

# Chapter 1

## Continuous Features Discretizaion for Anomaly Intrusion Detectors Generation

Amira Sayed A.Aziz, Ahmad Taher Azar, Aboul Ella Hassanien, Sanaa Al-Ola Hanafy

**Abstract** Network security is a growing issue, with the evolution of computer systems and expansion of attacks. Biological systems have been inspiring scientists and designs for new adaptive solutions, such as genetic algorithms. In this paper, an approach that uses the genetic algorithm to generate anomaly network intrusion detectors is used. An algorithm is proposed using a discretization method for the continuous features selection of intrusion detection, to create some homogeneity between values, which have different data types. Then, the intrusion detection system is tested against the NSL-KDD data set using different distance methods. A comparison is held amongst the results, and it is shown by the end that this proposed approach has good results, and recommendations are given for future experiments.

### 1.1 Introduction

With the evolution of computer networks during the past few years, security is a crucial issue and a basic demand for computer systems. Attacks are expanding and evolving as well, making it important to come up with new and advanced solutions for network security. Intrusion Detection Systems (IDS) have been around us for

---

Amira Sayed A.Aziz  
 French University in Egypt (UFE), Shorouk City, Egypt.; Scientific Research Group in Egypt (SRGE), Cairo, Egypt, e-mail: amira.abdelaziz@egyptscience.net

Ahmad Taher Azar  
 Misr University for Science & Technology (MUST), 6th of October City, Egypt.; Scientific Research Group in Egypt (SRGE), Cairo, Egypt, e-mail: ahmed.t.azar@yahoo.com

Aboul Ella Hassanien  
 Chairman of Scientific Research Group in Egypt (SRGE), Cairo, Egypt, e-mail: abo@egyptscience.net

Sanaa Al-Ola Hanafy  
 Cairo University, Faculty of Computers and Information, Cairo, Egypt.

a some time, as an essential mechanism to protect computer systems, where they identify malicious activities that occur in that protected system. Genetic Algorithms (GA) are a group of computational models inspired by natural selection [1][2]. This solution works on a group of chromosomes-like data structure (a population) where they reproduce new individuals that would be more fitting in the environment. These new generations are developed using selection and recombination functions such as crossover and mutation [3].

The GAs were first seen as optimization solutions, but now they are applied in a variety of systems, including the IDSs [4][5]. The GA is used as a machine learning technique to generate artificial intelligence detection rules. The rules are usually in the if-then forms, where the conditions are values that represent normal samples or values to indicate an intrusion is in the act [3][6]. For a Network-based IDS (NIDS), usually the network traffic is used to build a model and detect anomalous network activities. Many features can be extracted and used in a GA to generate the rules, and these features may be of different data types, and may have a wide range of values. So, this paper presents an approach that uses a discretization algorithm with continuous features to create homogeneity amongst features.

Discretization is simply a process of converting continuous attributes to discrete ones by partitioning them into intervals. Data sets used in intrusion detection systems are high-dimensional, hence discretization is needed as a preprocessing step before applying clustering, feature selection, training...etc processes on the data. Limited use and research was held concerning when to use which discretization algorithms in IDS. More details about discretization algorithms can be found in [7], [8], and [9].

The rest of this paper is organized as follows: Section 1.2 gives a background of the different algorithms used in this approach. Section 1.3 gives a review on some of the previous work done in the area. Section 1.4 describes the proposed approach, Section 1.5 gives an overview of the experimental analysis and results. And finally in Section 1.6, conclusion and directions for future research are presented.

## 1.2 Background

### 1.2.1 Anomaly Intrusion Detection

Intrusion Detection Systems (IDSs) are security tools used to detect anomalous or malicious activity from inside and outside intruders [10]. An IDS can be host-based or network-based, which is the concern in this paper [11]. They are classified by many axes, one of them is the detection methodology that classifies them to signature-based and anomaly-based IDS. The former detects attacks by comparing the data to patterns stored in a signature database of known attacks. The later detects anomalies by defining a model of normal behaviour of the monitored system, then considers any behaviour lying outside the model as anomalous or suspicious

activity. Signature-based IDS can detect well-known attacks with high accuracy but fails to detect or find unknown attacks. Anomaly-based IDS has the ability to detect new or unknown attacks but usually has high false positives rate (normal activities detected as anomalous). There are three types of anomaly detection techniques: statistical-based, knowledge-based, and machine learning-based [12]. IDS performance can be measured by two key aspects: the detection process efficiency and the involved cost of the operation [12].

### ***1.2.2 Genetic Algorithms***

Genetic Algorithm (GA) is an evolutionary computational technique that is used as a search algorithm, based on the concepts of natural selection and genetics. There are 3 meanings of search: 1- Search for stored data: where the problem is to retrieve some information stored in a computer memory efficiently. 2- Search for paths to goals: where one needs to find the best paths from an initial state to a goal. 3- Search for solutions: where one needs to find a solution or group of solutions in a large space of candidates. GA works on a population of individuals, where each individual is called a chromosome and is composed of a string of values called genes. The population goes through a process to find a solution or group of high quality solutions. The quality of an individual is measured by a fitness function that is dependant on the environment and application. The process starts with an initial population, that goes through transformation for a number of generations. During each generation, three major operations are applied sequentially to each individual: selection, crossover, and mutation until target is met [13][6].

### ***1.2.3 Negative Selection Approach***

Artificial Immune Systems (AIS) are inspired by the nature's Human Immune System (HIS), which is an adaptive, tolerant, self-protecting, and dynamic defence system [14]. AIS is a set of algorithms that mimic the different functionalities of the HIS, and they can perform a range of tasks. The major algorithms are: negative selection, clonal selection, and immunity networks. The Negative Selection Approach (NSA) is based on the concept of self-nonself discrimination, by first creating a profile of the self (normal) behaviour and components. Then use this profile to rule out any behaviour that doesn't match with that profile. The training phase goes on the self samples. And then, the detectors are exposed to different samples, and if a detector matches a self as nonself then it's discarded. The final group of detectors (mature detectors) are released to start the detection process [15][16].

### 1.2.4 Equal-Width Binning Algorithm

There are many algorithms for continuous features discretization for algorithmic purposes. These discretization algorithms are very important for machine learning, as it is required by some algorithms. But more importantly, the discretization increases the speed of induction algorithms. The discretization algorithms are classified in many ways: Local or Global, Supervised or Unsupervised Static or Dynamic, Top-down or Bottom-up, and Direct or Incremental [17] [18] [19].

Local methods apply partitions on localized regions of the instance space, where Global methods works on the entire instance space, that every feature is partitioned into number of regions independent of other attributes. Supervised methods make use of the class labels associated with the instances in the process, while unsupervised methods perform discretization regardless of class label. Static or Dynamic, where Static methods determine the number of bins for each feature independent of other features after performing one discretization pass of the data (performed before the classification). Dynamic methods determine the number of bins for all features simultaneously by a search through the data space (performed while the classifier is built). Top-down(Splitting) or Bottom-up (Merging), where Splitting methods start with an empty group of cut points, and build up during the discretization process. While in merging, the algorithm starts with a list of cut points, then discards unneeded ones during discretization by merging intervals. Direct or Incremental, where In direct methods, number of bins (intervals) is predefined either by user or using an algorithm. Incremental methods start with simple discretization that gets improved and refined until stopped by a condition (meeting a certain criterion).

### 1.3 Related Work

There have been several studies reported focusing on discretization algorithms [19] [20]. Dougherty et al. (1995) [17] Applied EWB, 1R, and Recursive Entropy Partitioning as preprocessing step before using C4.5 and Naive-Bayes classifiers on data. The data set they used was 16 data sets from the UC Irvine (UCI) Repository. C4.5 performance improved on 2 data sets using entropy discretization, but slightly decreased on some. At 95 % confidence level, Naive-Bayes with entropy discretization is better than C4.5 on 5 data sets and worse on 2 (with average accuracy 83.97% vs. 82.25% for C4.5). Clarke and Barton (2000) [21] applied Minimum Descriptive Length (MDL) to select number of intervals, and modified a version of the k2 method for one test, entropy based discretization for another test. They ran their experiment on NGHS and DISC from two epidemiological studies. The Dynamic Partitioning with MDL metric lead to more highly connected BBN than with only entropy partitioning. New proposed method lead to better representations of variable dependencies in both data sets. But generally, using entropy and MDL partitioning provided clarification and simplification in the BBN. Zhao and Zhou (2009) [22] suggested a rough set based heuristic method, enhanced in two ways: (1) decision

information is used in candidate cut computation (SACC), and (2) an estimation of cut selection probability is defined to measure cut significance (ABSP). The experiment was applied on continuous UCI data sets. Their SACC was compared to an algorithm known as UACC, and ABSP was compared to some typical rough set based discretization algorithm. SACC performs better with less number of cuts, and ABSP slightly improves predictive accuracies.

Gupta et al. (2010) [18] applied K-means clustering with euclidean distance similarity metric, and Shared Nearest Neighbour (SNN). MDL was used for discretization with  $\alpha=\beta=0.5$  (ME-MDL), and was applied on 11 data sets from UCI repository. Comparing their algorithm results with ME-MDL results: in all data sets, when SNN or k-means clustering was used, the proposed algorithm gave better results. In heart data set, SNN clustering performed better than ME-MDL. In other data sets, k-means was better than SNN. Joita (2010) [23] proposed a discretization algorithm based on the k-means clustering algorithm, that avoids the  $O(n \log n)$  time required for sorting. The algorithm was proposed to be tested in the future. Chen et al. (2011) [24] proposed an improved method of continuous attributes discretization by: (1) hierarchical clustering was applied to form initial division of the attribute, and (2) merging adjacent ranges based on entropy, taking into consideration not to affect level of consistency of the decision table. They used data of their provincial educational committee project for the experiment. The results were not listed, but mentioned to prove the validity of their algorithm. Ferreira and Figueirdo (2012) [25] used clustering with discretization for better results. Updated versions of the well-know Linde-Buzp-Gray (LBG) algorithm were proposed: U-LBG1 (used a variable number of bits) and U-LBG2 (used a fixed number of bits). For clustering, Relevance-Redundancy Feature Selection (RRFS) and Relevance Feature Selection (RFS) methods were used. Data sets from UCI, the five data sets of the NIP2003 FS challenge, and several micro-array gene expression data sets were used for their experiments (no normalization was applied on any of the used data sets). The proposed approaches allocated a small number of bits per feature. RRFS performs better than RS for eliminating redundant features.

## 1.4 Proposed Approach

### 1.4.1 Motivation

In [26] the algorithm was originally suggested with the application on real-valued features in the NSL-KDD data set. It was used with a variation parameter defining the upper and lower limits of the detectors values (conditions). It had very good results, but the real-valued features are not enough to detect all types of attacks, so the algorithm should expand to include features of different types. In [27], the algorithm was applied on the KDD data set, using a range of features to detect anomalies. The problem with using different features is that they have different data

types ranges: binary, categorical, and continuous (real and integer). This may lead to problems while applying the algorithm. First of all, a wide range of values need to be covered in a way that can represent each region uniquely. Secondly, there should be some sort of homogeneity between features values to apply the GA. So, the use of some discretization algorithm for continuous features lead to the suggestion of the following approach.

#### 1.4.2 Suggested Approach

Equal-width interval binning [17] is the simplest method for data discretization, where the range of values is divided into  $k$  equally sized bins, as  $k$  is a parameter supplied by the user as the required number of bins. The bin width is calculated as:

$$\delta = \frac{x_{max} - x_{min}}{k} \quad (1.1)$$

and the bin boundaries are set as:  $x_{min} + i\delta$ ,  $i=1, \dots, k-1$ .

The equal-width interval binning algorithm is a global, unsupervised, and static discretization algorithm. The suggested approach starts with binning the continuous features with a previously defined number of bins, Then, replace each feature value with its enclosing bin number. Finally, run the GA on the modified data set samples to generate the detectors (rules). Following the NSA concepts, this is applied on the normal samples through the training phase. The self samples are presented in the self space  $S$ . The process is shown in Algorithm I.

---

#### Algorithm 1 Proposed Algorithm

---

- 1: Run equal-width binning algorithm on continuous features.
  - 2: Initialize population by selecting random individuals from the space  $S$ .
  - 3: **for** The specified number of generations **do**
  - 4:     **for** The size of the population **do**
  - 5:         Select two individuals (with uniform probability) as  $parent_1$  and  $parent_2$ .
  - 6:         Apply crossover to produce a new individual ( $child$ ).
  - 7:         Apply mutation to child.
  - 8:         Calculate the distance between  $child$  and  $parent_1$  as  $d_1$ , and the distance between  $child$  and  $parent_2$  as  $d_2$ .
  - 9:         Calculate the fitness of  $child$ ,  $parent_1$ , and  $parent_2$  as  $f$ ,  $f_1$ , and  $f_2$  respectively.
  - 10:         **if** ( $d_1 < d_2$ ) and ( $f > f_1$ ) **then**
  - 11:             replace  $parent_1$  with  $child$
  - 12:         **else**
  - 13:             **if** ( $d_2 \leq d_1$ ) and ( $f > f_2$ ) **then**
  - 14:                 Replace  $parent_2$  with  $child$ .
  - 15:             **end if**
  - 16:         **end if**
  - 17:     **end for**
  - 18: **end for**
  - 19: Extract the best (highly-fitted) individuals as your final solution.
-

The fitness - which was inspired from [28] - is measured by calculating the matching percentage between an individual and the normal samples, as:

$$fitness(x) = \frac{a}{A} \quad (1.2)$$

where  $a$  is the number of samples matching the individual by 100% , and  $A$  is the total number of normal samples. Three different distance methods were tested (one at a time), to find the best results. The distances measured between a child  $X$  and a parent  $Y$  using the following formulas:

- The Euclidean distance as:

$$d(X, Y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 \dots (x_n - y_n)^2} \quad (1.3)$$

- The Hamming distance, which defines the difference between 2 strings (usually binary) as the number of places in which the strings have different values[29]. So it's calculated as (where  $n$  is number of features):

$$d(X, Y) = \sum_{i=0}^n |(x_i - y_i)| \quad (1.4)$$

- The Minkowski Distance, which is similar to the Euclidean distance but uses the  $p$ -norm dimension as the power value instead. So, the formula goes as:

$$d(X, Y) = \left( \sum_{i=0}^n (|x_i - y_i|^p) \right)^{1/p} \quad (1.5)$$

In the Minkowski distance case,  $p$  can be any value larger than 0 and up to infinity. It can be have real value between 0 and 1. If we are interested in finding the difference between objects, then we should aim for high  $p$  values. If we are interested in finding the how much the objects are similar, then we should go for low  $p$  values [30]. In our experiment, a small value of 0.5 was used, and a big values of 18 was used to compare results.

## 1.5 Experiment

### 1.5.1 Data Set

The NSL-KDD IDS data set [31] was proposed in [32] to solve some issues in the widely use KDD Cup 99 data set. These issues affect the performance of the systems that use the KDD data set and results in very poor evaluation of them. The resulted

data set is having a reasonable size and is unbiased, and it's affordable to use in the experiments without having to select a small portion of the data. The data sets used in our experiment are:

- KDD Train+\_20 Percent normal samples for training and generating the detectors.
- KDD Train+ and KDDTest+ for testing, where the difference between them is that the Test set include additional unknown attacks that are not included in the Train set.

### 1.5.2 Experiment Settings

In the proposed approach, the features were selected as in [17], which are shown in Table 1.1. Ports classification was performed manually, and they were classified into 9 categories as in [27]. The procedure was done manually because it is dependent on the network and system settings more than number ranges.

**Table 1.1** Features Selected from NSL-KDD Data Set

Feature	Data Type	No. of Bins
duration	Integer	8
protocol_type	Categorical	N/A
service	Integer	9
land	Binary	N/A
urgent	Integer	1
host	Integer	3
num_failed_logins	Integer	3
logged_in	Binary	N/A
root_shell	Binary	N/A
su_attempted	Binary	N/A
num_file_creations	Integer	4
num_shells	Integer	2
is_host_login	Binary	N/A
is_guest_login	Binary	N/A
count	Integer	10
same_srv_rate	Real	3
diff_srv_rate	Real	3
srv_diff_host_rate	Real	3

The values used for the GA parameters are summarized in Table 1.2:

Different values of population size and number of generations were used to compare the results to see which would lead to better results, and threshold value of 0.8 was used for the experiment.



**Table 1.2** GA Parameters

Population size	200, 400, 600
Number of generations	200, 500, 1000, 2000
Mutation rate	$2/L$ , where L is the number of features.
Crossover rate	1.0

### 1.5.3 Results

After running the algorithm on the train set normal samples, varieties of detectors (rules) were obtained, based on population size and number of generations. Running those detectors on the test set, the detection rates are shown in figure 1.1

**Fig. 1.1** Detection Rates

As shown in figure 1.1, the detection rates are all above 75 % - as the maximum detection rates realized are 81.93% and 81.54% obtained by the detectors generated by GA applying Euclidean and Minkowski (p=18) distances respectively. The overall average detection rates are 79.06, 78.62, 79.25, and 79.18% obtained using Euclidean, Hamming, and Minkowski distance functions respectively, and in most cases the Minkowski distance gave better performance. The rates obtained with detectors generated using population size 200 are generally better. To measure the IDS efficiency, true positives and true negatives rates (TPR and TNR respectively) are calculated and shown in figures 1.2 and 1.3.

Figure 1.2 indicated that the recognition of normal samples is high as the TPRs are mostly above 90% with low FPRs — all less than 9%. In figure 1.3, TNRs lie between 60% and 80%, which means that at least 20% of the attacks are detected as normal. Better TNRs are obtained using detectors generated using bigger population

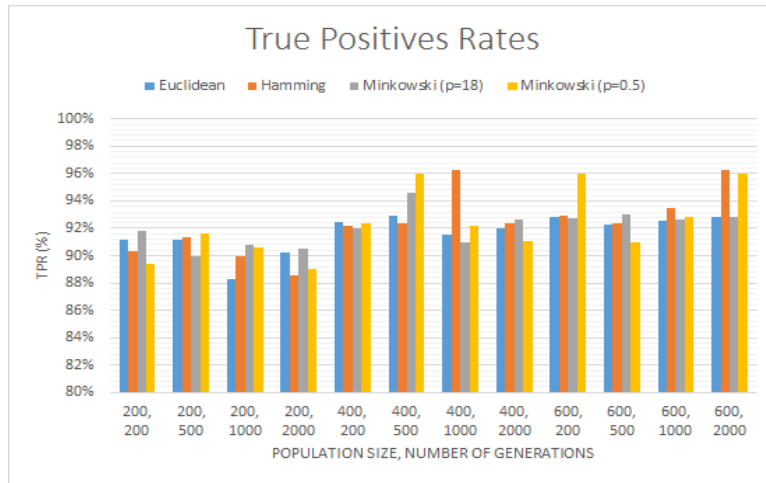


Fig. 1.2 True Positives Rates

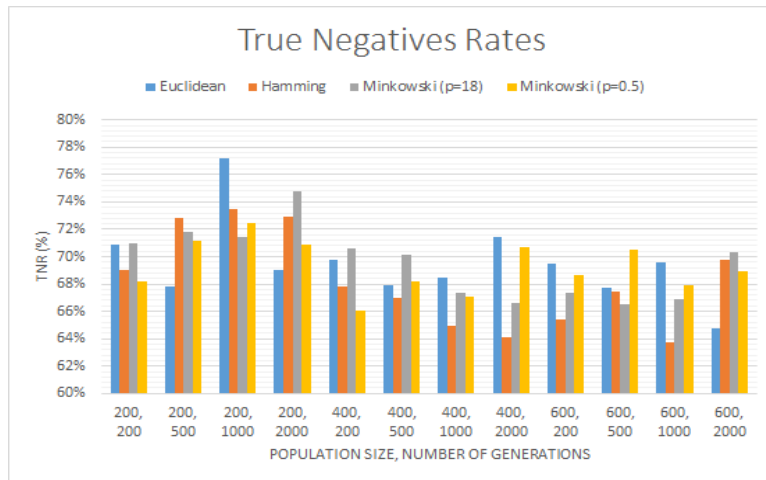


Fig. 1.3 True Negatives Rates

size, mostly with the Minkowski distance. Detectors generated using bigger populations give higher TPRs, while higher TNRs are obtained with detectors generated by smaller populations. Other experiments similar to this one use classifiers for multiple classification detection, but 2-classes detection (normal/anomaly) is applied in the algorithm. So, a real comparison can't be held in the moment until a multi-class classifier is applied for more precise results.

## 1.6 Conclusion and Future Work

In the paper, an algorithm is implemented to generate detectors that should be able to detect anomalous activities in the network. The data was pre-processed before using them in the algorithm, by discretizing the continuous features to create homogeneity between data values, and then replacing values with bin numbers. The results indicated that the equal-width interval binning algorithm that was used is very simple and has good results. As for the parameters of the GA, the detectors generated by GA with smaller population size gave better detection rates and true alarms than others generated using higher population sizes. The advantage of using smaller population sizes are: (1) less time-consuming while generating the detectors, and (2) less number of detectors are generated. The detectors generated using the Minkowski distance gave better results in most cases than those generated using the Euclidean distance (which is widely used) and the Hamming distance. Future work will be focused on applying other discretization algorithms that are more dynamic. Also, using classifiers to increase detection accuracy, and be able to define which type of anomalies have been detected.

## References

1. Haupt RL, Haupt SE (2004) Practical Genetic Algorithms, Wiley, 2nd Edition.
2. Polhlheim H (2006) Genetic and Evolutionary Algorithms: Principles, Methods and Algorithms. [Online] Available: <http://www.geatbx.com/docu/index.html>.
3. Whitley D (1994) A Genetic Algorithm Tutorial, *Statistics and Computing*, 4: 65-85.
4. Owais S, Snasel V, Abraham A (2008) Survey: Using Genetic Algorithm Approach in Intrusion Detection Systems Techniques, 7th Computer Information Systems and Industrial Management Applications, Ostrava, 26-28:300 - 307. DOI: 10.1109/CISIM.2008.49..
5. Li W (2004) Using Genetic Algorithm for Network Intrusion Detection, Proceedings of the United States Department of Energy Cyber Security Group.
6. Nitchell M (1998) An Introduction to Genetic Algorithms, MIT Press Cambridge, MA, USA. ISBN:0262631857.
7. Sengupta N and Sil J (2012) Evaluation of Rough Set Theory Based Network Traffic Data Classifier Using Different Discretization Method, *IJIEE*, 2(3):338-341.
8. Wa'el MM, Agiza HN, Radwan E (2009) Intrusion Detection Using Rough Sets based Parallel Genetic Algorithm Hybrid Model, Proceedings of the World Congress on Engineering and Computer Science 2009 (WCECS 2009), San Francisco, USA, II.
9. Ertoz, L, Eilertson, E, Lazarevic, A, Tan, PN, Kumar V, Srivastava, J, Dokas, P (2004) MINDS - Minnesota Intrusion Detection System. In *Data Mining - Next Generation Challenges and Future Directions*, MIT Press.
10. Aleksandar L, Vipin K, Jaideep S (2005) Intrusion Detection: A Survey, In: Vipin Kumar et al. (eds), *Managing Cyber Threats Issues, Approaches, and Challenges*, 5: 19-78.
11. A. Murali, M. Roa (2005) A survey on intrusion detection approaches, *Information and Communication Technologies, ICICT. 2005. First International Conference on.*, pp. 233-240, Aug 2005.
12. Garcia-Teodora P, Daz-Verdejo J, Maci-Fernandez G, Vzquez E (2009) Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1-2): 18-28.

- 12 Amira Sayed A.Aziz, Ahmad Taher Azar, Aboul Ella Hassanien, Sanaa Al-Ola Hanafy
13. Akbar S, Chandulal JA, Rao KN, Kumar S (2011) Troubleshooting Techniques for Intrusion Detection System using Genetic Algorithm. *International Journal of Wisdom Based Computing*, 1(3): 86-92.
  14. Dasgupta D (2006) Advances in artificial immune systems. *IEEE Computational Intelligence Magazine* 1(4): 40-49.
  15. Greensmith J, Whitbrook A, Aickelin U (2010) *Artificial Immune Systems, Handbook of Metaheuristics, International Series in Operations Research and Management Science*, Springer, Springer US, 146: 421-448.
  16. Aickelin U, Greensmith J, Twycross J (2004) Immune system approaches to intrusion detection - a review, In Proc. of the 3rd International Conference on Artificial Immune Systems (ICARIS), LNCS 3239, 316-329.
  17. Dougherty J, Kohavi R, Sahami M (1995) Supervised and Unsupervised Discretization of Continuous Features, *Machine Learning: Proceedings of the Twelfth Conference*, 95(10): 194-202.
  18. Gupta A, Mehrotra K, Mohan C (2010) A Clustering-based Discretization of Supervised Learning, *Statistics & Probability Letters*, Elsevier, 80(910): 816-824.
  19. Liu H, Hussain F, Tan CL, Dash M (2002) Discretization: An Enabling Technique, *Data Mining and Knowledge Discovery*, 6(4): 393-423.
  20. Kotsiantis S, Kanellopoulos D (2006) Discretization Techniques: A recent survey, *GESTS International Transactions on Computer Science and Engineering*, 32(1): 47-58.
  21. Clarke EJ; and Barton BA (2000) Entropy and MDL discretization of continuous variables for Bayesian belief networks, *International Journal of Intelligent Systems*, 15(61): 61-92.
  22. Zhao J, Zhou Y (2009) New heuristic method for data discretization based on rough set theory. *The Journal of China Universities of Posts and Telecommunications*, 16(6): 113-120.
  23. Joita D (2010) Unsupervised Static Discretization Methods in Data Mining, *Revista Mega Byte*, 9.
  24. Chen S, Tang L, Liu W, Li Y (2011) A Improved Method of Discretization of Continuous Attribute, 2011 2nd International Conference on Challenges in Environmental Science and Computer Engineering (CESCE 2011), Elsevier, 11(A): 213-217.
  25. Ferreira AJ and Figueiredo MA (2012) An unsupervised approach to feature discretization and selection, *Pattern Recognition*, 45(9): 3048-3060.
  26. A.Aziz AS, Salama M, Hassanien AE, Hanafi SO (2012) Detectors Generation using Genetic Algorithm for a Negative Selection Inspired Anomaly Network Intrusion Detection System, In proceeding of: *IEEE FedCSIS, Wroclaw, Poland*, 625-663, ISBN:978-83-60810-51-4.
  27. Powers ST and He J (2008) A Hybrid Artificial Immune System and Self-Organizing Map for Network Intrusion Detection, *Information Sciences: an International Journal*, Elsevier, 178(15): 3024-3042.
  28. Goyal A and Kumar C (2007) GA-NIDS: A Genetic Algorithm based Network Intrusion Detection System, A Project at Electrical Engineering and Computer Science, Northwestern University, Evanston, IL, <http://www.cs.northwestern.edu/~ago210/ganids/>.
  29. He MX, Peroukhov SV, Ricci PE (2004) Genetic Code, Hamming Distance, and Stochastic Matrices, *Bulletin of Mathematical Biology*, 66(5): 1405-1421, DOI: 10.1016/j.bulm.2004.01.002.
  30. Kotnarowski M. (2010) Measurement of distance between voters and political parties - Different approaches and their consequences, 3rd ECPR Graduate Conference, Dublin.
  31. NSL-KDD data set, <http://nsl.cs.unb.ca/NSL-KDD/>
  32. Tavallaee M, Nagheri E, Lu W, Ghorbani AA (2009) A detailed analysis of the KDD Cup 99 Data Set, In Proceedings of the 2009 IEEE Symposium Computational Intelligence for Security and Defense Applications, CISDA09.