A combination of discretization and filter methods for improving classification performance in KDD Cup 99 dataset

Conference Paper · June 2009

DOI: 10.1109/IJCNN.2009.5178622 · Source: DBLP

CITATIONS
26

READS
340

3 authors:

Verónica Bolón-Canedo
University of A Coruña
103 PUBLICATIONS **1,731** CITATIONS

SEE PROFILE

Noelia Sánchez-Maroño
University of A Coruña
81 PUBLICATIONS **1,553** CITATIONS

SEE PROFILE

Amparo Alonso-Betanzos
University of A Coruña
237 PUBLICATIONS **3,097** CITATIONS

SEE PROFILE

A Combination of Discretization and Filter Methods for Improving Classification Performance in KDD Cup 99 Dataset

V. Bolón-Canedo, N. Sánchez-Maroño and A. Alonso-Betanzos

Abstract— **KDD Cup 99 dataset is a classical challenge for computer intrusion detection as well as machine learning researchers. Due to the problematic of this dataset, several sophisticated machine learning algorithms have been tried by different authors. In this paper a new approach is proposed that consists in a combination of a discretizator, a filter method and a very simple classical classifier. The results obtained show the adequacy of the method, that achieves comparable or even better performances than those of other more complicated algorithms, but with a considerable reduction in the number of input features. The proposed method has also been tried over another two large datasets maintaining the same behavior as in the KDD Cup 99 dataset.**

I. Introduction

COMPUTER intrusion is a set of actions that violate the security of a system. Such a situation must be detected and corrected in order to guarantee the integrity, confidentially and/or availability of computing resources. Intrusion detection systems (IDS) have been designed that complement other security measures based on attack prevention (firewalls, antivirus, etc.). The aim of an IDS is to inform the system administrator of any suspicious activities and to recommend specific actions to prevent or stop the intrusion (for example, close network ports, kill doubtful processes, etc.). In order to be able to implement these actions, the IDS must, among other tasks, analyze network traffic data in order to determine whether there is evidence of an attack, or whether the data are anomalous with respect to normal traffic. Ideally, the system should be sufficiently generalized to be able to detect any type of attack all the while maintaining a low false positive rate.

This paper proposes a method combining discretization algorithms, filters and classifiers with the goal of enhancing capabilities in computer intrusion detection. Specifically, it was intended to deal with the KDD Cup 99 dataset, a standard and well explored reference in this field [1]. It is a hard dataset for the sake of classification because of its large size and high number of input features, some of them with unbalanced values. A preliminary study of this dataset suggests that there are some features that might be irrelevant, correlated and so on. Then, feature selection, the process of selecting a feature subset from the training examples and ignoring features not in this set during induction and classification, could be an effective way to improve its performance and decrease the training time of a supervised learning algorithm [2]. There are two main models that deal with feature selection: filter methods and wrapper methods [3]. Although wrapper methods tend to obtain better performances, they are very time consuming and it will be intractable to deal with the entire KDD Cup 99 dataset. Therefore, filters allow for reducing the dimensionality of the dataset without compromising the time and memory requirements of machine learning algorithms. There are many filters in the literature [3][4][5][6] with different properties and some of them were chosen for a comparative study.

Many filter algorithms are shown to work effectively on discrete data or even more strictly, on binary data [7], so discretization is necessary as a previous step. Most of the authors employed as discretizator EMD (Entropy Minimization Discretization), the method proposed by Fayyad & Irani [8]. However, other discretizators exist that are reported to behave better, as PKID (Proportional k-Interval Discretization) in combination with naive Bayes [9]. In this work, it will be demonstrated that the choice of the discretizator determines the features selected by the filter and, consequently, affects the results of the classifier. Furthermore, it will be shown that combining a discretizator followed by a filter method, a simple classification algorithm (such as C4.5 [10] or naive Bayes [11]) can obtain good performance results. In fact, the results obtained are similar or even better to those of the KDD winner and other more sophisticated machine learning methods, such as SVMs, functional networks, etc., but with a reduced set of features.

Finally, a preliminary comparative study of our combination method over two large datasets [12] was carried out, with the aim of extrapolating the results obtained with the KDD Cup 99 dataset. As a conclusion, the proposed method obtains similar performances, using a reduced set of features, than those obtained by the original machine learning method.

II. The KDD Cup 99 dataset

The KDD Cup 99 dataset, which derived from the DARPA dataset [13], was used for the KDD (Knowledge Discovery and Data Mining Tools Conference) Cup 99 Competition [14]. The complete dataset has almost 5 million input patterns and each record represents a TCP/IP connection that is composed of 41 features that are both qualitative and quantitative in nature [15]. The dataset used in our study is a smaller subset (10% of the original training set), it has 494 021 instances and it was employed as the training set in the original competition. For the test set, we used the original

V. Bolón-Canedo, N. Sánchez-Maroño and A. Alonso-Betanzos are with the Department of Computer Science, University of A Coruña, Spain (email: {vbolon, nsanchez, ciamparo}@udc.es).

This work was supported in part by Spanish Ministerio de Ciencia e Innovación under Project Code TIN 2006-02402, partially supported by the European Union ERDF.

978-1-4244-3553-1/09/$25.00 ©2009 IEEE

KDD Cup 99 dataset containing 331 029 patterns. Around 20% of the two datasets are normal patterns (no attacks). As for attack patterns, four categories were identified [16]: Denial of Service (DoS), Probe, Remote-to-local (R2L) and User-to-root (U2R).

In general, IDS's users are interested in distinguishing between attack and no-attack situations. Then, the type of attack being performed is just one more step in the detection. In this manner, KDD Cup 99 problem can be treated as a binary classification problem, that consists of detecting attacks versus normal patterns. Thus, it is necessary to change the four labels of the class (DoS, Probe, R2L and U2R) into a new one called *Attack*. The training and test set percentages of the binary case are shown in Table I.

Type	% Train Set	% Test set
Normal	19.69	19.48
Attack	80.31	80.52

TABLE I

PERCENTAGES OF DISTRIBUTION OF NORMAL ACTIVITIES AND ATTACKS IN THE KDD CUP 99 TRAINING AND TEST DATASETS

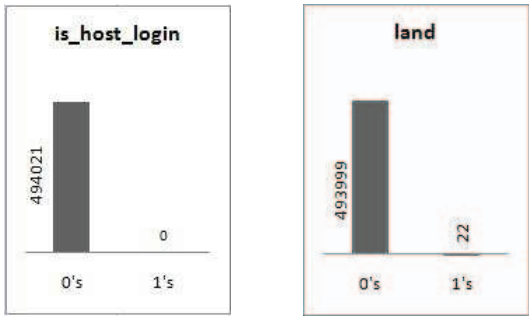

Fig. 1. KDD Cup 99 input attributes *is_host_login* and *land*. As can be seen, *is_host_login* is constant in the dataset and *land* has very few instances of the 1's type.

Feature	Min.	Max.	Mean	StdDev	Distinct
duration	0	58 329	47.98	707.75	2495
src_byes	0	693 375 640	3025.61	988 218.10	3300
dst_bytes	0	5 155 468	868.53	33 040.00	10 725

TABLE II

UNBALANCED CONTINUOUS ATTRIBUTES OF KDD CUP 99 DATASET

Since the amount of audit data that an IDS needs to examine is very large even for a small network, analysis is difficult even with computer assistance, because extraneous features can make it harder to detect suspicious behavior patterns [17]. Each record of the KDD Cup 99 dataset capture various features of the connections, as for example, the source and destination bytes of a TCP connection, the number of failed login attempts or the duration of a connection. Complex relationships exist between the features, which are difficult for human experts to discover. An IDS must therefore reduce the amount of data to be processed so as to maintain an accurate and real-time detection. Some input data may not be useful to the IDS and thus can be eliminated before processing. In complex classification domains, features may contain false correlations, which hinder the process of detecting intrusions. Furthermore, some features may be redundant since the information they add is contained in other features. Extra features can increase computation time, and can have an impact on the accuracy of the IDS. Feature selection improves classification by searching for the subset of features which best classifies the training data [18].

The KDD Cup 99 training dataset (494 021 instances) is a good candidate to feature selection because of the properties of its input attributes. It contains two features that are constant (*num_outbound_cmds* and *is_host_login*) and some that are almost constant (*land*, *root_shell*, *num_shells* ...). Two examples can be viewed at figure 1.

Apart from constant features, KDD Cup 99 dataset has continuous features that are very unbalanced and for which a possible solution can be to discretize numeric data. Some examples of these attributes can be viewed in Table II.

Besides the properties of its features, this dataset has problems because of its size: it is too large for some classifier algorithms. In fact, most of the previous studies about the KDD Cup 99 use a subset of the training set [19][20][21], usually containing less than 50 000 instances.

In this work, a new method is proposed with the aim of allowing the use of the whole training dataset (494 201 instances). The method consists in a combination of discretization and feature selection methods, achieving an important reduction in the number of input features. The proposed method is tested, obtaining results comparable to those of the KDD winner. Furthermore, simple classification methods are used (such as naive Bayes or C4.5) in order to demonstrate that they will obtain similar results to those of other more complicated methods.

III. FEATURE SELECTION

Feature selection consists on detecting the relevant features and discarding the irrelevant features. It has several advantages [22], such as:

- improving the performance of the machine learning algorithms.
- data understanding, gaining knowledge about the process and perhaps helping to visualize it.
- data reduction, limiting storage requirements and perhaps helping in reducing costs.
- simplicity, possibility of using simpler models and gaining speed.

There are two main models that deal with feature selection: filter methods and wrapper methods [3]. While wrapper models involve optimizing a predictor as part of the selection process, filter models rely on the general characteristics of the training data to select features with independence of any predictor. Wrapper models tend to give better results but filter methods are usually computationally less expensive than wrappers. So, in those cases in which the number of features is very large, filter methods are indispensable to

obtain a reduced set of features that then can be treated by other more expensive feature selection methods. In this paper, the filter approach was chosen because of the large size of the KDD Cup 99 dataset, in number of input features as well as in number of instances. Several filter methods were tested to check whether the results obtained were different.

A. Filter methods

CFS, INTERACT and Consistency-based filter were chosen for this study, with the aim of employing filters that use different performance measurements to select the final features. CFS is one of the most well-known and used filters, INTERACT is a new approach based in the interaction between features and finally, Consistency-based is a classical algorithm.

1) Correlation-based Feature Selection, CFS: Correlation based Feature Selection (CFS) is a simple filter algorithm that ranks feature subsets according to a correlation based heuristic evaluation function [4]. The bias of the evaluation function is toward subsets that contain features that are highly correlated with the class and uncorrelated with each other. Irrelevant features should be ignored because they will have low correlation with the class. Redundant features should be screened out as they will be highly correlated with one or more of the remaining features. The acceptance of a feature will depend on the extent to which it predicts classes in areas of the instance space not already predicted by other features. CFS's feature subset evaluation function is:

$$M_S = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}},$$

where M_S is the heuristic 'merit' of a feature subset S containing k features, $\overline{r_{cf}}$ is the mean feature-class correlation ($f \in S$) and $\overline{r_{ff}}$ is the average feature-feature intercorrelation. The numerator of this equation can be thought of as providing an indication of how predictive of the class a set of features is; and the denominator of how much redundancy there is among the features.

2) INTERACT: The INTERACT algorithm [5] is based on symmetrical uncertainty (SU) [23], which is defined as the ratio between the information gain (IG) and the entropy (H) of two features, x and y:

$$SU(x, y) = 2\frac{IG(x/y)}{H(x) + H(y)}, \quad (1)$$

where the information gain is defined as:

$$IG(x/y) = H(y) + H(x) - H(x, y),$$

being $H(x)$ and $H(x, y)$ the entropy and joint entropy, respectively.

Besides SU, INTERACT also includes the consistency contribution (c-contribution). C-contribution of a feature is an indicator about how significantly the elimination of that feature will affect consistency. The algorithm consists of two major parts. In the first part, the features are ranked in descending order based on their SU values. In the second part, features are evaluated one by one starting from the end of the ranked feature list. If c-contribution of a feature is less than an established threshold, the feature is removed, otherwise it is selected. The authors stated in [5] that INTERACT can thus handle feature interaction, and efficiently selects relevant features.

3) Consistency-based Filter: The consistency-based filter [6][24] evaluates the worth of a subset of features by the level of consistency in the class values when the training instances are projected onto the subset of attributes. The algorithm generates a random subset S from the number of features in every round. If the number of features of S is less than the current best, the data with the features prescribed in S is checked against the inconsistency criterion. If its inconsistency rate is below a pre-specified one, S becomes the new current best.

The inconsistency criterion, which is the key to the success of this algorithm, specifies to what extent the dimensionally reduced data can be accepted. If the inconsistency rate of the data described by the selected features is smaller than a pre-specified rate, it means the dimensionally reduced data is acceptable.

B. Discretization methods

Many filter algorithms are shown to work on discrete data [7]. In order to deal with numeric attributes, a common practice for those algorithms is to discretize the data before conducting feature selection. For both users and experts, discrete features are easier to understand, use, and explain and discretization can make learning more accurate and faster [25]. In general, the results obtained (decision trees, induction rules) using discrete features are usually more compact, shorter and more accurate than using continuous ones, hence the results can be more closely examined, compared, used and reused. In addition to the many advantages of having discrete data over continuous one, a suite of classification learning algorithms can only deal with discrete data. The filter methods described in the previous section cannot deal directly with numerical attributes, so we need to discretize the data before applying the filter method. For the INTERACT filter, a discretization of the numeric attributes is performed if needed. Something similar happens if we look to the source code in Weka for CFS or Consistency-based filter.

In essence, the process of discretization [26] involves the grouping of continuous values into a number of discrete intervals. However, the decision of which continuous values group together, how many intervals to generate, and thus where to position the interval cutpoints on the continuous scale of attribute values is not always identical for the different discretization methods. There exist many discretizators in the literature, but in this work the most suitable for large datasets have been chosen. In this manner, we chose EMD (Entropy Minimization Discretization), EWD (Equal Width Discretization) and EFD (Equal Frequency Discretization) because they are classic algorithms, and PKID because it is a new approach that works well with large datasets.

In order to discretize a numeric attribute X_i, there are *n* training instances for which the value of X_i is known, the minimum and the maximum value are v_{min} and v_{max} respectively. All the discretization methods first sort the values into ascending order.

The discretization methods used in this paper are the following [27]:

1) Entropy Minimization Discretization, EMD: This popular method was created by Fayyad & Irani [8]. EMD evaluates as a candidate cut point the midpoint between each successive pair of the sorted values. For evaluating each candidate cut point, the data are discretized into two intervals and the resulting class information entropy is calculated. A binary discretization is determined by selecting the cut point for which the entropy is minimal amongst all candidates. The binary discretization is applied recursively, always selecting the best cut point. A minimum description length criterion (MDL) is applied to decide when to stop discretization.

2) Proportional k-Interval Discretization, PKID: PKID is a method created by Yang & Webb [9]. The idea behind PKID is that discretization bias and variance relate to interval size and interval number. This strategy seeks an appropriate trade-off between the bias and variance of the probability estimation by adjusting the number and size of intervals to the number of training instances. The following compromise is adopted: given a numeric attribute, supposing we have *n* training instances with known values for the attribute, we discretize it into $\sqrt{n}$ intervals, with $\sqrt{n}$ instances in each interval. Thus we give equal weight to both bias and variance management. Further, with *N* increasing, both the number and size of intervals increase correspondingly, which means discretization can decrease both the bias and variance of the probability estimation. This is very desirable, because if a numeric attribute has more instances available, there is more information about it. PKID has greater capacity to take advantage of the additional information inherent in large volumes of training data.

3) Equal Width Discretization, EWD: EWD divides the number line between v_{min} and v_{max} into *k* intervals of equal width; *k* is a user predefined parameter and usually is set as 10.

A variant of this method is the Bin-log *l*. In this case, the number of intervals, *k*, is established as $k = \max\{1, 2{*}\log l\}$, where *l* is the number of distinct observed values for each attribute [28].

4) Equal Frequency Discretization, EFD: EFD divides the sorted values into *k* intervals so that each interval contains approximately the same number of training instances. Thus each interval contains n/k (possibly duplicated) adjacent values; *k* is a user predefined parameter and usually is set as 10.

IV. A PREVIOUS ANALYSIS OF DISCRETIZATION METHODS

Before applying discretization and feature selection methods to the KDD Cup 99 dataset, an analysis was carried out to check how discretization methods behave when they are combined with classifier methods. Our aim was to demonstrate that they have an influence on the classification errors. Then, we have to check if feature selection methods select the same subset of features independently of the discretizator applied.

Several authors have studied discretization methods in combination with naive Bayes classifier. In the Yang & Webb work [9] they conduct a wide study of distinct discretizators, like PKID and EMD over several datasets from the UCI machine learning repository [12] and KDD archive [29] with naive Bayes as the classifier. In this paper, we have extended this research including another discretization method, Bin-log *l* (BL) and another classifier, C4.5. The decision tree C4.5 classifier applies an internal discretization. However, for applying a filter, it is necessary to apply a discretizator before (especially in KDD Cup 99 dataset that has very unbalanced attributes). So, we have checked whether distinct discretizators have an influence on the C4.5 classification results.

For the comparative study, the longer datasets used in Yang & Webb [9] were chosen due to their size similarity to the KDD Cup 99 dataset. For each dataset, a 10-trial, 3-fold cross validation is used to train and test the classifier. In each fold, the dataset was discretized by the corresponding method and then a classifier was applied. The classification performance was evaluated in terms of average error in the testing data across trials.

The classification errors of PKID, EMD and BL on each dataset with naive Bayes and C4.5 as classifiers are listed in Table III. In each record, boldface font indicates the discretizator that achieves the best error for each dataset and classifier.

Dataset	naive Bayes			C4.5		
	PKID	BL	EMD	PKID	BL	EMD
Musk	**8.3**	20.9	9.4	6.5	4.6	**4.5**
Pioneer-1	**1.7**	10.2	14.8	2.2	6.5	**0.5**
Handwritten	**12.0**	13.4	13.5	38.6	**10.9**	12.0
Letter	**25.8**	40.2	30.4	24.8	**21.6**	23.2
Adult	**17.1**	18.3	17.2	13.8	16.1	**13.4**
Ipums.la.99	19.9	**16.7**	20.1	**6.5**	7.4	**6.5**
Mean Error	**14.1**	20.0	17.6	15.4	11.2	**10.0**

TABLE III
AVERAGE ERROR OVER EXPERIMENTAL DATASETS

As Table III shows, for naive Bayes the best results are obtained with PKID while EMD obtained the best performance for C4.5. PKID is a more suitable method for naive Bayes than EMD or BL because it gives importance both to the number of intervals and to the size of discretized intervals and adjusts it in relation to the amount of available training data. However, EMD was developed in the context of decision tree learning and it seeks to identify a small number of intervals, each dominated by a single class, so it is more adequate for C4.5.

In general, the best results are obtained with C4.5 classifier, so that is also our working hypothesis for the KDD

Cup 99 dataset. However, due to the variability of the results obtained with the different discretizators (Table III), several combinations will be tested.

V. KDD Cup 99 Results

The previous section shows the different results achieved by classifiers depending on the discretization method employed. It can be assumed that filter methods will exhibit a similar behavior, i.e. depending on the discretizator chosen the filter will select very different features subsets.

A. Study of different discretization and filter combinations

In order to prove the above statement, a comparative study was carried out using the KDD cup 99 training dataset of 494 021 samples. Five discretization methods (EMD, PKID, EWD, EFD and BL) were considered in combination with 3 filters (CFS, INTERACT and Consistency-based Filter). Therefore, 15 combinations of discretizador plus filter were considered, in order to check which subset of features is selected by each combination. This study showed in Table IV.

As expected, there are large differences between the combinations, both in number of features and in which set of features is selected. No two combinations exist that choose the same subset of characteristics, so it shows the importance of the selection of the discretizator. CFS (with any discretizator) chooses the minimum number of features (5), while combination between EWD and Consistency-based selects the maximum (26). It seems that Consistency-based is the filter that selects the largest number of features, while CFS selects the smallest. In terms of discretizators, when using PKID or EMD, the filters selected fewer features than when using EWD, EFD or BL.

		Filter		
		CFS	**INTERACT**	**Consistency**
Discretizator	**PKID**	3,6,12,31,37	3,5,6,12,23,31,32	1,3,5,32,34,39
	EMD	6,12,23,31,37	3,5,6,19,23,31,37	1,3,5,23,33,34,35
	EWD	12,23,31,32,35	3,10,12,23,28,31, 32,33,36,37,41	1,2,3,4,6,8,10,11, 12,14,17,18,19,23, 26,27,28,29,31,32, 33,34,35,36,37,40
	EFD	6,12,23,31,32	3,5,6,12,23,31,35, 37	1,3,5,6,10,12,16, 24,25,32,33,34,35, 36,37,39,40
	BL	12,23,31,32,35	3,10,12,28,29,31, 32,33,36,37,41	1,2,3,4,10,12,23, 26,27,28,29,31,32, 33,34,35,36,37,39, 40,41

TABLE IV
SELECTED FEATURES FOR THE DISTINCT COMBINATIONS BETWEEN DISCRETIZATORS AND FILTERS

Each combination leads to a different subset of features, and so a classifier could obtain different performance results. Hence, it is not possible to determine which combination of filter and discretizator is the best. A further study, including classifier, is required.

B. Combining discretizators, filters and simple classifiers

Results for the KDD Cup 99 binary classification problem were obtained combining 5 discretization algorithms (EMD, PKID, EWD (with 10 bins), EFD (with 10 bins) and BL), 3 feature selection methods (CFS, INTERACT and Consistency-based Filter) and 2 classifiers (naive Bayes and C4.5). With naive Bayes classifier, *kernel* parameter was tuned in order to obtain the best results. The same occurs with C4.5 classifier and its parameter *Confidence Factor*, set to 0.25 and 0.50.

All the methods described above are available in Weka tool [30], with the exception of Bin-log *l*, that was implemented in Matlab [31].

In order to check the good performance of the proposed method, it is necessary to do a 10-fold cross-validation over the KDD Cup 99 training data set. Later, the model will be applied to the KDD Cup 99 test dataset of 311 029 samples, with the aim of checking the performance over a dataset with new attacks that simulate the real world.

A total of 60 combinations of discretizator, filter and classifier were tested. For the sake of clarity, only the best result for each combination is shown. For example, for the combination between CFS filter and naive Bayes classifier, the best result was obtained using Bin-log *l* discretizator and only this combination is shown. The performance measures used are the error, true positive's rate and false positive's rate, defined as:

- **TP** shows the overall percentage of attacks detected.
- **FP** shows the false positive rate, that is the proportion of normal patterns erroneously classified as attacks.
- **Error** shows the overall percentage error rate for the two classes (Normal and Attack).

In Table V the validation results (using a 10-fold cross-validation) for this experiment are shown. Again, boldface font indicates the best value per column. The last column indicates the number of selected features.

Method	**Error**	**TP**	**FP**	**F. No.**
BL + CFS + NB	1.88	97.99	1.37	5
EMD + INT + NB	0.53	99.34	**0.02**	7
PKID + Cons + NB	0.70	99.14	0.04	6
EMD + CFS + C45	0.49	99.50	0.46	5
EMD + INT + C45	**0.03**	99.97	0.06	7
EMD + Cons + C45	**0.03**	**99.98**	0.06	7
PKID + Cons + C45	0.05	**99.98**	0.15	6

TABLE V
VALIDATION RESULTS USING DIFFERENT COMBINATIONS FOR KDD CUP 99 DATASET

As was expected, EMD appears to be the best discretizator for C4.5; however, for naive Bayes, PKID was not always the best option. The best TP rate was obtained with EMD + INT + NB (second row), but the best result in the table is the one achieved by EMD + Cons + C45 (sixth row), since it has the best error and TP rate, and FP rate suffers a small increase (from 0.02 to 0.06). Test results are listed in Table VI.

Method	Error	TP	FP
BL + CFS + NB	9.17	89.24	2.59
EMD + INT + NB	8.06	90.28	1.23
PKID + Cons + NB	7.99	90.18	**0.42**
EMD + CFS + C45	7.97	90.26	0.62
EMD + INT + C45	6.74	91.73	0.44
EMD + Cons + C45	6.29	92.62	1.78
PKID + Cons + C45	**5.15**	**94.07**	1.90

TABLE VI

TEST RESULTS USING DIFFERENT COMBINATIONS FOR KDD CUP 99 DATASET

Observing the test results, it can be confirmed that the system performance varies considerably. This is due to the fact that the test dataset has attack patterns that are not included in the training dataset. Best FP rate was achieved by PKID + Cons + NB (third row) but error and TP rate are not satisfactory. However, PKID + Cons + C4.5 has very good results in error and TP rate, but worse in FP rate. EMD + INT + C4.5 (fifth row) seems to have the more balanced results in the three measurements.

In general, best results are obtained with C4.5 classifier and with a small number of selected features (between 5 and 7); the combinations with higher number of features (26) achieved bad performance results (error of 7.72).

Next, a comparative study with the results from other authors [20][21] for the test dataset is shown in Table VII. It is important to note that the validation results are not included in this table because of the different validation techniques: they use a validation set that is a subset of the training dataset and in this study a 10-fold cross-validation was performed. Furthermore, the validation results of the KDD Winner are not available. Besides, a statistical test trying to establish whether the differences among the methods were statistically significant is not possible, since for the other author's methods only the results shown in Table VI were available in [20], [21].

Method	Error	TP	FP
PKID+Cons+C4.5(0.25)	5.15	94.07	1.90
PKID+Cons+C4.5(0.50)	**5.14**	**94.08**	1.92
EMD+INT+C4.5(0.25)	6.74	91.73	**0.44**
EMD+INT+C4.5(0.50)	6.69	91.81	0.49
KDD Winner	6.70	91.80	0.55
5FNs_poly	6.48	92.45	0.86
5FNs_fourier	6.69	92.72	0.75
5FNs_exp	6.70	92.75	0.75
SVM Linear	6.89	91.83	1.62
SVM 2poly	6.95	91.79	1.74
SVM 3poly	7.10	91.67	1.94
SVM RBF	6.86	91.83	1.43
ANOVA ens.	6.88	91.67	0.90
Pocket 2cl.	6.90	91.80	1.52
Pocket mcl.	6.93	91.86	1.96

TABLE VII

COMPARISON WITH OTHER AUTHORS OVER THE TEST DATASET

The first four rows in Table VII show the results obtained by using the combination proposed in this paper, while the remaining rows show results obtained by other authors (see [21]). Specifically, for different functional networks, SVM models, ANOVA (ANOVA ens.), and linear perceptrons (Pocket 2cl. and Pocket mcl.). Finally, the results obtained for the KDD Cup 99 by the competition winner are reproduced in the fifth line of the table.

As it can be seen in Table VII, the combination PKID + Cons + C4.5 (0.50) (second row), obtains the best error and TP rate employing only 6 features (14% of total). Nevertheless, this improvement has a negative impact on the FP rate , although it is not the worst value in the table (SVM 3poly and Pocket mcl).

The lower FP rate is achieved using EMD + INT + C4.5 (0.25) (third row). Similar results are obtained by varying the confidence factor (fourth row). It can be verified that these results outperform the results achieved by the KDD Winner. Error and TP rate are very similar, and in addition a decrement in the FP rate is obtained using this combination. It must be emphasized that the FP rate is a measure of immense importance in determining the quality of an IDS system [32]. Moreover, this combination uses just 7 features, while the KDD winner employs the whole features set (41). Therefore, a better result is obtained with a simpler model that only needs 17% of the total features.

VI. APPLYING THE PROPOSED COMBINATION TO OTHER DATASETS

In order to check if the results obtained (that is, maintaining or even decrementing the performance, but with a reduced number of input features) could be generalized, the method was applied to two large datasets. With KDD Cup 99 dataset, equal or better results than other authors were achieved but with 6 times fewer number of features. The target is to demonstrate that fewer features can obtain similar results than the whole set, with a computational time and storage requirements reduction.

- Census-Income dataset [12] has 41 features, a binary class, 199 523 training instances and 99 762 test instances. For the sake of comparison, Table VIII shows validation and testing results after applying C4.5 and naive Bayes on Census-Income dataset, then the proposed method is used and only the best results are shown. This dataset is highly unbalanced; the majority class contains 93.79% of the samples in the training set, and 93.90% in the test set. Then, an error rate of 6.21% is achieved by assigning all samples to the majority class and no features are required.

  Table VIII shows that filtering allows to outperform the results (Naive Bayes rows) or, at least, to achieve similar results but with a high reduction in the number of features. Performance results obtained with naive Bayes are very poor in both cases because better results can be achieved by assigning all samples to the majority class.

Method	Val. Error	Test Error	F. No.
C4.5	4.67	4.56	41
EMD + INT + C4.5	4.63	4.64	24
Naive Bayes	14.71	14.61	41
PKID + Cons + NB	11.54	11.58	14

TABLE VIII

10-FOLD CROSS VALIDATION AND TEST RESULTS ON CENSUS-INCOME DATASET

- Adult dataset [12] has 14 features, a binary class, 32 561 training instances and 16 281 test instances. Table IX shows validation and testing results after applying C4.5 and naive Bayes on Adult dataset. Then the proposed method is used and several results are included due to their being very similar and thus making it difficult to determine which is the better among them. This dataset is not as unbalanced as Census-Income dataset, since in this case, the majority class contains 75.92% of the samples in the training set, and 76.38% in the test set.

Method	Val. Error	Test Error	F. No.
C4.5	13.78	14.16	14
EMD + CFS + C45	14.19	14.07	5
PKID + CFS + C45	14.08	14.01	5
PKID + INT + C45	14.03	13.56	9
Naive Bayes	16.52	16.80	14
PKID + CFS + NB	16.78	16.71	5
EMD + CFS + NB	16.69	16.59	5
PKID + INT + NB	14.59	14.49	9

TABLE IX

10-FOLD CROSS VALIDATION AND TEST RESULTS ON ADULT DATASET

The decrement in the number of features employed by the proposed method with respect to the classifier is not as high as in the previous cases, however it maintains the performance by using just 37% of features.

VII. CONCLUSIONS AND FUTURE WORK

In this paper a combination of discretization and feature selection methods to improve binary classification performance in three large datasets has been presented. The method achieves good results over the KDD Cup 99 dataset, outperforming the KDD Winner results with only 17% of the total features. Specifically, the result obtained for the combination EMD+INT+C45(0.50) in Table VII is very important due to the fact that none of the other authors have achieved better results than KDD Winner in all three measures. It must be emphasized also, the low FP rate, while maintaining the Error and TP rates.

The good results obtained using only 7 features implies that only those features, rather than of the original 41, will be required in a real system. This will mean an important reduction in the data processing and the data storage costs besides obtaining an interesting improvement in performance.

In order to check if the results obtained could be generalized, the method has been tried over two large datasets with similar characteristics (Census-Income and Adult), and the results maintain the performance of the original machine learning methods but again reducing the number of features.

For future work, we plan to check the adequacy of more sophisticated classifiers and also of other methods of dimensionality reduction. The extension of the combination method to the KDD Cup 99 multiclass problem is also an interesting challenge. Besides the reported difficulties of the binary KDD Cup 99 dataset, some new problems need to be envisaged: the four types of attacks are heavily unbalanced, different features must be taken into account for correctly classifying each class, different types of classifiers might be needed for each class, etc.

REFERENCES

[1] C. Elkan, *Results of the KDD'99 Classifier Learning.* ACM SIGKDD Explorations Newsletter, vol. 1, no. 2, pp. 63-64, ACM Press, 2000

[2] S. Das, *Filters, Wrappers and a Boosting-Based Hybrid for Feature Selection.* Machine Learning -International Workshop then Conference-, pp. 74-81, 2001

[3] R. Kohavi and G. John, *Wrappers for Feature Subset Selection.* Artificial Intelligence Journal, Special issue on relevance, vol. 97, no. 1-2, pp. 273-324, Elsevier, 1997

[4] M.A. Hall, *Correlation-based Feature Selection for Machine Learning.* PhD thesis, University of Waikato, Hamilton, New Zealand, 1999

[5] Z. Zhao and H. Liu, *Searching for Interacting Features* Proceedings of International Joint Conference on Artificial Intelligence, IJCAI, 1991, pp. 1156-1167

[6] M. Dash and H. Liu, *Consistency-based Search in Feature Selection.* Artificial Intelligence Journal, vol. 151, no. 1-2, pp. 155-176, 2003

[7] H. Liu and R. Setiono, *Feature Selection via Discretization.* Journal of IEEE Transactions on Knowledge and Data Engineering, pp. 642-645, 1997

[8] U.M. Fayyad and K.B. Irani, *Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning* Proceedings of the 13th International Joint Conference on Artificial Intelligence, pp. 1022-1029, Morgan Kaufmann, 1993

[9] Y. Yang and G.I. Webb, *Proportional k-Interval Discretization for Naive-Bayes Classifiers.* EMCL '01: Proceedings of the 12th European Conference on Machine Learning, pp. 564-575, Springer-Verlag, 2001

[10] J.R. Quinlan, *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993

[11] I. Rish, *An Empirical Study of the Naive Bayes Classifier.* Proceedings of IJCAI-01 Workshop on Empirical Methods in Artificial Intelligence, vol. 335, 2001

[12] C.L. Blake and C.J. Merz, *UCI Repository of Machine Learning Databases.* `http://www.ics.uci.edu/~mlearn/MLRepository.html` [Last access: 1st December 2008] Department of Information and Computer Science, University of California, Irvine, 1998

[13] R.P. Lippmann, D.J. Fried, I. Graf, J.W. Haines, K.R. Kendall, D. McClung, D. Weber, S.E. Webster, D. Wyschogrod, R.K. Cunningham et al., *Evaluating Intrusion Detection Systems: the 1998 DARPA Off-line Intrusion Detection Evaluation.* Proc. DARPA Information Survivability Conference and Exposition, 2000, vol. 2.

[14] KDD Cup 99 Data., `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html` [Last access: 12th November 2008].

[15] S. Stolfo, W. Fan, W. Lee, A. Prodomidis and R.K. Chan, *Cost-based Modeling for Fraud and Intrusion Detection: Results from the JAM Project* In DARPA Information Survivability Conf., 2000

[16] S. Mukkamala, A.H. Sung and A. Abraham, *Intrusion Detection Using an Ensemble of Intelligent Paradigms* Journal of Network and Computer Applications, vol. 28, no. 2, pp. 167-182, Elseiver, 2005

[17] W. Lee, S. Stolfo and K. Mok, *A Data Mining Framework for Building Intrusion Detection Models* Proceedings of the IEEE symposium on security and privacy, pp. 120-132, 1999

[18] A.H. Sung and S. Mukkamala, *Identifying Important Features for Intrusion Detection Using Support Vector Machines and Neural Networks* Proceedings of International Symposium on Applications and the Internet (SAINT 2003), vol. 217, pp. 209-217, 2003

[19] A. Chebrolu, A. Abraham and J. Thomas, *Feature Deduction and Ensemble Design of Intrusion Detection Systems.* Computers & Security, vol. 24, no. 4, pp. 295-307, Elseiver, 2005
[20] M. Fugate and J.R. Gattiker, *Computer Intrusion Detection with Classification and Anomaly Detection, using SVMs.* International Journal of Pattern Recognition and Artificial Intelligence, vol. 17, no. 3, pp. 441-458, 2003
[21] A. Alonso-Betanzos, N. Sanchez-Maroño, F.M. Carballal-Fortes, J. Suarez-Romero and B. Perez-Sanchez, *Classification of Computer Intrusions Using Fuctional Networks. A Comparative Study.* ESANN '07: Proceedings of the European Symposium on Artificial Neural Networks, pp. 25-27, 2007
[22] I. Guyon, S. Gunn, M. Nikravesh and L. Zadeh, *Feature Extraction. Foundations and Applications.* Springer, 2006
[23] W.H. Press, B.P Flannery, S.A Teukolsky and W.T. Vetterling, *Numerical recipes in C.* Cambridge University Press, Cambridge, 1988
[24] H. Liu and R. Setiono, *A Probabilistic Approach to Feature Selection - a Filter Solution.* Proceedings of the 13th Internation Conference on Machine Learning, pp. 319-327, Morgan Kaufmann, 1996
[25] H. Liu, F. Hussain, C.L. Tan and M. Dash, *Discretization: An Enabling Technique.* Data Mining and Knowledge Discovery Journal, vol. 6, no. 4, pp. 393-423, Springer, 2002
[26] D. Janssens, T. Brijs, K. Vanhoof and G. Wets, *Evaluating the Performance of Cost-based Discretization versus Entropy and Error-based Discretization.* Computers and Operations Research Journal, vol. 33, no. 11, pp. 3107-3123, Elsevier, 2006
[27] Y. Yang and G.I. Webb, *A Comparative Study of Discretization Methods for Naive-Bayes Classifiers.* Proceedings of PKAW 2002: The 2002 Pacific Rim Knowledge Acquisition Workshop, pp. 159-173, 2002
[28] J. Dougherty, R. Kohavi and M. Sahami, *Supervised and Unsupervised Discretization of Continuous Features.* Proceedings of the 12th International Conference on Machine Learning, vol. 202, pp. 194-202, Morgan Kaufmann, 1995
[29] S.D Bay, *The UCI KDD Archive* `http://kdd.idc.uci.edu` [Last access: 1st December 2008] Department of Information and Computer Science, University of California, Irvine, 1999
[30] I.H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques.* 2nd Edition, Morgan Kaufmann, San Francisco, 2005 `http://www.cs.waikato.ac.nz/ml/weka/` [Last access: 1st December 2008]
[31] THE MATHWORKS, *Matlab, user's guide.* `http://www.mathworks.com/` [Last access: 1st December 2008]
[32] S. Axelsson, *The Base-Rate Fallacy and its Implications for the Difficulty of Intrusion Detection.* Proceedings of the 6 th ACM Conference on Computer and Communications Security, vol. 1, no. 4, pp. 1-7, 1999