Genetic Programming approach for multi-category pattern classification applied to network intrusions detection

K. M. Faraoun[1*] and A. Boukelif[2]

[1] Evolutionary Engineering and Distributed Information Systems Laboratory, EEDIS. Djillali Liabes University. Sidi Bel Abbes - Algeria

[2] Laboratoire des telecommunications et du traitement numerique de signal, equipe de recherche des techniques videes. Djillali Liabes University. Sidi Bel Abbes - Algeria.

Abstract:

The present paper describe a new approach of classification using genetic programming. The proposed technique consist of genetically coevolving a population of non-linear transformations on the input data to be classified, and map them to a new space with a reduced dimension, in order to get a maximum inter-classes discrimination. The classification of new samples is then performed on the transformed data, and so become much easier. Contrary to the existing GP-classification techniques, the proposed one use a dynamic repartition of the transformed data in separated intervals, the efficacy of a given intervals repartition is handled by the fitness criterion, with a maximum classes discrimination. Experiments were first performed using the Fisher's Iris dataset, and then, the KDD'99 Cup dataset was used to study the intrusion detection and classification problem. Obtained results demonstrate that the proposed genetic approach outperform the existing GP-classification methods and give a very accepted results compared to other existing techniques.

Keywords:

Genetic programming, patterns classification, intrusion detection

Introduction

Pattern classification concepts are important in the design of computerized information processing systems for many applications such as remote sensing, medical diagnosis, sonar, radar etc. Pattern classification involves the development of theory and techniques for the categorization of input data into identifiable classes [25]. A pattern class is a category determined by some common attributes. A pattern is the description of any member of a category representing a pattern class. The application determines the measurement of features. Classification typically involves the mapping of an N-dimensional feature vector to one of multiple classes. The N-dimensional feature vector is like a point in the N-dimensional feature space. The samples belonging to a particular class give rise to a data distribution of that class in some region of the feature space.

[*] Corresponding Author: Faraoun Kamel Mohamed
Tel: (+213) 75 32 36 50; Fax: (+213) 48 57 77 50; E-mail: Kamel_mh@yahoo.fr

It is possible for data distributions of two classes to be either overlapping or non-overlapping in the feature space. A pattern classifier determines the decision boundaries between different classes. The complexity of these boundaries may range from linear to non-linear surfaces. The significance of decision boundaries lies in the fact that they can usually be generated by utilizing representative patterns from each class. The pattern classifier uses these decision boundaries and determines the class for a new pattern. In the present work, we consider the problem of classifying real number vectors form R^N, where N is the features number of a given pattern.

The basic problem in pattern classification is to develop decision functions that partition the feature space into regions each of which contains sample patterns belonging to a class.

Intrusions in computer networks can be traced and detected by collecting information about the traffic in and out of the network. From a pattern classification point of view, the network intrusion detection problem can be formulated as follows: given the information about network connections between pairs of hosts, assign each connection to one out of N data classes representing normal traffic or different categories of intrusions (e.g., Denial of Service, access to root privileges, etc.). It is worth noting that various definitions of data classes are possible. The term "connection" refers to a sequence of data packets related to a particular service, e.g., the transfer of an image via the ftp protocol. The intrusion detection problem can then be viewed as a Multi-category pattern classification problem, when each connection features constitute one pattern to be assigned to one of the N existing classing (depending on the number of intrusions types taken into account).

In this paper, an attempt is made to show the use of a new GP-classification approach to perform network intrusion detection. Section 1 gives some theoretic background about genetic programming approaches and related works. The section 2 explains the method developed in the present work with its different elements and parameters. In the section 3, we give a description of the two datasets used for experiments and the codification of the different data elements. Section 4 summarizes the different obtained results and gives a comparison with the other approaches with discussions. Enhancements of the proposed method are explained in the section 5. The paper is finally concluded with a summary of the most important points and future works.

1. Theory

1.1. Genetic Programming paradigm

Genetic programming is an extension of genetic algorithms [9]. It is a general search method that uses analogies from natural selection and evolution. In contrast to GA, GP encodes multi-potential solutions for specific problems as a population of programs or functions. The programs can be represented as parse trees. Usually, parse trees are composed of internal nodes and leaf nodes. Internal nodes are called primitive functions, and leaf nodes are called terminals. The terminals can be viewed as the inputs to the specific problem. They might include the independent variables and the set of constants. The primitive functions are combined with the terminals or simpler function calls to form more complex function calls.

GP randomly generates an initial population of solutions. Then, the initial population is manipulated using various genetic operators to produce new populations. These operators include reproduction, crossover, mutation, dropping condition, etc. The whole process of evolving from one population to the next population is called a generation. A high-level description of GP algorithm can be divided into a number of sequential steps:

- Create a random population of programs, or rules, using the symbolic expressions provided as the initial population.
- Evaluate each program or rule by assigning a fitness value according to a predefined fitness function that can measure the capability of the rule or program to solve the problem.
- Use reproduction operator to copy existing programs into the new generation.
- Generate the new population with crossover, mutation, or other operators from a randomly chosen set of parents.
- Repeat steps 2 onwards for the new population until a predefined termination criterion has been satisfied, or a fixed number of generations have been completed.
- The solution to the problem is the genetic program with the best fitness within all the generations.

In GP, crossover operation is achieved firstly by reproduction of two parent trees; two crossover points are then randomly selected in the two offspring trees. Exchanging sub-trees, which are selected according to the crossover point in the parent trees, generates the final offspring trees. The obtained offspring trees are usually different from their parents in size and shape.

Mutation operation is also considered in GP. A single parental tree is firstly reproduced. Then a mutation point is randomly selected from the reproduction, which can be either a leaf node or a subtree. Finally, the leaf node or the sub-tree is replaced by a new leaf node or sub-tree generated randomly. Fitness functions ensure that the evolution is toward optimization by calculating the fitness value for each individual in the population. The fitness value evaluates the performance of each individual in the population.

1.2. Genetic Programming and Classification Task

Generally, GP trees can perform classification by returning numeric (real) values and then translating these values into class labels [10]. For binary classification problems the division between negative and non-negative numbers acts as a natural boundary for a division between two classes. This means that genetic programs can easily represent binary class problems. While evaluating the GP expression for an input data, if the result of the GP-expression is > 0, the input data is assigned to one class; else it is assigned to the other class. Thus, the desired output D is +1 for one class and is -1 for the other class in the training set. Hence, the output of a GP-expression is either +1 (indicating that the input data belong to that class) or -1 (indicating that the input sample dose not belong to that class). During the genetic evolution of individuals, the best individual is those who correctly classify the maximum of training samples, the positive samples must give a value of +1 for the output, and negative samples must give -1.

Given a set of training data $D_{Train}=\{X_1,X_2......X_p\}\subset R^N$, a binary classifier is a GP-expression T, so that:

$$T(X_i) < 0 \text{ if } X_i \text{ e Class 1 } (D=+1)\ T(X_i) > 0 \quad \text{otherwise} \quad (D=-1) \qquad (1)$$

GP is guided by the fitness function to search for the most efficient computer program to solve a given problem. A simple measure of fitness has been adopted for the binary classification problem:

$$\text{Fitness(T)}=\frac{\text{Number of samples classified correctly}}{\text{Number of samples used for training during evolution}} \qquad (2)$$

Each genetic expressing evolved map the samples space of the X_i's, to the real numbers set R, and attribute the interval]-∞, 0] to the class 1 and the interval]0,+∞ [to the class 2. This mapping is static, but it can achieve good results for 2-category classification problems. Unfortunately, when more than two classes are involved (n-classes problem), finding meaningful division points over the set of reals the genetic programs return is more difficult. If boundary regions are chosen at arbitrary points over the set of reals then genetic programs face the problem of not only containing the necessary elements to distinguish between classes, but also must perform a translation task to provide output in the necessary range pre-specified for a given class. Many alternatives were proposed by many authors to solve this problem. In [11], if there are n classes in a classification task, these classes are sequentially assigned n regions along the numeric output value space from some negative numbers to positive numbers by (n-1)*thresholds/boundaries. Class 1 is allocated to the region with all numbers less than the first boundary; class 2 is allocated to all numbers between the first and the second boundaries and class n to the region with all numbers greater than the last boundary n-1, as shown in the following:

$$\text{Classe } (X_i) = \begin{cases} \text{classe 1} & \text{if } T(X_i) \le b_1 \\ \text{classe 2} & \text{if } b_1 \le T(X_i) \le b_2 \\ \cdots\cdots & \\ \text{classe n - 1} & \text{if } b_{n-3} \le T(X_i) \le b_{n-2} \\ \text{classe n} & \text{if } b_{n-2} \le T(X_i) \le b_{n-1} \end{cases} \qquad (3)$$

In this equation, n refers to the number of object classes, T is the GP-expression evolved, $T(X_i)$ is the output value, and b_1, b_2, b_{n-1}, b_n are static, pre-defined classes boundaries.

An alternative approach to static range selection, where ranges are arbitrarily chosen to correspond to class boundaries that all programs for the run must adhere to, is to allow each program to use a separate set of ranges for class boundaries that are dynamically determined for each individual program. Given a classification problem with many training examples and an individual from a GP population it is possible to use a subset of the training examples and record the values that are returned when attributes for specific classes are used as inputs. Based upon these outputs the effectively infinite range of the reals can then be segmented into regions corresponding to class boundaries based upon areas the program has returned values for each class in the subset of training examples, this method was implemented in [11].

The GP employed for classification tasks do however have a requirement for long training times when compared to many other classification methods. It is also often quite difficult to extract a meaningful reason as to why a given class was chosen. Because of these factors the GP method is seen to be applicable to tasks where accuracy is the most important factor in classification, and training times and understand ability are seen as relatively unimportant. The major considerations in applying GP to pattern classification are:

- GP-based techniques are data distribution-free, so no a priori knowledge is needed about statistical distribution of the data;
- GP can directly operate on the data in its original form;
- GP can detect the underlying but unknown relationship that exists among data and express it as a mathematical expression;
- GP can discover the most important discriminating features of a class during training phase ;

The generated expression can be easily used in the application environment.

1.3. Related Works

The use of genetic programming to solve the multi-category classification and the intrusion detection problems has been attempted in many researches in different ways. In [12], Loveard et al. proposed five methodologies for multi-category classification problems. Of these five methodologies, they have shown that dynamic range selection method is more suitable for multi-class problems. In this dynamic range selection scheme, they record the real valued output returned by a classifier (tree or program) for a subset of training samples. The range of the recorded values is then segmented into regions to represent class boundaries. If the output of the classifier for a pattern falls in the region, then the class is assigned to. Once the segmentation of the output range has been performed, the remaining training samples can then be used to determine the fitness of an individual (or classifier). Chien et al. [13] used GP to generate discriminator functions using arithmetic operations with fuzzy attributes for a classification problem. In [14], Mendes et al. used

GP to evolve a population of fuzzy rule sets and a simple evolutionary algorithm to evolve the membership function definitions. These two populations are allowed to co-evolve so that both rule sets and membership functions can adapt to each other. For a C-class problem, the system is run C-times. Kishore et al. [3] proposed an interesting method which considers a class problem as a set of two-class problems. When a GP classifier expression (GPCE) is designed for a particular class, that class is viewed as the desired class and the remaining classes taken together are treated as a single undesired class. So, with GP runs, all GPCEs are evolved and can be used together to get the final classifier for the C-class problem. They have experimented with different function sets and incremental learning. In [15], Durga and Nikhil R. proposed a method to design classifiers for a C-class pattern classification problem using a single run of GP. For a class problem, a multi-tree classifier consisting of C-trees is evolved, where each tree represents a classifier for a particular class. The performance of a multi-tree classifier depends on the performance of its constituent trees. A new concept of unfitness of a tree was exploited in order to improve genetic evolution. Weak trees having poor performance are given more chance to participate in the genetic operations so that they get more chance to improve themselves.

In [10], Mengjie and Will proposed two new approaches to ameliorate the performances of genetic classification algorithms. Rather than using fixed static thresholds as boundaries to distinguish between different classes, this approach introduces two methods of classification where the boundaries between different classes can be dynamically determined during the evolutionary process. The two methods are centred dynamic class boundary determination and slotted dynamic class boundary determination. Their obtained results suggest that, while the static class boundary method works well on relatively easy object classification problems, the two dynamic classes boundary determination methods outperform the static method for more difficult, multiple class object classification problems. The mentioned approaches were tested on different dataset publicly available, like the IRIS dataset, the Cancer dataset, the Australian Credit Card and the Fisher's Iris data or the Heart Disease datasets, witch are relatively very small and limited compared to the intrusion detection problem ones. The most important work on GP-classification for intrusion detection is the one presented in [1] by Dong Song, where a Page-based Linear Genetic Programming is implemented with a two-layer Subset Selection scheme to address only the two-class intrusion detection classification problem, the same author introduce and hierarchical RSS-DSS algorithm for dynamically filtering large datasets to enhance the system performances in [2]. Less important works can be found in [16], [17] with the Chimera model, and [18].

1.2. Genetic Programming and Classification Task

Generally, GP trees can perform classification by returning numeric (real) values and then translating these values into class labels [10]. For binary classification problems the division between negative and non-negative numbers acts as a natural boundary for a division between two classes. This means that genetic programs can easily represent binary class problems. While evaluating the GP expression for an input data, if the result of the GP-expression is ≥ 0, the input data is assigned to one class; else it is assigned to the other class. Thus, the desired output D is +1 for one class and is -1 for the other class in the training set. Hence, the output of a GP-expression is either +1 (indicating that the input data belong to that class) or -1 (indicating that the input sample dose not belong to that class). During the genetic evolution of individuals, the best individual is those who correctly classify the maximum of training samples, the positive samples must give a value of +1 for the output, and negative samples must give -1.

Given a set of training data $D_{Train}=\{X_1,X_2,\ldots..X_p\}\subset R^N$, a binary classifier is a GP-expression T, so that:

$$\begin{array}{lll} T(X_i) \leq 0 & \text{if } X_i \in \text{Class 1} & (D=+1) \\ T(X_i) > 0 & \text{otherwise} & (D=-1) \end{array} \qquad (1)$$

GP is guided by the fitness function to search for the most efficient computer program to solve a given problem. A simple measure of fitness has been adopted for the binary classification problem:

$$\text{Fitness}(T)= \frac{\text{Number of samples classified correctly}}{\text{Number of samples used for training during evolution}} \qquad (2)$$

Each genetic expressing evolved map the samples space of the X_i's, to the real numbers set R, and attribute the interval $]-\infty, 0]$ to the class 1 and the interval $]0,+\infty[$ to the class 2. This mapping is static, but it can achieve good results for 2-category classification problems. Unfortunately, when more than two classes are involved (n-classes problem), finding meaningful division points over the set of reals the genetic programs return is more difficult. If boundary regions are chosen at arbitrary points over the set of reals then genetic programs face the problem of not only containing the necessary elements to distinguish between classes, but also must perform a translation task to provide output in the necessary range pre-specified for a given class. Many alternatives were proposed by many authors to solve this problem. In [11], if there are n classes in a classification task, these classes are sequentially assigned n regions along the numeric output value space from some negative numbers to positive numbers by (n-1)*thresholds/boundaries. Class 1 is allocated to the region with all numbers less than the first boundary; class 2 is allocated to all numbers between the first and the second boundaries and class n to the region with all numbers greater than the last boundary n-1, as shown in the following:

$$\text{Classe}(X_i) = \begin{cases} \text{classe } 1 & \text{if } T(X_i) \le b_1 \\ \text{classe } 2 & \text{if } b_1 \le T(X_i) \le b_2 \\ \ldots\ldots & \\ \text{classe } n-1 & \text{if } b_{n-3} \le T(X_i) \le b_{n-2} \\ \text{classe } n & \text{if } b_{n-2} \le T(X_i) \le b_{n-1} \end{cases} \qquad (3)$$

In this equation, n refers to the number of object classes, T is the GP-expression evolved, $T(X_i)$ is the output value , and b_1, b_2, b_{n-1}, b_n are static, pre-defined classes boundaries.

An alternative approach to static range selection, where ranges are arbitrarily chosen to correspond to class boundaries that all programs for the run must adhere to, is to allow each program to use a separate set of ranges for class boundaries that are dynamically determined for each individual program. Given a classification problem with many training examples and an individual from a GP population it is possible to use a subset of the training examples and record the values that are returned when attributes for specific classes are used as inputs. Based upon these outputs the effectively infinite range of the reals can then be segmented into regions corresponding to class boundaries based upon areas the program has returned values for each class in the subset of training examples, this method was implemented in [11].

The GP employed for classification tasks do however have a requirement for long training times when compared to many other classification methods. It is also often quite difficult to extract a meaningful reason as to why a given class was chosen. Because of these factors the GP method is seen to be applicable to tasks where accuracy is the most important factor in classification, and training times and understand ability are seen as relatively unimportant.

The major considerations in applying GP to pattern classification are:

- GP-based techniques are data distribution-free, so no a priori knowledge is needed about statistical distribution of the data;
- GP can directly operate on the data in its original form;
- GP can detect the underlying but unknown relationship that exists among data and express it as a mathematical expression;
- GP can discover the most important discriminating features of a class during training phase ;

The generated expression can be easily used in the application environment.

1.3. Related Works

The use of genetic programming to solve the multi-category classification and the intrusion detection problems has been attempted in many researches in different ways. In [12], Loveard et al. proposed five methodologies for multi-category classification problems. Of these five methodologies, they have shown that dynamic range selection method is more suitable for multi-class problems. In this dynamic range selection scheme, they record the real valued output returned by a classifier (tree or program) for a subset of training samples. The range of the recorded values is then segmented into regions to represent class boundaries. If the output of the classifier for a pattern falls in the region, then the class is assigned to. Once the segmentation of the output range has been performed, the remaining training samples can then be used to determine the fitness of an individual (or classifier). Chien et al. [13] used GP to generate discriminator functions using arithmetic operations with fuzzy attributes for a classification problem. In [14], Mendes et al. used GP to evolve a population of fuzzy rule sets and a simple evolutionary algorithm to evolve the membership function definitions. These two populations are allowed to co-evolve so that both rule sets and membership functions can adapt to each other. For a C-class problem, the system is run C-times. Kishore et al. [3] proposed an interesting method which considers a class problem as a set of two-class problems. When a GP classifier expression (GPCE) is designed for a particular class, that class is viewed as the desired class and the remaining classes taken together are treated as a single undesired class. So, with GP runs, all GPCEs are evolved and can be used together to get the final classifier for the C-class problem. They have experimented with different function sets and incremental learning. In [15], Durga and Nikhil R. proposed a method to design classifiers for a C-class pattern classification problem using a single run of GP. For a class problem, a multi-tree classifier consisting of C-trees is evolved, where each tree represents a classifier for a particular class. The performance of a multi-tree classifier depends on the performance of its constituent trees. A new concept of unfitness of a tree was exploited in order to improve genetic evolution. Weak trees having poor performance are given more chance to participate in the genetic operations so that they get more chance to improve themselves.

In [10], Mengjie and Will proposed two new approaches to ameliorate the performances of genetic classification algorithms. Rather than using fixed static thresholds as boundaries to distinguish between different classes, this approach introduces two methods of classification where the boundaries between different classes can be dynamically determined during the evolutionary process. The two methods are centred dynamic class boundary determination and slotted dynamic class boundary determination. Their obtained results suggest that, while the static class boundary method works well on relatively easy object classification problems, the two dynamic classes boundary determination methods outperform the static method for more difficult, multiple class object classification problems.

The mentioned approaches were tested on different dataset publicly available, like the IRIS dataset, the Cancer dataset, the Australian Credit Card and the Fisher's Iris data or the Heart Disease datasets, witch are relatively very small and limited compared to the intrusion detection problem ones. The most important work on GP-classification for intrusion detection is the one presented in [1] by Dong Song, where a Page-based Linear Genetic Programming is implemented with a two-layer Subset Selection scheme to address only the two-class intrusion detection classification problem, the same author introduce and hierarchical RSS-DSS algorithm for dynamically filtering large datasets to enhance the system performances in [2]. Less important works can be found in [16], [17] with the Chimera model, and [18].

2. Proposed GP-classification approach

The present work propose a new approach of a dynamic GP-based classifier witch consist of genetically coevolving a population of non-linear transformations on the input data to be classified, and map them to a new space with a reduced dimension (1-D), in order to get a maximum inter-classes discrimination. Let $D_{Train}=\{X_1,X_2,\ldots..X_p\}\subset R^N$ be the set of training data. Because the proposed approach belong to the supervised learning category, each sample X_i can be labelled with its class identifier j and become X_i^j. The set D_{Train} can then be subdivided into n sub-set corresponding to n learned classes, such that:

$$D_{Train} = \bigcup_{j\leq n} D^j_{Train} \quad , D^j_{Train} = \left\{X^j_i \in D_{Train} \;/\; class(X^j_i) = j\right\} \tag{4}$$

The output value for each sample from the each training sample is computed using the GP-expression T, this allow to compute the transformed map for each sub-set D_{Train}^j , using the GP-expression T, $T(D_{Train}^j)$ given by:

$$T(D^j_{Train}) = \left\{Y = T(X^j_i) \;\;/\;\; X^j_i \in D^j_{Train}\right\} \tag{5}$$

The classification approach assign to each class j, the region covered by the set $T(D_{Train}^j)$. When a new sample Y is presented to the classifier, the corresponding class is deduced according to the following:

$$Classe\,(Y) = \begin{cases} classe\,1 & if\;\; T(Y) \in T(D^1_{Train}) \\ classe\,2 & if\;\; T(Y) \in T(D^2_{Train}) \\ \ldots\ldots & \\ classe\,n-1 & if\;\; T(Y) \in T(D^{n-1}_{Train}) \\ classe\,n & if\;\; T(Y) \in T(D^n_{Train}) \end{cases} \tag{6}$$

If the value of T(Y) dose not appear in any set $T(D_{Train}^j)$, we assign Y to the nearest class using the algorithm presented below in the figure 4.

We can see that the proposed classification method transform the problem from an N-dimensional vectors classification to a 1-dimentional values classification one. The classification of the transformed vectors become much easier, but this is assured if a maximum discrimination exist between the sets $T(D_{Train}^i)$. It is role of the genetic programming system to assure such criteria, the fitness of each transformation T depend on its ability to give a maximum discrimination between the $T(D_{Train}^i)$'s.

There is a trade-off between the generality and power of this classification approach search. To perform a relatively unbiased search and allow the saliencies of the problem to emerge the proposed approach has many degrees of freedom in its representation of the solution. Rather than evolve the class predictors directly and further encumber the genetic program, features are evolved which are then passed to a simple classifier. This hybrid approach assists the global search of the genetic program with the local search of the simple classifier. The classifier, with its malleable decision boundaries, performs local tuning of the solution to compensate for the genetic program's difficulty with evolving constants. In the following, we present the different steps of the classification approach: the learning phase, witch consist to search for the best transformation of the raining data D_{Train} , and the test phase that classify each test sample from a set of new vectors D_{Test}.

2.1. The learning phase

2.1.1. Terminals and functions

The GP-transformations are built using a terminal set Tr and a function set Fn. The terminals are the fields of the used training dataset: $Tr=\{V_1,V_2,\ldots,V_N\}$, in addition, we also used constants as terminals. These constants are randomly generated using a uniform distribution. To be consistent with the feature terminals, we also set the range of the constants as [-100, 100]. The functions set include:

- Arithmetic operators: +, -, /, *, ^;
- Non-linear functions: Sin ,Cos ,Ln , Log ,Exp ,Tan;

The +, - , and * operators have their usual meanings: addition, subtraction and multiplication, while / represents "protected" division which is the usual division operator except that a divide by zero gives a result of zero. Each of these functions takes two arguments. The transformations T_i are represented by hierarchical S-expressions trees, like proposed in the standard Koza implementation.

2.1.2. The fitness function

For a given training set $D_{Train} \subset R^N$, the genetic programming system evolve a population of transformations T. In order to compute the fitness of each one, we need to define a distance between the mapped sets $T(D_{Train}^i)$. The value of the fitness must express the inter-classes discrimination and separation. During our experiments, we have tested many fitness measurements, such as the Maximum distance between gravity centres of the mapped classes and the inter-classes and intra-classes variance criterion. But theses functions assume that the transformed sets $T(D_{Train}^i)$ must be homogeneous and linearly separable, this condition is not always easy to achieve, so it can be better to give to the classification system the ability to generate separated but alternate transformed sets. The figure 1 illustrate the two situations: (a) represent two pointes sets linearly separable (in a one dimensional space), and (b) show two separated pointes sets but in an alternated situation.

For this reason, we have proposed a new fitness function formula, witch try to minimize the total intersection between point sets, and search for a minimum number of common points between the mapped classes. The fitness function is inversely proportional to the computed number of common points between transformed sets $T(D_{Train}^i)$. Height values of the fitness signify that the transformed sets have a very small intersection region, and then the discrimination between each set elements become easier. The fitness value is computed by:

$$\text{Fitness}(T) = \frac{\text{Card}(\bigcap_{i \le n} T(D^i_{\text{Train}}))}{\text{Card}(T(D_{\text{Train}}))} \tag{7}$$

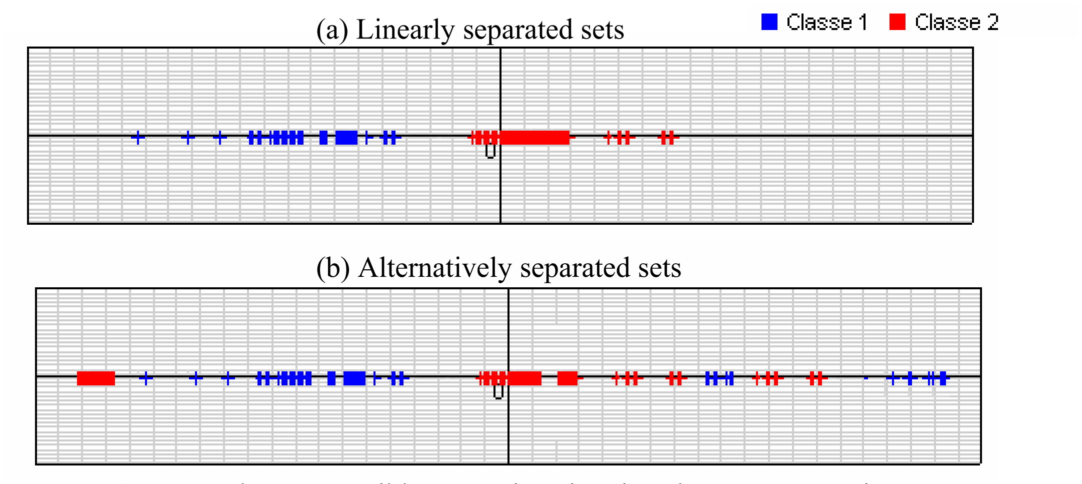


Figure 1. The two possible separation situations between two point sets.

When the function Card(X) give the cardinality of a given set X. The performed experiments show that this function give the best classification rates with respect to other fitness function mentioned above. The classification system become more flexible, and explore new solution unexploited by the other fitness functions.

2.1.3. Genetic operators and parameters

The standard crossover and mutation operators presented in the section 1.1 are used in this implementation. Each transformation T is represented by a binary tree and the genetic operators produce always valid binary expressions. To control the maximum depth of the generated expressions, we use a modifiable parameter to control the length of the generated expressions. The genetic evolution process stop when it reach a given generations count (termination criteria). The Table.1 give an overview of the parameters used in our implementation and the default value used for each one. During the evolution process, the result of each transformation T_i is bounded in a fixed interval ([-100,100] by default), to avoid to have scatter sets in R.

Parameter	**Value**
Generating constant probability	5%
Generating functions probability	70%
Crossover rate P_c	70%
Mutation rate P_m	10%
Population size	100
Maximum generations count	1000
Maximum individual's length	350
Minimum individual's length	30
Selection strategy	Roulette selection
Functions set	{+, - ,/ ,*, sin, cos, log, ln, tan, exp }
Terminals set	[-100,100]$\cup$ {input variables}

Table1. Set of parameters used to control the genetic evolution process

The result of the genetic evolution during the training phase is a the best generated transformation T, with the transformed sets $T(D_{Train}^{i})$. This output is used in the test phase to classify new samples.

2.2. The Test phase: classification of unseen samples

Let $D_{Test}=\{Y_1,Y_2,\ldots,Y_k\}$ be a new set of samples to be classified. Each vector $Y_i \in D_{Test}$ must be assigned to one of the n involved classes. To accomplish this task, the classification system operate like the following: First, a post-treatment algorithm is added to the classification system to compute a density array for the points of $T(D_{Train})$. This array is used with the transformation T during the test phase to deduce the class of each elements Y_i form D_{Test}. This algorithm is presented like the following (Fig.2):

```
Let T(D_Train) be the training Set;
Dens: array of density for T(D_Train) elements;
For each element p ∈ T(D_Train) do
        {For each sample X ∈ D_Train do
            {For each class i (i form1 to n) do
                {if (classe(X)=i)and(T(X)=p)
                    then Dens[p,i]:= Dens[p,i]+1;}

            }   }
```

Figure 2. A Post-treatment algorithm to generate density array, used during the testing phase

Then, For each new sample Y_i form D_{Test} , the corresponding class is determined using the following algorithm (Fig.3):

```
Let Y be any new sample from D_Test;
Ne: the nearest point from T(D_Train) to T(Y);
Max: a height random value;
Class_Y: the deduced class for the sample Y;
For each element p ∈ T(D_Train) do {d:=Distance (T(Y), p);
                        If d<Max then {Max:=d;
                                       Ne:=p;
                                       Class_Y:=class(p);
                                      }
                        Else if d=Max then
                                 {If Dens(p,classe(p))>Dens(p,Class_Y)
                                       then {Class_Y:=classe(p);
                                             Ne:=p
                                            }
                                 }
                        }
Result := Class_Y;
```

Figure 3. The proposed algorithm to deduce the class of new test samples Y_i, used during the testing phase

As shown by the experiments, these algorithms combined with the fitness function mentioned above, give much better results than using classical fitness measurement, this is due to the flexibility of the classes distribution accorded to the genetic classification system.

3. Datasets and experiments

The proposed classification approach is benchmarked using two different datasets: the Fisher IRIS [19] dataset and the MIT KDD99 dataset [20]. The first one is used just for comparison purpose, and to demonstrate the proposed method capabilities, it is relatively very small and limited compared to the intrusion detection problem datasets. The second one concern our problem of interest: the network intrusion detection. The KDD99 dataset is the most used one for intrusion detection problems, collected at the Lincoln Laboratory of MIT, under DARPA sponsorship, witch consists of about 5,000,000 connection records, with 41 data fields. The most important work on GP-classification using the KDD99 dataset is the one presented in [1, 2] by Dong Song, where a Page-based Linear Genetic Programming is implemented with a two-layer Subset Selection scheme to address only the two-class intrusion detection classification problem.

The first IRIS dataset was divided equally into a training set and a test validation set. The specific training sets for Iris setosa, versicolor and viginica are derived from the training set. To perform the experiments with the KDD99 dataset, the '10% KDD' set was sampled and only 24788 records are used to train our system. For the test purposes, we use the whole 'Corrected (Test)' used in almost all the implemented approaches. The Table.2 lists the class's distributions of our used normal sets.

	Training Set		**Testing Set**	
Normal	11673	47.09 %	60593	19.48 %
DOS	7829	31.58 %	229853	73.90 %
PBR	4107	16.56 %	4166	1.34 %
R2L	1119	4.51 %	16347	5.25 %
U2R	52	0.24 %	70	0.02 %

Table 2. Distribution of the normal and attack records in the used training and testing sets.

Attributes in the KDD datasets had all forms :continuous, discrete, and symbolic, with significantly varying resolution and ranges. Most pattern classification methods are not able to process data in such a format. Hence, pre-processing was required before pattern classification models could be built. Pre-processing consisted of two steps: first step involved mapping symbolic-valued attributes to numeric-valued attributes and second step implemented scaling. In the present work, we have used the data codification and scaling presented in [21]. All the resulting scaled fields belong to the interval [0, 1].

The Table.3 summarizes the 41 fields used in the KDD99 dataset regrouped in three mentioned categories. Each field is labelled with a symbolic notation (F_1, F_2,…, F_{41}) to be used as terminals during the genetic process.

Basic features of individual TCP connections		Content features suggested by domain knowledge		Traffic features computed using a two-second time window	
duration	**F1**	hot	**F10**	count	**F23**
protocol_type	**F2**	num_failed_logins	**F11**	srv_count	**F24**
service	**F3**	logged_in	**F12**	serror_rate	**F25**
flag	**F4**	num_compromised	**F13**	srv_serror_rate	**F26**
src_bytes	**F5**	root_shell	**F14**	rerror_rate	**F27**
dst_bytes	**F6**	su_attempted	**F15**	srv_rerror_rate	**F28**
land	**F7**	num_root	**F16**	same_srv_rate	**F29**
wrong_fragment	**F8**	num_file_creations	**F17**	diff_srv_rate	**F30**
urgent	**F9**	num_shells	**F18**	srv_diff_host_rate	**F31**
		num_access_files	**F19**	dst_host_count	**F32**
		num_outbound_cmds	**F20**	dst_host_srv_count	**F33**
		is_hot_login	**F21**	dst_host_same_srv_rate	**F34**
		is_guest_login	**F22**	dst_host_diff_srv_rate	**F35**
				dst_host_same_src_port_rate	**F36**
				dst_host_srv_diff_host_rate	**F37**
				dst_host_serror_rate	**F38**
				dst_host_srv_serror_rate	**F39**
				dst_host_rerror_rate	**F40**
				dst_host_srv_rerror_rate	**F41**

Table 3. The KDD99 used features, grouped in 3 categories

All tests were performed on an Intel-Pentium 4 CPU 2.66Ghz with 256 Mb Ram size. The performances of intrusion detection for the classifier are computed using the following expressions:

$$\text{Detection rate} \qquad DR = 1 - \frac{\text{False negatives number}}{\text{Total Number of Attaks}}$$

$$\text{False Positive Rate} \qquad FP = \frac{\text{False Positives}}{\text{Total Number of normal connections}} \qquad (8)$$

4. Results and comparison

This section presents the results of the proposed GP classification approach for the 2 n-classes pattern classification problems described above, using the set of parameters presented in the Table.1.

4.1. Fisher IRIS classification problem

The dataset was divided equally into a training set and a test validation set (75 samples in each set). The result of each test is a classification matrix C computed by the following algorithm (Fig.4):

```
Let T($D_{Test}$) be the training Set;
n is the number of classe
i is the true class of the sample and k is the assigned class.
For i=1 to n do
   For j=1 to n do {  C[i,j]:=0
                      For i=1 to Card(T($D_{Ttest}$)) do
                               {Apply the classifier and assign class k to input sample.
                                C[i,k]:=C[i,k]+1;
                               }
                   }
```

Figure 4. The algorithm used to compute the classification matrix

The classification rate is then computed using the following expression:

$$CR = \frac{\text{Number of samples classified correctly}}{\text{Number of samples used for training during evolution}} * 100 \qquad (9)$$

The table 4 give the classification matrix obtained using the proposed approach. Tables 5 and 6 give the results of the classification process using a maximum likelihood classifier and a GP-based classification approach proposed in [3].

	Setosa	Versicolor	Viginica
Setosa	25	0	0
Versicolor	0	25	0
Viginica	0	1	24

Table 4. Obtained classification matrix using the proposed approach

	Setosa	Versicolor	Viginica
Setosa	25	0	0
Versicolor	0	24	1
Viginica	0	2	23

Table 5. Classification matrix for iris data set with maximum likelihood classifier [3].

	Setosa	Versicolor	Viginica
Setosa	25	0	0
Versicolor	0	24	1
Viginica	0	2	23

Table 6. Classification matrix for GPCE with interleaved training sets for Iris data [3].

The following table (Table.7) give a comparison between the detection rate obtained with different classifiers as presented in [24, 3], and our proposed classification approach.

Method	NN	Naive Bayse	BayseNet	C4.5	GPCE [3]	Maximum Likelhood	GP-classification
Classification rate (DR)	96 %	96 %	94.667 %	94.67 %	96 %	97.3 %	**98.6 %**

Table 7. A summary of the detection rates obtained using different classifiers for the Fisher's Iris dataset

Best individual	Classification rate (DR)
(((((ln((-(((log2(exp((((F3)^2)/(F4))*(F3))))^2)-(F4)))^2))+(F4))*(F4))+(F3)))	**98.6 %**

Table 8. Best obtained individual using the proposed approach for the Fisher's Iris dataset

Table 8 give the optimal transformation T (best individual) obtained after the GP running, with the corresponding classification rate. The figures 5 show the distribution of the transformed training set T(D_{Train}) obtained with this transformation.

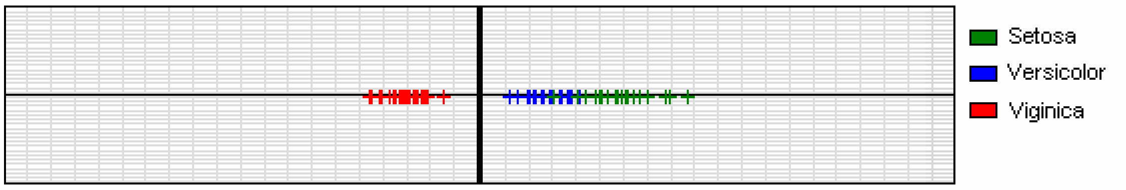


Figure 5. Distribution of the transformed training set T(D_{Train}) of the best obtained individual

From the Table.7, we can see that our proposed approach give best classification rate compared to other proposed approaches, one sample only from the "Viginica" set is misclassified.

4.2. KDD99 dataset: the intrusion detection problem

As we see in the Table.1, the KDD99 dataset is more voluminous than the Iris fisher's one, and contain more classes (5 classes). Discrimination is also very difficult in the intrusion detection case because the classes are not clearly separable, so the classification task will become harder. To evolve the GP classification system, the same parameters set presented in the Table.1 is used. In the Table.9, we present the classification matrix obtained. The figures 6 and 7 illustrates the transformed training set $T(D_{Train})$ repartition, and the fitness value evolution during the GP evolution. The best individual T is presented by the following expression:

T: ((((log2(tan(-(F3))))*(cos((tan((F5)+(((log2(tan(-(F3))))*(((log2(tan(-(F3))))*(F30))+(cos(F5))))*((tan(F13))+(F30)))))+((tan(log2(F2)))+(F30)))))*(cos(F5)))*((18)+(cos((tan(log2(F13)))+(F30)))))

	Normal	Prob	Dos	U2R	R2L	% Correct
Normal	59769	500	112	49	163	**98.64 %**
Probe	562	3443	113	3	45	**82.65 %**
Dos	8411	768	220662	0	11	**96.00 %**
U2R	25	11	6	19	9	**09.82 %**
R2L	10612	2107	8	2059	1611	**27.14 %**
% Correct	**75.29 %**	**50.42%**	**99.89%**	**0.89%**	**87.60%**	

Table 9. Classification matrix obtained using the proposed approach

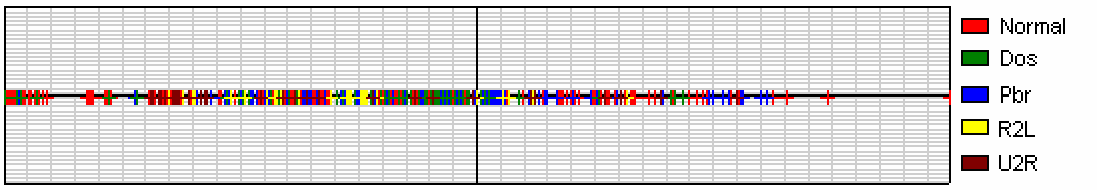


Figure 6. Distribution of the transformed training set $T(D_{Train})$ of the best individual

The following values of detection rate and the false positive rates were computed for the best obtained individual T:

Detection rate: DR = 0.925 (92.5 %)

False positive rate FP = 0.0135 (1.35 %)

Classification rate = 91.7 %

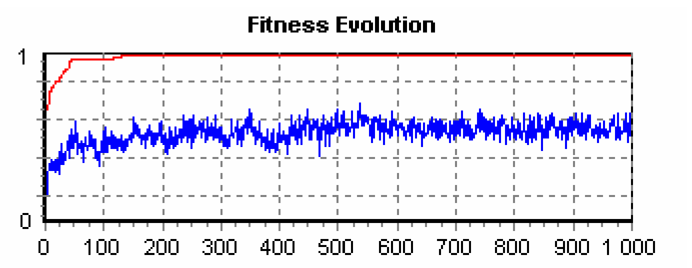


Figure 7. Evolution of the fitness value during the genetic process

The Table.10 summarize and compare the detection rates and false positive rates obtained using the approaches mentioned above, and some recent results on KDD benchmark presented in [7]and [8]. All the mentioned approaches were tested using the 'Corrected (Test)' set of the KDD99 cup competition.

Classification method	Detection rate	False Alarm rate
GP-classifier (proposed)	92.5 %	1.35 %
KDD99 wining entry[5]	91.0 %	**0.50 %**
KDD99 second place[6]	91.5 %	0.58 %
Linear GP classifier [1,2]	90.8 %	3.26 %
Data-mining techniques [22]	70%-90%	2.00 %
Support vector machine[7]	**98.0 %**	10.00 %
Self organized Maps [8]	89.0 %	4.60 %
Clustering techniques [7]	93.0 %	10.00 %
K-nearest neiberhood [7]	91.0 %	8.00 %

Table 10. Comparison of the detection performances between the proposed approaches and the existing techniques

We can see from the presented results that the proposed classification approach give very acceptable results compared to the other techniques. The highest detection rate is obtained using support vectors machine technique implemented in [7], but with a very height false positive rate (10 %) compared to 1.35% obtained with our proposed GP-classification approach.

It is reasonable to state that the set of pattern recognition and machine learning algorithms mentioned above offered an acceptable level of misuse detection performance for only two attack categories, namely Probing and DoS when tested on the KDD data sets, and failed to demonstrate an acceptable level of detection performance for the remaining two attack categories, which are U2R and R2L. To enhance the detection capabilities of our classification system, especially for the two categories R2L and U2R, we propose in the following an improvement of the proposed classification approach using a multi-transformation approaches. The obtained results demonstrate that the capabilities can be highly ameliorated compared to the standard approach.

5. GP-Classifier enhancement: The multi-transformations classification system

5.1. Method description

As explained in the section 2. the classification system use a single transformation (the best obtained individual) to transform each new sample, and then deduce the corresponding class using the algorithm presented in the figure 4. The main idea of the multi-transformation system is to use a set of multiple transformations $TR_{set} = \{T_1, T_2, \ldots, T_p\}$ obtained genetically (the best ones) on the sample to be classified. Each transformation will output a corresponding class with a confidence factor for each sample Y from the testing dataset computed using the following expression:

$$\text{Confidence}(Y,T) = \frac{\text{Dens}(Ne, \text{Class\_Y})}{\text{Card}(T(D_{\text{Train}}^{\text{Class\_Y}}))} * \text{Fitness}(T) \qquad (10)$$

when:

- Ne is the nearest point from the $T(D_{\text{Train}})$ set;
- Class_Y is the deduced class for the sample Y
- $\text{Card}(T(D_{\text{Train}}^{\text{Class\_Y}}))$ is the number of the samples from the training set belonging to Class_Y.
- Fitness(T) is the fitness value of the transformation T
- Dens(Ne, Class_Y) is the density value computed by the algorithm of Fig.3

It is clear from the equation (10) that the confidence factor of a given sample in relation to a transformation T_i range in the interval [0, 1]. All the mentioned parameters are taken from the algorithm of the figures 4. The equation (10) was introduced in the algorithm like the following (Fig.8):

```
Let Y be any new sample from $D_{Test}$;
Ne: the nearest point from $T(D_{Train})$ to T(Y);
Max: a height random value;
Class_Y: the deduced class for the sample Y;
**For** each element $p \in T(D_{Train})$ **do** {d:=Distance (T(Y), p);
                                  **If** d<Max **then** {Max:=d;
                                             Ne:=p;
                                             Class_Y:=class(p);
                                            }
                                  **Else if** d=Max **then**
                                        {**If** Dens(p,class(p))>Dens(p,Class_Y)
                                              **then** {Class_Y:=classe(p);
                                                    Ne:=p
                                                   }
                                        }
                                 }
**Confidence(Y, T) := Dens(Ne,Class_Y) /Card(T($D_{train}^{class\_y}$)** ;
**Ret_Class(Y,T) :=Class_Y ;**
```

Figure 8. The modified version of the algorithm used to deduce the class of new test samples, and compute their confidence factor.

This algorithm return for each sample Y, its corresponding class Ret_Class(Y, T), with the corresponding confidence factor Confidence(Y,T). The new classification system take the best individuals collected during the genetic evolution to construct a transformations set $TR_{set} = \{T_1, T_2, \ldots, T_p\}$. All this transformations are applied on each test sample Y during the testing phase to obtain p possible class and p corresponding confidence factor. These obtained outputs are combined to compute the membership factor of Y to each class c from the existing n classes using the following algorithm (Fig.9):

```
Membership(Y,c):=0;
**For** each transformation $T_i$ from $TR_{set}$ **do**
                       {**If** Ret_Class(Y,$T_i$) = c **then**
                         Membership(Y,c):= Membership(Y,c)+Confidence(Y,$T_i$);
                       }
Membership(Y,c):= Membership(Y,c) / p;
```

Figure 9. The algorithm proposed to compute the membership factor of a sample Y to a given class c.

It is clear from the formulas used above that the value of the membership factor range always in the interval [0, 1]. The classification system assign to Y the class with the highest membership factor:

$$\text{Class}(Y) = c \text{ such that Confidence}(Y, c) = \underset{1 \le i \le n}{\text{MAX}}(\text{confidence}(Y, i)) \qquad (11)$$

This method benefit from the detection capabilities of each transformation T from the generated set TR_{set}, it act like a rule system that average the obtained decisions to elaborate the final one. The following results demonstrate the improvement achieved by this technique compared to the single transformation one.

5.2. Results and comparison

This section summarize the results obtained using the multi-transformations classification system described above to detect and classify the intrusions in the KDD99 dataset. The test phase use the KDD99 'Corrected (Test)' set. The number of transformations p used in this experiment is fixed to 50 transformations collected during the learning phase realised by the genetic process. The following results give the average accuracy obtained for 40 GP trials conducted on the input training set. The classification rates, detection rates and the false positive rates were computed in each GP trial.

The figure.10 show the variations of the detection rate for each class with respect to the number of used transformation, it is clear that better classification rates are allowed for the two classes R2L and U2R. The classification is ameliorated when augmenting the number of the used transformations. For the classes Normal and Dos, the maximum classification performances are reached starting form 6 or 7 transformations. By the same way, it can be seen from the Figure.11 that the system reaches it's maximum performances when the number of used transformations is maximum (a higher detection rate and a lower false positive rate).

The Table.11 illustrate the classification matrix obtained with the best GP-trail using the multi-transformation method to classify the intrusions of the used KDD99 Test dataset with 50 collected transformations.

The performances rates obtained by the obtained solution are given by:

Detection rate: DR = 0.980 (98.0%)

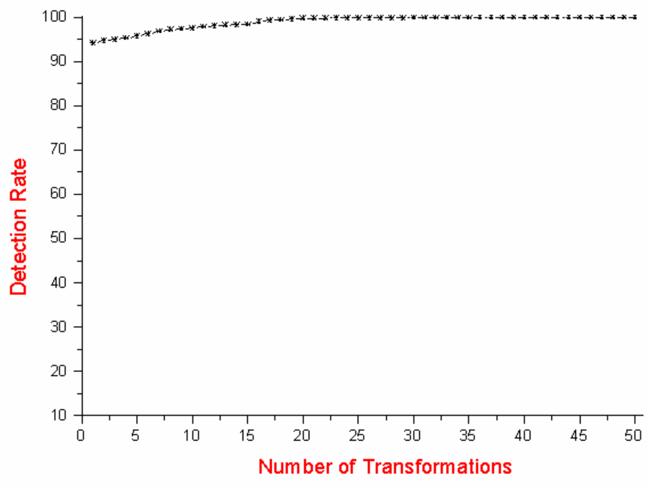
False positive rate FP = 7E-4 (0.07%)

Classification rate = 99.05 %

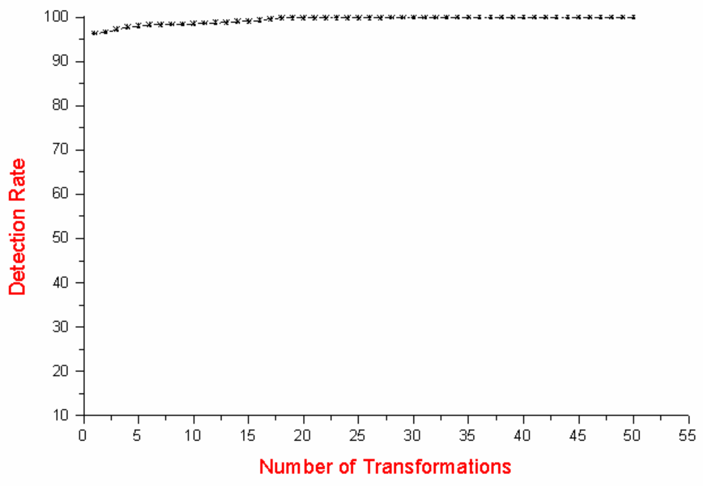
	Normal	**Prob**	**Dos**	**U2R**	**R2L**	**% Correct**
Normal	60550	21	10	4	8	**99.93%**
Probe	93	4053	15	0	5	**97.29%**
Dos	1792	911	227117	15	18	**98.81%**
U2R	21	6	2	38	3	**45.20%**
R2L	2973	154	21	85	13114	**80.22%**
% Correct	**92.54%**	**78.77%**	**99.97%**	**26.7%**	**99.74%**	

Table 11. Classification matrix obtained using the multi-transformations classification method with 50 transformations

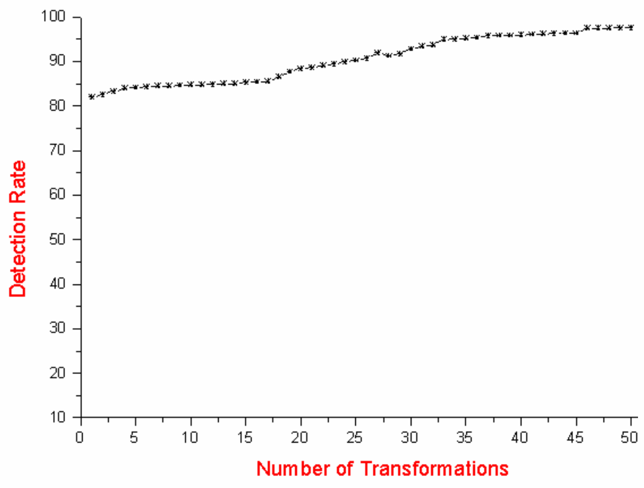
In the Table.12, obtained classification rates using the multi-transformations classification system are compared to the results presented in [23] using multiple classification systems such as Multilayer perceptron (MLP), Gaussian classifier (GAU), nearest cluster algorithm (NEA), incremental radial basis function, K-means clustering (K-M), C4.5 decision tree and many other techniques. The results shows that classification rates obtained using the multi-transformations classification system for the two classes R2L and U2R are very satisfactory with respect to the other techniques. The false positives detection rate of each attack class is not available for the SOM [8] and the linear GP [1, 2] techniques, since they are 2-category classifiers (normal and attack), their false positive rates can be given only in term of whole attacks classification.



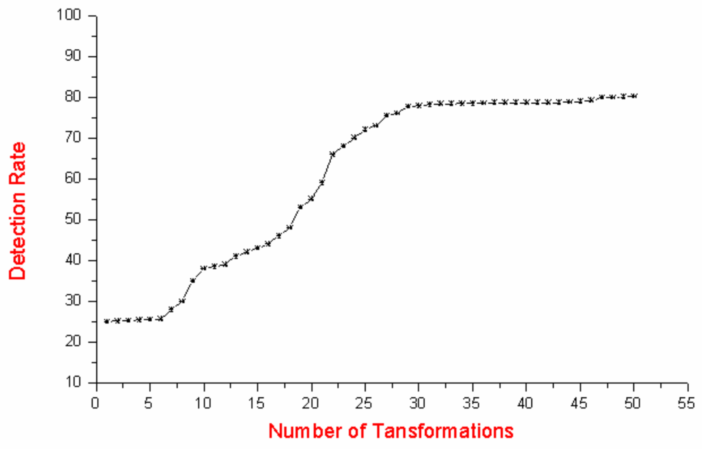
(a)



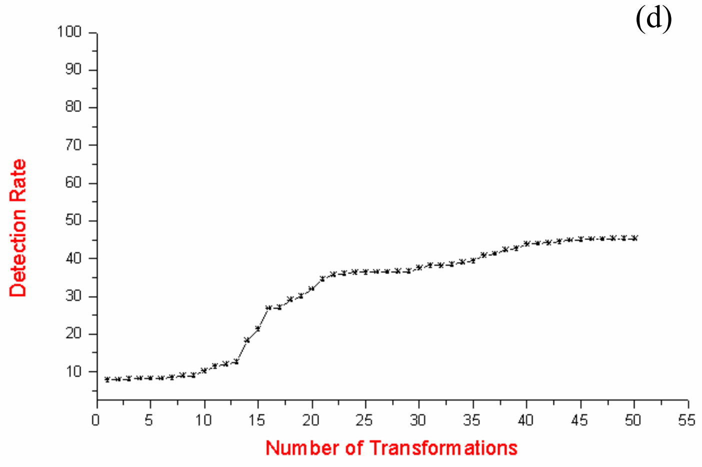
(b)



(c)



(d)



(e)

Figure 10. Evolution of the detection rate for each attack class with respect to the number of applied transformation: (a) Normal, (b) Dos, (c) Prob, (d) R2L and (e) U2R

Classification method	Dos		Prob		R2L		U2R	
	DR	FP	DR	FP	DR	FP	DR	FP
KDD cup Winner [4]	0.971	0.003	0.833	0.006	0.084	5E-5	0.123	3E-5
Agrawal and joshi [26]	0.969	0.001	0.730	8E-5	0.107	8E-4	0.066	4E-5
GP 1-transformation	0.960	7E-4	0.826	0.010	0.271	7E-4	0.100	0.006
GP multi-transformations	**0.988**	**1E-4**	**0.972**	0.003	**0.802**	1E-4	**0.452**	3E-4
SOM map [8]	0.951	-	0.643	-	0.113	-	0.229	-
Linear GP [1,2]	0.967	-	0.857	-	0.093	-	0.013	-
Multilayer Perceptron [23]	0.972	0.003	0.887	0.004	0.056	1E-4	0.132	5E-4
Gaussian classifier [23]	0.824	0.009	0.902	0.113	0.096	0.001	0.228	0.005
K-means clustering [23]	0.973	0.004	0.876	0.026	0.064	0.001	0.298	0.004
Nearest cluster algorithm [23]	0.971	0.003	0.888	0.005	0.034	1E-4	0.022	6E-6
Radial basis function [23]	0.730	0.002	0.932	0.188	0.059	0.003	0.061	4E-4
Leader algorithm [23]	0.972	0.003	0.838	0.003	0.001	3E-5	0.066	3E-4
Hypersphere algorithm [23]	0.972	0.003	0.848	0.004	0.010	5E-5	0.083	9E-5
Fuzzy ARTMAP [23]	0.970	0.003	0.808	0.007	0.037	4E-5	0.061	1E-5
C4.5 decision tree [23]	0.970	0.003	0.808	0.007	0.046	5E-5	0.018	2E-5

Table 12. Comparison of the classification rates obtained with different techniques using the 4 attack classes of the KDD99 dataset.

In the present work, the multi-transformations classification system requires approximately 1 hour and 48 minutes to generate a set of 50 optimal transformations, when addressing the problem of intrusions classification using the mentioned KDD99 dataset. Compared to other existing solutions, the proposed classification approach represents the potential to achieve best classification performances in shorter training time like illustrated by the Table.13.

Classification method	Training time	Solution complexity
KDD wining entry	≈ 24 Hours	500 decision tree
KDD second place	≈ 22 Hours	755 decision tree
Linear GP [2]	≈ 7 Hours and 30 minute	A linear program with 86 instruction in 2 address format
GP with Multi-transformation	≈ 1 Hours and 48 minute	50 non-linear transformation with an average length of 150 character

Table 13. Comparison of the time and solution complexity of different classification methods

6. Conclusion and Future Work

In this work, a new Genetic Programming classification system with a dynamic classes projection was implemented and tested on both Fisher's Iris dataset and the KDD'99 benchmark dataset, a problems involving a multi-category classification task. To do so, populations of non-linear transformations are evolved to transform the input training data to be classified to a new one dimensional space with a maximum discrimination between the projected classes. The classification task become much easier with the transformed data and the new testing samples are then transformed with the generated transformation and assigned to their corresponding class using a simple search algorithm (Figure 4). The technique is independent of the dataset and structure of GP employed. Moreover, the framework has no specialist hardware requirements, making use of the generic classifiers design already widely supported in computing systems. The proposed system is shown to be capable of learning attack and normal behaviour from the training data and make accurate predictions on the test data, which also contains new attacks that the system was not trained on.

In order to enhance the classification performances, especially for some bad handled categories, a multi-transformation system was implemented and tested to combine the classification decisions of a large transformations set. The obtained results show that the proposed system can achieve much better classification performances, without significant increasing of the learning and detection run time. The study of the proposed method show that increasing the number of combined transformations enhance significantly the system performances.

In comparison to artificial intelligence approaches currently proposed, the approach provides competitive performance whilst utilizing a relatively small set of training samples. The time complexity of the approach is independent from the number of used fields (41 in the case of the KDD dataset) and is very acceptable in relation to the other approaches (Table 16). The complexity of the generated solution is reduced in comparison to the solutions of other techniques. Each transformation is represented as a string with 150 characters (byte) at maximum, and can be easily transformed to an assembly routine and evaluated using a stack base schema, to be integrated in a real time detection system.

In terms of future work, the proposed classification approach can be extended to map the classes to a higher dimensionality space (especially for the 2D and 3D spaces). That is to say, a population of combinations of transformation $<T_1,T2,..,T_p>$ is evolved for the training dataset to get the optimal combination witch project the data to the specified space of dimensionality p. For example, in the 2D case, each individual is a couple $<T_1,T_2>$ that project each sample X_i from R^N to R^2 like the following:

$$T(X_i) = < T_1, T_2 > (X_i) = (y_1, y_2) \quad \text{such that} \quad \begin{cases} y_1 = T_1(X_i) \\ y_2 = T_2(X_i) \end{cases}, (y_1, y_2) \in \Re^2 \qquad (15)$$

The same principal can be used for any p-dimensional space. Such approach has the potential to reduce the information loss due to the transformation operation, since a higher dimension can handle more information and relationship between the different initial components. Another important advantage is the possibility to generate a graphical visualisation of the transformed data (in the 2D or 3D case) witches allow to have different possible profiles of the classes' distribution, and to give some interpretations like inter-classes proximity and intersections.

References:

1. Dong Song, Malcolm I. Heywood, and A. Nur Zincir-Heywood. "Training Genetic Programming on Half a Million Patterns: An Example from Anomaly Detection", IEEE Transactions on Evolutionary Computation, 9(3), pp 225-240, 2005
2. Dong Song, Malcolm I. Heywood, and A. Nur Zincir-Heywood. “A Linear Genetic Programming Approach to Intrusion Detection ». E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2724, pp. 2325–2336, 2003. © Springer-Verlag Berlin Heidelberg 2003
3. J. K. Kishore, L. M. Patnaik, V. Mani, and V. K. Agrawal, “Application of genetic programming for multicategory pattern classification,” IEEE Trans. Evol. Comput., vol. 4, pp. 242–258, Sept. 2000.
4. Pfahringer B.: Winning the KDD99 Classification Cup: Bagged Boosting. SIGKDD Explorations. ACM SIGKDD. 1(2) (2000) 65- 66
5. Levin I.: KDD-99 Classifier Learning Contest LLSoft’s Results Overview. SIGKDD Explorations. ACM SIGKDD. 1(2) (2000) 67- 75
6. Vladimir M., Alexei V., Ivan S.: The MP13 Approach to the KDD'99 Classifier Learning Contest. SIGKDD Explorations. ACM SIGKDD. 1(2) (2000) 76-77
7. Eskin E., Arnold A., Prerau M., Portnoy L., and Stolfo S. A Geometric Framework for Unsupervised Anomaly Detection: Detecting intrusions in unlabeled data. In D. Barbara and S. Jajodia, editors, Applications of Data Mining in Computer Security. Kluwer, 2002. ISBN 1-4020-7054-3, 2002.
8. Kayacik G., Zincir-Heywood N., and Heywood M. On the Capability of an SOM based Intrusion Detection System. In Proceedings of International Joint Conference on Neural Networks, 2003.
9. Koza, J. R. 1994. Genetic Programming II: Automatic Discovery of Reusable Programs. The MIT Press.
10. Mengjie Zhang and Victor Ciesielski. Genetic programming for multiple class object detection. In Norman Foo (editor), Proceedings of the 12th Australian Joint Conference on Artificial Intelligence, Volume 1747, Lecture Notes in Artificial Intelligence, pages 180–191. Springer, Heidelberg, Dec 1999.
11. Mengjie Zhang, Will Smart. “Multiclass Object Classification Using Genetic Programming“. Technical Report CS-TR-04/2, Feb 2004, School of Mathematical and Computing Sciences, Victoria University.
12. Loveard, T. & Ciesielski, V. Representing classification problems in genetic programming, in 'Proceedings of the Congress on Evolutionary Computation', Vol. 2, IEEE Press, COEX, World Trade Center, 159 Samseongdong, Gangnam-gu, Seoul, Korea, pp. 1070—1077 (2001). http://goanna.cs.rmit.edu.au/toml/cec2001.ps
13. B.-C. Chien, J. Y. Lin, and T.-P. Hong, “Learning discriminant functions with fuzzy attributes for classification using genetic programming,” Expert Syst. Applicat., vol. 23, pp. 31–37, 2002.
14. R. R. F. Mendes, F. B.Voznika, A. A. Freitas, and J. C. Nievola, “Discovering fuzzy classification rules with genetic programming and co-evolution,” in Lecture Notes in Artificial Intelligence, vol. 2168, Proc. 5th Eur. Conf. PKDD, 2001, pp. 314–325.
15. Durga Prasad Muni, Nikhil R. Pal, Senior Member, IEEE, and Jyotirmoy Das, “A Novel Approach to Design Classifiers Using Genetic Programming », ieee transactions on evolutionary computation, vol. 8, no. 2, pp. 183-196. April 2004.
16. Crosbie, Mark and Spafford, Gene, Applying Genetic Programming Techniques to Intrusion Detection, In Proceedings of the AAAI 1995 Fall Symposium, November 1995.
17. Bob Adolf . New Paradigms for Intrusion Detection Using Genetic Programming. Technical report January 2004.
18. Cosbie M. Gene Spafford. Applying genetic programming to intrusion detection October 1998. In procedding of the 18 th NISSC Conference October 1998.
19. R. A. Fisher, “The use of multiple measurements in taxonomic problems,” Ann. Eugenics, pt. II, vol. 7, pp. 179–188, 1936.

20. KDD data set, 1999; http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, cited April 2003
21. C. Elkan, "Results of the KDD'99 Classifier Learning", SIGKDD Explorations, ACM SIGKDD, Jan 2000.
22. Lee W. and Stolfo S. A Framework for Constructing Features and Models for Intrusion Detection Systems. Information and System Security, 3(4):227–261, 2000.
23. Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context, Maheshkumar Sabhnani, Gursel Serpen, Proceedings of the International Conference on Machine Learning, Models, Technologies and Applications (MLMTA 2003), Las Vegas, NV, June 2003, pages 209-215.
24. Ayse Küçükyılmaz; Pattern Classification: A Survey and Comparison. Department of Computer Engineering, Bilkent University, 06800, Ankara, Turkey. http://www.cs.bilkent.edu.tr/~guvenir/courses/cs550/Workshop/Ayse_Kucukyilmaz.pdf . April 7, 2005
25. Anil K. Jain, Robert P.W. Duin, and Jianchang Mao, Statistical pattern recognition: a review, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22. No. 1, pp. 4-37, January 2000.
26. R. Agarwal, and M. V. Joshi, "PNrule: A New Framework for Learning Classifier Models in Data Mining", Technical Report TR 00-015, Department of Computer Science, University of Minnesota, 2000.

Article received: 2006-01-02