MDPI

Article

Improving the generalization abilities of constructed neural networks with the addition of local optimization techniques

Ioannis G. Tsoulos[1,*], Vasileios Charilogis[2], Dimitrios Tsalikakis[3], Alexandros Tzallas[4]

[1] Department of Informatics and Telecommunications, University of Ioannina, Greece;itsoulos@uoi.gr
[2] Department of Informatics and Telecommunications, University of Ioannina, Greece; v.charilog@uoi.gr
[3] Department of Engineering Informatics and Telecommunications, University of Western Macedonia, 50100 Kozani, Greece;tsalikakis@gmail.com
[4] Department of Informatics and Telecommunications, University of Ioannina, Greece; tzallas@uoi.gr
* Correspondence: itsoulos@uoi.gr

Abstract: Constructed neural networks with the assistance of Grammatical Evolution have been widely used in a series of classification and data fitting problems appeared recently. Application areas of this innovative machine learning technique include solving differential equations, autism screening, measuring motor function in Parkinson's disease. Although this technique has given excellent results, in many cases it is trapped in local minimum and cannot perform satisfactorily in many problems. For this purpose, it is considered necessary to find techniques to avoid local minima and one technique is the periodic application of local minimization techniques that will undertake to adjust the parameters of the constructed artificial neural network, but maintaining the already existing architecture created by Grammatical Evolution. Periodic application of local minimization techniques has shown a significant reduction in both classification and data fitting problems found in the relevant literature.

Keywords: Grammatical Evolution; Genetic Programming; Neural networks; Local Optimization

Citation: Tsoulos, I.G.; Charilogis, V.; Tsalikakis, D.; Tzallas A. Improving the generalization abilities of constructed neural networks with the addition of local optimization techniques. *Journal Not Specified* **2024**, *1*, 0. https://doi.org/

Received:
Revised:
Accepted:
Published:

1. Introduction

Among the parametric machine learning models one can find Artificial neural networks (ANNs) [1,2], in which a set of parameters, called also weights, must be estimated in order for this model to adapt to classification or regression data. Neural networks have been used problems derived from physics [3–5], problems involving differential equations [6], solar radiation prediction [7], agriculture problems [8], problems derived from chemistry [9,10], wind speed prediction [11], economics problems [12,13], problems related to medicine [14,15] etc. A common way to express a neural network is as a function $N(\overrightarrow{x}, \overrightarrow{w})$. The vector $\overrightarrow{x}$ represents the input pattern to the neural network and the vector $\overrightarrow{w}$ stands for the vector of parameters that must be computed. The set of parameters is calculated by minimizing the training error:

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

In equation 1 the set $(\overrightarrow{x_i}, y_i)$, $i = 1, ..., M$ stands for the train set. The values y_i denote the target outputs for patterns $\overrightarrow{x_i}$. Recently, various methods have appeared that minimize this equation, such as the Back Propagation method [16,17], the RPROP method [18,19], the ADAM method [20] etc. Additionally, global optimization techniques were also used, such as the Simulated Annealing method [21], genetic Algorithms [22], Particle Swarm Optimization (PSO) [23], Differential Evolution [24], Ant Colony Optimization [25], Gray Wolf Optimizer [26], Whale optimization [27] etc.

In many cases, especially when the data is large in volume or has a high number of features, significant times are observed during the process of training artificial neural

networks. For this reason, techniques have been presented in recent years that exploit modern parallel computing structures for faster training of these machine learning models [28].

Another important aspect in artificial neural networks is the initialization of parameters. Various techniques have been proposed in this area, such as usage of polynomial bases [29], initialization based on decision trees [30], usage of intervals [31], discriminant learning [32] etc. Also, recently Chen et al. proposed a new weight initialization method [33].

Identifying the optimal architecture of an artificial neural network could be an extremely important factor to determine the generalization abilities of an artificial neural network. Networks with a few number of neurons can be trained faster and may have good generalization abilities, but in many cases the optimization method can not escape from local minima of the error function. Furthermore, networks with many neurons can have a significantly reduced training error but require a large computational time for their training and many times do not perform significantly when applied to data that is not present in the training set. In this direction, many researchers proposed various methods to discover the optimal architecture, such as usage of genetic algorithms [34,35], usage of the Particle Swarm Optimization method [36], usage of reinforcement learning [37] etc. Also, Islam et al. proposed a new adaptive merging and growing technique used to design the optimal structure of an artificial neural network [38].

Recently, a technique was presented that utilizes the Grammatical Evolution [39] for the efficient construction of the architecture of an artificial neural network as well as the calculation of the optimal values of the parameters [40]. In this technique, the architecture of the neural network is identified. Also, the method performs feature selection, since only those features are selected which will reduce the training error. Using this technique the number of features are reduced, leading to neural networks that are faster in response and with better generalization abilities. The neural construction technique has been applied in a wide series of problems, such as location of amide I bonds [41], solving differential equations [42], application in data collected for Parkinson's disease [43], prediction of performance for higher education students [44], autism screening [45] etc. A software which implements the previously mentioned method can be downloaded freely from https://github.com/itsoulos/NNC (accessed on 28 September 2024) [46].

Although the neural network construction technique has been successfully used in a variety of applications and is able to identify the optimal structure of a neural network as well as find satisfactory values for the model parameters, it can not avoid local minima of the error function, which results in reduced performance in the problems to be solved. In this research paper, the periodic application of local optimization techniques is proposed in randomly selected artificial neural networks constructed by Grammatical Evolution. Local optimization does not alter the generated neural network structure but can more efficiently identify values of the network parameters with lower values of the training error. The current work was applied on a many classification and data fitting datasets, and it seems to reduce the test error obtained by the original neural construction technique.

The current paper have the following organization: section 2 discusses the main aspects of the suggested method, section 3 outlines the series of performed experiments and section 4 provide some conclusions and provided various guidelines for future research.

2. Method description

This section initiates with a short description of the Grammatical Evolution method and continues with the detail description of the proposed algorithm.

2.1. Grammatical evolution

Grammatical evolution can be considered as genetic algorithm with integer chromosomes. These chromosomes stand for series of production rules of a provided in a grammar expressed in the Backus–Naur (BNF) form [47]. The method was used in data fitting [48,49],

trigonometric problems [50], automatic composition of music [51], production of numeric constants with an arbitrary number of digits [52], video games [53,54], energy problems [55], combinatorial optimization [56], cryptography [57], production of decision trees [58], electronics [59], Wikipedia taxonomies [60], economics [61], bioinformatics [62], robotics [63] etc. A BNF grammar is commonly defined as a set $G = (N, T, S, P)$. The following definitions are hold for any BNF grammar:

- The set N contains the symbols denoted as non-terminal.
- The set T has the terminal symbols.
- The symbol $S \in N$ stands for the start symbol of the grammar.
- The set P has all the production rules of the underlying grammar. These rules are used to produce terminal symbols from non - terminal symbols, and they are in the form $A \to a$ or $A \to aB$, $A, B \in N$, $a \in T$.

The production algorithm starts from the symbol S and through a series of steps, produces valid programs by replacing non-terminal symbols with the right hand of the selected production rule. The selection of the production rules is done in two steps:

- **Read** the the next element V from the processed chromosome.
- **Select** the production rule that will be applied using the equation: Rule = V mod N_R, where the N_R stands for the number of production rules that contains the current current non – terminal symbol.

The incorporated grammar for the neural construction method is outlined in Figure 1. The number in parentheses denote the sequence number of each rule for every non - terminal symbol. The symbol n represents the number of features for the used dataset.

```
S:=<sigexpr>                          (0)
<sigexpr>::=<Node>                    (0)
           | <Node> + <sigexpr>       (1)
<Node>::=<number>*sig(<sum>+<number>) (0)
<sum>::= <number>*<xxlist>            (0)
           |    <sum>+<sum>           (1)
<xxlist>::= x1        (0)
             |    x2  (1)
             ..............
             |    xn  (n-1)
<number>::= (<digitlist>.<digitlist>)        (0)
             |    (-<digitlist>.<digitlist>) (1)
<digitlist>::= <digit>            (0)
             | <digit><digitlist> (1)
<digit>::= 0      (0)
             |  1 (1)
             ...........
             |  9 (9)
```

Figure 1. The grammar used by the neural construction method.

This grammar can produce artificial neural networks in the form:

$$N(\vec{x}, \vec{w}) = \sum_{i=1}^{H} w_{(n+2)i-(n+1)} \sigma\left(\sum_{j=1}^{n} x_j w_{(n+2)i-(n+1)+j} + w_{(n+2)i} \right) \quad (2)$$

The symbol H corresponds to the number of weights or processing units. The function $\sigma(x)$ denotes the sigmoid function, and it has the following definition:

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \quad (3)$$

The grammar of the present method can construct artificial neural networks with a hidden processing layer and with a variable number of computing units. This kind of architecture is sufficient to approach any problem, as according to Hornik's theorem [64]. According to this theorem, an artificial neural network with a sufficient number of computing units and a processing level can approximate any function.

As an example consider a problem with 3 inputs: x_1, x_2, x_3 An example neural network that can be constructed by the Grammatical Evolution procedure could be the following:

$$N(x) = 2.1\text{sig}(10.5x_2 + 9.2x_3 + 5.7) + 3.2\text{sig}(2.2x_1 - 4.3x_3 + 6.2) \quad (4)$$

This previously artificial neural network has two processing nodes, and not all inputs are necessarily connected to each processing node, since the Grammatical Evolution process may miss some connections.

2.2. The proposed algorithm

The steps of the current method are derived from the steps of the original neural network construction method with the addition of the periodical application of the local optimization algorithm and it has as follows:

1. **Initialization step**.
   (a) **Set** $k = 0$ the generation counter.
   (b) **Set** as N_g the maximum number of generations and as N_c the number of chromosomes.
   (c) **Set** as p_s the selection rate and as p_m the mutation rate of the genetic algorithm.
   (d) **Set** as N_T the number of chromosomes the will be selected to apply the local search optimization method on them.
   (e) **Set** as N_I the number of generations that should pass before the application of the suggested local optimization technique.
   (f) **Set** as F the range of values within which the local optimization method can vary the parameters of the neural network.
   (g) **Perform** a random initialization of the chromosomes g_i, $i = 1, \ldots, N_c$ as sets of random integers.
2. **Fitness Calculation step**.
   (a) **For** $i = 1, \ldots, N_c$ **do**
      i. **Produce** the corresponding neural network $N_i(\overrightarrow{x}, \overrightarrow{w_i})$ for the chromosome g_i, The production is performed using the procedure of subsection 2.1. The vector $\overrightarrow{w_i}$ denotes the set of parameters produced for the chromosome g_i.
      ii. **Set** as $f_i = \sum_{j=1}^{M} \left(N_i(\overrightarrow{x}_j, \overrightarrow{w_i}) - y_j\right)^2$ the fitness of chromosome i. The set $(\overrightarrow{x_j}, y_j)$, $j = 1, ..., M$ represents the train set.
   (b) **EndFor**
3. **Genetic operations step**.
   (a) **Copy** the best $(1 - p_s) \times N_c$ chromosomes according to their fitness values to the next generation.
   (b) Apply the crossover procedure. This procedure creates $p_s \times N_c$ offsprings from the current population. Two chromosomes (z, w) are selected through tournament selection for every created couple $(\tilde{z}, \tilde{w})$ of ofssprings. These offsprings are created using the one - point crossover procedure, which is demonstrated in Figure 2.
   (c) **Apply** the mutation procedure. During this procedure a random number $r \in [0, 1]$ is selected from uniform distribution for every element of each chromosome. If $r \leq p_m$, then the current element is altered randomly.
4. **Local search step.**

(a) **If** k mod N_I=0 **then**

i. **Set** $S = \left\{g_{r_1}, g_{r_2}, \ldots, g_{r_{N_T}}\right\}$ a set of chromosomes of the current population that are selected randomly.

ii. **For** $j = 1, \ldots, N_T$ apply the procedure described in subsection 2.3 on chromosome g_{r_j}.

(b) **Endif**

5. **Termination check step**.

(a) **Set** $k = k + 1$

(b) **If** $k \leq N_g$ goto Fitness Calculation Step, **else**

i. **Obtain** the chromosome g^* that has the lowest fitness value among the population.

ii. **Produce** the corresponding neural network $N^*\left(\overrightarrow{x}, \overrightarrow{w^*}\right)$ for this chromosome. The vector $\overrightarrow{w^*}$ denotes the set of parameters for the chromosome g^*.

iii. **Obtain** the corresponding test error for $N^*\left(\overrightarrow{x}, \overrightarrow{w^*}\right)$.

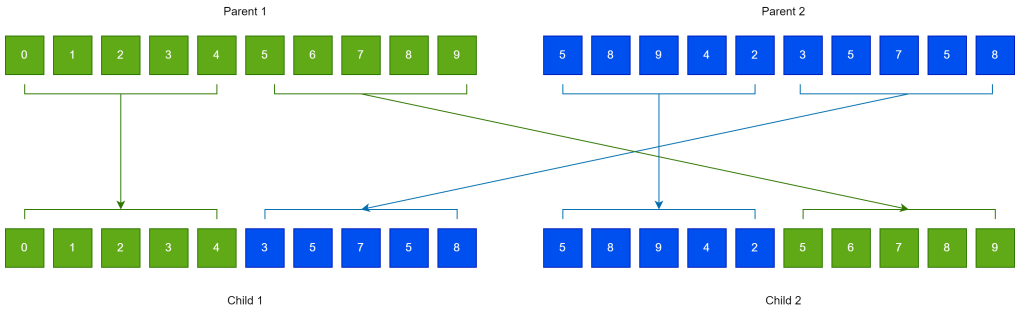

Figure 2. An example of the method of one - point crossover, used in the Grammatical Evolution procedure.

The flowchart for the proposed method is depicted in Figure 3.

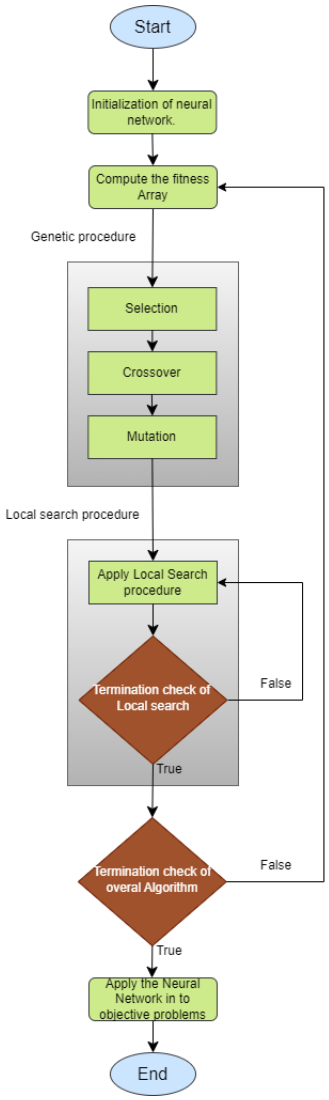

Figure 3. The flowchart of the proposed algorithm.

2.3. The local search procedure

The local search procedure initiates from the vector $\overrightarrow{w}$ that is produced for any given chromosome g using the procedure described in subsection 2.1. The procedure minimizes the error of equation 1 with respect to vector $\overrightarrow{w}$. The minimization is done within a value interval created around the initial point $\overrightarrow{w}$ keeping the network structure intact. The main steps of this procedure are given below:

1. **Set** $d = (n+2)H$. This value denotes the total number of parameters for $N(\overrightarrow{x}, \overrightarrow{w})$.
2. **For** $i = 1, \ldots, d$ **do**
   (a) **Set** $L_i = -F \times |w_i|$, the left bound of the minimization for the parameter i.
   (b) **Set** $R_i = F \times |w_i|$, the right bound of the minimization for the parameter i.
3. **EndFor**
4. **Minimize** the error function of equation 1 for vector $\overrightarrow{w}$ inside the bounding box $\left[\overrightarrow{L}, \overrightarrow{R}\right]$ using a local optimization procedure $\mathcal{L}(w)$. The BFGS method as modified by Powell [65] was incorporated in the conducted experiments.

As an example consider a dataset with $n = 2$ and the following constructed neural network:

$$N(\overrightarrow{x}, \overrightarrow{w}) = 2\sigma(1.5x_2 - 2.0) + 2.5\sigma(-1.2x_1 + 3.7) \quad (5)$$

where the number of nodes is $H = 2$. In this case the vector $\overrightarrow{w}$ has the elements $\overrightarrow{w} = (2, 0, 1.5, -2, 2.5, -1.2, 0, 3.7)$. If the parameter F has the value $F = 2$,then the bound vectors $\overrightarrow{L}$ and $\overrightarrow{R}$ are defined as:

$$\begin{aligned} \overrightarrow{L} &= (-4, 0, -3, -4, -5, -2.4, 0, 7.4) \\ \overrightarrow{R} &= (4, 0, 3, 4, 5, 2.4, 0, 7.4) \end{aligned}$$

Hence the minimization method could not change the elements w_2 and w_7 and as a consequence the architecture of the network remains intact.

3. Experimental results

A series of classification and regression datasets was used in order to test the current work. Also, the suggested technique was compared against other established machine learning methods and the results are reported. Furthermore, a series of experiments were executed to verify the sensitivity of the proposed technique with respect to the critical parameters presented earlier. The following URLs provide the used datasets:

1. **The UCI dataset repository, https://archive.ics.uci.edu/ml/index.php(accessed on 28 September 2024)[66]**
2. **The Keel repository, https://sci2s.ugr.es/keel/datasets.php(accessed on 28 September 2024)[67].**
3. **The Statlib URL http://lib.stat.cmu.edu/datasets/(accessed on 28 September 2024).**

3.1. The used classification datasets

The descriptions of the used datasets are:

1. **Appendictis** a dataset originated in [68].
2. **Australian** dataset [69], which is related to bank transactions.
3. **Balance** dataset [70], that has been used in a series of psychological experiments.
4. **Circular** dataset, that contains artificially generated data.
5. **Cleveland** dataset [71,72].
6. **Dermatology** dataset [73], a dataset that contains measurements for dermatological deceases.
7. **Ecoli** dataset, a dataset contains measurements about proteins[74].
8. **Fert** dataset, used to detect the relation between sperm concentration and demographic data.
9. **Haberman** dataset, a medical dataset related to breast cancer.
10. **Hayes roth** dataset [75].
11. **Heart** dataset [76], a medical dataset used for the prediction of heart diseases.
12. **HeartAttack** dataset, used for the detection of heart diseases.
13. **HouseVotes** dataset [77].
14. **Liverdisorder** dataset [78], a medical dataset.
15. **Ionosphere** dataset, a climate dataset [79,80].
16. **Mammographic** dataset [81], a dataset related to the breast cancer.
17. **Parkinsons** dataset, used in the detection of Parkinson's disease (PD)[82].
18. **Pima** dataset [83].
19. **Popfailures** dataset [84].
20. **Regions2** dataset, used in the detection of hepatitis C [85].
21. **Saheart** dataset [86], a medical dataset used for the detection of heart diseases.
22. **Segment** dataset [87], a dataset related to image processing.
23. **Spiral** dataset, which is an artificial dataset.
24. **Student** dataset [88], which contains data from experiments conducted in Portuguese schools.
25. **Transfusion** dataset[89], a medical dataset.
26. **Wdbc** dataset [90].

27. **Wine** dataset, used for the detection of quality of wines. [91,92].
28. **Eeg** datasets, a dataset contains EEG experiments [93]. The following distinct cases were used from this dataset: Z_F_S, Z_O_N_F_S, ZO_NF_S and ZONF_S.
29. **Zoo** dataset [94], used to classify animals.

The number of inputs and classes for every classification dataset is given in Table 1.

DATASET	INPUTS	CLASSES
APPENDICITIS	7	2
AUSTRALIAN	14	2
BALANCE	4	3
CIRCULAR	5	2
CLEVELAND	13	5
DERMATOLOGY	34	6
ECOLI	7	8
FERT	9	2
HABERMAN	3	2
HAYES ROTH	5	3
HEART	13	2
HEART ATTACK	13	2
HOUSEVOTES	16	2
LIVERDISORDER	6	2
IONOSPHERE	34	2
MAMMOGRAPHIC	5	2
PARKINSONS	22	2
PIMA	8	2
POPFAILURES	18	2
REGIONS2	18	5
SAHEART	9	2
SEGMENT	19	7
SPIRAL	2	2
STUDENT	5	4
TRANSFUSION	4	2
WDBC	30	2
WINE	13	3
Z_F_S	21	3
Z_O_N_F_S	21	5
ZO_NF_S	21	3
ZONF_S	21	2
ZOO	16	7

Table 1. Number of inputs and distinct classes for every classification dataset.

3.2. The used regression datasets

The descriptions for these datasets are provided below:

1. **Abalone** dataset [95].
2. **Airfoil** dataset, a dataset derived from by NASA [96].
3. **BK** dataset [97], that contains data from various basketball games.
4. **BL** dataset, which contains data from an electricity experiment.
5. **Baseball** dataset, used to predict the average income of baseball players.
6. **Concrete** dataset [98].
7. **Dee** dataset, that contains measurements about the price of electricity.
8. **FY,** this dataset used to measure the longevity of fruit flies.
9. **HO** dataset, appeared in the STALIB repository.
10. **Housing** dataset, originated in [99].
11. **Laser** dataset, that contains data from physics experiments

12. **LW** dataset, hat contains measurements from low weight babies.
13. **MORTGAGE** dataset, related to some economic measurements from USA.
14. **MUNDIAL,** appeared in the STALIB repository.
15. **PL** dataset, included in the STALIB repository.
16. **QUAKE** dataset, contains data about earthquakes.
17. **REALESTATE,** included in the STALIB repository.
18. **SN** dataset, that contains measurements from an experiment related to trellising and pruning.
19. **Treasury** dataset, that deals with some factors of the USA economy.
20. **VE** dataset, included in the STALIB repository.
21. **TZ** dataset, included in the STALIB repository.

The number of inputs for each dataset is provided in Table 2.

Table 2. Number of inputs for every regression dataset.

DATASET	INPUTS
ABALONE	8
AIRFOIL	5
BASEBALL	16
BK	4
BL	7
CONCRETE	8
DEE	6
HO	13
HOUSING	13
FY	4
LASER	4
LW	9
MORTGAGE	15
MUNDIAL	3
PL	2
QUAKE	3
REALESTATE	5
SN	11
TREASURY	15
TZ	60
VE	7

3.3. Experimental results

All the methods used here was coded in ANSI C++. The freely available optimization of Optimus, which can be downloaded from https://github.com/itsoulos/GlobalOptimus/(accessed on 28 September 2024) was also utilized for the optimization process. Each experiment was conducted 30 times and the average classification or regression value was measured. In each run a different seed for the random number generator was used and the drand48() function of the C programming language was incorporated. For the case of classification datasets the displayed classification error for a model $M(x)$ and the test dataset T is calculated as:

$$E_C(M(x)) = 100 \times \frac{\sum_{i=1}^{N} (\text{class}(M(x_i)) - y_i)}{N} \tag{6}$$

The test set T is defined as $T = (x_i, y_i),\ i = 1, \ldots, N$ For the case of regression problems the regression error $E_R(M(x))$ is defined as:

$$E_R(M(x)) = \frac{\sum_{i=1}^{N} (M(x_i) - y_i)^2}{N} \tag{7}$$

The method of ten - fold cross validation was incorporated to validate the experimental results. The experiments were carried out on an AMD Ryzen 5950X with 128GB of RAM, running the Debian Linux operating system. The values for the parameters of the algorithms are shown in Table 3. In all tables, the bold font is used to mark the machine learning method that have achieved the lowest classification or regression error.

Table 3. The values of the parameters used in the experiments.

PARAMETER	MEANING	VALUE
N_c	Chromosomes	500
N_g	Generations	200
p_s	Crossover rate	0.10
p_m	Mutation rate	0.05
N_T	Number of chromosomes that selected randomly	20
N_I	Number of iterations before local search	10
F	Magnitude of changes from local search	4.0

Table 4 contains the experimental results for the classification datasets and Table 5 the experimental results for the regression datasets. The neural network used in the experimental results has one processing unit with $H = 10$ processing nodes and the sigmoid function as activation function. The following notation was used in these tables:

1. The column BFGS represents the results obtained by the application of the BFGS method to train a neural network with $H = 10$ processing nodes.
2. The column GENETIC represents the results produced by the training of a neural network with $H = 10$ processing nodes using a Genetic Algorithm. The parameters of this algorithm are mentioned in Table 3.
3. The column NNC represents the results obtained by the neural construction technique.
4. The column INNC represents the results obtained by the proposed method.
5. The last row AVERAGE contains the average classification or regression error, as measured on all datasets.

The methods BFGS and GENETIC train the neural network $N(\vec{x}, \vec{w})$ by obtain the global minimum of the the error function defined in Equation 1.

Table 4. Results from the application of machine learning models on the classification datasets. Numbers in cells represent average classification error for the corresponding test set.

DATASET	BFGS	GENETIC	NNC	INNC
APPENDICITIS	18.00%	24.40%	**13.70%**	14.70%
AUSTRALIAN	38.13%	36.64%	**14.51%**	14.80%
BALANCE	8.64%	**8.36%**	22.11%	8.66%
CIRCULAR	6.08%	**5.13%**	13.64%	5.32%
CLEVELAND	77.55%	57.21%	50.10%	**47.93%**
DERMATOLOGY	52.92%	**16.60%**	25.06%	20.89%
ECOLI	69.52%	54.67%	**47.82%**	48.21%
FERT	23.20%	28.50%	**19.00%**	20.50%
HABERMAN	29.34%	28.66%	28.03%	**26.70%**
HAYES-ROTH	37.33%	56.18%	35.93%	**31.77%**
HEART	39.44%	26.41%	15.78%	**14.74%**
HEARTATTACK	46.67%	29.03%	**19.33%**	20.43%
HOUSEVOTES	7.13%	7.00%	3.65%	**3.26%**
IONOSPHERE	15.29%	15.14%	**11.12%**	11.92%
LIVERDISORDER	42.59%	37.09%	33.71%	**31.77%**
MAMMOGRAPHIC	17.24%	19.88%	17.78%	**15.81%**
PARKINSONS	27.58%	16.58%	**12.21%**	12.53%
PIMA	35.59%	34.21%	27.99%	**24.00%**
POPFAILURES	5.24%	**4.17%**	6.74%	6.44%
REGIONS2	36.28%	33.53%	25.52%	**23.18%**
SAHEART	37.48%	34.85%	30.52%	**28.09%**
SEGMENT	68.97%	46.30%	54.99%	**43.12%**
SPIRAL	47.99%	47.67%	48.39%	**43.99%**
STUDENT	7.14%	5.61%	5.78%	**4.55%**
TRANSFUSION	25.80%	25.84%	25.34%	**23.43%**
WDBC	29.91%	7.87%	6.95%	**4.41%**
WINE	59.71%	22.88%	14.35%	**9.77%**
Z_F_S	39.37%	24.60%	14.17%	**8.53%**
Z_O_N_F_S	65.67%	64.81%	49.18%	**38.58%**
ZO_NF_S	43.04%	21.54%	14.14%	**6.84%**
ZONF_S	15.62%	4.36%	3.14%	**2.52%**
ZOO	10.70%	9.50%	9.20%	**7.20%**
AVERAGE	**33.91%**	**26.73%**	**22.50%**	**19.52%**

Table 5. Results from the conducted experiments on the regression datasets. Numbers in cells denote average regression error as calculated on the corresponding test set.

DATASET	BFGS	GENETIC	NNC	INNC
ABALONE	5.69	7.17	5.05	**4.33**
AIRFOIL	0.003	0.003	0.003	**0.002**
BASEBALL	119.63	64.60	59.85	**48.42**
BK	0.36	0.26	2.32	**0.07**
BL	1.09	2.23	0.021	**0.002**
CONCRETE	0.066	0.01	0.008	**0.005**
DEE	2.36	1.01	0.26	**0.23**
HO	0.62	0.37	0.017	**0.01**
HOUSING	97.38	43.26	26.35	**16.01**
FY	0.19	0.65	0.058	**0.042**
LASER	0.015	0.59	0.024	**0.005**
LW	2.98	0.54	**0.011**	0.012
MORTGAGE	8.23	0.40	0.30	**0.026**
MUNDIAL	6.48	1.22	4.47	**0.034**
PL	0.29	0.28	0.045	**0.022**
QUAKE	0.42	0.12	0.045	**0.04**
REALESTATE	128.94	81.19	76.78	**70.99**
SN	3.89	0.40	0.026	**0.023**
TREASURY	9.91	2.93	0.47	**0.066**
TZ	3.27	5.38	5.04	**0.03**
VE	1.92	2.43	6.61	**0.025**
AVERAGE	**18.75**	**10.24**	**8.94**	**6.69**

The original technique of constructing artificial neural networks has significantly lower classification or regression errors than the other two techniques and the proposed method managed to improve the performance of this technique in the majority of the datasets. The percentage improvement in error is even greater in regression problems. In some cases the improvement exceeds 50% in the test error. This effect is evident in the box plots for classification and regression errors outlined in Figures 4 and 5 respectively.

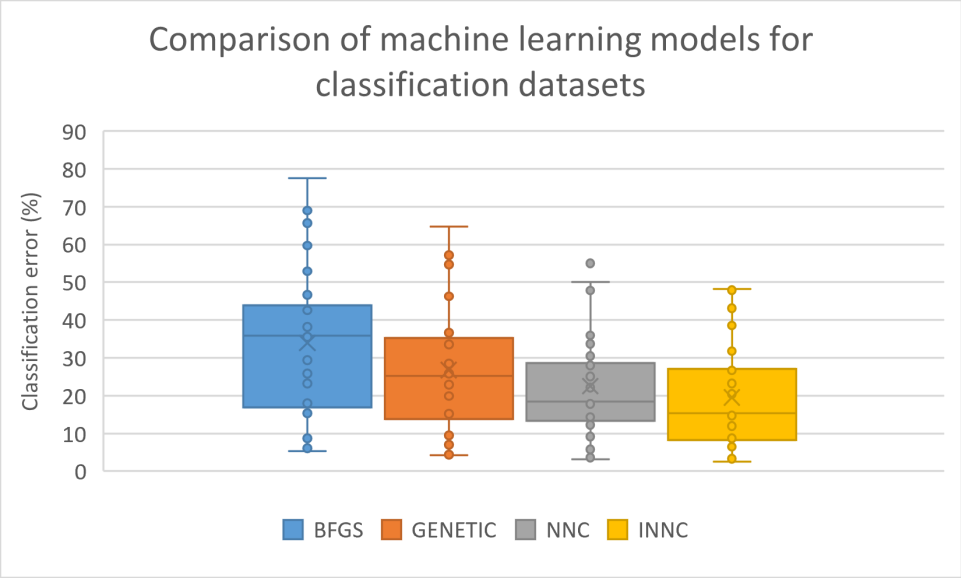

Figure 4. Box plot for the comparison between the machine learning methods applied on the classification datasets.

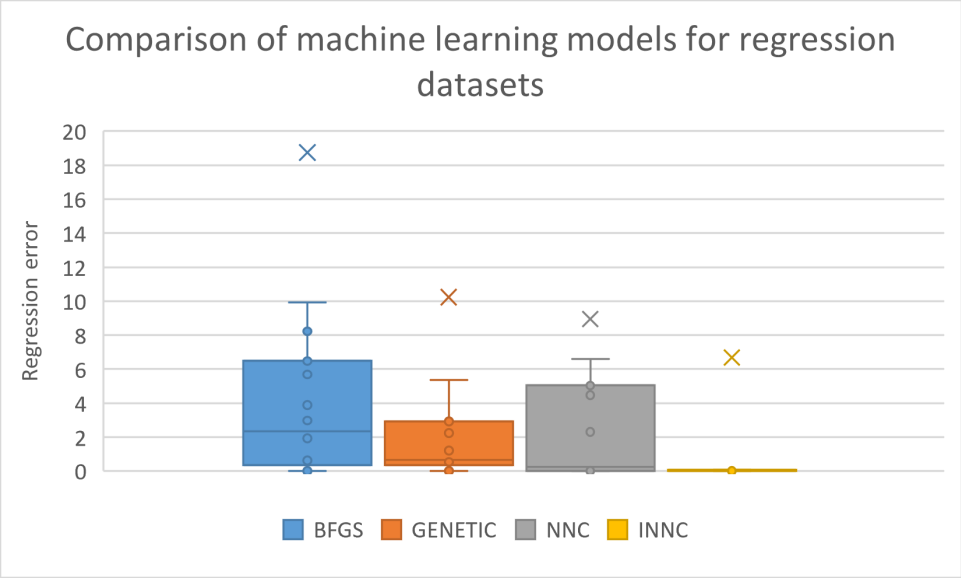

Figure 5. Box plot for the comparison between the machine learning methods applied on the regression datasets.

Furthermore, the same reduction in test error is validated from the statistical tests performed for classification and regression problems. These tests are depicted in Figures 6 and 7.

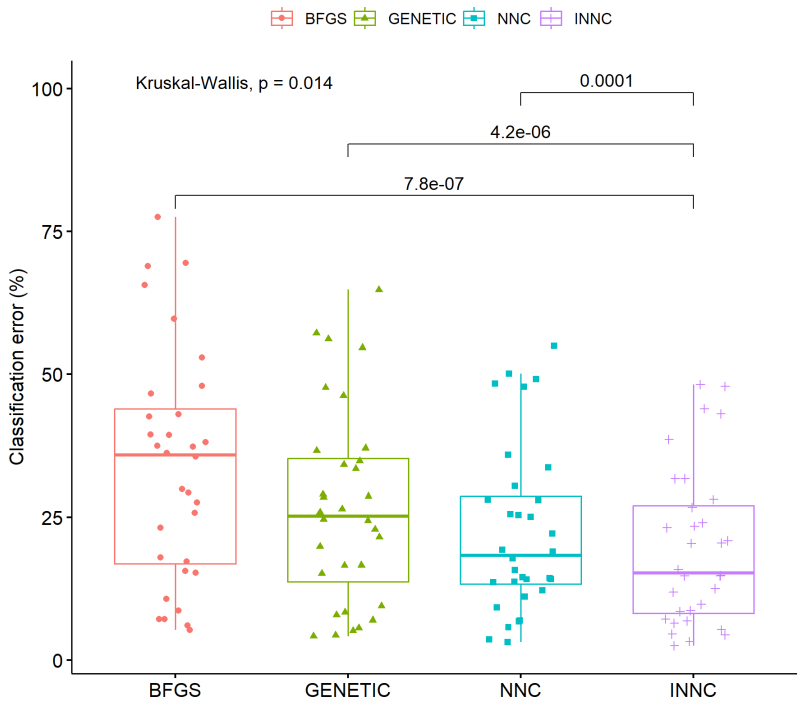

Figure 6. Statistical test for all machine learning models that was applied on the classification datasets.

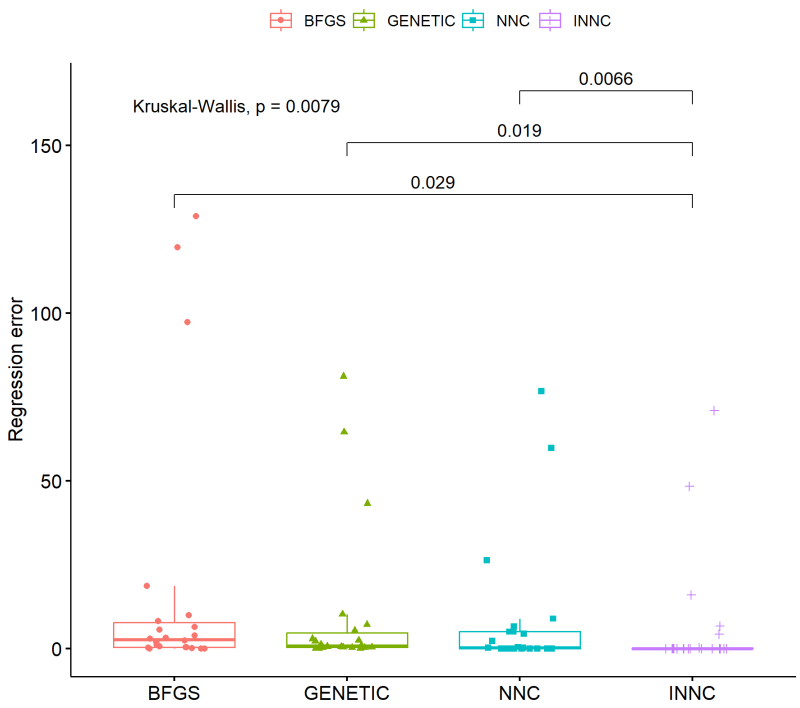

Figure 7. Statistical test for all machine learning models that was applied on the regression datasets.

3.3.1. Experiments with the parameter N_I

To verify the robustness of the proposed technique as well as its sensitivity to parameter changes, a new experiment was carried out in which the critical N_I parameter was varied from 5 to 20. This parameter determines the number of generations intervening before the local optimization method is executed on randomly selected chromosomes. The results from this experiment for the classification datasets are depicted in Table 6 and the results for the regression datasets in Table 7.

Table 6. Experimental results using a variety of values for the parameter N_I of the current work with application on the classification datasets.

DATASET	**INNC**($N_I = 5$)	**INNC**($N_I = 10$)	**INNC**($N_I = 20$)
APPENDICITIS	14.50%	14.70%	**14.20%**
AUSTRALIAN	**14.48%**	14.80%	14.58%
BALANCE	**7.63%**	8.66%	11.71%
CIRCULAR	**5.25%**	5.32%	5.87%
CLEVELAND	48.41%	**47.93%**	49.66%
DERMATOLOGY	**19.66%**	20.89%	23.11%
ECOLI	**47.39%**	48.21%	48.09%
FERT	20.80%	**20.50%**	20.90%
HABERMAN	26.87%	**26.70%**	27.27%
HAYES-ROTH	**31.69%**	31.77%	35.92%
HEART	14.85%	**14.74%**	15.56%
HEARTATTACK	20.10%	20.43%	**20.07%**
HOUSEVOTES	3.87%	**3.26%**	3.78%
IONOSPHERE	11.51%	11.92%	**11.09%**
LIVERDISORDER	**31.35%**	31.77%	31.85%
MAMMOGRAPHIC	16.25%	**15.81%**	16.06%
PARKINSONS	13.16%	**12.53%**	12.58%
PIMA	**23.95%**	24.00%	24.67%
POPFAILURES	**6.06%**	6.44%	6.48%
REGIONS2	23.36%	**23.18%**	24.21%
SAHEART	**27.68%**	28.09%	29.41%
SEGMENT	43.79%	**43.12%**	46.91%
SPIRAL	**43.63%**	43.99%	45.15%
STUDENT	**3.65%**	4.55%	3.88%
TRANSFUSION	**22.62%**	23.43%	23.89%
WDBC	**4.41%**	4.41%	5.46%
WINE	10.18%	**9.77%**	12.00%
Z_F_S	**7.70%**	8.53%	9.63%
Z_O_N_F_S	**36.64%**	38.58%	41.08%
ZO_NF_S	**6.84%**	6.84%	6.90%
ZONF_S	**2.24%**	2.52%	2.66%
ZOO	**6.30%**	7.20%	7.70%
AVERAGE	**19.28%**	**19.52%**	**20.39%**

Table 7. Experimental results with a variety of values for the parameter N_I. The current work was applied on the regression datasets.

DATASET	INNC($N_I = 5$)	INNC($N_I = 10$)	INNC($N_I = 20$)
ABALONE	4.38	**4.33**	4.45
AIRFOIL	0.002	**0.002**	0.002
BASEBALL	**47.50**	48.42	49.99
BK	0.64	**0.07**	1.86
BL	0.014	**0.002**	0.003
CONCRETE	**0.005**	0.005	0.005
DEE	**0.22**	0.23	0.23
HO	**0.01**	0.01	0.011
HOUSING	21.09	**16.01**	16.96
FY	0.046	**0.042**	0.19
LASER	**0.004**	0.005	0.005
LW	**0.011**	0.012	0.011
MORTGAGE	**0.02**	0.026	0.03
MUNDIAL	0.031	0.034	**0.029**
PL	**0.022**	0.022	0.022
QUAKE	0.045	0.04	**0.037**
REALESTATE	**68.85**	70.99	70.59
SN	**0.023**	0.023	0.024
TREASURY	**0.063**	0.066	0.073
TZ	**0.029**	0.03	0.031
VE	0.028	**0.025**	0.116
AVERAGE	**6.81**	**6.69**	**6.89**

The method appears to have similar results for each variation of the critical N_I parameter. In some cases the error is smaller for low values of this parameter but not to a great extent. This effect is also evident in the box plot presented for the classification datasets in Figure 8 as well as in the statistical comparison of Figure 9.

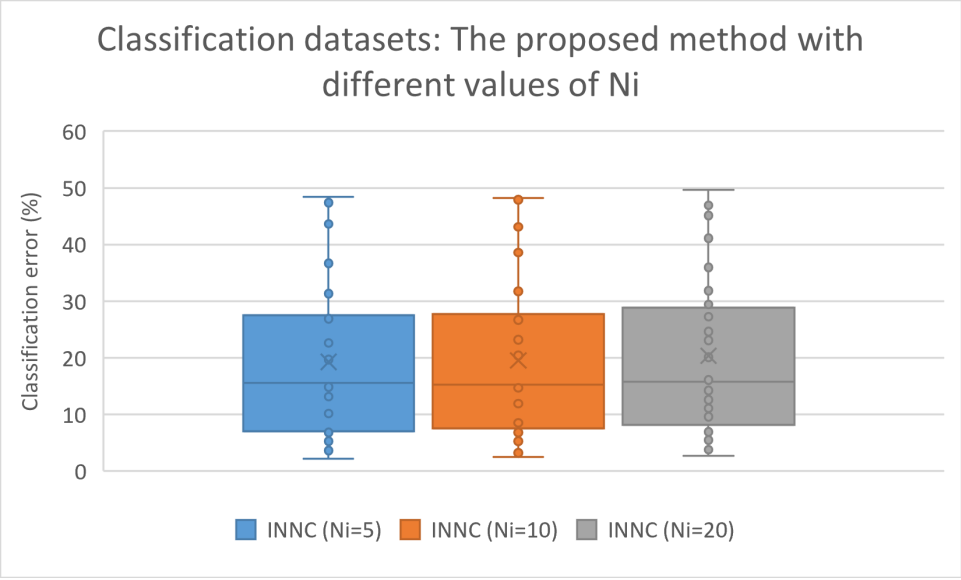

Figure 8. Box plot for the experiments with the parameter N_I and the current work. The method was applied on the classification datasets.

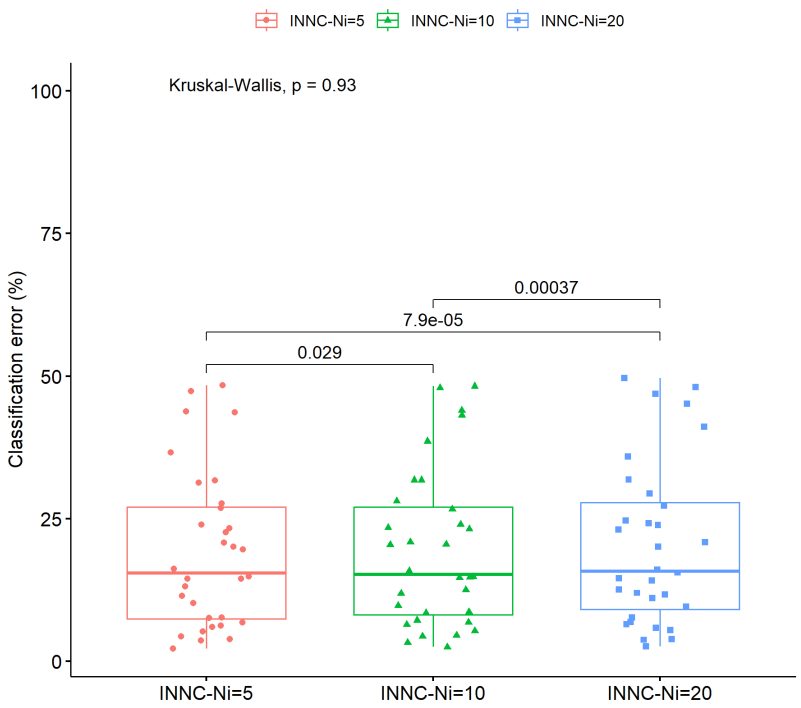

Figure 9. Statistical comparison for the experiment with the current work and a variety of values for parameter N_I. The method was applied on the classification datasets.

In addition, the average execution time of the methods was recorded for the various values of the parameter N_I and the results are shown in graphs 10 and 11 for classification and regression problems respectively.

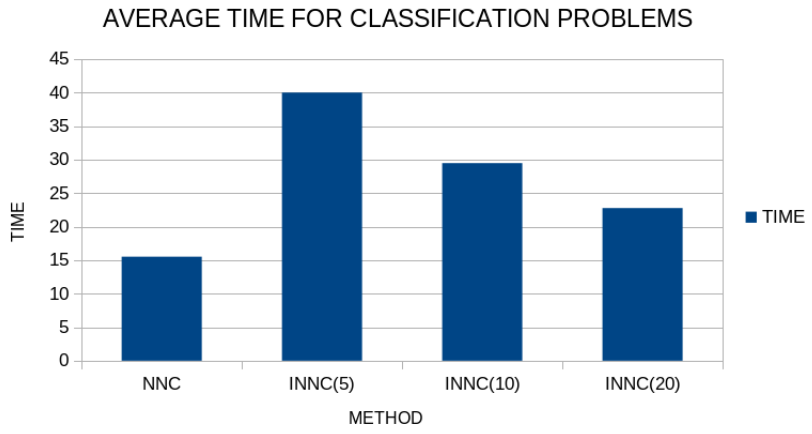

Figure 10. Average execution time for the original method and the modified one. The time was recorded for different values of the parameter N_I and for the classification datasets.

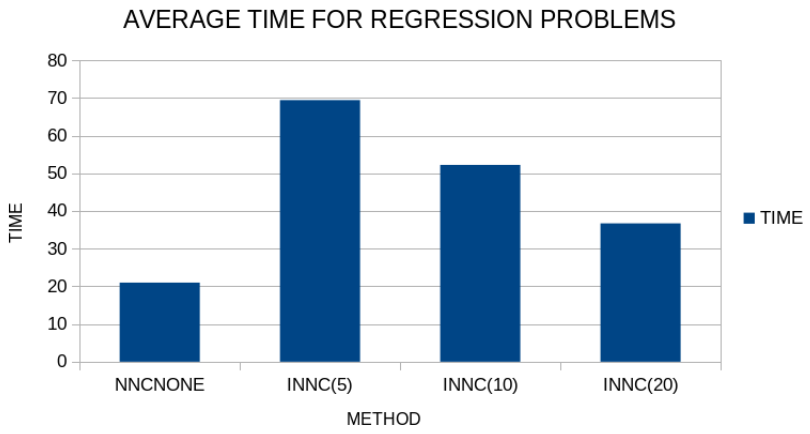

Figure 11. Average execution time for the original method and the modified one. The time was measured for different values of the parameter N_I. The execution times were recorded for the regression problems.

As expected, the method requires more computational time than the original one and in fact the smaller the value of the N_I parameter the more time has to be spent since more local optimizations have to be performed. Of course, this additional computing time can be significantly reduced by using parallel computing techniques.

3.3.2. Experiments with the parameter F

Another important parameter for the proposed method is the parameter F. This parameter identifies the range of changes that the local optimization method can cause on randomly selected chromosomes. In this experiment, the parameter F changed from 1.5 to 8.0 and the results for the classification datasets are depicted in Table 8 while the experimental results for the regression datasets are presented in Table 9.

Table 8. Experimental results with a variety of values for the parameter F, with application on the classification datasets.

DATASET	**INNC**$(F = 1.5)$	**INNC**$(F = 2.0)$	**INNC**$(F = 4.0)$	**INNC**$(F = 8.0)$
APPENDICITIS	14.80%	**14.20%**	14.70%	15.00%
AUSTRALIAN	14.58%	14.56%	14.80%	**14.54%**
BALANCE	**8.52%**	8.68%	8.66%	9.35%
CIRCULAR	5.92%	5.46%	**5.32%**	5.43%
CLEVELAND	48.35%	**47.38%**	47.93%	49.62%
DERMATOLOGY	**19.80%**	20.09%	20.89%	21.66%
ECOLI	**47.76%**	48.30%	48.21%	48.42%
FERT	21.20%	21.20%	**20.50%**	20.90%
HABERMAN	26.43%	**25.93%**	26.70%	26.70%
HAYES-ROTH	32.15%	**31.31%**	31.77%	35.69%
HEART	14.81%	15.41%	**14.74%**	15.41%
HEARTATTACK	21.20%	**19.90%**	20.43%	20.40%
HOUSEVOTES	3.35%	3.65%	**3.26%**	3.65%
IONOSPHERE	11.31%	**10.40%**	11.92%	11.23%
LIVERDISORDER	32.09%	**31.21%**	31.77%	33.18%
MAMMOGRAPHIC	16.15%	16.05%	**15.81%**	16.07%
PARKINSONS	12.63%	**12.47%**	12.53%	13.05%
PIMA	**23.58%**	23.80%	24.00%	23.79%
POPFAILURES	**5.85%**	6.29%	6.44%	6.04%
REGIONS2	24.08%	23.48%	**23.18%**	25.25%
SAHEART	28.41%	28.83%	**28.09%**	28.30%
SEGMENT	45.12%	43.37%	**43.12%**	43.93%
SPIRAL	**43.40%**	43.72%	43.99%	44.61%
STUDENT	4.25%	**4.15%**	4.55%	4.52%
TRANSFUSION	23.09%	**22.93%**	23.43%	23.05%
WDBC	5.27%	5.30%	**4.41%**	4.84%
WINE	10.53%	10.24%	**9.77%**	11.59%
Z_F_S	9.20%	**7.57%**	8.53%	8.63%
Z_O_N_F_S	40.60%	38.70%	**38.58%**	39.78%
ZO_NF_S	6.90%	7.68%	6.84%	**6.56%**
ZONF_S	2.64%	2.72%	**2.52%**	2.88%
ZOO	8.60%	7.40%	**7.20%**	8.90%
AVERAGE	**19.77%**	**19.45%**	**19.52%**	**20.09%**

Table 9. Experimental results using a variety of values for the parameter F, with application on the the regression datasets.

DATASET	INNC($F = 1.5$)	INNC($F = 2.0$)	INNC($F = 4.0$)	INNC($F = 8.0$)
ABALONE	4.49	4.43	**4.33**	4.66
AIRFOIL	0.002	0.002	**0.002**	0.002
BASEBALL	**48.31**	50.38	48.42	48.52
BK	0.06	0.21	0.07	**0.02**
BL	0.003	0.004	**0.002**	0.003
CONCRETE	**0.005**	0.005	0.005	0.005
DEE	**0.23**	0.23	0.23	0.23
HO	**0.01**	0.01	0.01	0.01
HOUSING	16.91	16.34	**16.01**	16.96
FY	0.042	0.043	**0.042**	0.043
LASER	0.005	0.005	0.005	**0.004**
LW	**0.011**	0.012	0.012	0.012
MORTGAGE	0.027	0.025	0.026	**0.024**
MUNDIAL	0.035	**0.033**	0.034	0.23
PL	**0.022**	0.022	0.022	0.022
QUAKE	**0.036**	0.037	0.04	0.036
REALESTATE	72.40	71.16	70.99	**70.76**
SN	0.024	0.024	**0.023**	0.024
TREASURY	0.07	0.068	**0.066**	0.068
TZ	**0.029**	0.031	0.03	0.031
VE	0.032	0.03	**0.025**	0.028
AVERAGE	**6.80**	**6.81**	**6.69**	**6.75**

Once again, there are no significant differences in the performance of the proposed technique as the critical factor F varies. In the case of the classification data, however, there is a small increase in the classification error as this factor increases, which may be due to the fact that, as we move away from the solution created by the method of creating neural networks, the performance of the method decreases. Also, the box plot for the classification datasets is depicted in Figure 12.

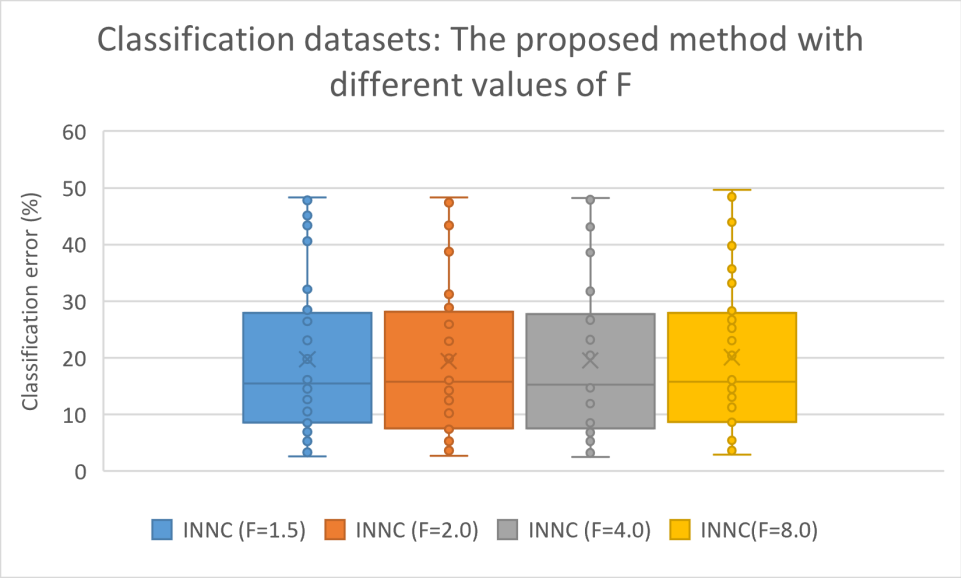

Figure 12. Box plot for the experiments using the current work and different values of the parameter F. The experiments were conducted on the classification datasets.

3.3.3. Experiments with the used local search optimizer

An important issue of the proposed method is the selection of the local search optimizer, that will be applied periodically to chromosomes selected randomly. In the current work, a modified version of the BFGS method [65] was chosen since it can satisfactorily handle the constraints placed on the network parameters for the optimization. However, an additional experiment was executed using different local optimization techniques and the results for the classification datasets are presented in Table 10 and the results for the regression datasets are shown in Table 11. The following notation is used in the experimental tables:

1. The column ADAM denotes the incorporation of the ADAM local optimization method [100] in the current technique.
2. The column LBFGS denotes the incorporation of the Limited Memory BFGS (L-BFGS) method [101] as the local search procedure.
3. The column BFGS represents the incorporation of the BFGS method, modified by Powell [65] as the local search procedure.

Table 10. Experimental results with different local optimization techniques in the proposed method, with application on the classification datasets.

DATASET	ADAM	LBFGS	BFGS
APPENDICITIS	**14.60%**	14.60%	14.70%
AUSTRALIAN	**14.78%**	15.29%	14.80%
BALANCE	16.89%	11.47%	**8.66%**
CIRCULAR	10.79%	8.15%	**5.32%**
CLEVELAND	49.10%	48.45%	**47.93%**
DERMATOLOGY	22.32%	23.03%	**20.89%**
ECOLI	**48.15%**	49.48%	48.21%
FERT	**18.10%**	20.10%	20.50%
HABERMAN	**26.27%**	26.63%	26.70%
HAYES-ROTH	36.23%	36.31%	**31.77%**
HEART	15.71%	15.00%	**14.74%**
HEARTATTACK	21.33%	20.07%	20.43%
HOUSEVOTES	3.39%	3.39%	**3.26%**
IONOSPHERE	**10.88%**	11.29%	11.92%
LIVERDISORDER	31.80%	33.18%	**31.77%**
MAMMOGRAPHIC	16.71%	16.30%	**15.81%**
PARKINSONS	**12.10%**	12.32%	12.53%
PIMA	25.49%	24.97%	**24.00%**
POPFAILURES	**6.20%**	6.31%	6.44%
REGIONS2	24.20%	24.06%	**23.18%**
SAHEART	28.87%	28.96%	**28.09%**
SEGMENT	50.16%	46.63%	**43.12%**
SPIRAL	46.33%	45.24%	43.99%
STUDENT	4.05%	**3.98%**	4.55%
TRANSFUSION	24.74%	24.56%	**23.43%**
WDBC	6.30%	5.66%	**4.41%**
WINE	11.47%	10.06%	**9.77%**
Z_F_S	12.33%	10.92%	**8.53%**
Z_O_N_F_S	41.40%	45.77%	**38.58%**
ZO_NF_S	11.26%	9.83%	**6.84%**
ZONF_S	2.64%	**2.35%**	2.52%
ZOO	8.50%	8.70%	**7.20%**
AVERAGE	**21.03%**	**20.72%**	**19.52%**

Table 11. Experimental results with local optimization techniques in the current work, with application on the regression datasets.

DATASET	ADAM	LBFGS	BFGS
ABALONE	4.73	4.49	**4.33**
AIRFOIL	0.003	0.003	**0.002**
BASEBALL	53.98	51.88	**48.42**
BK	**0.05**	0.08	0.07
BL	0.013	0.007	**0.002**
CONCRETE	0.006	0.006	**0.005**
DEE	0.24	**0.23**	0.23
HO	0.012	0.012	**0.01**
HOUSING	20.20	20.49	**16.01**
FY	**0.042**	0.042	0.042
LASER	0.012	**0.005**	0.005
LW	**0.011**	0.011	0.012
MORTGAGE	0.068	0.084	**0.026**
MUNDIAL	0.033	**0.029**	0.034
PL	0.031	0.023	**0.022**
QUAKE	**0.036**	0.037	0.04
REALESTATE	74.66	74.49	**70.99**
SN	0.025	0.024	**0.023**
TREASURY	0.114	0.084	**0.066**
TZ	**0.03**	0.08	0.03
VE	**0.024**	0.132	0.025
AVERAGE	**7.35**	**7.25**	**6.69**

The BFGS method achieved lower values for the test error in the majority of cases and this is also evident in the Figure 13, where a box plot is depicted for the classification datasets.

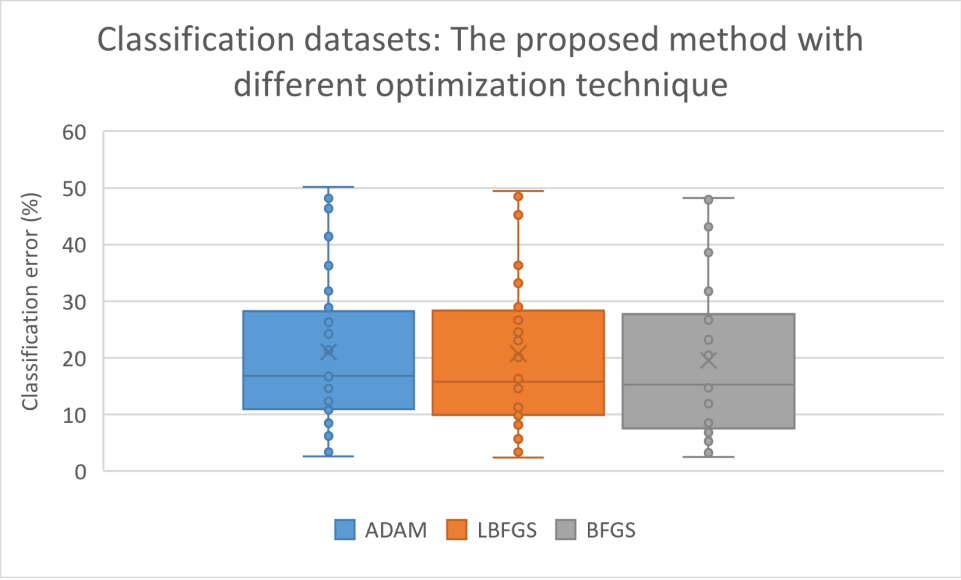

Figure 13. Box plot for the experiment with the proposed method and different local optimization methods. The current work was executed on the classification datasets.

4. Conclusions

An extension of the artificial neural network construction technique was presented in the present work, in which the continuous application of a local optimization method to chromosomes that selected randomly, was introduced. The application of the local optimization method is done in such a way as not to alter the architecture of the neural

network constructed by Grammatical Evolution. The proposed modified method was applied on a series of benchmark datasets found in the relevant literature and, judging from the experimental results, it reduced significantly the test error of the original method in most datasets.

Moreover, to establish the stability of the proposed technique, additional experiments were carried out in which a number of critical parameters were varied over a range of values. After the completion of these experiments, it became clear that there is no significant difference in the effectiveness of the proposed method, even if these critical parameters change significantly from execution to execution. The only case where a significant difference in the effectiveness of the proposed technique was found was when different local optimization techniques were used, where the BFGS variant appeared to achieve the best results in the majority of cases.

Nevertheless, one major drawback of the current work is the additional execution time required from the execution of the local search optimization techniques. Since the Grammatical Evolution procedure is a modified genetic algorithm, the generated artificial neural networks are independent of themselves, parallel programming techniques may be used in order to improve the speed of the method, such as the usage of MPI [102] or the OpenMP library [103].

Author Contributions: V.C. and I.G.T. performed the mentioned experiments. I.G.T. wrote the used software and D.T., A.T. and V.C. executed all the statistical comparisons and prepared the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Institutional Review Board Statement: Not applicable.

Institutional Review Board Statement: Not applicable.

Acknowledgments: This research has been financed by the European Union : Next Generation EU through the Program Greece 2.0 National Recovery and Resilience Plan , under the call RESEARCH – CREATE – INNOVATE, project name “iCREW: Intelligent small craft simulator for advanced crew training using Virtual Reality techniques" (project code:TAEDK-06195).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. Heliyon, 4(11).
2. Suryadevara, S., & Yanamala, A. K. Y. (2021). A Comprehensive Overview of Artificial Neural Networks: Evolution, Architectures, and Applications. Revista de Inteligencia Artificial en Medicina, 12(1), 51-76.
3. P. Baldi, K. Cranmer, T. Faucett et al, Parameterized neural networks for high-energy physics, Eur. Phys. J. C **76**, 2016.
4. Baldi, P., Cranmer, K., Faucett, T., Sadowski, P., & Whiteson, D. (2016). Parameterized neural networks for high-energy physics. The European Physical Journal C, 76(5), 1-7.
5. G. Carleo,M. Troyer, Solving the quantum many-body problem with artificial neural networks, Science **355**, pp. 602-606, 2017.
6. Khoo, Y., Lu, J., & Ying, L. (2021). Solving parametric PDE problems with artificial neural networks. European Journal of Applied Mathematics, 32(3), 421-435.
7. A. Kumar Yadav, S.S. Chandel, Solar radiation prediction using Artificial Neural Network techniques: A review, Renewable and Sustainable Energy Reviews **33**, pp. 772-781, 2014.
8. A. Escamilla-García, G.M. Soto-Zarazúa, M. Toledano-Ayala, E. Rivas-Araiza, A. Gastélum-Barrios, Abraham,Applications of Artificial Neural Networks in Greenhouse Technology and Overview for Smart Agriculture Development, Applied Sciences **10**, Article number 3835, 2020.
9. 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.
10. 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.
11. Khosravi, A. K. R. M. L. P. J., Koury, R. N. N., Machado, L., & Pabon, J. J. G. (2018). Prediction of wind speed and wind direction using artificial neural network, support vector regression and adaptive neuro-fuzzy inference system. Sustainable Energy Technologies and Assessments, 25, 146-160.

12. Lukas Falat and Lucia Pancikova, Quantitative Modelling in Economics with Advanced Artificial Neural Networks, Procedia Economics and Finance **34**, pp. 194-201, 2015.
13. 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.
14. 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.
15. Ronadl Bartzatt, Prediction of Novel Anti-Ebola Virus Compounds Utilizing Artificial Neural Network (ANN), Chemistry Faculty Publications **49**, pp. 16-34, 2018.
16. Vora, K., & Yagnik, S. (2014). A survey on backpropagation algorithms for feedforward neural networks. International Journal of Engineering Development and Research, 1(3), 193-197.
17. K. Vora, S. Yagnik, A survey on backpropagation algorithms for feedforward neural networks, International Journal of Engineering Development and Research **1**, pp. 193-197, 2014.
18. Pajchrowski, T., Zawirski, K., & Nowopolski, K. (2014). Neural speed controller trained online by means of modified RPROP algorithm. IEEE transactions on industrial informatics, 11(2), 560-568.
19. Hermanto, R. P. S., & Nugroho, A. (2018). Waiting-time estimation in bank customer queues using RPROP neural networks. Procedia Computer Science, 135, 35-42.
20. D. P. Kingma, J. L. Ba, ADAM: a method for stochastic optimization, in: Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), pp. 1–15, 2015.
21. C.L. Kuo, E.E. Kuruoglu, W.K.V. Chan, Neural Network Structure Optimization by Simulated Annealing, Entropy **24**, 348, 2022.
22. Reynolds, J., Rezgui, Y., Kwan, A., & Piriou, S. (2018). A zone-level, building energy optimisation combining an artificial neural network, a genetic algorithm, and model predictive control. Energy, 151, 729-739.
23. Das, G., Pattnaik, P. K., & Padhy, S. K. (2014). Artificial neural network trained by particle swarm optimization for non-linear channel equalization. Expert Systems with Applications, 41(7), 3491-3496.
24. Wang, L., Zeng, Y., & Chen, T. (2015). Back propagation neural network with adaptive differential evolution algorithm for time series forecasting. Expert Systems with Applications, 42(2), 855-863.
25. K.M. Salama, A.M. Abdelbar, Learning neural network structures with ant colony algorithms, Swarm Intell **9**, pp. 229–265, 2015.
26. S. Mirjalili, How effective is the Grey Wolf optimizer in training multi-layer perceptrons, Appl Intell **43**, pp. 150–161, 2015.
27. I. Aljarah, H. Faris, S. Mirjalili, Optimizing connection weights in neural networks using the whale optimization algorithm, Soft Comput **22**, pp. 1–15, 2018.
28. M. Zhang, K. Hibi, J. Inoue, GPU-accelerated artificial neural network potential for molecular dynamics simulation, Computer Physics Communications **285**, 108655, 2023.
29. T.M. Varnava, A.J.Meade, An initialization method for feedforward artificial neural networks using polynomial bases, Advances in Adaptive Data Analysis **3**, pp. 385-400, 2011.
30. I. Ivanova, M. Kubat, Initialization of neural networks by means of decision trees, Knowledge-Based Systems **8**, pp. 333-344, 1995.
31. S.S. Sodhi, P. Chandra, Interval based Weight Initialization Method for Sigmoidal Feedforward Artificial Neural Networks, AASRI Procedia **6**, pp. 19-25, 2014.
32. K. Chumachenko, A. Iosifidis, M. Gabbouj, Feedforward neural networks initialization based on discriminant learning, Neural Networks **146**, pp. 220-229, 2022.
33. Q. Chen, W. Hao, J. He, A weight initialization based on the linear product structure for neural networks, Applied Mathematics and Computation **415**, 126722, 2022.
34. J. Arifovic, R. Gençay, Using genetic algorithms to select architecture of a feedforward artificial neural network, Physica A: Statistical Mechanics and its Applications **289**, pp. 574-594, 2001.
35. P.G. Benardos, G.C. Vosniakos, Optimizing feedforward artificial neural network architecture, Engineering Applications of Artificial Intelligence **20**, pp. 365-382, 2007.
36. B.A. Garro, R.A. Vázquez, Designing Artificial Neural Networks Using Particle Swarm Optimization Algorithms, Computational Intelligence and Neuroscience, 369298, 2015.
37. B. Baker, O. Gupta, N. Naik, R. Raskar, Designing neural network architectures using reinforcement learning. arXiv preprint arXiv:1611.02167, 2016.
38. M. M. Islam, M. A. Sattar, M. F. Amin, X. Yao, K. Murase, A New Adaptive Merging and Growing Algorithm for Designing Artificial Neural Networks, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **39**, pp. 705-722, 2009.
39. M. O'Neill, C. Ryan, Grammatical evolution, IEEE Trans. Evol. Comput. **5,**pp. 349–358, 2001.
40. I.G. Tsoulos, D. Gavrilis, E. Glavas, Neural network construction and training using grammatical evolution, Neurocomputing **72**, pp. 269-277, 2008.
41. G.V. Papamokos, I.G. Tsoulos, I.N. Demetropoulos, E. Glavas, Location of amide I mode of vibration in computed data utilizing constructed neural networks, Expert Systems with Applications **36**, pp. 12210-12213, 2009.
42. I.G. Tsoulos, D. Gavrilis, E. Glavas, Solving differential equations with constructed neural networks, Neurocomputing **72**, pp. 2385-2391, 2009.

43. I.G. Tsoulos, G. Mitsi, A. Stavrakoudis, S. Papapetropoulos, Application of Machine Learning in a Parkinson's Disease Digital Biomarker Dataset Using Neural Network Construction (NNC) Methodology Discriminates Patient Motor Status, Frontiers in ICT 6, 10, 2019.
44. V. Christou, I.G. Tsoulos, V. Loupas, A.T. Tzallas, C. Gogos, P.S. Karvelis, N. Antoniadis, E. Glavas, N. Giannakeas, Performance and early drop prediction for higher education students using machine learning, Expert Systems with Applications **225**, 120079, 2023.
45. E.I. Toki, J. Pange, G. Tatsis, K. Plachouras, I.G. Tsoulos, Utilizing Constructed Neural Networks for Autism Screening, Applied Sciences **14**, 3053, 2024.
46. I.G. Tsoulos, A.Tzallas, D. Tsalikakis, NNC: A tool based on Grammatical Evolution for data classification and differential equation solving, SoftwareX **10**, 100297, 2019.
47. 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.
48. C. Ryan, J. Collins, M. O'Neill, Grammatical evolution: Evolving programs for an arbitrary language. In: Banzhaf, W., Poli, R., Schoenauer, M., Fogarty, T.C. (eds) Genetic Programming. EuroGP 1998. Lecture Notes in Computer Science, vol 1391. Springer, Berlin, Heidelberg, 1998.
49. M. O'Neill, M., C. Ryan, Evolving Multi-line Compilable C Programs. In: Poli, R., Nordin, P., Langdon, W.B., Fogarty, T.C. (eds) Genetic Programming. EuroGP 1999. Lecture Notes in Computer Science, vol 1598. Springer, Berlin, Heidelberg, 1999.
50. C. Ryan, M. O'Neill, J.J. Collins, Grammatical evolution: Solving trigonometric identities, proceedings of Mendel. Vol. 98. 1998.
51. A.O. Puente, R. S. Alfonso, M. A. Moreno, Automatic composition of music by means of grammatical evolution, In: APL '02: Proceedings of the 2002 conference on APL: array processing languages: lore, problems, and applications July 2002 Pages 148–155.
52. I. Dempsey, M.O' Neill, A. Brabazon, Constant creation in grammatical evolution, International Journal of Innovative Computing and Applications **1** , pp 23–38, 2007.
53. E. Galván-López, J.M. Swafford, M. O'Neill, A. Brabazon, Evolving a Ms. PacMan Controller Using Grammatical Evolution. In: , et al. Applications of Evolutionary Computation. EvoApplications 2010. Lecture Notes in Computer Science, vol 6024. Springer, Berlin, Heidelberg, 2010.
54. N. Shaker, M. Nicolau, G. N. Yannakakis, J. Togelius, M. O'Neill, Evolving levels for Super Mario Bros using grammatical evolution, 2012 IEEE Conference on Computational Intelligence and Games (CIG), 2012, pp. 304-31.
55. D. Martínez-Rodríguez, J. M. Colmenar, J. I. Hidalgo, R.J. Villanueva Micó, S. Salcedo-Sanz, Particle swarm grammatical evolution for energy demand estimation, Energy Science and Engineering **8**, pp. 1068-1079, 2020.
56. N. R. Sabar, M. Ayob, G. Kendall, R. Qu, Grammatical Evolution Hyper-Heuristic for Combinatorial Optimization Problems, IEEE Transactions on Evolutionary Computation **17**, pp. 840-861, 2013.
57. C. Ryan, M. Kshirsagar, G. Vaidya, G. et al. Design of a cryptographically secure pseudo random number generator with grammatical evolution. Sci Rep **12**, 8602, 2022.
58. P.J. Pereira, P. Cortez, R. Mendes, Multi-objective Grammatical Evolution of Decision Trees for Mobile Marketing user conversion prediction, Expert Systems with Applications **168**, 114287, 2021.
59. F. Castejón, E.J. Carmona, Automatic design of analog electronic circuits using grammatical evolution, Applied Soft Computing **62**, pp. 1003-1018, 2018.
60. L. Araujo, J. Martinez-Romo, A. Duque, Discovering taxonomies in Wikipedia by means of grammatical evolution. Soft Comput **22**, pp. 2907–2919, 2018.
61. C. Martín, D. Quintana, P. Isasi, Grammatical Evolution-based ensembles for algorithmic trading, Applied Soft Computing **84**, 105713, 2019.
62. Moore, J.H., Sipper, M. (2018). Grammatical Evolution Strategies for Bioinformatics and Systems Genomics. In: Ryan, C., O'Neill, M., Collins, J. (eds) Handbook of Grammatical Evolution. Springer, Cham. https://doi.org/10.1007/978-3-319-78717-6_16
63. Peabody, C., & Seitzer, J. (2015, March). GEF: a self-programming robot using grammatical evolution. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 29, No. 1).
64. Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. Neural networks, 2(5), 359-366.
65. M.J.D Powell, A Tolerant Algorithm for Linearly Constrained Optimization Calculations, Mathematical Programming **45**, pp. 547-566, 1989.
66. M. Kelly, R. Longjohn, K. Nottingham, The UCI Machine Learning Repository. 2023. Available online: https://archive.ics.uci.edu (accessed on 18 February 2024).
67. J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. Journal of Multiple-Valued Logic and Soft Computing 17, pp. 255-287, 2011.
68. Weiss, Sholom M. and Kulikowski, Casimir A., Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems, Morgan Kaufmann Publishers Inc, 1991.
69. J.R. Quinlan, Simplifying Decision Trees. International Journal of Man-Machine Studies **27**, pp. 221-234, 1987.
70. T. Shultz, D. Mareschal, W. Schmidt, Modeling Cognitive Development on Balance Scale Phenomena, Machine Learning **16**, pp. 59-88, 1994.

71. Z.H. Zhou,Y. Jiang, NeC4.5: neural ensemble based C4.5," in IEEE Transactions on Knowledge and Data Engineering **16**, pp. 770-773, 2004.
72. R. Setiono , W.K. Leow, FERNN: An Algorithm for Fast Extraction of Rules from Neural Networks, Applied Intelligence **12**, pp. 15-25, 2000.
73. 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.
74. P. Horton, K.Nakai, A Probabilistic Classification System for Predicting the Cellular Localization Sites of Proteins, In: Proceedings of International Conference on Intelligent Systems for Molecular Biology **4**, pp. 109-15, 1996.
75. 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.
76. I. Kononenko, E. Šimec, M. Robnik-Šikonja, Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF, Applied Intelligence **7**, pp. 39–55, 1997
77. R.M. French, N. Chater, Using noise to compute error surfaces in connectionist networks: a novel means of reducing catastrophic forgetting, Neural Comput. **14**, pp. 1755-1769, 2002.
78. J. Garcke, M. Griebel, Classification with sparse grids using simplicial basis functions, Intell. Data Anal. **6**, pp. 483-502, 2002.
79. J.G. Dy , C.E. Brodley, Feature Selection for Unsupervised Learning, The Journal of Machine Learning Research **5**, pp 845–889, 2004.
80. S. J. Perantonis, V. Virvilis, Input Feature Extraction for Multilayered Perceptrons Using Supervised Principal Component Analysis, Neural Processing Letters **10**, pp 243–252, 1999.
81. M. Elter, R. Schulz-Wendtland, T. Wittenberg, The prediction of breast cancer biopsy outcomes using two CAD approaches that both emphasize an intelligible decision process, Med Phys. **34**, pp. 4164-72, 2007.
82. M.A. Little, P.E. McSharry, E.J. Hunter, J. Spielman, L.O. Ramig, Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. IEEE Trans Biomed Eng. **56**, pp. 1015-1022, 2009.
83. J.W. Smith, J.E. Everhart, W.C. Dickson, W.C. Knowler, R.S. Johannes, Using the ADAP learning algorithm to forecast the onset of diabetes mellitus, In: Proceedings of the Symposium on Computer Applications and Medical Care IEEE Computer Society Press, pp.261-265, 1988.
84. 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.
85. N. Giannakeas, M.G. Tsipouras, A.T. Tzallas, K. Kyriakidi, Z.E. Tsianou, P. Manousou, A. Hall, E.C. Karvounis, V. Tsianos, E. Tsianos, A clustering based method for collagen proportional area extraction in liver biopsy images (2015) Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS, 2015-November, art. no. 7319047, pp. 3097-3100.
86. T. Hastie, R. Tibshirani, Non-parametric logistic and proportional odds regression, JRSS-C (Applied Statistics) **36**, pp. 260–276, 1987.
87. M. Dash, H. Liu, P. Scheuermann, K. L. Tan, Fast hierarchical clustering and its validation, Data & Knowledge Engineering **44**, pp 109–138, 2003.
88. P. Cortez, A. M. Gonçalves Silva, Using data mining to predict secondary school student performance, In Proceedings of 5th FUture BUsiness TEChnology Conference (FUBUTEC 2008) (pp. 5–12). EUROSIS-ETI, 2008.
89. I-Cheng Yeh, King-Jang Yang, Tao-Ming Ting, Knowledge discovery on RFM model using Bernoulli sequence, Expert Systems with Applications **36**, pp. 5866-5871, 2009.
90. W.H. Wolberg, O.L. Mangasarian, Multisurface method of pattern separation for medical diagnosis applied to breast cytology, Proc Natl Acad Sci U S A. **87**, pp. 9193–9196, 1990.
91. 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.
92. P. Zhong, M. Fukushima, Regularized nonsmooth Newton method for multi-class support vector machines, Optimization Methods and Software **22**, pp. 225-236, 2007.
93. 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, Phys. Rev. E **64**, pp. 1-8, 2001.
94. M. Koivisto, K. Sood, Exact Bayesian Structure Discovery in Bayesian Networks, The Journal of Machine Learning Research **5**, pp. 549–573, 2004.
95. Nash, W.J.; Sellers, T.L.; Talbot, S.R.; Cawthor, A.J.; Ford, W.B. The Population Biology of Abalone (_Haliotis_ species) in Tasmania. I. Blacklip Abalone (_H. rubra_) from the North Coast and Islands of Bass Strait; Sea Fisheries Division, Technical Report 48; Sea Fisheries Division, Department of Primary Industry and Fisheries: Orange, NSW, Australia, 1994.
96. T.F. Brooks, D.S. Pope, A.M. Marcolini, Airfoil self-noise and prediction. Technical report, NASA RP-1218, July 1989.
97. J.S. Simonoff, Smooting Methods in Statistics, Springer - Verlag, 1996.
98. I.Cheng Yeh, Modeling of strength of high performance concrete using artificial neural networks, Cement and Concrete Research. **28**, pp. 1797-1808, 1998.

99. D. Harrison and D.L. Rubinfeld, Hedonic prices and the demand for clean ai, J. Environ. Economics & Management **5**, pp. 81-102, 1978.
100. D. P. Kingma, J. L. Ba, ADAM: a method for stochastic optimization, in: Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), pp. 1–15, 2015.
101. D.C. Liu, J. Nocedal, On the Limited Memory Method for Large Scale Optimization, Mathematical Programming B. **45**, pp. 503–528, 1989.
102. Gropp, W.; Lusk, E.; Doss, N.; Skjellum, A. A high-performance, portable implementation of the MPI message passing interface standard. Parallel Comput. 1996, 22, 789–828.
103. Chandra, R. Parallel Programming in OpenMP; Morgan Kaufmann: Cambridge, MA, USA, 2001.