A data reduction strategy and its application on scan and backscatter detection using rule-based classifiers

Herrera-Semenets, Vitali; Pérez-García, Osvaldo Andrés; Hernández-León, Raudel; van den Berg, Jan; Doerr, Christian

DOI
10.1016/j.eswa.2017.11.041

Publication date
2018

Document Version
Final published version

Published in
Expert Systems with Applications

Citation (APA)
Herrera-Semenets, V., Pérez-García, O. A., Hernández-León, R., van den Berg, J., & Doerr, C. (2018). A data reduction strategy and its application on scan and backscatter detection using rule-based classifiers. *Expert Systems with Applications*, *95*, 272-279. https://doi.org/10.1016/j.eswa.2017.11.041

Expert Systems With Applications 95 (2018) 272–279

Contents lists available at ScienceDirect

Expert Systems With Applications

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

A data reduction strategy and its application on scan and backscatter detection using rule-based classifiers

Vitali Herrera-Semenets[a,*], Osvaldo Andrés Pérez-García[a], Raudel Hernández-León[a], Jan van den Berg[b], Christian Doerr[b]

[a] Advanced Technologies Application Center (CENATAV), 7a ♯ 21406, Playa, C.P., Havana, 12200, Cuba
[b] Intelligent Systems Department, Delft University of Technology, Mekelweg 4, CD Delft 2628, The Netherlands

ARTICLE INFO

Article history:
Received 1 August 2017
Revised 20 October 2017
Accepted 16 November 2017
Available online 21 November 2017

Keywords:
Data mining
Data reduction
Intrusion detection
Scan
Backscatter

ABSTRACT

In the last few years, the telecommunications scenario has experienced an increase in the volume of information generated, as well as in the execution of malicious activities. In order to complement Intrusion Detection Systems (IDSs), data mining techniques have begun to play a fundamental role in data analysis. On the other hand, the presence of useless information and the amount of data generated by telecommunication services (leading to a huge dimensional problem), can affect the performance of traditional IDSs. In this sense, a data preprocessing strategy is necessary to reduce data, but reducing data without affecting the accuracy of IDSs represents a challenge. In this paper, we propose a new data preprocessing strategy which reduces the number of features and instances in the training collection without greatly affecting the achieved accuracy of IDSs. Finally, our proposal is evaluated using four different rule-based classifiers, which are tested on real scan and backscatter data collected by a network telescope.

1. Introduction

Cyber scanning (scan) has been used for a long time as a method of discovering communication channels that can be exploited. The main objective is to scan as many listening ports as possible and store information from those ones that are receptive or useful for a specific purpose. In practice, several packets are sent for different protocols and it is inferred which services are "listening" by the responses received or not received. This kind of network reconnaissance represents a growing cyber security concern, because it is the first stage of an intrusion attempt (Bou-Harb, Debbabi, & Assi, 2014), which allows an attacker to locate a target and subsequently exploit its vulnerable system.

In order to remain hidden, the attackers falsify the source address and source port of the packets that make up the attack. Servers that receive such packets after processing the request send a response to the false address. This activity is known as backscatter. Backscatter traffic can be easily mistaken for a scan, because there can be enough falsified addresses that fall within the network space, such that replies from the victim will make it seem like a scanner, *i.e.* the victim sends unsolicited packets to hosts that may not even exist (Moore, Shannon, Brown, Voelker, & Savage, 2006).

Nowadays, the volume of data generated from using telecommunication services is considerably large, causing Big Data challenges in the network traffic (Zuech, Khoshgoftaar, & Wald, 2015). For example, a recent research shows that in the second quarter of the year 2016, a record-setting 21 attacks measured more than 30 million packets per second (Mpps).

Also, each packet can have tens of features, which can lead to a high dimensionality problem caused by too many unnecessary features for the intrusion detection task. Additionally, events representing attacks, executed by malicious users known as intruders, cause millions in losses and damage the prestige of the affected companies (Yar, 2013). In order to analyze such volume of data and quickly detect which events are associated to a specific attack, it is necessary to use an intrusion detection system (IDS) (Liao, Lin, Lin, & Tung, 2013). An IDS can be a software or device used to monitor the system or activities in a telecommunication network in order to detect policy violations or malicious activities, and consequently generates reports to the management system. However, due to the high amount of data required for analysis, the performance of such systems can be affected, and in some cases they may exhaust the available random access memory (RAM) and stop working.

* Corresponding author.
E-mail addresses: vherrera@cenatav.co.cu (V. Herrera-Semenets), osvaldo.perez@cenatav.co.cu (O. Andrés Pérez-García), rhernandez@cenatav.co.cu (R. Hernández-León), J.vandenBerg@tudelft.nl (J. van den Berg), c.doerr@tudelft.nl (C. Doerr).

https://doi.org/10.1016/j.eswa.2017.11.041
0957-4174/

In literature, there are two main approaches based on data mining for the intrusion detection problem: (1) the supervised approach (signature-based IDS), which requires a supervised training set T to create a classification model (Kotsiantis, 2007), and (2) the unsupervised approach (anomaly-based IDS), where there is no previous knowledge (Ghahramani, 2004). Our proposal is focused on improving the performance of the supervised approach, where the signature-based IDSs are one of the most used.

From the standpoint of classification, the main objective of building a signature-based IDS is to train a rule-based classifier that can categorize data as normal or attacks (Herrera-Semenets, Pérez-García, Gago-Alonso, & Hernández-León, 2017). In intrusion detection scenarios, the training set reduction could be useful to minimize the consumption of computer resources, such as RAM, allowing to apply computationally expensive algorithms.

Obviously, data reduction does result in some loss of data during the training phase (Aggarwal, 2015). This fact results in some loss of useful information, and therefore the classifier accuracy can be affected during the classification phase. On the other hand, the runtime of data reduction strategies is usually high when they use large datasets.

In this paper, we propose a novel data reduction strategy (DRS) for improving the supervised classifiers performance. DRS combines features reduction with instances reduction to obtain a reduced training set $S \subset T$, providing a high efficiency in the training phase without greatly affecting the accuracy of classifiers.

Additionally, we evaluate the performance of a set of rule-based classifiers (DTNB, PART, OneR and NNge) using S as the training set. Many works evaluate their proposals by applying cross-validation over a specific training dataset, or by using a testing dataset. If we consider the characteristics of the intrusion detection scenarios, where data describing malicious activities can be modified in order to make the detection system fail; it would be very interesting to see the performance of the classification models in a real scenario, during a certain period of time.

In our work, we try to simulate a real scenario. To do this, we take a period of 24 h of real network data associated to scan and backscatter traffic. The classification models obtained after the training stage are evaluated at every hour. This gives us a more precise idea of how these models behave in a real scenario. In addition, we show how our proposal increases the speed of the classifiers during the detection process, which is essential for real-time detection.

This paper is organized as follows. Related work about data reduction strategies is described in Section 2. The proposed strategy is introduced in Section 3. In Section 4, the experimental results using different classifiers are discussed. Also, a study case is presented, where the proposed strategy is used as a preprocessing step to improve the classifiers performance for scan and backscatter detection. Finally, our conclusions and future work are outlined in Section 5.

2. Related work

Data reduction may be in terms of the number of rows (instances) or in terms of the number of columns (features) (Aggarwal, 2015). In this sense, three main approaches have been proposed: (1) feature selection (Cheng, Cai, Zhang, Xu, & Su, 2015; Ganapathi & Duraivelu, 2015; Xia, Fang, & Zhang, 2014), (2) instance selection (García, Luengo, & Herrera, 2015; de Oliveira Moura, de Freitas, Cardoso, & Cavalcanti, 2014; Silva, Souza, & Motta, 2016) and (3) hybrid, where feature selection and instance selection are combined (Chen, Zhang, Jin, & Kim, 2014).

The feature selection algorithms look up the most relevant features of the dataset. In this way, only a subset of features from the underlying data is used in the analytical process. It facilitates the understanding of the extracted patterns and increases the performance of the training phase.

Most of the proposals that reduce data in these scenarios use only one feature selection technique. In our opinion, these proposals do not take advantage of the possibilities that the combination of such techniques can offer, since different metrics could measure different information in the features. Therefore, each metric could select different features as a final result, with the same level of data representativeness. In this sense, our hypothesis is that a better management of these techniques could lead to a better selection of the final set of features.

On the other hand, the presence of redundant or repetitive information (represented as instances) in training dataset is common in these scenarios. In our case, the goal is not to archive the information, in which case, the duplicate instances would have a positive use. The duplication to which we refer is due to errors that occur as a result of poor data quality, often because the services provided are not integrated. The feature selection methods do not allow addressing this problem. For this, it is necessary to use instance selection algorithms.

The instance selection (IS) algorithms obtain a reduced subset S from the original training set T, so that S does not contain superfluous instances. IS methods can either start with $S = \emptyset$ (incremental method) or $S = T$ (decremental method) (Olvera-López, Carrasco-Ochoa, Martínez-Trinidad, & Kittler, 2010). Incremental methods obtain S by selecting instances from T (Chou, Kuo, & Chang, 2006; Raicharoen & Lursinsap, 2005), while decremental ones obtain S by deleting instances from T (Wilson & Martinez, 2000).

According to the strategy used for selecting instances, the algorithms can be divided into two groups: Wrapper and Filter methods (Olvera-López, Carrasco-Ochoa, Martínez-Trinidad, et al., 2010). In Wrapper methods, the classifier is used in the selection process, the instances which do not affect the classification accuracy are removed from T. On the other hand, Filter methods are independent from the classifiers and the selection criterion is based on different heuristics.

Some Filter methods use the clustering approach for instance selection (Lumini & Nanni, 2006; Olvera-López, Carrasco-Ochoa, & Martínez-Trinidad, 2010). Clustering is performed by transforming T into clusters, then the selected instances are defined as centers of clusters. Filter methods end up being more efficient than Wrapper methods (Olvera-López, Carrasco-Ochoa, Martínez-Trinidad, et al., 2010). Moreover, since their selection criteria is not based on the used classifier, the achieved accuracy with S using different classifiers can be acceptable.

In these scenarios, we can not say that instance selection is more advantageous than feature selection or vice versa. If only feature selection is applied, unnecessary information may remain in the instances, and if only instance selection is applied, unrepresentative features of the dataset may remain. A solution to this problem has been the proposal of hybrid strategies, which combine both techniques mentioned above.

In Chen et al. (2014) a hybrid method for intrusion detection is proposed. In this case, the feature selection process is performed by the OneR (Holte, 1993) algorithm. Then, a clustering method of Affinity Propagation (Frey & Dueck, 2007) (AP) is used for the instance selection process. The problem here is that the AP clustering runs out of memory for 6000 training data instances. Therefore, in order to improve the performance and scalability of the method, they have implemented a distributed solution for AP clustering using MapReduce. For a proper performance, such method requires to use MapReduce with 8 nodes; also each node is equipped with a quad core at 2.5 GHz CPU with 4 GB RAM. If we take into account the resources required to process 12872 instances (few data compared to a real scenario), this makes its proposal somewhat expensive in terms of computational resources. In this sense, they use a

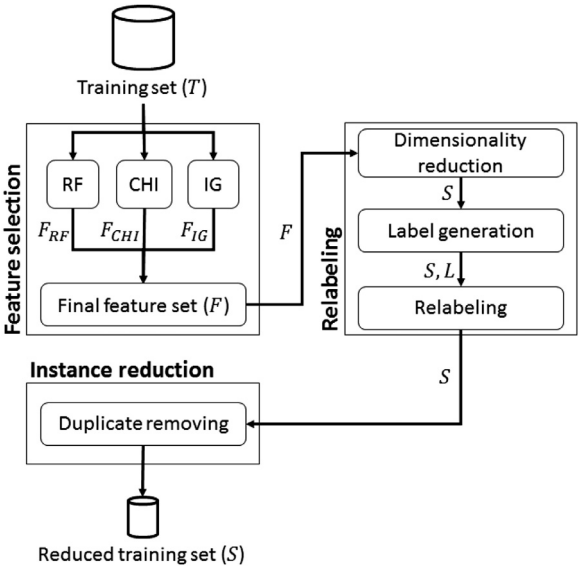

Fig. 1. Scheme of data reduction strategy.

network traffic dataset of 12872 instances for training and another 115848 instances for detection, which represents a small dataset for this scenario; also, it is not possible to observe its scalability with larger datasets, more characteristic of intrusion detection scenarios.

However, the main problem of the instance selection methods is the high runtime when large datasets are processed, making their application unfeasible in some cases, and directly affecting the hybrid approaches. In this sense, we propose a new hybrid approach including a feature selection step, where different metrics are combined to select the final feature set, and a decremental Filter method for instance selection, which incorporates a fast step to reduce the number of instances based on a prior relabeling process. This reduces the training collection without greatly affecting the accuracy of the classifier, with low runtime.

3. Data reduction strategy

In this section, we introduce our data reduction strategy (DRS). As shown in Fig. 1, DRS comprises three main processes: feature selection, relabeling, and instance reduction.

First, the feature selection (FS) process is performed in order to reduce the dimensionality of the training set. Following our hypothesis mentioned above, it was taken into account that the selected methods will use different measures to estimate the correlation between an attribute and a class, indicating the relevance of a feature to the class. After a study, we determined that the most commonly used measures can be grouped into three categories: entropy based (Information Gain, Gain Ratio, and Symmetric Uncertainty), statistical based (Chi-square), and instance based (Relief and ReliefF) (Liu et al., 2016).

In this sense, we use three different algorithms (one representative from each category): ReliefF (RF), Chi-squared (CHI) Ranking Filter and Information Gain (IG) Ranking Filter. Additionally, CHI and IG algorithms have been reported in some comparative studies as the most effective methods of feature selection for classification (Forman, 2003). Thus, our strategy selects features based on three different measures. This ensures that a single measure is not sufficient to rule out a feature.

A FS algorithm A obtain a set of scores P_A, by assigning a score $p_f \in P_A$ to each feature f according to its relevance. In this sense, when A gets P_A, its score mean $\overline{p}_A$ is computed. Later, if a feature f satisfies $p_f > \overline{p}_A$, then f is assigned to a feature set F_A. It is important to note that the three algorithms are executed in parallel, which allows to gain more efficiency. Next, the features set $F = F_{RF} \cup F_{CHI} \cup F_{IG}$ is selected as the most representative set.

After selecting the final features set F, the relabeling process is carried out to generate new labels for the selected features values. For dimensionality reduction, the columns representing all features $f \notin F$ are removed from T, and consequently a reduced set S is obtained.

The high volume of data generated in these scenarios require efficient processing strategies. During the relabeling process, continuous features are discretized. This allows us to map data from a wide range of numerical values to a very small subset of discrete values. There are two approaches widely used to perform discretization tasks: the supervised and unsupervised one.

Supervised discretization methods make intensive use of the class to partition continuous features, while unsupervised discretization methods are class independent. Failure to take into account class information makes unsupervised discretization methods significantly faster than supervised ones (Joiţa, 2010). Note, that in malicious activities detection scenarios there may be different types of classes, just to give an example, in the KDD99 dataset (Olusola, Oladele, & Abosede, 2010) there are 23 different types of classes (attacks).

There are several comparative studies where the k-means algorithm is used as an unsupervised discretization method (Dash, Paramguru, & Dash, 2011; Maslove, Podchiyska, & Lowe, 2012; Vannucci & Colla, 2004). In these studies it is concluded that the process of discretization using k-means produces results more consistent and favorable than other unsupervised methods. In addition, k-means is an algorithm that uses a minimal quadratic error partitioning to generate an arbitrary number of partitions reflecting the original distribution of the feature, which makes the results similar to those achieved by the supervised discretization methods. In order to gain efficiency, we use the k-means algorithm to generate the labels during the label generation step.

The purpose of applying a clustering algorithm as part of the relabeling process is to search for similar values and group them into clusters, such that the distance between values within a cluster is as small as possible, and the distance between clusters is as large as possible. In this sense, given an integer k, the algorithm is implemented in four steps:

1. Split the objects in k nonempty subsets.
2. Compute the seed as the centroid (middle point) of the cluster.
3. Assign each object to the closest cluster.
4. When it is not possible to do more assignments, go to step 2.

For each selected numerical feature f_i, the k-means algorithm is executed over the set of values taken by f_i in S, denoted by V_i. Each of the obtained clusters contains a range of numerical values, which are represented by a unique numerical label. The use of these clusters allow us to cover feature values that do not exist in S and are included in the classification stage.

For example, suppose that in the training phase a feature f_1 takes values in the set $V_1 = \{0, 0.2, 0.3, 0.6, 0.7, 1.0\}$, and during the relabeling process, clusters $c_1 = [0, 0.2, 0.3, 0.6]$ and $c_2 = [0.7, 1.0]$ are obtained. Then, in the classification stage, it is required to classify a new transaction, in which the feature f_1 takes the value $0.9 \notin V_1$, but 0.9 is in the range of c_2, therefore it can be classified.

After label generation, the relabeling step is performed. Here, the numeric features values are replaced by their corresponding labels, and a unique numerical label is assigned for each non-numerical feature value.

Finally, the instance reduction process is performed over the relabeled training collection. Here, duplicate removing step, as its name suggests, removes duplicate instances from S. Notice that an

instance is a duplicate instance if there is at least another instance with the same feature values and class. The result is a reduced training collection. This collection is used by a classifier to build a classification model.

4. Experimentation

In this section, the classifiers used to evaluate our proposal are introduced. Next, DRS performance is evaluated and compared regarding another hybrid strategy using a benchmark dataset. Finally, a study case is presented, showing the application of DRS in real scenarios and its advantages. Note that DRS was performed in a single PC equipped with an Intel Quad Core at 3.5 GHz CPU with 12 GB of RAM.

4.1. Classifiers

In our experiments we used four different classifiers that have been applied in intrusion detection tasks (Azad & Jha, 2014; MeeraGandhi, Appavoo, & Srivasta, 2010; Panda & Patra, 2009). We run such classifiers using the Weka[1] framework. In this section, we briefly introduce each classifiers.

Non-Nested generalized exemplars (Sylvain, 2002) (NNge) generalizes instances without nesting or overlap. The generalization is performed by merging instances, forming hyperrectangles in feature space that represent conjunctive rules with internal disjunction. NNge forms a generalization each time a new instance is added to the dataset, by joining it to its nearest neighbour of the same class.

Decision Table/Naive Bayes (Hall & Frank, 2008) (DTNB) splits the feature set into two groups: one group assigns class probabilities based on Naive Bayes (NB), and the other group based on a Decision Table (DT). Then, the class probability estimates of the DT and NB must be combined to generate overall class probability estimates.

OneR (Holte, 1993) is a basic rule-based classifier. It generates a one-level decision tree represented as a rule set, where all test one attribute. OneR is a simple, cheap method that often comes up with quite good rules.

PART (Frank & Witten, 1998) combines the divide-and-conquer strategy with separate-and-conquer strategy of rule learning. It builds a partial decision tree in each iteration, and the leaf with the largest coverage is made into a rule.

4.2. Experimental results

In this section, we evaluate and compare the performance of DRS against the hybrid method proposed by Chen et al. (2014). The experiments were conducted using KDD'99 dataset. [2] This dataset has been widely used for testing network intrusion detection approaches, and it is considered a benchmark dataset (Özgür & Erdem, 2016). The training set T consists of 494021 instances, while the testing set contains 311029 instances. Each instance contains 41 features out of which 9 are discrete and 32 are continuous. Also, each instance is labeled as either normal or an attack, with exactly one specific attack type. In our experiments, we classify all the instances into two categories, “normal” or “attack”.

In Chen et al. (2014), the KDD'99 dataset was evaluated using a specific configuration, which was the same used in our experiments. But, taking into account that in Chen et al. (2014) it is necessary to predefine the number of features to be selected, in order to make a fair comparison, we decided that it will be the same as the amount of features selected by our proposal.

[1] http://cs.waikato.ac.nz/ml/weka/.

[2] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

Table 1
Comparative results using KDD99 dataset.

Classifier	TP	FP	Acc.	Strategy
OneR	0889	0016	9073	baseline
	0864	0018	8884	DRS
	0858	0020	8816	(Chen et al., 2014)
DTNB	0911	0005	9274	baseline
	0892	0006	9097	DRS
	0888	0006	9091	(Chen et al., 2014)
NNge	0.917	0018	9298	baseline
	0901	0019	9104	DRS
	0895	0020	9082	(Chen et al., 2014)
PART	0.915	0006	9307	baseline
	0901	0008	9192	DRS
	0897	0009	9155	(Chen et al., 2014)

After applying DRS on T, a reduced dataset S was obtained with 60540 instances and 17 features. This represents approximately a 12 % of the number of instances in T, and a 41 % of the number of features in T. On the other hand, after applying (Chen et al., 2014) on T, a reduced dataset was obtained with 63234 instances and 17 features. Note that DRS gets a smaller set, specifically 2694 instances less than the (Chen et al., 2014) proposal.

In Table 1, we show the true positives (TP) and false positives (FP) rates, as well as accuracy achieved by each classifier for DRS and (Chen et al., 2014) strategies. Also, we include the results using the original training set without any reduction (baseline).

Note that despite the considerable reduction of the dataset, the TP, FP and accuracy achieved using DRS do not vary greatly regarding to those obtained using the baseline. Although DRS further reduces the training dataset than (Chen et al., 2014) proposal, the classifiers reaches better results with DRS.

Fig. 2 shows the time taken to build the classification model by each classifier. The time difference between using baseline and a reduction strategy is significant. On the other hand, the time difference between using the dataset reduced by DRS and the dataset reduced by Chen et al. (2014) is practically imperceptible. However, it is worth clarifying that in all cases DRS slightly exceeds the proposal presented by Chen et al. (2014).

The time taken to build the classification model is not the only advantage offered by DRS, it also provides a notable improvement during the classification process (see Fig. 3). DRS allows classifiers to practically double their speed during the classification. The improvement is more noticeable in classifiers with higher computational cost. For example, in Fig. 3(b), the classification time of NNge is reduced by 46 %, making it 41 s faster regarding the baseline. In this experiment, our proposal also surpasses the results achieved by Chen et al. (2014).

4.3. Scan and backscatter detection: A study case

In this section we present a study case, where DRS is used to reduce the original training set and improve the performance of classifiers to detect scan and backscatter packets. The dataset used in this experiment is a subset of data collected by a network telescope (Moore et al., 2006) of the Delft University of Technology over a period of 16 months. Specifically, the dataset represents 24 h (since November 13, 2015 at 14:11:45 until November 14, 2015 at 14:21:47) of scan and backscatter TCP traffic.

From this dataset, each network packet was processed and represented as an instance, where each element represents a feature of the packet (see Table 2). Note that we only process valid TCP segment. A segment is valid unless an invalid part of it was found. An example of invalid part could be: bad checksum, an illegal value for a field, or if the length of the packet is too short for a header or according to a length field.

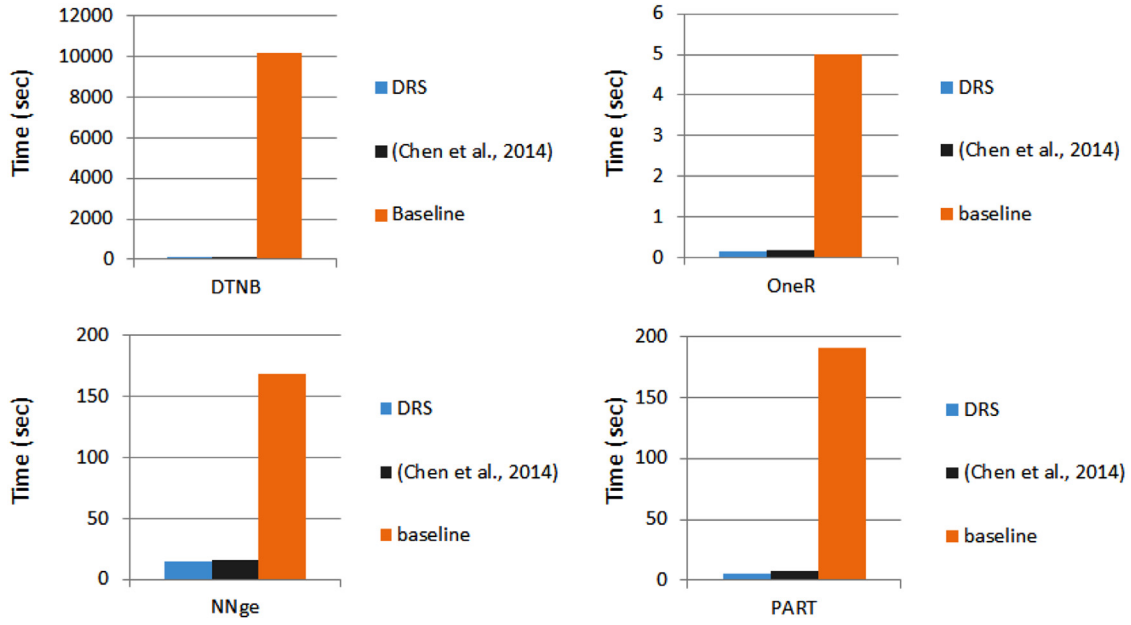

Fig. 2. Time taken to build the classification models for KDD'99 dataset.

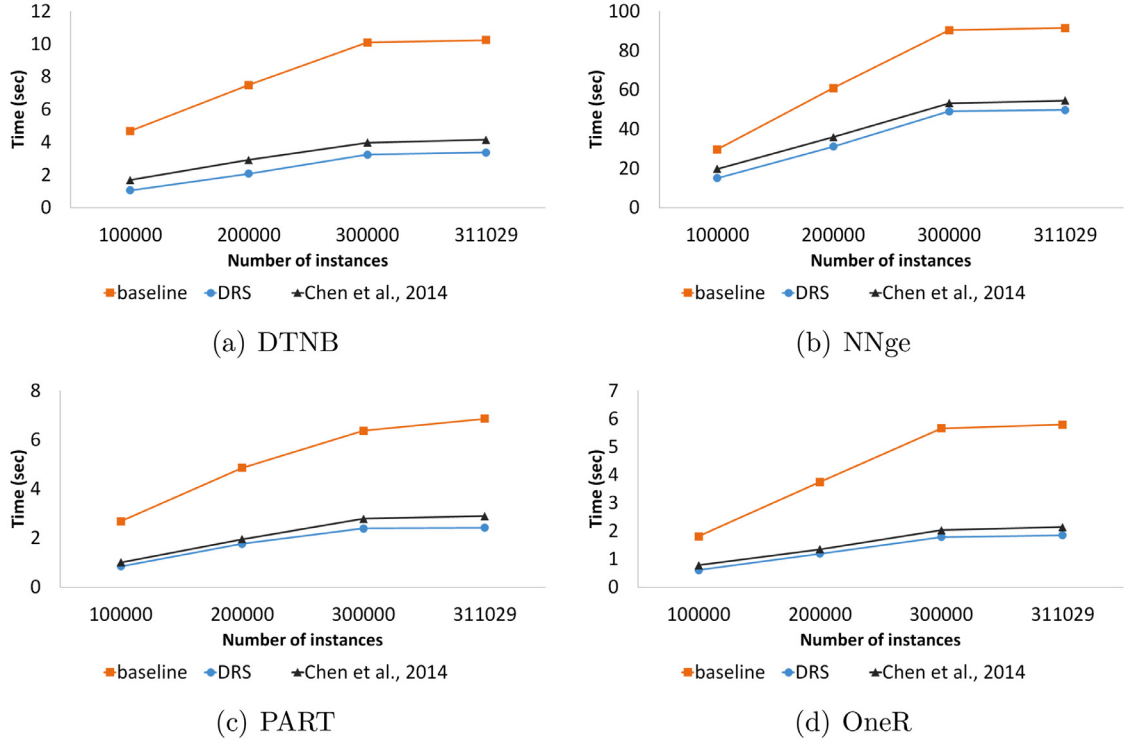

Fig. 3. Comparative results about the detection speed of the classifiers for KDD'99 dataset.

The network traffic arriving at the network telescope is sent to unused IP addresses. Therefore, the packets arriving could come from: scans, backscatter or IP misconfigurations. In order to label the dataset, we rely on the work of Blenn, Ghiëtte, and Doerr (2017), where they propose a strategy to identify when a package (collected by a network telescope) belongs to the scan or backscatter traffic. Basically, the idea is that the scan could be separated from backscatter through the SYN+ACK and RST flags (TCP flag feature). In this sense, network scans from clients to a server will feature the SYN flag; on the other hand, backscatters must thus contain a SYN+ACK for an open port or RST for a closed port.

After labeling the dataset, we removed the TCP flag feature, in order to extract other patterns from the data that could also discriminate between backscatter and scans. After all packets were processed, a labeled dataset with 21470669 instances and 12 features was obtained.

To make our experiment closer to reality, we split the dataset in hours, and use the first hour subset T for the training, and the remaining subsets for the testing. The training set T consists of 794544 instances and 12 features. After applying DRS on T, a reduced dataset S was obtained with 62768 instances and 8 features (source address, source port, destination address, destination port,

Table 2
Features used to represent the dataset.

Feature	Description
Source address	Source IP-address of the sending host.
Source port	Source port of the sending host.
Destination address	Destination IP-address of the receiving host.
Destination port	Destination port of the receiving host.
Source MAC	Source MAC-address of the sending host.
Destination MAC	Destination MAC-address of the receiving host.
Protocol	Protocol used in the data portion of the IP datagram.
Packet length	The number of bytes this packet take.
IP length	The number of bytes in this segment.
TCP flag	Indicates the active control bits in the TCP segment.
ICMP message	ICMP control message.
TTL	The time-to-live (TTL) value can be thought of as an upper bound on the time that an IP datagram can exist in an Internet system.
ToS	The type of service (ToS) field could specify a datagram's priority and request a route for low-delay, high-throughput, or highly-reliable service.

Table 3
Comparative results using *S* and *T* for training.

Classifier	TP	FP	Acc.	Class	Training set
OneR	0995	0018	9844	backscatter	S
	0982	0005	9844	scan	S
	0995	0017	9852	backscatter	T
	0983	0005	9852	scan	T
DTNB	1000	0020	9841	backscatter	S
	0980	0000	9841	scan	S
	1000	0020	9843	backscatter	T
	0980	0000	9843	scan	T
NNge	0.992	0015	9847	backscatter	S
	0985	0008	9847	scan	S
	0995	0015	9858	backscatter	T
	0985	0005	9858	scan	T
PART	0.995	0018	9844	backscatter	S
	0982	0005	9844	scan	S
	0995	0015	9858	backscatter	T
	0985	0005	9858	scan	T

packet length, IP length, ICMP message and TTL). This represents an 8 % of the number of instances in *T*, and a 77 % of the number of features in *T*.

After obtaining the reduced dataset *S*, we proceeded to check if the classifiers performance were affected. So we build the classification models using *S* for training, and compared their performances regarding the models built using *T* for training (see Table 3). The classification models were tested using the first hour dataset.

As shown in Table 3, the OneR, NNge and PART classifiers gets pretty much the same results using the original and reduced training sets. Moreover, the DTNB classifier achieves the same true positives (TP) and false positives (FP) rates, with only a negligible difference of 0, 02 % in accuracy (Acc.). After this evaluation, we can say that there is no big difference between the results obtained using the training sets *S* and *T*.

However, using the training set *S* brings certain benefits to the classifiers. The Fig. 4 shows the time taken to build a classification model by each classifier using *S* and *T*. In this figure, we can see that the time is reduced using *S*, which becomes more significant in classifiers such as NNge and DTNB that require more processing time. In addition, the time taken by PART classifier to build its classification model is reduced by approximately 90 %.

Fig. 5 shows another contribution of our proposal, the improvement of detection speed. We can see an improvement of detection speed in the classifiers when the reduced training set *S* was used to build the classification model. This advantage can be very useful for models that require real-time data analysis.

Our last experiment was to analyze over 24 h, how the classifiers used in our experiments behave. To do this, we evaluated the classification model generated by each classifier using the training set *S*. As shown in Fig. 6, the classifiers did not perform well. The main cause of this poor performance is that very specific rules were generated, mainly for the backscatter class, where the rules refer to specific IP addresses.

Apparently, PART classifier shows adequate performance, however this is not due directly to the rules generated, but rather to its evaluation strategy. Such evaluation strategy defines a default rule to classify those instances that are not covered by any of the rules generated. The default rule consists in assigning the most frequent class in the training set (in our case scan class). For example, during the 10th hour and the 21st hour there is an abrupt drop in the accuracy of PART. This is because these datasets contain a large number of backscatter packets that could not be covered by the existing rules, and therefore the default rule did not cover them either.

IDS are considered as systems under attack, where attackers change the way to execute their attacks in order to make the detection system fail. This fact affects the data associated with the attacks, which constantly change. Rule-based IDS must be able to adapt to these changes in the data. Therefore, it is necessary to consider a mechanism to periodically update the existing rules, which allows to maintain an adequate accuracy level, thus avoiding poor performance of the classifiers over the course of time, as shown in Fig. 6.

5. Conclusions

The training set reduction represents an important step for supervised classification on large volumes of data. But, the dataset reduction without significantly affecting the accuracy of the classifiers represents a challenge. In this paper, we present a novel data reduction strategy for improving the IDS performance. Experimental results show that our proposal can significantly reduce the amount of training data, without affecting the classifiers accuracy too much. This makes feasible the use of DRS as a preprocessing method for the application of expensive techniques on large volumes of data.

One of the advantages that DRS offers to rule-based IDS is that it considerably reduces the time taken to build the classification model. Another advantage is that the training set reduction allows classifiers to generate rules using only the most representative features of the training set. This contributes to reduce the amount of rules generated and the amount of conditions in them, making the evaluation process less costly, and thus increasing the speed of the detection process. This contribution is very useful for real-time detection. Another advantage of DRS is that it runs very efficiently on a computer with discrete features, such as the one used in our experiments, unlike other proposals which require a more expensive deployment to achieve an efficiency similar to ours.

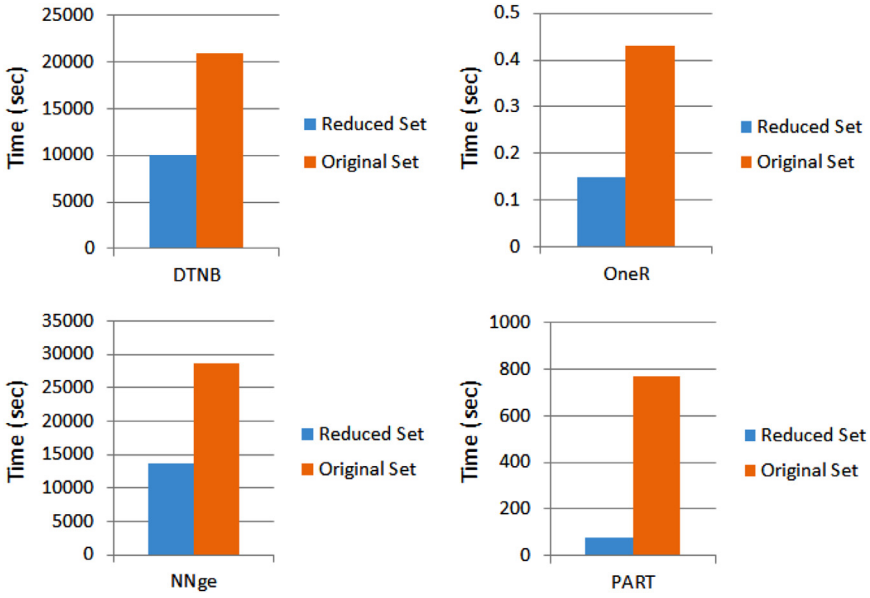

Fig. 4. Time taken to build the classification models.

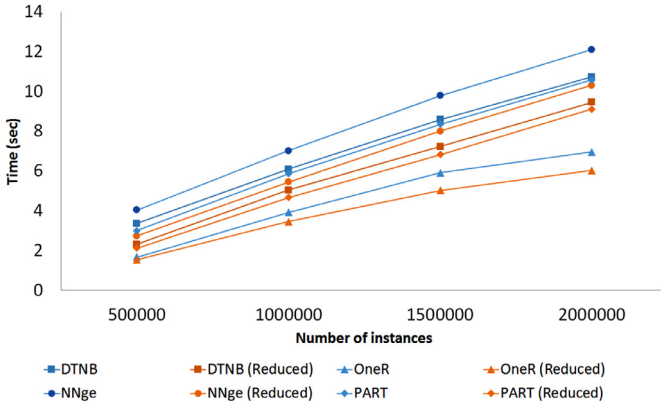

Fig. 5. Comparative results about the detection speed of the classifiers.

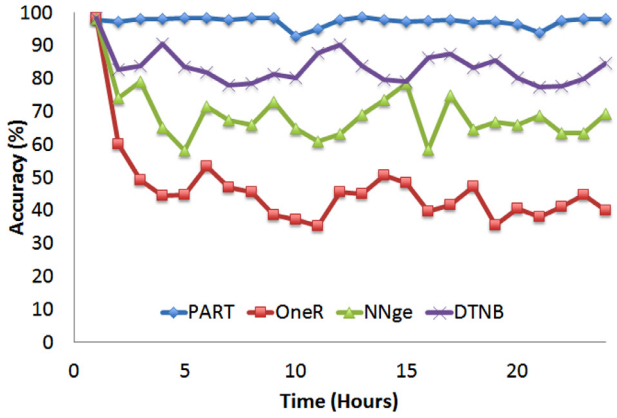

Fig. 6. Accuracy reported by different classifiers during 24 h.

As shown in the experiments, our proposal is not limited to scan and backscatter detection, DRS can also be applied in environments with similar characteristics to those present in this scenario. For example, intrusions detection, fraud detection in telecommunications services, and fraud detection in banking transactions are environments where DRS could be used to improve the performance of detection systems.

Our last experiment demonstrated that over the course of time, classifiers begin to lose accuracy, mainly because there is no regular updating of existing rules, besides the fact that very specific rules are generated. As a future work, we will integrate our proposal with a rule-based IDS that allows us to update the existing rules periodically, and evaluate the performance of our proposal in other scenarios such as fraud detection in telecommunications services, where the volume of data to be processed is considerably large, reaching some millions of instances.

References

Aggarwal, C. C. (2015). *Data mining: The textbook*. Springer.

Azad, C., & Jha, V. K. (2014). Data mining based hybrid intrusion detection system. *Indian Journal of Science and Technology, 7*(6), 781–789.

Blenn, N., Ghiëtte, V., & Doerr, C. (2017). Quantifying the spectrum of denial-of-service attacks through internet backscatter. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*. Reggio Calabria, Italy: ACM.

Bou-Harb, E., Debbabi, M., & Assi, C. (2014). Cyber scanning: A comprehensive survey. *IEEE Communications Surveys & Tutorials, 16*(3), 1496–1519.

Chen, T., Zhang, X., Jin, S., & Kim, O. (2014). Efficient classification using parallel and scalable compressed model and its application on intrusion detection. *Expert Systems with Applications, 41*(13), 5972–5983.

Cheng, X., Cai, H., Zhang, Y., Xu, B., & Su, W. (2015). Optimal combination of feature selection and classification via local hyperplane based learning strategy. *BMC Bioinformatics, 16*(1), 2–19.

Chou, C.-H., Kuo, B.-H., & Chang, F. (2006). The generalized condensed nearest neighbor rule as a data reduction method. In *18th international conference on pattern recognition (icpr'06): 2* (pp. 556–559). IEEE.

Dash, R., Paramguru, R. L., & Dash, R. (2011). Comparative analysis of supervised and unsupervised discretization techniques. *International Journal of Advances in Science and Technology, 2*(3), 29–37.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *The Journal of Machine Learning Research, 3*, 1289–1305.

Frank, E., & Witten, I. H. (1998). Generating accurate rule sets without global optimization. In *International conference on machine learning* (pp. 144–151).

Frey, B. J., & Dueck, D. (2007). Clustering by passing messages between data points. *science, 315*(5814), 972–976.

Ganapathi, N. P., & Duraivelu, V. (2015). A knowledgeable feature selection based on set theory for web intrusion detection system. In *Artificial intelligence and evolutionary algorithms in engineering systems: 325* (pp. 51–59). Springer.

García, S., Luengo, J., & Herrera, F. (2015). *Data preprocessing in data mining*. Springer.

Ghahramani, Z. (2004). Unsupervised learning. In *Advanced lectures on machine learning, LNCS: 3176* (pp. 72–112). Springer.

Hall, M. A., & Frank, E. (2008). Combining naive bayes and decision tables. In *Flairs conference: 2118* (pp. 318–319).

Herrera-Semenets, V., Pérez-García, O. A., Gago-Alonso, A., & Hernández-León, R. (2017). Classification rule-based models for malicious activity detection. *Intelligent Data Analysis, 21*(5), 1141–1154.

Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning, 11*(1), 63–90.

Joiţa, D. (2010). Unsupervised static discretization methods in data mining. *Titu Maiorescu University, Bucharest, Romania*.

Kotsiantis, S. B. (2007). Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 conference on emerging artificial intelligence applications in computer engineering: Real word ai systems with applications in ehealth, hci, information retrieval and pervasive technologies* (pp. 3–24). Amsterdam, The Netherlands: IOS Press.

Liao, H.-J., Lin, C.-H. R., Lin, Y.-C., & Tung, K.-Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications, 36*(1), 16–24.

Liu, W., Liu, S., Gu, Q., Chen, J., Chen, X., & Chen, D. (2016). Empirical studies of a two-stage data preprocessing approach for software fault prediction. *IEEE Transactions on Reliability, 65*(1), 38–53.

Lumini, A., & Nanni, L. (2006). A clustering method for automatic biometric template selection. *Pattern Recognition, 39*(3), 495–497.

Maslove, D. M., Podchiyska, T., & Lowe, H. J. (2012). Discretization of continuous features in clinical datasets. *Journal of the American Medical Informatics Association, 20*(3), 544–553.

MeeraGandhi, G., Appavoo, K., & Srivasta, S. (2010). Effective network intrusion detection using classifiers decision trees and decision rules. *International Journal of Advanced Network and Application, Vol 2*.

Moore, D., Shannon, C., Brown, D. J., Voelker, G. M., & Savage, S. (2006). Inferring internet denial-of-service activity. *ACM Transactions on Computer Systems (TOCS), 24*(2), 115–139.

de Oliveira Moura, S., de Freitas, M. B., Cardoso, H. A., & Cavalcanti, G. D. (2014). Choosing instance selection method using meta-learning. In *2014 ieee international conference on systems, man, and cybernetics (smc)* (pp. 2003–2007). IEEE.

Olusola, A. A., Oladele, A. S., & Abosede, D. O. (2010). Analysis of kdd99 intrusion detection dataset for selection of relevance features. In *Proceedings of the world congress on engineering and computer science: 1* (pp. 20–22).

Olvera-López, J. A., Carrasco-Ochoa, J. A., & Martínez-Trinidad, J. F. (2010). A new fast prototype selection method based on clustering. *Pattern Analysis and Applications, 13*(2), 131–141.

Olvera-López, J. A., Carrasco-Ochoa, J. A., Martínez-Trinidad, J. F., & Kittler, J. (2010). A review of instance selection methods. *Artificial Intelligence Review, 34*(2), 133–143.

Özgür, A., & Erdem, H. (2016). A review of kdd99 dataset usage in intrusion detection and machine learning between 2010 and 2015, *4*, e1954v1. PeerJ PrePrints.

Panda, M., & Patra, M. R. (2009). Ensembling rule based classifiers for detecting network intrusions. In *International conference on advances in recent technologies in communication and computing, 2009. artcom'09.* (pp. 19–22). IEEE.

Raicharoen, T., & Lursinsap, C. (2005). A divide-and-conquer approach to the pairwise opposite class-nearest neighbor (poc-nn) algorithm. *Pattern Recognition Letters, 26*(10), 1554–1567.

Silva, D. A., Souza, L. C., & Motta, G. H. (2016). An instance selection method for large datasets based on markov geometric diffusion. *Data & Knowledge Engineering, 101*, 24–41.

Sylvain, R. (2002). Nearest neighbor with generalization. *University of Canterbury, Christchurch, New Zealand*.

Vannucci, M., & Colla, V. (2004). Meaningful discretization of continuous features for association rules mining by means of a som. In *European symposium on artificial neural networks (esann)* (pp. 489–494).

Wilson, D. R., & Martinez, T. R. (2000). Reduction techniques for instance-based learning algorithms. *Machine Learning, 38*(3), 257–286.

Xia, J., Fang, A. C., & Zhang, X. (2014). A novel feature selection strategy for enhanced biomedical event extraction using the turku system. *BioMed Research International, 2014*, 1–12.

Yar, M. (2013). *Cybercrime and society* (2nd ed.). Sage.

Zuech, R., Khoshgoftaar, T. M., & Wald, R. (2015). Intrusion detection and big heterogeneous data: A survey. *Journal of Big Data, 2*(1), 1–41.