Genetic Programming Approach for Multi-Category Pattern Classification Applied to Network Intrusions Detection

Kamel Faraoun[1] and Aoued Boukelif[2]
[1]Evolutionary Engineering and Distributed IS Laboratory, University of Sidi Bel Abbès, Algeria
[2]Communication Networks, Architectures, and Multimedia Lab, University of Sidi Bel Abbès, Algeria

Abstract: *This paper describes a new approach of classification using genetic programming. The proposed technique consists 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 then 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.*

Received December 28, 2005; accepted April 21, 2006

1. 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 [9]. 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.

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 Genetic Programming (GP)-classification approach to perform network intrusion detection.

Section 1 gives some theoretical background about genetic programming approaches and related works. Section 2 explains the method developed in the present work with its different elements and parameters. In 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 section 5. The paper is finally concluded with a summary of the most important points and future works.

2. Theory

2.1. Genetic Programming Paradigm

Genetic programming is an extension of genetic algorithms [12]. 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 step 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.

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.

2.2. Genetic Programming and Classification Task

Generally, GP trees can perform classification by returning numeric (real) values and then translating these values into class labels [25]. 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 belongs to that class) or -1 (indicating that the input sample does 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:

$$\begin{aligned} &T(X_i) \le 0 \text{ if } X_i \in \text{Class 1} \quad (D = +1) \\ &T(X_i) > 0 \quad \text{otherwise} \quad (D = -1) \end{aligned} \tag{1}$$

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:

$$\text{Fitness}(T) = \frac{\text{No. of samples classified correctly}}{\text{No. of samples used for training}} \tag{2}$$

Each genetic expression 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 [26], 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:

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

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 class's 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 [26].

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 prior 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.

2.3. 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 [16], 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.* [3] used GP to generate discriminator functions using arithmetic operations with fuzzy attributes for a classification problem. In [18], 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.* [11] 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 [19], 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.

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, which 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 [23] 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, and the same author introduced hierarchical RSS-DSS algorithm for dynamically filtering large datasets to enhance the system performances in [22]. Less important works can be found in [1, 4], and with the Chimera model [5].

3. Proposed GP-Classification Approach

The present work propose a new approach of a dynamic GP-based classifier which consists 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 belongs 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:

$$D_{Train} = \bigcup_{j \le n} D_{Train}^{j}$$
$$D_{Train}^{j} = \left\{ X_i^j \in D_{Train} \,/\, class(X_i^j) = j \right\} \quad (4)$$

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

$$T(D_{Train}^{j}) = \left\{ Y = T(X_i^j) \;/\; X_i^j \in D_{Train}^{j} \right\} \quad (5)$$

The classification approach assigns 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:

$$Classe\ (Y) = \begin{cases} Classe\ 1 & if\ T(Y) \in T(D_{Train}^{1}) \\ Classe\ 2 & if\ T(Y) \in T(D_{Train}^{2}) \\ \ldots\ldots & \\ Classe\ n-1 & if\ T(Y) \in T(D_{Train}^{n-1}) \\ Classe\ n & if\ T(Y) \in T(D_{Train}^{n}) \end{cases} \quad (6)$$

If the value of T (Y) dose not appears in any set $T(D_{Train}^{j})$, we assign Y to the nearest class using the algorithm presented below in Figure 2.

We can see that the proposed classification method transforms the problem from an N-dimensional vectors classification to a 1-dimentional values classification one. The classification of the transformed vectors becomes much easier, but this is assured if a maximum discrimination exists between the sets T (D_{Train}^{i}). It is the role of the genetic programming system to assure such criteria, the fitness of each transformation T depends 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. In the following, we present the different steps of the classification approach: Learning phase, which search for the best transformation of the raining data D_{Train}, and test phase that classifies each test sample from a set of new vectors D_{Test}.

3.1. Learning Phase

3.1.1. 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.

3.1.2. Fitness Function

For a given training set $D_{Train} \subset R^N$, the genetic programming system evolves a population of transformations T_i. 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.

For this reason, we have proposed a new fitness function formula, which 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 becomes easier. The fitness value is computed by:

$$Fitness\ (T) = \frac{Card(\bigcap_{i \le n} T(D_{Train}^{i}))}{Card(T(D_{Train}))} \quad (7)$$

When the function Card (X) gives the cardinality of a given set X. The performed experiments show that this function gives the best classification rates with respect to other fitness function mentioned above.

3.1.3. Genetic Operators and Parameters

The standard crossover and mutation operators presented below 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 stops when it reach a given generations count (termination criteria). Table 1 gives 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.

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

3.2. Test Phase: Classifying 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 operates 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 presented in Figure 1. Then, for each new sample Y_i form D_{Test}, the corresponding class is determined using the algorithm presented in Figure 2.

Table 1. Set of parameters used to control the genetic evolution process.

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] ∪ {Input Variables}

```
Let T (D_Train) be the training set;
Dens: Array ofdensity 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 1. A Post-treatment algorithm to generate density array, used during the testing phase.

```
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 2. 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.

4. Datasets and Experiments

The proposed classification approach is benchmarked using two different datasets: Fisher IRIS [8] dataset and MIT KDD99 dataset [17]. 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 concerns our problem of interest: The network intrusion detection. The most important work on GP-classification using the KDD99 dataset is the one presented in [22, 23] 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. Table 2 lists the class's distributions of our used sets.

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

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 [6]. The resulting scaled fields belong to the interval [0, 1].

The 41 fields used in the KDD99 dataset were labelled with a symbolic notation (F_1, F_2, …, F_{41}), respectively, to be used as terminals during the genetic process.

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

$$Detection\ Rate\ \ DR = 1 - \frac{False\ Negatives\ Number}{Total\ Number\ of\ Attaks}$$

$$False\ Positive\ Rate\ FP = \frac{False\ Positives}{Total\ Number\ of\ Normal\ Connections} \quad (8)$$

5. 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 Table 1.

5.1. Fisher IRIS Classification Problem

The dataset was divided equally into a training set and a test validation set (75 samples in each set). The classification rate is then computed using the following expression:

$$CR = \frac{\text{Number of Samples Classified Correctly}}{\text{Number of Samples Used for Training}} * 100 \quad (9)$$

Table 3 gives a comparison between the detection rate obtained with different classifiers as presented in [11, 13], and our proposed classification approach.

Table 3. A summary of the detection rates obtained using different classifiers for the Fisher's Iris dataset.

Method	Classification Rate
NN	96 %
Naive Bayse	96 %
Bayse Net	94.667 %
C4.5	94.67 %
GPCE [1]	96 %
Maximum Likelihood	97.3 %
Proposed GP-Classification	98.6 %

Figure 3 shows the distribution of the transformed training set T (D_{Train}) obtained with this transformation.

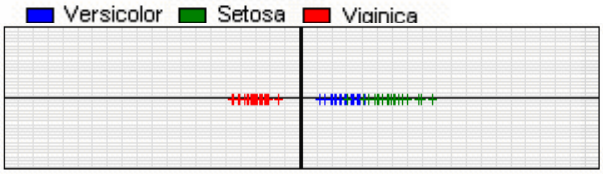

Figure 3. Distribution of the transformed training set T (D_{Train}) of the best obtained individual.

From Table 3, we can see that our proposed approach gives best classification rate compared to other proposed approaches, one sample only from the Viginica set is misclassified.

5.2. KDD99 Dataset: The Intrusion Detection Problem

As we see in Table 2, the KDD99 dataset is more voluminous than the Iris Fisher's one, and contains 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 Table 1 is used. In Table 4, we present the classification matrix obtained. Figure 4 illustrates the transformed training set T (D_{Train}) repartition, 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)))))

Table 4. Classification matrix obtained using the proposed approach.

	Normal	Prob	Dos	U2R	R2L	%
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%	

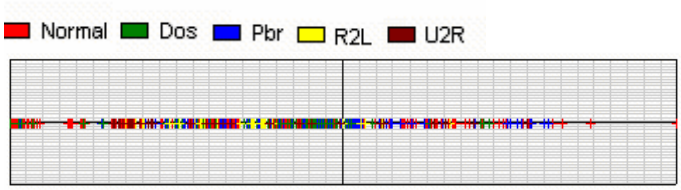

Figure 4. Distribution of the transformed training set T(D_{Train}) of the best individual.

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 %

Table 5 summarizes and compares the detection rates and false positive rates obtained using the approaches mentioned above, and some recent results on KDD benchmark presented in [7, 10]. All the mentioned approaches were tested using the 'corrected (test)' set of the KDD99 cup competition.

We can see from the presented results that the proposed classification approach gives 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 datasets, 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.

6. GP-Classifier Enhancement: Multi-Transformations Classification System

6.1. Method Description

The main idea of the multi-transformation system is to use a set of multiple transformations TR_{set} = {T_1, T_2, …., 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:

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

When:

Ne: Is the nearest point from the T (D_{Train}) set.
Class_Y: Is the deduced class for the sample Y.
$Card(T(D_{Train}^{Class\_Y}))$: Is the number of the samples from the training set belonging to Class_Y.
Fitn (T): Is the fitness value of the transformation T.
Dens (Ne, Class_Y): Is the density value computed by the algorithm of Figure 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]. The equation 10 was introduced in the algorithm presented in Figure 5.

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 takes the best individuals collected during the genetic evolution to construct a transformations set TR_{set} = {T_1, T_2, …., 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 as shown in Figure 6.

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 assigns to Y the class with the highest membership factor:

Class (Y) = c, such that:

$$Confidence\ (Y,c) = \max_{1 \le i \le n} (confidence\ (Y,i)) \quad (11)$$

Table 5. Comparison of the detection performances between the proposed approaches and the existing techniques.

Classification Method	Detection Rate	False Alarm Rate
GP-Classifier (Proposed)	92.5 %	1.35 %
KDD99 Wining Entry [15]	91.0 %	0.50 %
KDD99 Second Place [24]	91.5 %	0.58 %
Linear GP Classifier [22, 23]	90.8 %	3.26 %
Data-Mining Techniques [14]	70%-90%	2.00 %
Support Vector Machine [7]	98.0 %	10.00 %
Self Organized Maps [10]	89.0 %	4.60 %
Clustering Techniques [7]	93.0 %	10.00 %
K-Nearest Neiberhood [7]	91.0 %	8.00 %

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

6.2. Results and Comparison

This section summarizes 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 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 %

In Table 6, obtained classification rates using the multi-transformations classification system are compared to the results presented in [21] 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 show 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 positive detection rate of each attack class is not available for the SOM [10] and the linear GP [22, 23] techniques, since they are 2-category classifiers (normal and attack), their false positive rates can be given only in term of whole attacks classification.

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 which illustrated in Table 7.

Table 6. Comparison of the classification rates obtained with different techniques using the KDD99 dataset.

Classification Method	Dos		Prob	
	DR	FP	DR	FP
KDD Cup Winner [20]	0.971	0.003	0.833	0.006
Agrawal and Joshi [1]	0.969	0.001	0.730	8E-5
GP 1-Transformation	0.960	7E-4	0.826	0.010
Multi-Transformations	0.988	1E-4	0.972	0.003
SOM Map [10]	0.951	-	0.643	-
Linear GP [22, 23]	0.967	-	0.857	-
Multilayer Perceptron [21]	0.972	0.003	0.887	0.004
Gaussian Classifier [21]	0.824	0.009	0.902	0.113
K-Means Clustering [21]	0.973	0.004	0.876	0.026
Nearest Cluster [21]	0.971	0.003	0.888	0.005
Radial Basis Function [21]	0.730	0.002	0.932	0.188
Leader Algorithm [21]	0.972	0.003	0.838	0.003
Hypersphere Algo.[21]	0.972	0.003	0.848	0.004
Fuzzy ARTMAP [21]	0.970	0.003	0.808	0.007
C4.5 Decision Tree [21]	0.970	0.003	0.808	0.007

Table 7. Comparison of the classification rates obtained using R2L and U2R classes.

Classification Method	R2L		U2R	
	DR	FP	DR	FP
KDD Cup Winner [20]	0.084	5E-5	0.123	3E-5
Agrawal and Joshi [1]	0.107	8E-4	0.066	4E-5
GP 1-Transformation	0.271	7E-4	0.100	0.006
Multi-Transformations	0.802	1E-4	0.452	3E-4
SOM Map [10]	0.113	-	0.229	-
Linear GP [22, 23]	0.093	-	0.013	-
Multilayer Perceptron [21]	0.056	1E-4	0.132	5E-4
Gaussian Classifier [21]	0.096	0.001	0.228	0.005
K-Means Clustering [21]	0.064	0.001	0.298	0.004
Nearest Cluster [21]	0.034	1E-4	0.022	6E-6
Radial Basis Function [21]	0.059	0.003	0.061	4E-4
Leader Algorithm [21]	0.001	3E-5	0.066	3E-4
Hypersphere Algo.[21]	0.010	5E-5	0.083	9E-5
Fuzzy ARTMAP [21]	0.037	4E-5	0.061	1E-5
C4.5 Decision Tree [21]	0.046	5E-5	0.018	2E-5

7. Conclusion

In this work, a new GP-classification system with a dynamic class's 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. 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 our proposed method shows that increasing the number of combined transformations enhance significantly the system performances.

References

[1] Adolf B., "New Paradigms for Intrusion Detection Using Genetic Programming," *Technical Report*, Northwestern University Technological Institute of Evanston, Illinois, USA, January 2004.

[2] Agarwal R. and Joshi M. V., "PNrule: A New Framework for Learning Classifier Models in Data Mining," *Technical Report*, TR 00-015, Department of Computer Science, University of Minnesota, 2000.

[3] Chien B. C., Lin J. Y., and Hong T. P., "Learning Discriminant Functions with Fuzzy Attributes for Classification Using Genetic Programming," *Expert Systems Applications*, vol. 23, no. 1 , pp. 31-37, 2002.

[4] Crosbie M., and Spafford G., "Applying Genetic Programming Techniques to Intrusion Detection," *in Proceedings of the AAAI'1995 Fall Symposium*, Cambridge, Massachusetts, pp. 1-8, November 1995.

[5] Cosbie M. and Spafford G., "Applying Genetic Programming to Intrusion Detection," *in Proceeding of the 18th NISSC Conference*, Baltimore, USA, pp. 194-204, October 1998.

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

[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 Barbara D. and Jajodia S. (Eds), *Applications of Data Mining in Computer Security*, Kluwer, 2002.

[8] Fisher R. A., "The Use of Multiple Measurements in Taxonomic Problems," *Annals of Eugenics part II*, vol. 7, pp. 179-188, 1936.

[9] Jain A. K., Duin R. P. W., and Mao J., "Statistical Pattern Recognition: A Review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4-37, January 2000.

[10] 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*, Portland, USA, pp. 1808-1813, 2003.

[11] Kishore J. K., Patnaik L. M., Mani V., and Agrawal V. K., "Application of Genetic Programming for Multicategory Pattern Classification," *IEEE Transactions Evolutionary Computation*, vol. 4, no. 3, pp. 242-258, September 2000.

[12] Koza J. R., *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press, 1994.

[13] Küçükyılmaz A., "Pattern Classification: A Survey and Comparison," *Technical Report*, Department of Computer Engineering, Bilkent University, Ankara, Turkey, April 2005.

[14] Lee W. and Stolfo S., "A Framework for Constructing Features and Models for Intrusion Detection Systems," *Information and System Security*, vol. 3, no. 4, pp. 227-261, 2000.

[15] Levin I., "KDD-99 Classifier Learning Contest LLSoft's Results Overview," *SIGKDD Explorations*, *ACM SIGKDD*, vol. 1, no. 2, pp. 67-75, 2000.

[16] Loveard T. and Ciesielski V., "Representing Classification Problems in Genetic Programming," *in Proceedings of the Congress on Evolutionary Computation*, Korea, vol. 2, pp. 1070-1077, 2001.

[17] Matthew V. Mahoney P. K. C., An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection," *in Proceeding of Recent Advances in Intrusion Detection (RAID'2003)*, 2003.

[18] Mendes R. R. F., Voznika F. B., Freitas A. A., and Nievola J. C., "Discovering Fuzzy Classification Rules with Genetic Programming and Co-Evolution," *in Proceedings of the 5th European Conference PKDD*, Freiburg, Germany, pp. 314-325, 2001.

[19] Muni D. P., Pal N. R., and Das J., "A Novel Approach to Design Classifiers Using Genetic Programming," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 183-196, April 2004.

[20] Pfahringer B., "Winning the KDD99 Classification Cup: Bagged Boosting," *ACM SIGKDD*, vol. 1, no. 2, pp. 65-66, 2000.

[21] Sabhnani M. and Serpen G., "Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection

Context," *in Proceedings of the International Conference on Machine Learning, Models, Technologies and Applications (MLMTA'2003)*, Las Vegas, USA, pp. 209-215, June 2003.

[22] Song D., Heywood M. I., and Zincir-Heywood N., "A Linear Genetic Programming Approach to Intrusion Detection," in Cantú-Paz E., Cantu-Paz E., James A., and Foster K. D. (Eds), GECCO 2003, LNCS 2724, Springer-Verlag, Berlin Heidelberg, pp. 2325-2336, 2003.

[23] Song D., Heywood M. I., and Zincir-Heywood N., "Training Genetic Programming on Half a Million Patterns: An Example from Anomaly Detection," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp 225-240, 2005.

[24] Vladimir M., Alexei V., and Ivan S., "The MP13 Approach to the KDD'99 Classifier Learning Contest," *SIGKDD Explorations, ACM SIGKDD*, vol. 1, no. 2, pp. 76-77, 2000.

[25] Zhang M. and Ciesielski V., "Genetic Programming for Multiple Class Object Detection," *in Proceedings of the 12th Australian Joint Conference on Artificial Intelligence*, vol. 1747, pp. 180-191, December 1999.

[26] Zhang M. and Smart W., "Multiclass Object Classification Using Genetic Programming," *Technical Report*, CS-TR-04/2, School of Mathematical and Computing Sciences, Victoria University, February 2004.

Kamel Faraoun received his Master's degree in computer science from Djilali Liabbes University Sidi-Bel Abbes, Algeria in 2002. Currently, he is a lecturer at the Computer Sciences Institute of Djilali Liabess University, and preparing his PhD thesis. He has published several papers in international journals. He is a member of the Evolutionary Engineering and Distributed Information Systems Laboratory (EEDIS). His research interests include computer safety systems, genetic algorithms, fractal images compression, evolutionary programming, grammatical inferences, and physical materials structures modeling.

Aoued Boukelif received his Bachelor of science in electrical engineering from the University of Pittsburgh and an honour degree in electrical engineering, image processing option. Currently, he is an assistant professor at the University of Sidi Bel Abbes and head of a research team dealing with information and communication technologies applied to distance learning. He is the author of many publications, he is a member of the Communication Networks, Architectures and Multimedia Laboratory at Sidi bel Abbes University. His research interests include digital television, digital image compression, satellite communications information, and communication technologies.