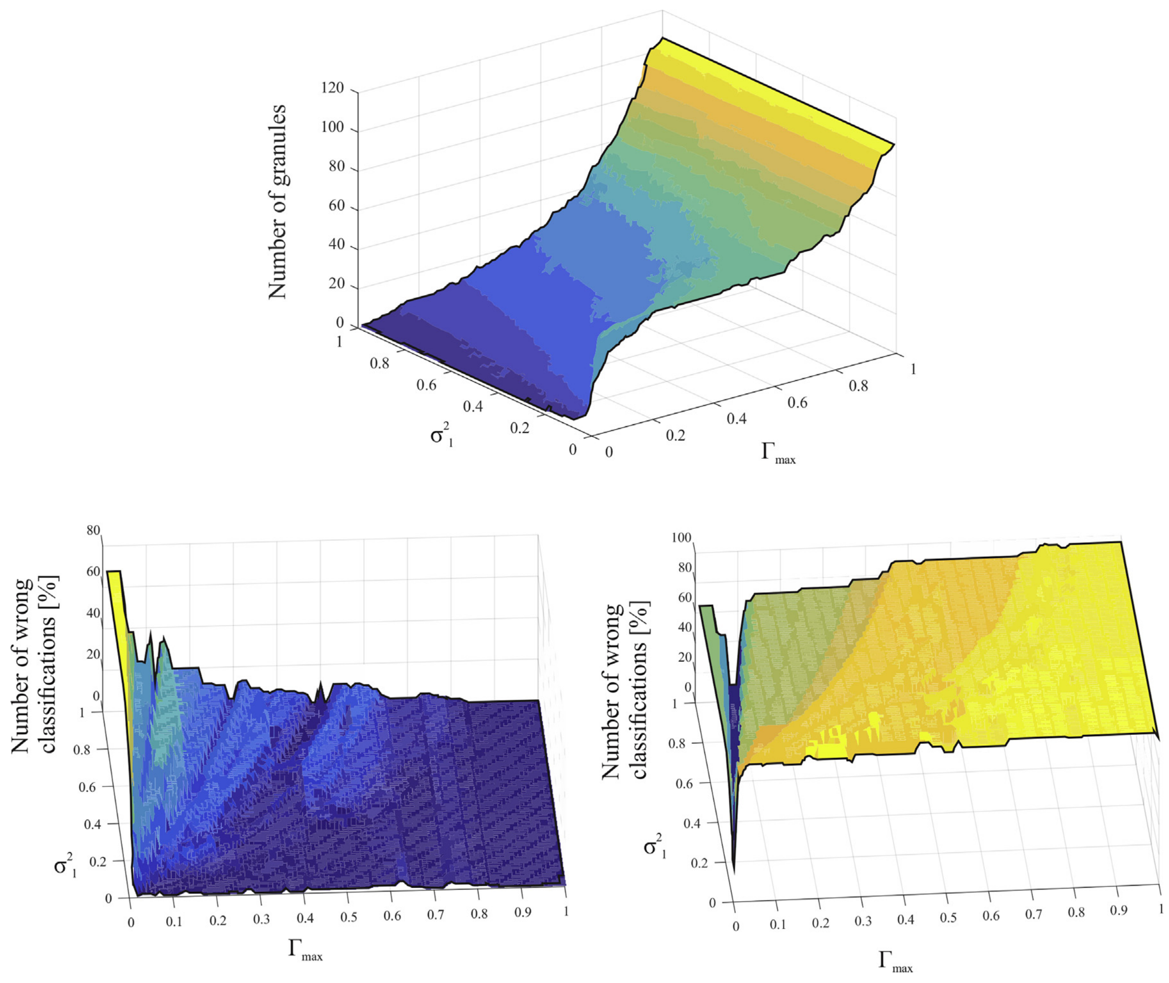


Fig. 7. Effect of parameters Γ_{max} and σ_l on the classifier for Iris dataset. Top graph – number of generated granules, bottom left graph – the percentage of wrongly classified samples in evolving mode, bottom right – percentage of wrongly classified samples on test set in “freeze” mode.

parameters are usually tuned in an off-line procedure on a smaller dataset. They can be tuned by using optimization procedure where the criterion is in the form of common goodness measure. For the classification case one of the typical measures could be used: recall (TPR), precision (PPV), false positive rate (FPR), false negative rate (FNR), false discovery rate (FDR). When tuning the parameters one must find the compromise among number of created granules (clusters), wrong classification rate on learning set and wrong classification rate on testing set. Although that the classifier is meant for constant on-line adaptation it must still retain a good performance in “freeze” mode. An example of the classifier performance for the Iris dataset is given in Fig. 7.

4.6. Threats to validity and comparison to other approaches

The presented method was tested on a KDD network intrusion data sets. It is argued in [61] that this data set is not the best set for testing the learning methods. Due to the properties of the data set the performance of the classifiers might be lower in real life applications than on these data sets. According to [61] there are several reasons for that. One is that the around 78 percent and 75 percent of the records are duplicated in the train and test set, respectively, making algorithms more biased towards more frequent records. However, this should not affect the evolving methods since they function in an on-line and must handle such cases. It was also reported [61] that usually classification methods achieve accuracy on the learning set of 98 percent and the testing set about 86 percent to 98 percent. It was pointed out that some researchers do not use the complete testing data set to estimate the performance of the classifier but randomly selected samples. In this way, it is possible to achieve better accuracies. Therefore they proposed a new version of the data set called NLS-KDD, which was successfully analyzed with the k-means clustering in [64]. Despite drawbacks of the KDD data set, pointed out in [61], there are papers that use this data set as a benchmark in intrusion detection learning. It should also be noted that the results given in this paper and results given in other literature cannot be directly compared due to several reasons. Usually, researchers use different portions of data for learning and testing. For example, the results presented in [50,65,66] were obtained based on validation of the methods on a test set of the KDD challenge, however different data sets are used for learning. In [67] 10 percent data set was used and in [52] the authors partitioned the data by themselves into training (1220 samples) and testing set (1219 samples). In [51] 10% data set was used for learning and testing. The majority of the method presented in the existing literature are batch methods, meaning that once the classifier is learned it does not adapt to new samples. This makes the comparison a bit biased towards evolving methods since they are adapting the model to new process conditions on-line. On the other hand batch methods usually produce more accurate models and classifiers than evolving methods in the standard experiment setup (learning the model on learning data set and then validating the model on test

Table 7
Comparison of methods detection rates.

Detection rate/Recall					
Method	Normal	U2R	DoS	R2L	Probe
PKID + Cons + C4.5[50]	97.08	25.00	96.08	8.12	73.62
EMD + INT + C4.5 [50]	96.36	19.30	94.07	32.87	79.48
EMD + CFS + C4.5(1) [50]	96.76	21.05	92.54	26.29	86.08
EMD + CFS + C4.5(2) [50]	96.56	32.89	93.35	5.32	78.18
PKID + Cons + NB(1)[50]	96.63	24.12	90.20	29.98	89.94
PKID + Cons + NB(2) [50]	96.13	11.40	95.12	13.85	96.67
KDD winner [50]	99.45	13.16	97.12	8.40	83.32
5FNs_p oly [50]	92.45	8.33	96.77	29.43	85.96
5FNs_f ourier [50]	92.72	10.97	96.86	23.75	85.74
5FNs_e xp [50]	92.75	13.60	96.85	23.77	85.60
SVM linear [50]	91.83	21.93	97.38	16.55	81.76
SVM2poly [50]	91.79	1.75	97.41	14.74	86.44
SVM3poly [50]	91.67	2.63	97.62	9.35	88.45
SVM RBF [50]	91.83	25.88	97.30	18.29	79.26
ANOVA ens. [50]	91.67	53.94	97.64	8.51	87.52
Pocket 2cl. [50]	91.80	29.82	97.40	14.77	85.84
Pocket mcl. [50]	91.86	54.38	97.65	11.45	86.79
Hi – SOM – 6f, 2l [67]	92.4	22.9	96.5	11.3	72.8
Hi – SOM – 6f, 3l [67]	95.4	10	95.1	9.9	64.3
Hi – SOM – 41f, 1l [67]	98.5	0	96.8	0.15	63.4
PNrule [65]	–	11.4	21.74	13.05	89.01
RIPPER [65]	–	22.06	11.84	8.33	81.16
C. 45 [65]	–	4.82	15.5	5.23	73.04
eChaucy_evol_test	**98.12**	**70**	**97.58**	**26.96**	**94.26**
eChaucy_evol_learn	**99.49**	**50**	**99.93**	**23.62**	**94.1**
eChaucy_evol_10%	**99.14**	**55.77**	**99.57**	**92.54**	**97.59**
eChaucy_10%	**99.32**	**31.43**	**96.72**	**1.58**	**79.96**
eChaucy_learn	**99.46**	**10**	**96.45**	**1.13**	**71.32**

data). This is due to two reasons: their iterative functioning means that more sweeps trough the data is made; in the case of labeled learning they use all the data labels for learning, whereas with the presented evolving methodology only labels that are needed for the creation of new clusters are used. Since the idea behind the presented approach is very different from the concept of standard off-line approaches found in the literature, the comparison of the results is not entirely fair. It is also difficult to find common measures of goodness for the classifier in the literature. Nevertheless, a Table 7 is given that gathers some of the results from other literature and results presented in this paper to give the reader some insights about the accuracies of other classifiers. The results from this paper are presented in bold text where *eChaucy_evol_test*, *eChaucy_evol_learn*, *eChaucy_evol_10%* denotes full evolving mode on KDD test, learning and 10% data set, respectively. The notation *eChaucy_learn* denotes results obtained when learning on learning KDD set and testing on test KDD set. Notation *eChaucy_10%* denotes results obtained when learning on 10% KDD set and testing on test KDD set. In both cases, the learning of the classifier was stopped during the testing phase.

It can be seen from Table 7 that the presented approach gives comparable results to other methods. The results presented in Table 7 vary from method to method. As mentioned, this is due to the use of different learning data sets and some of them also use prior variable/feature selection methods. The plus sign in the table entries from [50] denote that beside the learning algorithm also feature selection and discretization algorithm was used. Similar the values before letters *f* and *l* with the entries from [67] denote the number of features and number of neural network layers, respectively. Based on the results from [51,50,68] the use of variable selection and discretization methods can further improve the classifier accuracy. In our case, normalized raw data was used to test the method (without any variable selection and prior data transformation) therefore there is still much room for improvement of the results.

5. Conclusion

The usefulness of the evolving approaches for different tasks has been shown in the literature many times. They are usually implemented for regression problems. Such approaches can cope with big-data problems and can be effectively used with time-varying systems. In this paper, a novel idea of evolving algorithm is presented for network intrusion system. As shown in the paper the evolving methods, such as presented one, could be successfully used for detecting cyber-attacks. The obtained results are comparable to the results reported in the literature where classical (off-line generated models) classifiers are used. The presented approach retains a relatively good performance compared to classical approaches even when functioning in a "freeze mode". The advantage of the presented approach is that the classifier adapts in an on-line manner to new attacks (patterns) therefore there is no need for retraining the whole classifier after a certain time period. This feature could in practice reduce costs of maintaining the network intrusion system since retraining of models/classifiers is an expensive task. It was also shown that the presented approach uses only a fraction of all labels for training. This reduces the cost of labeling the data sets for learning the classifier. The results presented in this paper were obtained on a raw data with no additional filtering and discretization. According to the existing literature, using such techniques can further improve the intrusion detection results. The current drawback of the method is that it does not implement the cluster merging mechanism that could decrease the number of created clusters. At the current stage algorithm still, has two parameters that need to be tuned by the user. This could be in future avoided by applying automatic tuning on a batch of data. Further research will be focused on eliminating these drawbacks and further improving the classification results. Additional a cosine distance is being investigated as a distance measurement.

Acknowledgement

The author Igor Škrjanc would like to thank to Japan Society for the Promotion of Science (JSPS) which has awarded him a fellowship under the JSPS Invitation Fellowship for Research in Japan with the identification number L 16533, which enables the collaborative research at Kobe University as the host institution.

References

[1] P.P. Angelov, D.P. Filev, An approach to on-line identification of Takagi–Sugeno fuzzy models, IEEE Trans. Syst. Man Cybern. – Part B 34 (1) (2004) 484–497.
[2] P. Werbos, Beyond Regression: New TOOLS for Prediction and Analysis in the behavioral Sciences (Ph.D. dissertation), Harvard University, Cambridge, 1974.
[3] A. Valdes, K. Skinner, Adaptive, model-based monitoring for cyber attack detection, Proceedings of Recent Advances in Intrusion Detection: Third International Workshop, RAID 2000 (2000) 80–93.
[4] M. Asif, P. Angelov, H. Ahmed, An approach to real-time color-based object tracking, Proceedings of 2006 International Symposium on Evolving Fuzzy Systems (2006) 86–91, http://dx.doi.org/10.1109/ISEFS.2006.251169.
[5] P. Angelov, X. Zhou, Evolving fuzzy systems from data streams in real-time, Proceedings of 2006 International Symposium on Evolving Fuzzy Systems (2006) 29–35, http://dx.doi.org/10.1109/ISEFS.2006.251157.
[6] P. Angelov, D. Filev, *Simpl_e TS*: a simplified method for learning evolving Takagi–Sugeno fuzzy models, in: Proceedings of The 2005 IEEE International Conference on Fuzzy Systems, Reno, Nevada, USA, 2005, pp. 1068–1073.
[7] P. Angelov, Evolving Takagi–Sugeno fuzzy systems from streaming data (eTS+), in: Evolving Intelligent Systems: Methodology and Applications, Wiley, New Jersey, 2010, pp. 21–50.
[8] E. Lughofer, E.P. Klement, FLEXFIS: a variant for incremental learning of Takagi–Sugeno fuzzy systems, in: Proceedings of The 2005 IEEE International Conference on Fuzzy Systems, Reno, Nevada, USA, 2005, pp. 915–920.
[9] E. Lughofer, J.-L. Bouchot, A. Shaker, On-line elimination of local redundancies in evolving fuzzy systems, Evolv. Syst. 2 (3) (2011) 165–187.
[10] J.-S.R. Jang, C.-T. Sun, Functional equivalence between radial basis function networks and fuzzy inference systems, IEEE Trans. Neural Netw. 4 (1993) 1139–1150.

[11] N.K. Kasabov, Q. Song, DENFIS: Dynamic Evolving Neural-Fuzzy Inference System and its application for time-series prediction, IEEE Trans. Fuzzy Syst. 10 (2) (2002) 144–154.
[12] H.J. Rong, N. Sundararajan, G.B. Huang, P. Saratchandran, Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction, Fuzzy Sets Syst. 157 (9) (2006) 1260–1275.
[13] S.G. Tzafestas, K.C. Zikidis, NeuroFAST: On-line neuro-fuzzy ART-based structure and parameter learning TSK model, IEEE Trans. Syst. Man Cybern. – Part B 31 (5) (2001) 797–802.
[14] H. Soleimani-B, C. Lucas, B.N. Araabi, Recursive Gath-Geva clustering as a basis for evolving neuro-fuzzy modeling, Evolv. Syst. 1 (1) (2010) 59–71.
[15] S. Blažič, D. Dovžan, I. Škrjanc, Cloud-based identification of an evolving system with supervisory mechanisms, Proceedings of IEEE Control Systems Society Multiconference on Systems and Control (2014) 1906–1911.
[16] S. Blažič, I. Škrjanc, Problems of identification of cloud-based fuzzy evolving systems, in: L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. Zadeh, J. Zurada (Eds.), Artificial Intelligence and Soft Computing. ICAISC 2016, Lecture Notes in Computer Science, vol. 9692, Springer, Cham, 2016, pp. 173–182.
[17] J. Shing, R. Jang, ANFIS: Adaptive-Network-Based Fuzzy Inference System, IEEE Trans. Syst. Man Cybern. 23 (3) (1993) 665–685.
[18] D. Dovžan, I. Škrjanc, Recursive fuzzy c-means clustering for recursive fuzzy identification of time-varying processes, ISA Trans. 50 (2) (2011) 159–169.
[19] D. Dovžan, I. Škrjanc, Recursive clustering based on a Gustafson–Kessel algorithm, Evolv. Syst. 2 (2011) 15–24.
[20] L. Maciel, F. Gomide, R. Ballini, Recursive possibilistic fuzzy modeling, 2014 IEEE Symposium on Evolving and Autonomous Learning Systems (EALS) (2014) 9–16.
[21] F. Bocklisch, S.F. Bocklisch, M. Beggiato, J.F. Krems, Adaptive fuzzy pattern classification for the online detection of driver lane change intention, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.02.089.
[22] M. Pratama, S.G. Anavatti, P. Angelov, E. Lughofer, A novel incremental learning machines, IEEE Trans. Neural Netw. Learn. Syst. 25 (1) (2014) 55–68.
[23] D. Leite, F. Gomide, Incremental granular fuzzy modeling using imprecise data streams, in: Fifty Years of Fuzzy Logic and its Applications, Springer International Publishing, 2015, pp. 107–124.
[24] I. Škrjanc, D. Dovžan, F. Gomide, Evolving fuzzy-model-based on c-regression clustering, in: Proceedings of 2014 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS), IEEE, 2014, pp. 1–7.
[25] A. Lemos, W. Caminhas, F. Gomide, Fuzzy evolving linear regression trees, Evolv. Syst. 2 (1) (2011) 1–14.
[26] G. Leng, G. Prasad, T.M. McGinnity, An on-line algorithm for creating self-organizing fuzzy neural networks, Neural Netw. 17 (2004) 1477–1493.
[27] E. Lughofer, Flexible Evolving Fuzzy Inference Systems from Data Streams (FLEXFIS++), in: Learning in Non-Stationary Environments: Methods and Applications, Springer, 2012, pp. 205–245.
[28] A. Kalhor, B. Araabi, C. Lucas, An online predictor model as adaptive habitually linear and transiently nonlinear model, Evolv. Syst. 1 (1) (2010) 29–41.
[29] J. Rubio, SOFMLS: online self-organizing fuzzy modified least-squares network, IEEE Trans. Fuzzy Syst. 17 (6) (2009) 1296–1309.
[30] L. Maciel, A. Lemos, R. Ballini, F. Gomide, Adaptive fuzzy c-regression modeling for time series forecasting, in: 2015 Conference of the International Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology (IFSA-EUSFLAT-15), Atlantis Press, 2015, pp. 917–924.
[31] E. Lima, M. Hell, R. Ballini, F. Gomide, Evolving fuzzy modeling using participatory learning, in: Evolving Intelligent Systems: Methodology and Applications, Wiley, New Jersey, 2010, pp. 67–86.
[32] O.D.R. Filho, G.L. de Oliveira Serra, Recursive fuzzy instrumental variable based evolving neuro-fuzzy identification for non-stationary dynamic system in a noisy environment, Fuzzy Sets Syst. (2017), http://dx.doi.org/10.1016/j.fss.2017.05.016.
[33] D. Filev, O. Georgieva, An extended version of the Gustafson–Kessel algorithm for evolving data stream clustering, in: Evolving Intelligent Systems: Methodology and Applications, Wiley, New Jersey, 2010, pp. 273–300.
[34] L. Maciel, F. Gomide, R. Ballini, Recursive possibilistic fuzzy modeling, in: Proceedings of 2014 IEEE Symposium on Evolving and Autonomous Learning Systems (EALS), IEEE SSCI, 2014, pp. 9–16.
[35] P. Angelov, E. Lughofer, X. Zhou, Evolving fuzzy classifiers using different model architectures, Fuzzy Sets Syst. 159 (23) (2008) 3160–3182.
[36] D. Dovžan, V. Logar, I. Škrjanc, Implementation of an evolving fuzzy model (eFuMo) in a monitoring system for a waste-water treatment process, IEEE Trans. Fuzzy Syst. 23 (5) (2015) 1761–1776.
[37] E. Lughofer, Evolving fuzzy systems—methodologies, advanced concepts and applications Studies in Fuzziness and Soft Computing Series, vol. 266, 1st ed., Springer-Verlag, Berlin Heidelberg, 2011, pp. 476.
[38] R. Krishnapura, J.M. Keller, Possibilistic approach to clustering, IEEE Trans. Fuzzy Syst. 2 (1) (1993) 98–100.
[39] N.R. Pal, K. Pal, J.M. Keller, J.C. Bezdek, A possibilistic fuzzy c-means clustering algorithm, IEEE Trans. Fuzzy Syst. 13 (4) (2005) 517–530.
[40] H. Timm, C. Borgelt, C. Doering, R. Kruse, An extension to possibilistic fuzzy cluster analysis, Fuzzy Sets Syst. 147 (1) (2004) 3–16.
[41] B. Ojeda-Magana, R. Ruelas, M.A. Corona-Nakamura, D. Andina, An improvement to the possibilistic fuzzy c-means clustering algorithm, Intell. Autom. Soft Comput. 20 (1) (2006) 585–592.
[42] P. Angelov, R. Yager, A simple fuzzy rule-based system through vector membership and kernel-based granulation, Proceedings of 5th IEEE International Conference on Intelligent Systems (IS) (2010) 349–354.
[43] P. Angelov, R. Yager, Simplified fuzzy rule-based systems using non-parametric antecedents and relative data density, 2011 IEEE Workshop on Evolving and Adaptive Intelligent Systems (EAIS) (2011) 62–69.
[44] G. Klančar, I. Škrjanc, Evolving principal component clustering with a low run-time complexity for LRF data mapping, Appl. Soft Comput. 35 (2015) 349–358.
[45] S.J. Stolfo, W. Fan, W. Lee, A. Prodromidis, P.K. Chan, Cost-based modeling and evaluation for data mining with application to fraud and intrusion detection: Results from the JAM project, Proceedings of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX 2000) (2000) 1–15.
[46] J. Zhang, M. Zulkernine, A. Haque, Random-forests-based network intrusion detection systems, IEEE Trans. Syst. Man Cybern. – Part C: Appl. Rev. 38 (5) (2008) 649–659.
[47] V. Jyothsna, V.V.R. Prasad, K.M. Prasad, A review of anomaly based intrusion detection systems, Int. J. Comput. Appl. 28 (7) (2011) 26–35.
[48] M. Shyu, S. Chen, K. Sarinnapakorn, L. Chang, A novel anomaly detection scheme based on principal component classifier, Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop, in Conjunction with the Third IEEE International Conference on Data Mining (ICDM03) (2003) 172–179.
[49] V. Bolon-Canedo, N. Sanchez-Marono, A. Alonso-Betanzos, A combination of discretization and filter methods for improving classification performance in KDD Cup 99 dataset, Proceedings of International Joint Conference on Neural Networks (2009) 359–366.
[50] V. Bolon-Canedo, N. Sanchez-Marono, A. Alonso-Betanzos, Feature selection and classification in multiple class dataset: an application to KDD Cup 99 dataset, Expert Syst. Appl. 38 (2011) 5947–5957.
[51] S. Parsazad, E. Saboori, A. Allahyar, Fast feature reduction in intrusion detection datasets, Proceedings of MIPRO 2012 (2012) 1023–1029.
[52] S. Devaraju, S. Ramakrishnan, Performance comparison for intrusion detection system using neural network with KDD dataset, ICTACT J. Soft Comput. 4 (3) (2014) 743–752.
[53] A.M. Chandrashekhar, K. Raghuveer, Performance evaluation of data clustering techniques using KDD Cup-99 intrusion detection data set, Int. J. Inf. Netw. Secur. 1 (4) (2012) 294–305.
[54] R. Zhang, S. Zhang, Y. Lan, J. Jiang, Network anomaly detection using one class support vector machine, Proceedings of the International Multi Conference of Engineers and Computer Scientists 2008, vol. I (2008) 1–16.
[55] G. Liu, Z. Yi, S. Yang, A hierarchical intrusion detection model based on the PCA neural networks, Neurocomputing 70 (7–9) (2007) 1561–1568.
[56] R.A.R. Ashfaq, X.-Z. Wang, J.Z. Huang, H. Abbas, Y.-L. He, Fuzziness based semi-supervised learning approach for intrusion detection system, Inf. Sci. 378 (2017) 484–497, http://dx.doi.org/10.1016/j.ins.2016.04.019.
[57] P. Mishra, E.S. Pilli, V. Varadharajan, U. Tupakula, Intrusion detection techniques in cloud environment: a survey, J. Netw. Comput. Appl. 77 (2017) 18–47, http://dx.doi.org/10.1016/j.jnca.2016.10.015.
[58] A.L. Buczak, E. Guven, A survey of data mining and machine learning methods for cyber security intrusion detection, IEEE Commun. Surv. Tutor. 18 (2) (2016) 1153–1176, http://dx.doi.org/10.1109/COMST.2015.2494502.
[59] P.G. Jeya, M. Ravichandran, C.S. Ravichandran, Efficient classifier for R2L and U2R attacks, Int. J. Comput. Appl. 45 (21) (2012) 28–32.
[60] S. Chebrolu, A. Abraham, J.P. Thomas, Feature deduction and ensemble design of intrusion detection systems, Comput. Secur. 24 (4) (2005) 295–307.
[61] M. Tavallaee, E. Bagheri, W. Lu, A.-A. Ghorbani, A detailed analysis of the KDD Cup 99 data set, Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009 (2009) 53–58.
[62] S. Blažič, P. Angelov, I. Škrjanc, Comparison of approaches for identification of all-data cloud-based evolving systems, IFAC-PapersOnLine 48 (10) (2015) 129–134, 2nd IFAC Conference on Embedded Systems, Computer Intelligence and Telematics CESCIT 2015.
[63] D. Škrjanc, Dovžan, Evolving Gustafson–Kessel possibilistic c-means clustering, Procedia Comput. Sci. 94 (2015) 191–198.
[64] V. Kumar, H. Chauhan, D. Panwar, K-means clustering approach to analyze NSL-KDD intrusion detection dataset, Int. J. Soft Comput. Eng. 3 (4) (2013) 1–4.
[65] R. Agarwal, M.V. Joshi, PNrule: A New Framework for Learning Classifier Models in Data Mining (a Case-Study in Network Intrusion Detection), Technical report RC-21719, IBM Research Division, 2000.
[66] A. Jahanbani, H. Karimi, A new approach for detecting intrusions based on PCA neural networks, J. Basic Appl. Sci. Res. 2 (1) (2012) 672–679.
[67] H.G. Kayacik, A.N. Zincir-Heywood, M.I. Heywood, A hierarchical SOM-based intrusion detection system, Eng. Appl. Artif. Intell. 20 (4) (2007) 439–451.
[68] Y. Zhu, J. Liang, J. Chen, Z. Ming, An improved NSGA-III algorithm for feature selection used in intrusion detection, Knowl.-Based Syst. 116 (2017) 74–85, http://dx.doi.org/10.1016/j.knosys.2016.10.030.