Constructing features using a hybrid genetic algorithm

Ioannis G. Tsoulos*

Department of Informatics and Telecommunications, University of Ioannina, Greece

Abstract

A hybrid procedure that incorporates Grammatical Evolution and a weight decaying technique is proposed here for various problems, classification or regression. The proposed method has two main phases: the creation of features and the evaluation of these features. During the first phase, using Grammatical Evolution, new features are created as non-linear combinations of the original features of the datasets. In the second phase, based on the characteristics of the first phase, the original data set is modified and a neural network trained with a genetic algorithm is applied to this dataset. The proposed method was applied to an extremely wide set of datasets from the relevant literature and the experimental results were compared with 4 other techniques.

Keywords: Genetic algorithm, machine learning, neural networks, Grammatical Evolution.

1 Introduction

Artificial Neural networks (ANNs) are programming tools [1, 2], based on a series of parameters that are commonly called weights or processing units. They have been used in a variety of problems from different scientific areas, such as physics [3, 4, 5], chemistry [6, 7, 8], economics [9, 10, 11], medicine [12, 13] etc. A common way to express a neural network is a function $N(\overrightarrow{x}, \overrightarrow{w})$, with $\overrightarrow{x}$ the input vector (commonly called pattern) and $\overrightarrow{w}$ the weight vector. A method that trains a neural network should be used to estimate the vector $\overrightarrow{w}$ for a certain problem. The training procedure can be formulated also as an optimization problem, where the target is to minimize the so-called error function:

$$E\left(N\left(\overrightarrow{x}, \overrightarrow{w}\right)\right) = \sum_{i=1}^{M}\left(N\left(\overrightarrow{x}_i, \overrightarrow{w}\right) - y_i\right)^2 \qquad (1)$$

*Corresponding author. Email: itsoulos@uoi.gr

In equation 1 the set $(\overrightarrow{x_i}, y_i)$, $i = 1, ..., M$ is the dataset used to train the neural network, with y_i being the actual output for the point $\overrightarrow{x_i}$. The neural network form used here is considered also in [14]. Suppose we have a neural network with a processing level that uses the sigmoid function as an output function. Every output of the network is defined as:

$$o_i(x) = \sigma\left(p_i^T x + \theta_i\right), \qquad (2)$$

with p_i the weight vector and θ_i the bias for the output i. For a neural network with H hidden nodes the final output function can be written:

$$N(x) = \sum_{i=1}^{H} v_i o_i(x), \qquad (3)$$

with v_i the output weight for processing unit i. Hence, by using one vector for all the parameters (weights and biases) the neural network can be written in fthe following form:

$$N\left(\overrightarrow{x}, \overrightarrow{w}\right) = \sum_{i=1}^{H} w_{(d+2)i-(d+1)}\sigma\left(\sum_{j=1}^{d} x_j w_{(d+2)i-(d+1)+j} + w_{(d+2)i}\right) \qquad (4)$$

where H is the number of processing units of the neural network and d is the dimension of vector $\overrightarrow{x}$. The function $\sigma(x)$ is the sigmoid function defined as:

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \qquad (5)$$

From the equation 4 one can obtain that the dimension of the weight vector w is computed as: $w = (d + 2)H$. The function of equation 1 has been minimized with a variety of optimization methods during the past years, such as: the Back Propagation method [15, 16], the RPROP method [17, 18, 19], Quasi Newton methods [20, 21], Particle Swarm Optimization [22, 23] etc. All the previously mentioned methods have to overcome two major problems:

- Excessive computational times, because they require processing time proportional to the dimension of the objective problem and the number of processing units as well. For example, a neural network of $H = 10$ processing units applied to a test data with $d = 3$, is considered as an optimization problem with dimension $w = (d + 2)H = 50$. This means that the total number of network parameters is growing extremely fast, which results in a longer computation time than the corresponding universal optimization method. An extensive discussion on the problems caused by the dimensionality of neural networks is presented in [24]. A common approach to overcome this problem is to use the PCA technique to reduce the dimensionality of the objective problem [25, 26, 27] i.e. the parameter d.

- The overfitting problem. It is quite common for these methods to produce poor results when they are applied to data (test data) not previously used in the training procedure. This problem is discussed in detail in the article by Geman et all [28] as well as in the article of Hawkins [29]. A variety of methods have been proposed to overcome this problem, such as weight sharing [30], pruning [31, 32, 33], the dropout technique [34], early stopping [35, 36], and weight decaying [37, 38].

This article proposes a method that tackle both the above problems using two major steps. During the first step a new set of features is created from the initial features using a procedure based on the Grammatical Evolution technique [39]. A feature is a measurement that defines a property of the objective problem and the series of all measurements forms a pattern. A feature can be a integer value, a double precision value or even a string literature. In our case we consider only numeric values for the features. The number of features of each pattern is the dimensionality of the problem defined as d in this work. The procedure of Feature Construction with Grammatical Evolution was introduced in the work of Gavrilis et al [40] and it has been used with success in Spam Identification [41], Fetal heart classification [42], epileptic oscillations in clinical intracranial electroencephalograms [43] etc. The outcomes of the first phase are the modified training and testing data according to the created features. During the second step, a genetic algorithm that incorporates a weight decaying procedure is used to train a neural network on the modified data of the first step.

Genetic algorithms are methods based on biological observations such as reproduction and mutation [44, 45]. The genetic algorithms work by creating and maintaining a population of candidate solutions (chromosomes). This population is altered iteratively though operations such as crossover and mutation until some stopping criteria are met. They have many advantages, such as simplicity in the implementation, endurance in noise, they can be easily parallelized etc. Also, they have been applied to many problems, such as aerodynamic optimization [46], steel structure optimization [47], brain images [48] etc. They have been used to train neural networks in various research papers, such as the work of Leung et al [49] which estimates the structure and the weights of a neural network through a genetic algorithm, the evolution of a neural networks for daily rainfall–runoff forecasting [50], the evolution of neural networks to predict the deformation modulus of rock masses [51] etc.

The idea of feature construction has been examined by various researchers in the relevant literature, such as the work of Smith and Bull [52], who used tree genetic programming approach to construct features from the original ones. Another approach to constructing features using genetic programming was proposed by Neshatian et al [53], where the genetic programming utilizes an entropy based fitness function that maximizes the purity of class intervals. Another evolutionary approach was proposed by Li and Yin for feature selection using gene expression data [54]. Lastly, a recent work that utilizes a genetic programming approach and the information gain ratio (IGR) was proposed by Ma and Teng[55] to construct features from the original ones.

In the problems of classification and regression, as the number of features increases, additional examples are needed in order to achieve good results in training a model but also to maintain good generalization skills in unknown data. Of course, adding new examples to the training process is almost never possible and this results in poor performance in the control data. For this reason, the original data set must be transformed into a new one, which gives better generalization skills to the learning models. According to the Cover's theorem[56], there is at least one non-linear extension of the original feature vector, so that with this extension a linear separation of the set of patterns can be made. Many techniques have been proposed in this direction that try to detect such nonlinear extensions. The proposed method uses a hybrid approach, in which first new features are constructed using Grammatical Evolution and then these features are evaluated by a neural network that appropriately trains a genetic algorithm. In the first phase, the creation of new features is done in such a way as to achieve the best possible learning accuracy. The methods that can be used to convert attributes are grouped into three categories: feature selection, feature construction, and feature reduction. The second case is the most difficult, as it does not simply require reducing the size of the problem, but also the non-linear creation of new features from old ones.

The proposed technique can overcome other techniques from the modern literature, as it does not require prior knowledge of the objective problem, it can be applied with the exact same procedure to both categorization problems and function learning problems. In addition, it can be used to discover hidden function dependencies between the original features of the problem and, because it is based on Grammatical Evolution, the user can add and subtract functions or even allow the algorithm to construct new functions to better learn the data set. Still, the final characteristics of the method can be evaluated by any computational intelligence model without any additional processing. In the present method, these characteristics are evaluated by an artificial neural network, but something that could change.

The rest of this paper is organized as follows: in section 2 the proposed method is described in detail, in section 3 the proposed method is tested on a series of well known datasets from the relevant literature and the results are compared to those of a simple genetic algorithm and finally in section 4 some conclusions are presented.

2 Method description

The proposed method has two major phases. In the first phase, a procedure that exploits the Grammatical Evolution technique is used, in order to create new features from the old ones. The new features are evaluated using a Radial Basis Function (RBF) [57] neural network with H hidden nodes. The RBF network is used during this phase instead of a neural network because the training procedure for RBF networks are must faster than those of neural networks. In the second phase, a hybrid genetic algorithm trains a neural network using the constructed features of the first phase.

2.1 The usage of Grammatical Evolution

Grammatical evolution is an evolutionary procedure where the chromosomes represent production rules from a BNF (Backus–Naur form) grammar[58], which is used very often to describe the syntax of programming languages, document formats etc. These grammars are defined as the set $G = (N, T, S, P)$, where

- N is the set of non-terminal symbols, which produce through production rules series of terminal symbols.
- T is the set of terminal symbols.
- S is a non-terminal symbol also called the start symbol
- P is a set of production rules in the form $A \to a$ or $A \to aB,\ A, B \in N,\ a \in T$.

The production procedure initiates from the start symbol of the BNF grammar and iteratively produces programs by replacing non-terminal symbols with the right hand of the production rules that will be selected according to the value of each element in the chromosome. In the proposed method, the BNF grammar of Figure 1 was used to create a new feature from the initial features. The symbols that are in <> are considered as non-terminal symbols. The parameter N denotes the number of original features. Typically, a chromosome x in grammatical evolution is expressed as a series of binary values 0 or 1. In the current work, in order to simplify the mapping procedure and to increase the speed of the algorithm, every element of each chromosome is considered as an integer in a predefined range. In our case, the range $[0, 255]$ was used but of course, this could be easily changed.

Take, for example, the chromosome $x = [9, 8, 6, 4, 16, 10, 17, 23, 8, 14]$ and $N = 3$. The valid expression $f(x) = x_2 + \cos(x_3)$ is created using a series of production steps shown in Table 1. An expression is considered as valid if it contains only terminal symbols. Each number in the parentheses stands for the sequence number of the production rule. Hence, the process to produce N_f features from the original is as follows:

1. Every chromosome Z is split into N_f parts. Each part g_i will be used to construct a feature.
2. For every part g_i construct a feature t_i using the grammar given in Figure 1
3. Create a mapping function

$$G(\overrightarrow{x}, Z) = \left(t_1(\overrightarrow{x}, Z), t_2(\overrightarrow{x}, Z), \ldots, t_{N_f}(\overrightarrow{x}, Z)\right) \quad (6)$$

where $\overrightarrow{x}$ is a pattern from the original set and Z is the chromosome.

Figure 1: BNF grammar of the proposed method.

```
S::=<expr>   (0)
<expr> ::=  (<expr> <op> <expr>)  (0)
           | <func> ( <expr> )    (1)
           |<terminal>            (2)
<op> ::=     +      (0)
           | -      (1)
           | *      (2)
           | /      (3)
<func> ::=   sin  (0)
           | cos  (1)
           |exp   (2)
           |log   (3)
<terminal>::=<xlist>                (0)
           |<digitlist>.<digitlist> (1)
<xlist>::=x1    (0)
           | x2 (1)
           .........
           | xN (N)
<digitlist>::=<digit>                  (0)
           | <digit><digit>            (1)
           | <digit><digit><digit>     (2)
<digit>  ::= 0 (0)
           | 1 (1)
           | 2 (2)
           | 3 (3)
           | 4 (4)
           | 5 (5)
           | 6 (6)
           | 7 (7)
           | 8 (8)
           | 9 (9)
```

Table 1: Steps to produce a valid expression from the BNF grammar.

String	Chromosome	Operation
<expr>	9,8,6,4,16,10,17,23,8,14	$9 \mod 3 = 0$
(<expr><op><expr>)	8,6,4,16,10,17,23,8,14	$8 \mod 3 = 2$
(<terminal><op><expr>)	6,4,16,10,17,23,8,14	$6 \mod 2 = 0$
(<xlist><op><expr>)	4,16,10,17,23,8,14	$4 \mod 3 = 1$
(x2<op><expr>)	16,10,17,23,8,14	$16 \mod 4 = 0$
(x2+<expr>)	10,17,23,8,14	$10 \mod 3 = 1$
(x2+<func>(<expr>))	17,23,8,14	$17 \mod 4 = 1$
(x2+cos(<expr>))	23,8,14	$23 \mod 2 = 1$
(x2+cos(<terminal>))	8,14	$8 \mod 2 = 0$
(x2+cos(<xlist>))	14	$14 \mod 3 = 2$
(x2+cos(x3))		

2.2 Feature construction

The first phase of feature construction presented below, has also been used as a feature construction mechanism in the initial work of Gavrilis et al[40]. In this phase a genetic algorithm constructs new features from the original and as a fitness functionthe training error of an RBF network on the data set created with the new features is used. The steps of the algorithm for the first phase are:

1. **Initialization step**

   (a) **Set** iter=0, generation number.

   (b) **Construct** the set $\text{TR} = \left\{ \left(\overrightarrow{x_1}, y_1\right), \left(\overrightarrow{x_2}, y_2\right), \ldots, \left(\overrightarrow{x_M}, y_M\right) \right\}$, which is the original train set.

   (c) **Set** N_c the number of chromosomes and N_f the number of desired constructed features. These options are defined by the user.

   (d) **Initialize** randomly in range $[0, 255]$ the integer chromosomes $Z_i, i = 1 \ldots N_c$

   (e) **Set** N_g as the maximum number of generations allowed.

   (f) **Set** $p_s \in [0, 1]$ as the selection rate and $p_m \in [0, 1]$ the mutation rate.

2. **Termination check. If** iter>=N_g **goto** step 6.

3. **Estimate** the fitness f_i of every chromosome Z_i with the following procedure:

   (a) **Use** the procedure described in subsection 2.1 and create N_f features.

   (b) **Create** a modified training set

$$\text{TN} = \left\{ \left(G\left(\overrightarrow{x_1}, Z_i\right), y_1\right), \left(G\left(\overrightarrow{x_2}, Z_i\right), y_2\right), \ldots, \left(G\left(\overrightarrow{x_M}, Z_i\right), y_M\right) \right\} \quad (7)$$

(c) **Train** an RBF neural network C with H processing units on the modified training set TN using the following train error

$$f_i = \sum_{j=1}^{M} \left(C\left(G\left(\overrightarrow{x_j}, Z_i\right)\right) - y_j\right)^2 \tag{8}$$

4. **Genetic Operators**

   (a) **Selection procedure:** Initially, the chromosomes are sorted according to their fitness value. The best chromosomes are placed in the beginning of the population and the worst at the end. The best $(1-p_s) \times N_c$ chromosomes are transferred to the next generation intact. The remaining chromosomes are substituted by offsprings created through the crossover and mutation procedures.

   (b) **Crossover procedure:** In this process, for every produced offspring, two mating chromosomes (parents) are selected from the previous population using tournament selection. The tournament selection is a rather simple selection mechanism defined as: First a set of $K > 1$ randomly selected chromosomes is constructed and subsequently the chromosome with the best fitness value in the previous set is selected as the mating chromosome. Having selected the two parents for the offrpsing, the offspring is formed using the one point crossover. In one - point crossover a random point is selected for the two parents and their right-hand side subchromosomes are exchanged.

   (c) **Mutation procedure:** For every element of each chromosome a random number $r \in [0,1]$ is taken. If $r \leq p_m$ then this element is altered randomly by producing a new integer number.

5. **Set** iter=iter+1 and **goto** Step 2.

6. **Get** the best chromosome in the population defined as Z_l with the corresponding fitness value f_l and **Terminate**.

2.3 Weight decay mechanism

The quantity x in the equation 5 of the sigmoid function is calculated through many calculations involving the input patterns as well as the weight vector. If the value within the function is too large, then the sigmoid function tends to 1 and this will result in the neural network losing what generalization possibilities it has. In order to estimate the effect of this issue, the quantity $B\left(N\left(\overrightarrow{x}, \overrightarrow{w}\right), F\right)$ is defined as shown in the Algorithm 1.

Algorithm 1 Calculation of the bounding quantity for neural network $N(x,w)$.

```
1. Define $b = 0$
2. For $i = 1..K$ Do
    (a) For $j = 1..M$ Do
        i. Define $v = \sum_{kT=1}^{d} w_{(d+2)i-(d+i)+k} x_{jk} + w_{(d+2)i}$
        ii. If $|v| > F$ set $b = b + 1$
    (b) EndFor
3. EndFor
4. Return $\frac{b}{K \star M}$
```

2.4 Application of genetic algorithm

The following is a hybrid genetic algorithm used to train artificial neural networks in the modified dataset. The purpose of this algorithm is to train the artificial neural network in such a way that it does not lose its generalizing abilities. For this purpose, it uses a fitness function that consists of the neural network training error, but also a punitive factor is added. This penalty factor aims to ensure that network weights do not get too high values during training. This technique can be applied directly to a neural network without having previously performed the first phase of feature construction. The main steps of the hybrid genetic algorithm used in the second phase are:

1. **Initialization step**
   (a) **Set** iter=0, the generation number.
   (b) **Set** TN the modified training set, where
   $$\mathrm{TN} = \left\{ \left(G\left(\overrightarrow{x_1}, Z_l\right), y_1\right), \left(G\left(\overrightarrow{x_2}, Z_l\right), y_2\right), \ldots, \left(G\left(\overrightarrow{x_M}, Z_l\right), y_M\right) \right\} \quad (9)$$
   (c) **Initialize** randomly the double precision chromosomes $D_i, i = 1 \ldots N_c$ in range $[L_N, R_N]$. The size of each chromosome is set to $W = (N_f + 2)\, H$
2. **Termination check. If** iter>=N_g **goto** step 6
3. **Fitness calculation step**.
   (a) **For** every chromosome D_i
      i. **Calculate** the quantity $B_i = \sum_{x \in \mathrm{TN}} \left(B\left(N\left(x, D_i\right), F\right)\right)$ using the algorithm 1.

```
            ii. Calculate the quantity $E_i = \sum_{(x,y)\in \mathrm{TN}} \left(N\left(x, D_i\right) - y\right)^2$ , the training error of the neural network where the chromosome $D_i$ is used as the weight vector.
            iii. Set $f_i = -E_i\left(1 + \lambda B_i^2\right)$, where $\lambda > 0$ as the fitness of $D_i$.
        (b) End For
    4. Genetic operations Step. Apply the same genetic operations as in the first algorithm of subsection 2.2.
    5. Set iter=iter+1 and goto step 2
    6. Local Search step.
        (a) Get the best chromosome $D^*$ of the population.
        (b) For i=1..W Do
            i. Set $p_i = D_i^*$
            ii. Set $LM_i = -\alpha\left|p_i\right|$
            iii. Set $RM_i = \alpha\left|p_i\right|$ , $\alpha > 1$ .
        (c) EndFor
        (d) Set $L^* = \mathcal{L}\left(D^*, LM, RM\right)$ where $\mathcal{L}()$ is a local optimization method procedure that searches for a local optimum of $N\left(x, D^*\right)$ inside the bounds $\left[\overrightarrow{LM}, \overrightarrow{RM}\right]$. The TOLMIN [59] local optimization procedure used in the above algorithm, which is modified version BFGS local optimization procedure[60].
        (e) Apply the optimized neural network $N\left(x, D^*\right)$ to the test set, that has been modified using the same transformation procedure as in the train set and report the final results.
```

3 Experiments

The software for the algorithm was coded using ANSI C++ and, for all the OpenMP library [61] was utilized for parallelization and to speed up the genetic algorithm. Every experiment was executed 30 times with a different seed for the random generator each time and averages were measured and reported. The function used for random numbers was the drand48() function of the C programming language. The classification error is reported for classification datasets on the test set and the average mean squared error for regression datasets. Also, for more reliability in the results, the common used method of 10 - fold cross validation was used. The values for the parameters of the used algorithms are reported in Table 2.

Table 2: Experimental parameters.

PARAMETER	VALUE
H	10
N_c	500
N_f	2
p_s	0.10
p_m	0.05
N_g	200
L_N	-10.0
R_N	10.0
F	20.0
λ	100.0
α	5.0

3.1 Experimental datasets

The method was tested on a series of regression and classification datasets obtained mostly from two repositories:

1. the Machine Learning Repository `http://www.ics.uci.edu/~mlearn/MLRepository.html`

2. The Keel repository `https://sci2s.ugr.es/keel/`

The description for these datasets has as follows:

1. **Balance** dataset [62], used in psychological experiments.

2. **Dermatology** dataset [63], which is used for differential diagnosis of erythemato-squamous diseases.

3. **Glass** dataset. This dataset contains glass component analysis for glass pieces that belong to 6 classes.

4. **Hayes Roth** dataset [64].

5. **Heart** dataset [65], used to detect heart disease.

6. **Ionosphere** dataset, a meteorological dataset used in various research papers [66, 67].

7. **Parkinsons** dataset,[68] which is created using a range of biomedical voice measurements from 31 people, 23 with Parkinson's disease (PD).The dataset has 22 features.

8. **Pima** dataset, related to the diabetes disease.

9. **PopFailures** dataset [69], used in meteorology.

10. **Spiral** dataset, which is an artificial dataset with two classes. The features in the first class are constructed as:: $x_1 = 0.5t\cos\left(0.08t\right),\ x_2 = 0.5t\cos\left(0.08t+\frac{\pi}{2}\right)$ and for the second class the used equations are : $x_1 = 0.5t\cos\left(0.08t+\pi\right),\ x_2 = 0.5t\cos\left(0.08t+\frac{3\pi}{2}\right)$

11. **Wine**: dataset, which is related to chemical analysis of wines and it has been used in comparison in various research papers [70, 71].

12. **Wdbc** dataset: which contains data for breast tumors.

13. As an real word example, consider an EEG dataset described in [72, 73] is used here. The dataset consists of five sets (denoted as Z, O, N, F and S) each containing 100 single-channel EEG segments each having 23.6 sec duration. With different combinations of these sets the produced datasets are Z_F_S, ZO_NF_S, ZONF_S.

The regression datasets are available from the Statlib URL `ftp://lib.stat.cmu.edu/datasets/index.html` and other sources:

1. **BK** dataset. This dataset comes from Smoothing Methods in Statistics [74] and is used to estimate the points scored per minute in a basketball game. The dataset has 96 patterns of 4 features each.

2. **BL** dataset: This dataset can be downloaded from StatLib. It contains data from an experiment on the affects of machine adjustments on the time to count bolts. It contains 40 patters of 7 features each.

3. **Housing** dataset, described in [75].

4. **Laser** dataset, which is related to laser experiments.

5. **NT** dataset [76], which is related to the body temperature measurements.

6. **Quake** dataset, used to estimate the strength of a earthquake.

7. **FA** dataset, which contains percentage of body fat and ten body circumference measurements. The goal is to fit body fat to the other measurements.

8. **PY** dataset [77], used to learn Quantitative Structure Activity Relationships (QSARs).

The numbers of features and patterns for every dataset used in the experiments are listed in Table 3.

3.2 Experimental results

The table 4 represents the comparative results for the classification datasets and the table 5 shows the results for the regression problems. For the case of classification problems, the average classification error is reported, while for the regression problem the average per point error is reported. The proposed method is denoted as FC MLP and it compared against four other approaches from the relevant literature:

Table 3: Features and patterns for every experimental dataset.

DATASET	FEATURES	PATTERNS
BALANCE	4	625
BK	4	96
BL	7	41
DERMATOLOGY	34	359
GLASS	9	214
HAYES ROTH	5	132
HEART	13	270
HOUSING	13	506
IONOSPHERE	34	351
LASER	4	993
NT	2	131
PARKINSONS	22	195
PIMA	8	768
POPFAILURES	18	540
PY	27	74
QUAKE	3	2178
FA	18	252
SPIRAL	2	2000
WINE	13	179
WDBC	30	569
Z_F_S	21	300
Z_O_N_F_S	21	500
ZO_NF_S	21	500

1. The Minimum Redundancy Maximum Relevance Feature Selection method[78, 79] with two selected features. This approach is denoted as MRMR in the experimental tables. The features selected by MRMR are evaluated using an artificial neural network trained by a genetic algorithm with N_c chromosomes.

2. The Principal Component Analysis (PCA) method as implemented in Mlpack software[80]. The PCA method is used to construct two features from the original dataset. Subsequently, these features are evaluated using an artificial neural network trained by a genetic algorithm with N_c chromosomes.

3. A genetic algorithm with N_c chromosomes and the parameters of Table 2 used to train a neural network with H hidden nodes. This approach is denoted as MLP GEN in the experimental tables.

4. A Particle Swarm Optimization (PSO) with N_c particles and N_g number of generations used to train a neural network with H hidden nodes. This method is denoted as MLP PSO in the experimental tables.

Also, the average classification errors for some of the classification datasets are illustrated graphically in Figure 2. An additional experiment was performed, where the number of chromosomes for the genetic algorithm of feature construction (subsection 2.2) varied from 50 to 500. This experiment was performed on 4 data sets and the results are presented in Table 6. This table shows the reliability and durability of the proposed method, as well as because of a low number of chromosomes, it achieves quite good generalization results.

As the experimental results clearly show, the proposed method is significantly superior to the other techniques and in many cases the percentage gain reaches 90%. The proposed technique for each data set created two artificial features with non-linear combinations of the original features. This process is based on Grammatical Evolution. Because the previous procedure is extremely time consuming, it was chosen to evaluate the characteristics to train a radial base network, which has fast training time. Then another genetic algorithm is used to train an artificial neural network on the new features. The overall process is the same regardless of the type of data and this means that the method can be applied to a wide range of datasets. However, because the method requires the presence of two phases using genetic algorithms, it is considered a very slow method compared to other techniques in the literature. Execution times, however, could be drastically reduced by using parallel techniques such as the OpenMP technique used during the experiments. Also, as was clear from the additional experiments performed with the number of chromosomes, the method is quite robust even for a small number of chromosomes.

Table 4: Experimental results for classification datasets.

DATASET	MRMR	PCA	MLP GEN	MLP PSO	FC MLP
BALANCE	56.80%	56.48%	8.23%	8.07%	0.30%
DERMATOLOGY	68.54%	62.11%	10.01%	17.57%	4.98%
GLASS	58.35%	50.16%	58.03%	57.35%	45.84%
HAYES ROTH	61.21%	61.13%	35.26%	36.69%	23.26%
HEART	38.04%	35.84%	25.46%	25.67%	17.71%
IONOSPHERE	12.93%	21.22%	13.67%	15.14%	8.42%
PARKINSONS	17.16%	16.96%	17.47%	18.35%	10.10%
PIMA	26.29%	39.43%	32.98%	30.45%	23.76%
POPFAILURES	7.04%	31.42%	7.66%	6.24%	4.66%
SPIRAL	44.87%	45.94%	45.71%	42.10%	26.53%
WINE	30.73%	30.39%	20.82%	19.31%	7.31%
WDBC	12.91%	10.28%	6.32%	6.95%	3.47%
Z_F_S	32.71%	44.81%	9.42%	10.38%	5.52%
Z_O_N_F_S	43.04%	56.45%	60.38%	63.56%	31.20%
ZO_NF_S	33.79%	40.02%	8.06%	8.84%	4.00%

Table 5: Experiments for regression datasets.

DATASET	MRMR	PCA	MLP GEN	MLP PSO	FC MLP
BK	0.03	0.17	0.21	0.15	0.03
BL	0.15	0.19	0.84	2.15	0.005
Housing	67.97	319.08	30.05	33.43	10.77
Laser	0.031	0.145	0.003	0.038	0.002
NT	1.79	0.69	1.11	0.03	0.01
Quake	0.06	0.59	0.07	0.28	0.03
FA	0.02	0.08	0.04	0.08	0.01
PY	1.56	0.30	0.21	0.07	0.02

Figure 2: Graphic representation for some of the classification datasets.

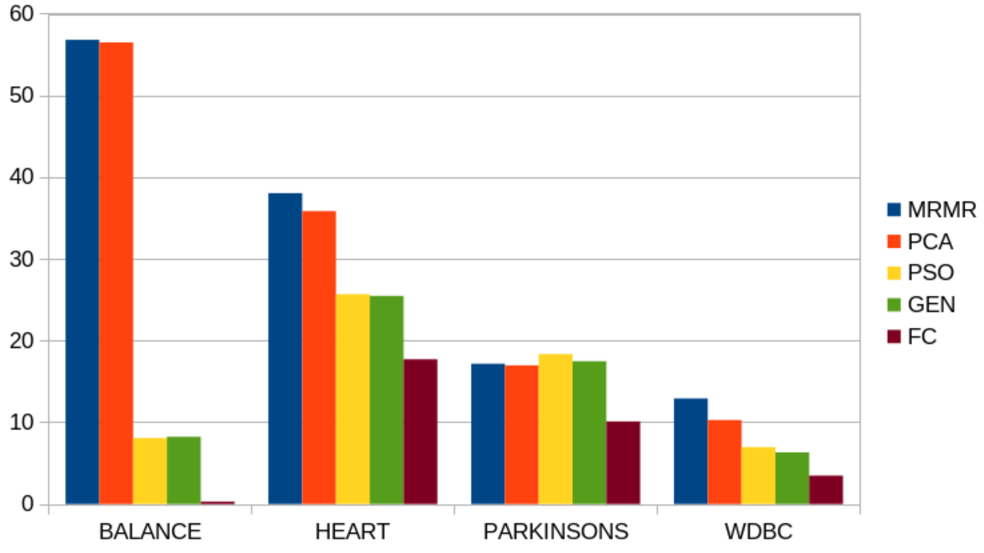

Table 6: Experiments with the number of chromosomes for the algorithm of subsection 2.2

DATASET	$N_g = 50$	$N_g = 100$	$N_g = 200$	$N_g = 500$
HEART	21.26%	21.34%	18.73%	17.71%
BK	0.02	0.02	0.02	0.03
BL	0.02	0.02	0.01	0.005
FA	0.01	0.01	0.01	0.01

4 Conclusions

In the present work, a hybrid feature construction technique was presented with two phases: a) feature construction and b) feature evaluation. In the first phase, new features were created as non-linear combinations of old features using Grammatical Evolution and radial base networks. In the second phase, the original data set was transformed based on the new features and an artificial neural network with a genetic algorithm was trained to learn the new data set. The genetic algorithm used tried to train the artificial neural network in such a way that it did not lose its generalizing abilities. The proposed technique was applied to a number of data sets from the relevant literature and the results were more than satisfactory. Also, with a series of additional experiments, the stability of the proposed methodology was shown, since even with a small number of chromosomes it produces satisfactory results. However, the proposed technique is much slower than other processes, as it requires two computational phases to reach a conclusion. However, with the use of parallel techniques, acceleration can be achieved. The method can be made more efficient in a number of ways. For example, by using parallel genetic algorithms, by using smarter evaluators to construct features instead of radial base networks such as SVM, by using more sophisticated termination techniques for genetic algorithms to achieve acceleration of the export of the results etc.

References

[1] C. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.

[2] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of Control Signals and Systems **2**, pp. 303-314, 1989.

[3] P. Baldi, K. Cranmer, T. Faucett et al, Parameterized neural networks for high-energy physics, Eur. Phys. J. C **76**, 2016.

[4] J. J. Valdas and G. Bonham-Carter, Time dependent neural network models for detecting changes of state in complex processes: Applications in earth sciences and astronomy, Neural Networks **19**, pp. 196-207, 2006

[5] G. Carleo,M. Troyer, Solving the quantum many-body problem with artificial neural networks, Science **355**, pp. 602-606, 2017.

[6] Lin Shen, Jingheng Wu, and Weitao Yang, Multiscale Quantum Mechanics/Molecular Mechanics Simulations with Neural Networks, Journal of Chemical Theory and Computation **12**, pp. 4934-4946, 2016.

[7] Sergei Manzhos, Richard Dawes, Tucker Carrington, Neural network-based approaches for building high dimensional and quantum dynamics-friendly potential energy surfaces, Int. J. Quantum Chem. **115**, pp. 1012-1020, 2015.

[8] Jennifer N. Wei, David Duvenaud, and Alán Aspuru-Guzik, Neural Networks for the Prediction of Organic Chemistry Reactions, ACS Central Science **2**, pp. 725-732, 2016.

[9] Lukas Falat and Lucia Pancikova, Quantitative Modelling in Economics with Advanced Artificial Neural Networks, Procedia Economics and Finance **34**, pp. 194-201, 2015.

[10] Mohammad Namazi, Ahmad Shokrolahi, Mohammad Sadeghzadeh Maharluie, Detecting and ranking cash flow risk factors via artificial neural networks technique, Journal of Business Research **69**, pp. 1801-1806, 2016.

[11] G. Tkacz, Neural network forecasting of Canadian GDP growth, International Journal of Forecasting **17**, pp. 57-69, 2001.

[12] Igor I. Baskin, David Winkler and Igor V. Tetko, A renaissance of neural networks in drug discovery, Expert Opinion on Drug Discovery **11**, pp. 785-795, 2016.

[13] Ronadl Bartzatt, Prediction of Novel Anti-Ebola Virus Compounds Utilizing Artificial Neural Network (ANN), Chemistry Faculty Publications **49**, pp. 16-34, 2018.

[14] I.G. Tsoulos, D. Gavrilis, E. Glavas, Neural network construction and training using grammatical evolution, Neurocomputing **72**, pp. 269-277, 2008.

[15] D.E. Rumelhart, G.E. Hinton and R.J. Williams, Learning representations by back-propagating errors, Nature **323**, pp. 533 - 536 , 1986.

[16] T. Chen and S. Zhong, Privacy-Preserving Backpropagation Neural Network Learning, IEEE Transactions on Neural Networks **20**, , pp. 1554-1564, 2009.

[17] M. Riedmiller and H. Braun, A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP algorithm, Proc. of the IEEE Intl. Conf. on Neural Networks, San Francisco, CA, pp. 586–591, 1993.

[18] T. Pajchrowski, K. Zawirski and K. Nowopolski, Neural Speed Controller Trained Online by Means of Modified RPROP Algorithm, IEEE Transactions on Industrial Informatics **11**, pp. 560-568, 2015.

[19] Rinda Parama Satya Hermanto, Suharjito, Diana, Ariadi Nugroho, Waiting-Time Estimation in Bank Customer Queues using RPROP Neural Networks, Procedia Computer Science **135**, pp. 35-42, 2018.

[20] B. Robitaille and B. Marcos and M. Veillette and G. Payre, Modified quasi-Newton methods for training neural networks, Computers & Chemical Engineering **20**, pp. 1133-1140, 1996.

[21] Q. Liu, J. Liu, R. Sang, J. Li, T. Zhang and Q. Zhang, Fast Neural Network Training on FPGA Using Quasi-Newton Optimization Method,IEEE Transactions on Very Large Scale Integration (VLSI) Systems **26**, pp. 1575-1579, 2018.

[22] C. Zhang, H. Shao and Y. Li, Particle swarm optimisation for evolving artificial neural network, IEEE International Conference on Systems, Man, and Cybernetics, , pp. 2487-2490, 2000.

[23] Jianbo Yu, Shijin Wang, Lifeng Xi, Evolving artificial neural networks using an improved PSO and DPSO **71**, pp. 1054-1060, 2008.

[24] Verleysen M., Francois D., Simon G., Wertz V., On the effects of dimensionality on data analysis with neural networks. In: Mira J., Álvarez J.R. (eds) Artificial Neural Nets Problem Solving Methods. IWANN 2003. Lecture Notes in Computer Science, vol 2687. Springer, Berlin, Heidelberg. 2003.

[25] Burcu Erkmen, Tülay Yıldırım, Improving classification performance of sonar targets by applying general regression neural network with PCA, Expert Systems with Applications **35**, pp. 472-475, 2008.

[26] Jing Zhou, Aihuang Guo, Branko Celler, Steven Su, Fault detection and identification spanning multiple processes by integrating PCA with neural network, Applied Soft Computing **14**, pp. 4-11, 2014.

[27] Ravi Kumar G., Nagamani K., Anjan Babu G., A Framework of Dimensionality Reduction Utilizing PCA for Neural Network Prediction. In: Borah S., Emilia Balas V., Polkowski Z. (eds) Advances in Data Science and Management. Lecture Notes on Data Engineering and Communications Technologies, vol 37. Springer, Singapore. 2020

[28] S. Geman, E. Bienenstock and R. Doursat, Neural networks and the bias/variance dilemma, Neural Computation 4 , pp. 1 - 58, 1992.

[29] Douglas M. Hawkins, The Problem of Overfitting, J. Chem. Inf. Comput. Sci. **44**, pp. 1–12, 2004.

[30] S.J. Nowlan and G.E. Hinton, Simplifying neural networks by soft weight sharing, Neural Computation 4, pp. 473-493, 1992.

[31] S.J. Hanson and L.Y. Pratt, Comparing biases for minimal network construction with back propagation, In D.S. Touretzky (Ed.), Advances in Neural Information Processing Systems, Volume 1, pp. 177-185, San Mateo, CA: Morgan Kaufmann, 1989.

[32] M.C. Mozer and P. Smolensky, Skeletonization: a technique for trimming the fat from a network via relevance assesment. In D.S. Touretzky (Ed.), Advances in Neural Processing Systems, Volume 1, pp. 107-115, San Mateo CA: Morgan Kaufmann, 1989.

[33] M. Augasta and T. Kathirvalavakumar, Pruning algorithms of neural networks — a comparative study, Central European Journal of Computer Science, 2003.

[34] Nitish Srivastava, G E Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan R Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, Journal of Machine Learning Research **15**, pp. 1929-1958, 2014.

[35] Lutz Prechelt, Automatic early stopping using cross validation: quantifying the criteria, Neural Networks **11**, pp. 761-767, 1998.

[36] X. Wu and J. Liu, A New Early Stopping Algorithm for Improving Neural Network Generalization, 2009 Second International Conference on Intelligent Computation Technology and Automation, Changsha, Hunan, 2009, pp. 15-18.

[37] N. K. Treadgold and T. D. Gedeon, Simulated annealing and weight decay in adaptive learning: the SARPROP algorithm,IEEE Transactions on Neural Networks **9**, pp. 662-668, 1998.

[38] M. Carvalho and T. B. Ludermir, Particle Swarm Optimization of Feed-Forward Neural Networks with Weight Decay, 2006 Sixth International Conference on Hybrid Intelligent Systems (HIS'06), Rio de Janeiro, Brazil, 2006, pp. 5-5.

[39] M. O'Neill, C. Ryan, Grammatical evolution, IEEE Trans. Evol. Comput. **5,**pp. 349–358, 2001.

[40] Dimitris Gavrilis, Ioannis G. Tsoulos, Evangelos Dermatas, Selecting and constructing features using grammatical evolution, Pattern Recognition Letters **29**,pp. 1358-1365, 2008.

[41] Dimitris Gavrilis, Ioannis G. Tsoulos, Evangelos Dermatas, Neural Recognition and Genetic Features Selection for Robust Detection of E-Mail Spam, Advances in Artificial Intelligence Volume 3955 of the series Lecture Notes in Computer Science pp 498-501, 2006.

[42] George Georgoulas, Dimitris Gavrilis, Ioannis G. Tsoulos, Chrysostomos Stylios, João Bernardes, Peter P. Groumpos, Novel approach for fetal heart rate classification introducing grammatical evolution, Biomedical Signal Processing and Control **2**,pp. 69-79, 2007

[43] Otis Smart, Ioannis G. Tsoulos, Dimitris Gavrilis, George Georgoulas, Grammatical evolution for features of epileptic oscillations in clinical intracranial electroencephalograms, Expert Systems with Applications **38**, pp. 9991-9999, 2011

[44] D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Publishing Company, Reading, Massachussets, 1989.

[45] Z. Michaelewicz, Genetic Algorithms + Data Structures = Evolution Programs. Springer - Verlag, Berlin, 1996.

[46] D. J. Doorly , J. Peiró, Supervised Parallel Genetic Algorithms in Aerodynamic Optimisation, Artificial Neural Nets and Genetic Algorithms, pp. 229-233, 1997.

[47] Kamal C. Sarma, Hojjat Adeli, Bilevel Parallel Genetic Algorithms for Optimization of Large Steel Structures, Computer-Aided Civil and Infrastructure Engineering **16**, pp. 295-304, 2001.

[48] Yong Fan, Tianzi Jiang, Evans, D.J., Volumetric segmentation of brain images using parallel genetic algorithms, IEEE Transactions on Medical Imaging **21**, pp. 904-909, 2002.

[49] F. H. F. Leung, H. K. Lam, S. H. Ling and P. K. S. Tam, Tuning of the structure and parameters of a neural network using an improved genetic algorithm, IEEE Transactions on Neural Networks **14**, pp. 79-88, 2003.

[50] A. Sedki, D. Ouazar, E. El Mazoudi, Evolving neural network using real coded genetic algorithm for daily rainfall–runoff forecasting, Expert Systems with Applications **36**, pp. 4523-4527, 2009.

[51] A. Majdi, M. Beiki, Evolving neural network using a genetic algorithm for predicting the deformation modulus of rock masses, International Journal of Rock Mechanics and Mining Sciences **47**, pp. 246-253, 2010.

[52] M.G. Smith, L. Bull, Genetic Programming with a Genetic Algorithm for Feature Construction and Selection, Genet Program Evolvable Mach **6**, pp. 265–281, 2005.

[53] K. Neshatian, M. Zhang, P. Andreae, A Filter Approach to Multiple Feature Construction for Symbolic Learning Classifiers Using Genetic Programming, IEEE Transactions on Evolutionary Computation **16**, pp. 645-661, 2012.

[54] X. Li, M. Yin, Multiobjective Binary Biogeography Based Optimization for Feature Selection Using Gene Expression Data, IEEE Transactions on NanoBioscience **12**, pp. 343-353, 2013.

[55] J. Ma, G. Teng, A hybrid multiple feature construction approach for classification using Genetic Programming, Applied Soft Computing **80**, pp. 687-699, 2019.

[56] T.M. Cover, Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition, IEEE Trans. Electron. Comput. EC **14**, pp. 326–334, 1965.

[57] J. Park and I. W. Sandberg, Universal Approximation Using Radial-Basis-Function Networks, Neural Computation 3, pp. 246-257, 1991.

[58] J. W. Backus. The Syntax and Semantics of the Proposed International Algebraic Language of the Zurich ACM-GAMM Conference. Proceedings of the International Conference on Information Processing, UNESCO, 1959, pp.125-132.

[59] M.J.D. Powell, A Tolerant Algorithm for Linearly Constrained Optimization Calculations, Mathematical Programming **45**, pp 547, 1989.

[60] R. Fletcher, A new approach to variable metric algorithms, Computer Journal **13**, pp. 317-322, 1970.

[61] R. Chandra, L. Dagum, D. Kohr, D. Maydan,J. McDonald and R. Menon, Parallel Programming in OpenMP, Morgan Kaufmann Publishers Inc., 2001.

[62] T. Shultz, D. Mareschal, W. Schmidt, Modeling Cognitive Development on Balance Scale Phenomena, Machine Learning **16**, pp. 59-88, 1994.

[63] G. Demiroz, H.A. Govenir, N. Ilter, Learning Differential Diagnosis of Eryhemato-Squamous Diseases using Voting Feature Intervals, Artificial Intelligence in Medicine. **13**, pp. 147–165, 1998.

[64] B. Hayes-Roth, B., F. Hayes-Roth. Concept learning and the recognition and classification of exemplars. Journal of Verbal Learning and Verbal Behavior **16**, pp. 321-338, 1977.

[65] I. Kononenko, E. Šimec, M. Robnik-Šikonja, Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF, Applied Intelligence **7**, pp. 39–55, 1997.

[66] J.G. Dy , C.E. Brodley, Feature Selection for Unsupervised Learning, The Journal of Machine Learning Research **5**, pp 845–889, 2004.

[67] S. J. Perantonis, V. Virvilis, Input Feature Extraction for Multilayered Perceptrons Using Supervised Principal Component Analysis, Neural Processing Letters **10**, pp 243–252, 1999.

[68] Max A. Little, Patrick E. McSharry, Eric J. Hunter, Lorraine O. Ramig (2008), 'Suitability of dysphonia measurements for telemonitoring of Parkinson's disease', IEEE Transactions on Biomedical Engineering **56**, pp. 1015-1022, 2009.

[69] D.D. Lucas, R. Klein, J. Tannahill, D. Ivanova, S. Brandon, D. Domyancic, Y. Zhang, Failure analysis of parameter-induced simulation crashes in climate models, Geoscientific Model Development **6**, pp. 1157-1171, 2013.

[70] M. Raymer, T.E. Doom, L.A. Kuhn, W.F. Punch, Knowledge discovery in medical and biological datasets using a hybrid Bayes classifier/evolutionary algorithm. IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society, **33** , pp. 802-813, 2003.

[71] P. Zhong, M. Fukushima, Regularized nonsmooth Newton method for multi-class support vector machines, Optimization Methods and Software **22**, pp. 225-236, 2007.

[72] R. G. Andrzejak, K. Lehnertz, F.Mormann, C. Rieke, P. David, and C. E. Elger, "Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: dependence on recording region and brain state," Physical Review E, vol. 64, no. 6, Article ID 061907, 8 pages, 2001.

[73] A. T. Tzallas, M. G. Tsipouras, and D. I. Fotiadis, "Automatic Seizure Detection Based on Time-Frequency Analysis and Artificial Neural Networks," Computational Intelligence and Neuroscience, vol. 2007, Article ID 80510, 13 pages, 2007. doi:10.1155/2007/80510

[74] J.S. Simonoff, Smooting Methods in Statistics, Springer - Verlag, 1996.

[75] D. Harrison and D.L. Rubinfeld, Hedonic prices and the demand for clean ai, J. Environ. Economics & Management **5**, pp. 81-102, 1978.

[76] Mackowiak, P.A., Wasserman, S.S., Levine, M.M., 1992. A critical appraisal of 98.6 degrees f, the upper limit of the normal body temperature, and other legacies of Carl Reinhold August Wunderlich. J. Amer. Med. Assoc. 268, 1578–1580

[77] R.D. King, S. Muggleton, R. Lewis, M.J.E. Sternberg, Proc. Nat. Acad. Sci. USA **89**, pp. 11322–11326, 1992.

[78] Hanchuan Peng, Fuhui Long, and Chris Ding, Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy, IEEE Transactions on Pattern Analysis and Machine Intelligence **27**, pp.1226-1238, 2005.

[79] Chris Ding, and Hanchuan Peng, Minimum redundancy feature selection from microarray gene expression data, Journal of Bioinformatics and Computational Biology **3**, pp.185-205, 2005

[80] Ryan R. Curtin, James R. Cline, N. P. Slagle, William B. March, Parikshit Ram, Nishant A. Mehta, Alexander G. Gray, MLPACK: A Scalable C++ Machine Learning Library, Journal of Machine Learning Research **14**, pp. 801−805, 2013.