Preprints.org

Article

Not peer-reviewed version

Constructing the Bounds for Neural Network Training Using Grammatical Evolution

Ioannis G. Tsoulos *, Alexandros Tzallas, Evangelos Karvounis

Posted Date: 5 October 2023

doi: 10.20944/preprints202310.0274.v1

Keywords: Neural networks; Genetic algorithms; Grammatical Evolution

Preprints.org is a free multidiscipline platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2023 doi:10.20944/preprints202310.0274.v1

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Article

Constructing the Bounds for Neural Network Training Using Grammatical Evolution

Ioannis G. Tsoulos *,†, Alexandros Tzallas † and Evangelos Karvounis

Department of Informatics and Telecommunications, University of Ioannina, Greece; tzallas@uoi.gr (A.T.); ekarvounis@uoi.gr (E.K.)

* Correspondence: itsoulos@uoi.gr

† Current address: Department of Informatics and Telecommunications, University of Ioannina, Greece.

‡ These authors contributed equally to this work.

Abstract: Artificial neural networks are widely established models of computational intelligence that have been tested for effectiveness in a variety of real-world applications. These models require fitting a set of parameters through the use of some optimization technique. However, an issue that researchers often face is finding an efficient range of values for the parameters of the artificial neural network. This paper proposes an innovative technique of generating a promising range of values for the parameters of the artificial neural network. Finding the value field is done by a series of rules for partitioning the original set of values or expanding it, which rules are generated using Grammatical Evolution. After finding a promising interval of values, any optimization technique such as a genetic algorithm can be used to train the artificial neural network on that interval of values. The new technique was tested on a wide range of problems from the relevant literature and the results were extremely promising.

Keywords: neural networks; genetic algorithms; Grammatical Evolution

1. Introduction

Artificial neural networks (ANNs) are widespread machine learning models, which base their dynamics on a series of parameters which are also called computational units or weights in the relevant literature [1,2]. Neural networks are widely used in a variety of scientific applications, such as problems from physics [3–5], solutions of differential equations [6,7], agriculture problems [8,9], chemistry problems [10–12], problems that appear in the area of economic sciences [13–15], problems related to medicine [16,17] etc. Commonly, the neural networks are expressed as a function $N(\overrightarrow{x}, \overrightarrow{w})$, with the assumption that the vector $\overrightarrow{x}$ stands for the input vector to the neural network and the vector $\overrightarrow{w}$ is the parameter vector that should be estimated. The first vector is called pattern in the bibliography and the second one weight vector. Techniques that adjust the parameter vector $\overrightarrow{w}$ minimize the so-called training error, defined as:

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

In Equation (1) the set $\left(\overrightarrow{x_i}, y_i\right)$, $i = 1, ..., M$ is denoted as the train set of the neural network. The y_i are considered the target outputs for the $\overrightarrow{x_i}$ points . As proposed in [18], neural networks can be expressed as the following summation:

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

The number of processing units is denoted by H and the value d stands for the dimension of input vector $\overrightarrow{x}$. The function $\sigma(x)$ denotes the well - known sigmoid function expressed as:

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

check for updates

Citation: Tsoulos, I,G.; Tzallas, A.; Karvounis, E. Constructing the bounds for neural network training using Grammatical Evolution. *Preprints* **2023**, *1*, 0. https://doi.org/

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

© 2023 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2023 doi:10.20944/preprints202310.0274.v1

From the above equations one can calculated the total number of parameters in the neural network as:

$$n = (d + 2)H \tag{4}$$

The training of neural networks has been performed by a variety of numerical methods in recent bibliography, such as: the Back Propagation method [19,20], the RPROP method [21–23], the Adam optimizer [24,25] etc. Furthermore, more advanced optimization methods have been incorporated to optimize the parameter vector of neural networks by minimizing Equation (1) such as Quasi Newton methods [26,27], the Tabu Search method [28], Simulated Annealing [29,30], Genetic Algorithms [31,32], Particle Swarm Optimization [33,34], Differential Evolution [35,36], Ant Colony Optimization [37] etc.

Also, recently a series of hybrid optimization methods have been proposed to tackle the minimization of the training error, such as a method that combines PSO and genetic algorithms [38,39], incorporation of a particle swarm optimization algorithm and gravitational search algorithm [40], a combination of a genetic algorithm and controlled gradient method [41] etc.

In addition, an important topic for artificial neural networks such as parameter initialization has been studied by several researchers in recent years. In this area appeared papers such as initialization with decision trees [42], incorporation of the Cauchy's inequality [43], discriminant learning [44] etc. Also, another interesting topic in the area of neural networks is weight decaying, where some regularization is applied to the parameters of the neural network to avoid the overfitting problem. For this topic, several research papers have been suggested, such as a method that incorporates positive correlation [45], the SarProp algorithm [46], usage of pruning techniques [47] etc.

This paper proposes a two-stage method for efficient training of the parameters of artificial neural networks. In the first stage, a value interval is constructed for the parameters of the neural network through Grammatical Evolution [48]. In the first stage of this work, after an initial estimate of the interval for the parameters of the neural network is made, a series of rules for partitioning and expanding the initial value interval are applied with the usage of Grammatical Evolution, until a promising value interval is found. In the second stage of the method, a genetic algorithm is applied to train the parameters of the artificial neural network. Training is performed within the optimal interval of values found in the first stage of the method.

The rest of this article is organized as follows: in Section 2 the steps of the proposed method are outlined in detail, in Section 3 the datasets used in the conducted experiments as well the experimental results are presented and finally in Section 4 some conclusions are presented.

2. Method description

In this section a detailed description of the Grammatical Evolution technique is provided, accompanied by the proposed language that creates intervals for the parameters of the neural networks. Subsequently, the first phase of the proposed technique is described and finally, the second phase with the application of a Genetic Algorithm to the outcome of the first phase, is thoroughly analyzed.

2.1. Grammatical evolution

Grammatical evolution is an evolutionary process, where the chromosomes represent production rules of some provided BNF (Backus–Naur form) grammar [49] and this evolutionary process can evolve programs in the underlying language. Grammatical Evolution has been used successfully in many real world problems, such as function approximation[50,51], solution of trigonometric equations [52], automatic music composition [53], neural network construction [54,55], creating numeric constraints [56], video games [57,58], estimation of energy demand [59], combinatorial optimization [60], cryptography [61], evolution of decision trees [62], automatic design of analog electronic circuits [63] etc. Any BNF grammar can be described as the set $G = (N, T, S, P)$ where

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2023 doi:10.20944/preprints202310.0274.v1

- N is the set of the non-terminal symbols. Every symbol in N has a series of production rules, used to produce terminal symbols.
- T is the set of terminal symbols.
- S denotes the start symbol of the grammar, with $S \in N$.
- P is the set of production rules, to create terminal symbols non - terminal symbols. These rules are in the form $A \to a$ or $A \to aB$, $A, B \in N$, $a \in T$.

The algorithm initiates from the symbol S and iteratively produces terminal symbols by replacing non-terminal symbols with the right hand of the selected production rule. Every production rule is is selected using the following steps:

- Get the next element V from the current chromosome.
- Select the next production rule according to: Rule = V mod N_R, where N_R is the total number of production rules for the current non – terminal symbol.

The grammar used in the current work is shown in Figure 1.

```
S::=<expr>   (0)
<expr> ::=  (<xlist> , <lcommand>,<rcommand>)  (0)
           |<expr>,<expr>                (1)
<xlist>::=x1    (0)
           | x2 (1)
           .........
           | xn (n)
<left_command>  ::= NOTHING (0)
           |        EXPAND  (1)
           |        DIVIDE  (2)
<right_command> ::= NOTHING (0)
           |        EXPAND  (1)
           |        DIVIDE  (2)
```

Figure 1. BNF grammar used in the current work. The value n stands for the number of total parameters in the neural network.

The symbols in <> represent the non-terminal symbols of the grammar. The numbers in parentheses in the right part of each rule, indicate production rule sequence numbers. In the proposed language described by the grammar of the scheme, three distinct operations can be applied to parameters of the artificial neural network either on the left end of the parameter's value range (described by <lcommand>) or on the right end of the value range (described by the <rcommand> symbol). These commands are

1. NOTHING. This command means that no action takes place.
2. EXPAND. With this command, the corresponding end of the value field is extended by 50% of the width of the field.
3. DIVIDE. With this command, the corresponding end of the value field is shrunk by 50% of the width of the field.

As a complete example consider a neural network with $H = 2$ hidden nodes and the dimension of the objective problem is set to $d = 2$. Hence, the total number of parameters in the neural network is $n = (d + 2)H = 8$. Also, for reasons of simplicity, we consider that the initial range of values for all parameters of the artificial neural network is the interval [-2,2] and is denotes as a the set of intervals

$$I_N = \{[-2,2],[-2,2],[-2,2],[-2,2],[-2,2],[-2,2],[-2,2],[-2,2]\} \quad (5)$$

and consider also the chromosome

$$x = [9,8,6,4,15,9,16,23,7]$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2023 doi:10.20944/preprints202310.0274.v1

The steps to produce the final program

$$p_{\text{test}} = (x7, \text{EXPAND}, \text{NOTHING}), (x1, \text{DIVIDE}, \text{EXPAND})$$

are shown in Table 1.

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

Expression	Chromosome	Operation
	9,8,6,4,15,9,16,23,8	9 mod 2=1
<expr>,<expr>	8,6,4,15,9,16,23,8	8 mod 2=0
(<xlist>,<lcommand>,<rcommand>),<expr>	6,4,15,9,16,23,8	6 mod 8=6
(x7,<lcommand>,<rcommand>),<expr>	4,15,9,16,23,8	4 % 3=1
(x7,EXPAND,<rcommand>),<expr>	15,9,16,23,8	15%3=0
(x7,EXPAND,NOTHING),<expr>	9,16,23,8	9 %2 =1
(x7,EXPAND,NOTHING),(<xlist>,<lcommand>,<rcommand>)	16,23,8	16%8=0
(x7,EXPAND,NOTHING),(x1,<lcommand>,<rcommand>)	23,8	23%3=2
(x7,EXPAND,NOTHING),(x1,DIVIDE,<rcommand>)	8	8%3=2
(x7,EXPAND,NOTHING),(x1,DIVIDE,EXPAND)		

After the application of the program p_{test} to the original interval I_N of Equation (5), the new set has as follows:

$$I_N = \{[-4,2],[-2,2],[-2,2],[-2,2],[-2,2],[-2,2],[0,4],[-2,2]\} \tag{6}$$

2.2. The first phase of the proposed method

Before starting the process of creating partitioning rules and expanding the value interval of the neural network weights, an initial estimate of this interval should be made. In the present work, only a few steps of a genetic algorithm are applied to make an initial estimate of the value interval. At the end of the genetic algorithm, the chromosome with the best value is the

$$\text{IX} = (\text{IX}_1, \text{IX}_2, \ldots, \text{IX}_n) \tag{7}$$

So, the initial interval set for the parameters of the neural network is calculated as

$$I_N = \{[-\text{IX}_1, \text{IX}_1], \{-\text{IX}_2, \text{IX}_2\}, \ldots, [-\text{IX}_n, \text{IX}_n]\} \tag{8}$$

The first phase of the proposed technique will accept the set I_N as input and seek to compute a new set by reducing the training error of the artificial neural network. The first step is mainly a genetic algorithm and the corresponding steps are as follows:

1. **Set** N_c as the number of chromosomes for the Grammatical Evolution.
2. **Set** as H the number of weights for the neural network.
3. **Set** N_g the maximum number of allowed generations.
4. **Set** as p_s the selection rate , with $p_s \leq 1$.
5. **Set** as p_m the mutation rate, with $p_m \leq 1$.
6. **Set** N_s as the number of randomly created neural networks, that will be used in the fitness calculation.
7. **Initialize** randomly the N_c chromosomes. Every chromosome is a set of integer numbers used to produce valid programs through Grammatical Evolution and the associated grammar of Figure 1.
8. **Set** $f^* = [\infty, \infty]$, the best discovered fitness. For this algorithm, we consider the fitness function f_g of any given chromosome g as an interval $f_g = \left[f_{g,\textbf{low}}, f_{g,\textbf{upper}}\right]$
9. **Set** iter=0.
10. **For** $i = 1, \ldots, N_c$ **do**

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2023 doi:10.20944/preprints202310.0274.v1

(a) **Create** for the chromosome c_i the corresponding program p_i using the grammar of Figure 1.
(b) **Apply** the program p_i to I_N in order to produce the bounds $\left[\vec{L_{p_i}}, \overrightarrow{R_{p_i}}\right]$.
(c) **Set** $E_{\min} = \infty$, $E_{\max} = -\infty$
(d) **For** $j = 1, \ldots, N_S$ **do**
   i. **Create** randomly $g_j \in \left[\vec{L_{p_i}}, \overrightarrow{R_{p_i}}\right]$ as a set for the parameters of neural network.
   ii. **Calculate** the associated training error $E_{g_j} = \sum_{k=1}^{M} \left(N(\vec{x}_k, \vec{g_j}) - y_k\right)^2$
   iii. **If** $E_{g_j} \leq E_{\min}$ **then** $E_{\min} = E_{g_j}$
   iv. **If** $E_{g_j} \geq E_{\max}$ **then** $E_{\max} = E_{g_j}$
(e) **EndFor**
(f) **Set** $f_i = [E_{\min}, E_{\max}]$ as the fitness value for the chromosome c_i.

11. **EndFor**
12. **Apply** the selection procedure. Firstly, the chromosomes are sorted with correspondence to their fitness values. Since fitness is considered an interval, a fitness comparison function is required. For this reason the operator $L^*(f_a, f_b)$ to compare two fitness values $f_a = [a_1, a_2]$ and $f_b = [b_1, b_2]$ as follows:

$$L^*(f_a, f_b) = \begin{cases} \text{TRUE}, & a_1 < b_1, \text{OR } (a_1 = b_1 \text{ AND } a_2 < b_2) \\ \text{FALSE}, & \text{OTHERWISE} \end{cases} \quad (9)$$

In practice this means that the fitness value f_a is considered smaller than f_b if $L^*(f_a, f_b) = \text{TRUE}$. The first $(1 - p_s) \times N_c$ chromosomes with the lowest fitness values are copied to the next generation. The remaining chromosomes are substituted by chromosomes produced by the crossover procedure. During the selection process, for every new offspring, two chromosomes are selected as parents from the population using well - known procedure of tournament selection.
13. **Apply** the crossover procedure. For each pair (z, w) of parents, two new chromosomes $\tilde{z}$ and $\tilde{w}$ are created using the one - point crossover, graphically shown in Figure 2.
14. **Apply** the mutation procedure. For each element of every chromosome alter the corresponding element with probability p_m.
15. **Set** iter=iter+1
16. **If** iter$\leq N_g$ goto step 10.

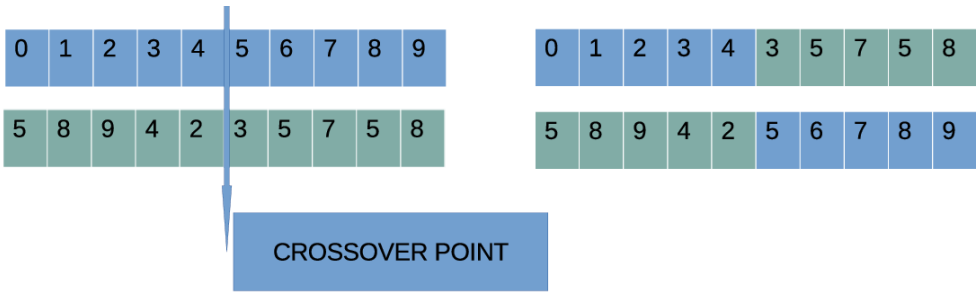

Figure 2. The one point crossover, used in the Grammatical Evolution.

2.3. The second phase of the proposed method

1. **Initialization Step**
   (a) **Set** N_c as the number of chromosomes that participate in the genetic algorithm.
   (b) **Set** N_g the maximum number of allowed iterations.
   (c) **Set** H the number of weights for the neural network.
   (d) **Get** the best interval S from the previous step of Subsection 2.2.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2023 doi:10.20944/preprints202310.0274.v1

(e) **Initialize** using uniform distribution the N_C chromosomes in S.
(f) **Set** as p_s the selection rate, with $p_s \leq 1$.
(g) **Set** as p_m the mutation rate, with $p_m \leq 1$.
(h) **Set** iter=0.

2. **Fitness calculation Step**
   (a) **For** $i = 1, \ldots, N_g$ **do**
      i. **Calculate** the fitness f_i of chromosome g_i as $f_i = \sum_{j=1}^{M} \left(N(\overrightarrow{x}_j, \overrightarrow{g_i}) - y_j \right)^2$
   (b) **EndFor**
3. **Genetic operations step**
   (a) **Selection procedure.** Initially, the chromosomes are sorted according to their fitness values. The first $(1 - p_s) \times N_c$ chromosomes with the lowest fitness values are copied to the next generation. The remaining chromosomes are substituted by chromosomes produced by the crossover procedure. During the selection process, for every new offspring, two chromosomes are selected as parents from the population using well - known procedure of tournament selection.
   (b) **Crossover procedure**: For each pair (z, w) of selected parents, two chromosomes $\tilde{z}$ and $\tilde{w}$ are constructed using the following equations:

$$\begin{aligned} \tilde{z}_i &= a_i z_i + (1 - a_i) w_i \\ \tilde{w}_i &= a_i w_i + (1 - a_i) z_i \end{aligned} \tag{10}$$

   Where a_i is random number with the property $a_i \in [-0.5, 1.5]$ [64].
   (c) **Mutation procedure**: For each element of every chromosome alter the corresponding element with probability p_m.
4. **Termination Check Step**
   (a) **Set** $iter = iter + 1$
   (b) **If** iter$\leq N_g$ goto step 2. else apply a local search procedure to the best chromosome of the population. In the current work the BFGS variant of Powell [65] was used.

3. Experiments

The effectiveness of the proposed method was evaluated on a series of classification and regression datasets, used in the relevant literature. These datasets can be downloaded from the following online databases:

1. UCI dataset repository, https://archive.ics.uci.edu/ml/index.php [66]
2. Keel repository, https://sci2s.ugr.es/keel/datasets.php [67].
3. The Statlib URL ftp://lib.stat.cmu.edu/datasets/index.html.

3.1. Experimental datasets

The classification datasets used in this paper are the following:

1. **Appendictis** a medical purpose dataset, suggested in [68].
2. **Australian** dataset [69], a dataset related to credit card transactions.
3. **Balance** dataset [70], related to psychological states.
4. **Cleveland** dataset, which is a medical dataset [71,72].
5. **Dermatology** dataset [73], a medical dataset related to erythemato-squamous diseases.
6. **Heart** dataset [74], a medical dataset related to heart diseases.
7. **Hayes roth** dataset [75].
8. **HouseVotes** dataset [76], related to votes in the U.S. House of Representatives Congressmen.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2023 doi:10.20944/preprints202310.0274.v1

9. **Ionosphere** dataset, used for classification of radar returns from the ionosphere [77,78].
10. **Liverdisorder** dataset [79], a medical dataset related to liver disorders.
11. **Mammographic** dataset [80], used to identify breast tumors.
12. **Parkinsons** dataset, a medical dataset related to Parkinson's disease (PD) [81].
13. **Pima** dataset [82], used to detect the presence of diabetes.
14. **Popfailures** dataset [83], a dataset related to climate measurements.
15. **Regions2** dataset, medical dataset related to hepatitis C [84].
16. **Saheart** dataset [85], a medical dataset related to heart diseases.
17. **Segment** dataset [86], an image processing dataset.
18. **Wdbc** dataset [87], a medical dataset related to breast tumors.
19. **Wine** dataset, used to detect the origin of wines [88,89].
20. **Eeg** datasets, a medical dataset related to EEG measurements [90]. There are 3 different cases from this dataset used here denoted as Z_F_S, ZO_NF_S, ZONF_S.
21. **Zoo** dataset [91], used to classify animals.

The description of the used regression datasets has as follows:

1. **Abalone** dataset [93], used to to predict the age of abalone from physical measurements.
2. **Airfoil** dataset, derived from NASA [94].
3. **Baseball** dataset, used to estimate the salary of baseball players.
4. **BK** dataset [95], used to predict the points scored in a basketball game.
5. **BL** dataset, an electrical engineering dataset.
6. **Concrete** dataset [96].
7. **Dee** dataset, used to predict the price of electricity.
8. **Diabetes** dataset, a medical dataset.
9. **Housing** dataset, provided in [97].
10. **FA** dataset, used to predict the fit body fat.
11. **MB** dataset, available from from Smoothing Methods in Statistics [98].
12. **MORTGAGE** dataset, related to Economic data from USA.
13. **PY** dataset, (Pyrimidines problem) [99].
14. **Quake** dataset, used to predict the strength of a earthquakes.
15. **Treasure** dataset, related to Economic data from USA.
16. **Wankara** dataset, a dataset related to weather.

3.2. Experimental results

The proposed method was coded in ANSI C++ and the optimization methods were obtained from the freely available from the OPTIMUS computing environment, downloaded from https://github.com/itsoulos/OPTIMUS/(accessed on 9 September 2023). The results were validated using the 10 - fold validation technique in all datasets. The experiments were executed 30 times for each dataset using a different seed for the random generator each time. The average classification error is reported for the case of classification datasets and the average mean test error for the regression datasets. The machine used in the experiments was an AMD Ryzen 5950X with 128GB of RAM, running the Debian Linux operating system. The experimental settings are listed in the Table 2. The experimental results for the classification datasets are shown in Table 3, while for the regression datasets the results are shown in Table 4. The following applies to the results tables:

1. A genetic algorithm where the parameters have the values of Table 2 used to train a neural network with H hidden nodes. The results in the experimental tables are denoted by the label GENETIC.
2. The Adam optimization method is used to train a neural network with H hidden nodes. The column ADAM denotes the results for this method.
3. The RPROP method is used to train a neural network with H hidden nodes. The corresponding results are denoted by RPROP in the relevant tables.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2023 doi:10.20944/preprints202310.0274.v1

4. The NEAT method (NeuroEvolution of Augmenting Topologies) [100], where the maximum number of allowed generations is the same as in the case of the Genetic algorithm.
5. The proposed method (denoted as PROPOSED) was used with the experimental settings and are shown in Table 2.
6. An extra line was also added to the experimental tables under the title AVERAGE. This line represents the average classification or regression error for all datasets.

Table 2. Experimental parameters.

PARAMETER	VALUE
H	10
N_C	200
N_S	50
N_t	200
p_s	0.10
p_m	0.01

Table 3. Experiments for classification datasets

DATASET	GENETIC	ADAM	RPROP	NEAT	PROPOSED
Appendicitis	18.10%	16.50%	16.30%	17.20%	17.00%
Australian	32.21%	35.65%	36.12%	31.98%	24.55%
Balance	8.97%	7.87%	8.81%	23.14%	16.71%
Cleveland	51.60%	67.55%	61.41%	53.44%	47.91%
Dermatology	30.58%	26.14%	15.12%	32.43%	8.93%
Hayes Roth	56.18%	59.70%	37.46%	50.15%	32.21%
Heart	28.34%	38.53%	30.51%	39.27%	17.40%
HouseVotes	6.62%	7.48%	6.04%	10.89%	3.48%
Ionosphere	15.14%	16.64%	13.65%	19.67%	7.14%
Liverdisorder	31.11%	41.53%	40.26%	30.67%	28.90%
Lymography	23.26%	29.26%	24.67%	33.70%	17.86%
Mammographic	19.88%	46.25%	18.46%	22.85%	17.32%
Parkinsons	18.05%	24.06%	22.28%	18.56%	14.35%
Pima	32.19%	34.85%	34.27%	34.51%	25.58%
Popfailures	5.94%	5.18%	4.81%	7.05%	4.58%
Regions2	29.39%	29.85%	27.53%	33.23%	28.32%
Saheart	34.86%	34.04%	34.90%	34.51%	27.43%
Segment	57.72%	49.75%	52.14%	66.72%	20.68%
Wdbc	8.56%	35.35%	21.57%	12.88%	5.23%
Wine	19.20%	29.40%	30.73%	25.43%	5.35%
Z_F_S	10.73%	47.81%	29.28%	38.41%	6.56%
ZO_NF_S	8.41%	47.43%	6.43%	43.75%	3.60%
ZONF_S	2.60%	11.99%	27.27%	5.44%	2.21%
ZOO	16.67%	14.13%	15.47%	20.27%	6.10%
AVERAGE	**23.60%**	**31.54%**	**25.65%**	**29.42%**	**16.23%**

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2023 doi:10.20944/preprints202310.0274.v1

Table 4. Experiments for regression datasets.

DATASET	GENETIC	ADAM	RPROP	NEAT	PROPOSED
ABALONE	7.17	4.30	4.55	9.88	4.48
AIRFOIL	0.003	0.005	0.002	0.067	0.002
BASEBALL	103.60	77.90	92.05	100.39	51.39
BK	0.027	0.03	1.599	0.15	0.02
BL	5.74	0.28	4.38	0.05	0.002
CONCRETE	0.0099	0.078	0.0086	0.081	0.004
DEE	1.013	0.63	0.608	1.512	0.23
DIABETES	19.86	3.03	1.11	4.25	0.41
HOUSING	43.26	80.20	74.38	56.49	24.55
FA	1.95	0.11	0.14	0.19	0.01
MB	3.39	0.06	0.055	0.061	0.048
MORTGAGE	2.41	9.24	9.19	14.11	0.65
PY	5.41	0.09	0.039	0.075	0.025
QUAKE	0.040	0.06	0.041	0.298	0.038
TREASURY	2.929	11.16	10.88	15.52	0.84
WANKARA	0.012	0.02	0.0003	0.005	0.0002
AVERAGE	**12.30**	**11.70**	**12.44**	**12.70**	**5.17**

In general, the proposed technique outperforms the rest of the techniques in the experiments. In fact, in many cases, the test error is reduced by more than 70%. The numerical superiority of the proposed technique over the others in the performed experiments is presented graphically and in the Figures 3 and 4, where Wilcoxon signed-rank tests were performed on the executed experiments. However, since the proposed technique consists exclusively of a series of genetic algorithms, it can be significantly accelerated by the use of parallel computing techniques, since one of the main characteristics of genetic algorithms is their ease of parallelization [101,102]. Programming techniques that can be used to parallelize the method could be the MPI programming interface [103] or the OpenMP programming library [104].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2023 doi:10.20944/preprints202310.0274.v1

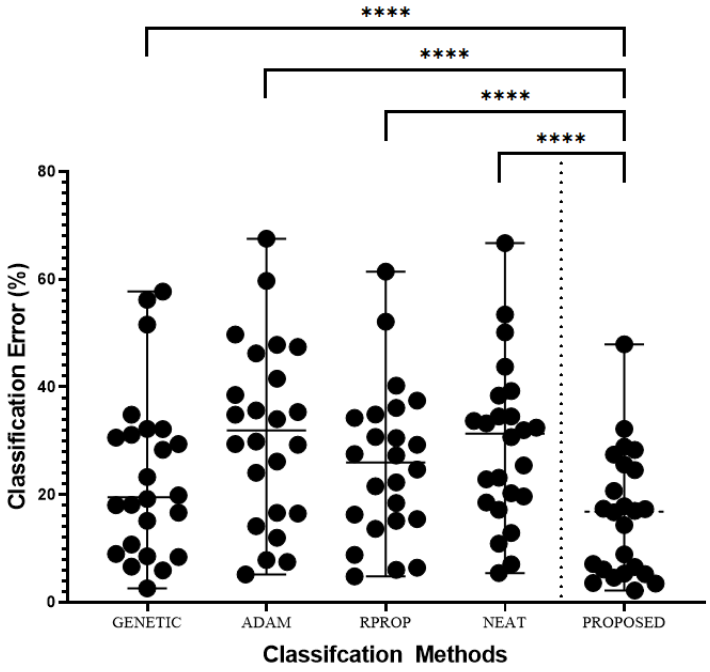

Figure 3. Scatter plot representation and the two-sample paired (Wilcoxon) signed-rank test results of the comparison for each of the four (4) classification methods (GENETIC, ADAM, RPROP, and NEAT) with the PROPOSED method regarding the classification error in twenty-four (24) different public available classification datasets. The stars only intend to flag significance levels for the two most used groups. A p-value of less than 0.0001 is flagged with four stars (****).

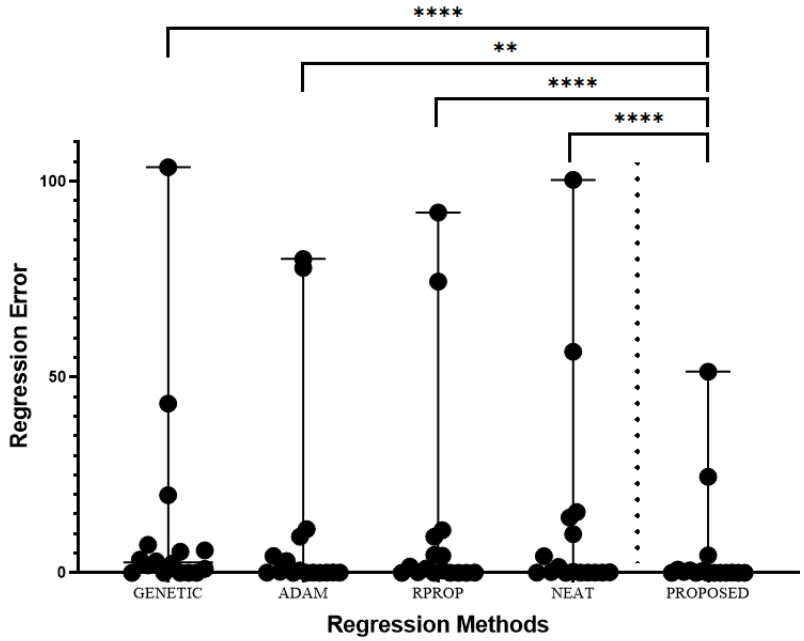

Figure 4. Scatter plot representation and the Wilcoxon signed-rank test results of the comparison for each of the four (4) regression methods (GENETIC, ADAM, RPROP, and NEAT) with the PROPOSED method regarding the regression error in sixteen (16) different publicly available regression datasets. Star links join significantly different values; one star (*) stands for $p<0.001$, and four stars (***) stand for $p<0.0001$.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2023 doi:10.20944/preprints202310.0274.v1

4. Conclusions

This paper proposed a two-step technique to efficiently find a reliable interval of values for the parameters of artificial neural networks. In the first stage with partitioning techniques and using Grammatical Evolution, a range of parameter values was searched and, in the second stage, a genetic algorithm trained the artificial neural network within the optimal range of values of the first stage. The proposed technique is highly general and has been successfully applied to both data fitting and classification problems. In both cases, the experimental results demonstrate its superiority over other techniques appearing in the relevant literature on the training of artificial neural networks. In fact, in many cases in the performed experiments, the proposed method outperforms others that were tested in percentages that exceed 70%. In the future, a series of actions can be carried out to improve the results and speed up the proposed method, such as:

1. There is a need for more efficient techniques for initializing the value space for artificial neural network parameters. In the present work, the optimal result from the execution of a limited number of steps by a genetic algorithm was used as an initial estimate of the value interval.
2. In the present work, the same techniques as in any whole-chromosome genetic algorithm were used to perform the crossover and mutation operations. Research could be done at this point to find more focused crossover and mutation techniques for this particular problem.
3. The present technique consists of two phases, in each of which a problem-adapted genetic algorithm is executed. This means that significant computational time is required to complete the algorithm. However, since genetic algorithms are inherently parallelizable, modern parallel programming techniques could be used here.

Author Contributions: I.G.T., A.T. and E.K. conceived the idea and methodology and supervised the technical part regarding the software. I.G.T. conducted the experiments, employing several datasets, and provided the comparative experiments. A.T. performed the statistical analysis. E.K. and all other authors prepared the manuscript. E.K. and I.G.T. organized the research team and A.T. supervised the project. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Informed Consent 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 conflict of interest.

Sample Availability: Not applicable.

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. Y. Shirvany, M. Hayati, R. Moradian, Multilayer perceptron neural networks with novel unsupervised training method for numerical solution of the partial differential equations, Applied Soft Computing **9**, pp. 20-29, 2009.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2023 doi:10.20944/preprints202310.0274.v1

7. A. Malek, R. Shekari Beidokhti, Numerical solution for high order differential equations using a hybrid neural network—Optimization method, Applied Mathematics and Computation **183**, pp. 260-271, 2006.
8. A. Topuz, Predicting moisture content of agricultural products using artificial neural networks, Advances in Engineering Software **41**, pp. 464-470, 2010.
9. 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.
10. 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.
11. 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.
12. 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.
13. Lukas Falat and Lucia Pancikova, Quantitative Modelling in Economics with Advanced Artificial Neural Networks, Procedia Economics and Finance **34**, pp. 194-201, 2015.
14. 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.
15. G. Tkacz, Neural network forecasting of Canadian GDP growth, International Journal of Forecasting **17**, pp. 57-69, 2001.
16. 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.
17. Ronadl Bartzatt, Prediction of Novel Anti-Ebola Virus Compounds Utilizing Artificial Neural Network (ANN), Chemistry Faculty Publications **49**, pp. 16-34, 2018.
18. I.G. Tsoulos, D. Gavrilis, E. Glavas, Neural network construction and training using grammatical evolution, Neurocomputing **72**, pp. 269-277, 2008.
19. D.E. Rumelhart, G.E. Hinton and R.J. Williams, Learning representations by back-propagating errors, Nature **323**, pp. 533 - 536 , 1986.
20. T. Chen and S. Zhong, Privacy-Preserving Backpropagation Neural Network Learning, IEEE Transactions on Neural Networks **20**, , pp. 1554-1564, 2009.
21. 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.
22. 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.
23. 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.
24. 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.
25. Y. Xue, Y. Tong, F. Neri, An ensemble of differential evolution and Adam for training feed-forward neural networks. Information Sciences **608**, pp. 453-471, 2022.
26. 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.
27. 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.
28. R.S. Sexton, B. Alidaee, R.E. Dorsey, J.D. Johnson, Global optimization for artificial neural networks: A tabu search application. European Journal of Operational Research **106**, pp. 570-584, 1998.
29. A. Yamazaki, M. C. P. de Souto,T. B. Ludermir, Optimization of neural network weights and architectures for odor recognition using simulated annealing, In: Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 **1**, pp. 547-552 , 2002.
30. Y. Da, G. Xiurun, An improved PSO-based ANN with simulated annealing technique, Neurocomputing **63**, pp. 527-533, 2005.
31. 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
32. X. Yao, Evolving artificial neural networks, Proceedings of the IEEE, 87(9), pp. 1423-1447, 1999.
33. 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.
34. Jianbo Yu, Shijin Wang, Lifeng Xi, Evolving artificial neural networks using an improved PSO and DPSO **71**, pp. 1054-1060, 2008.
35. J. lonen, J.K. Kamarainen, J. Lampinen, Differential Evolution Training Algorithm for Feed-Forward Neural Networks, Neural Processing Letters **17**, pp. 93–105, 2003.
36. M. Baioletti, G. Di Bari, A. Milani, V. Poggioni, Differential Evolution for Neural Networks Optimization, Mathematics **8**, 2020.
37. K.M. Salama, A.M. Abdelbar, Learning neural network structures with ant colony algorithms, Swarm Intell **9**, pp. 229–265, 2015.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2023 doi:10.20944/preprints202310.0274.v1

38. J.R. Zhang, J. Zhang, T.M. Lok, M.R. Lyu, A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training, Applied Mathematics and Computation **185**, pp. 1026-1037, 2007.
39. S. Mishra, S. K. Patra, Short Term Load Forecasting Using Neural Network Trained with Genetic Algorithm & Particle Swarm Optimization, In: 2008 First International Conference on Emerging Trends in Engineering and Technology, Nagpur, India, 2008, pp. 606-611, doi: 10.1109/ICETET.2008.94.
40. S. Mirjalili, S.Z. Mohd Hashim, H. Moradian Sardroudi, Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm, Applied Mathematics and Computation **218**, pp. 11125-11137, 2012.
41. A. Kobrunov, I. Priezzhev, Hybrid combination genetic algorithm and controlled gradient method to train a neural network, Geophysics **81**, pp,35–43, 2016.
42. I. Ivanova, M. Kubat, Initialization of neural networks by means of decision trees, Knowledge-Based Systems **8**, pp. 333-344, 1995.
43. J.Y.F. Yam, T.W.S. Chow, A weight initialization method for improving training speed in feedforward neural network, Neurocomputing **30**, pp. 219-232, 2000.
44. K. Chumachenko, A. Iosifidis, M. Gabbouj, Feedforward neural networks initialization based on discriminant learning, Neural Networks **146**, pp. 220-229, 2022.
45. M.D. Shahjahan, M. Kazuyuki, Neural network training algorithm with possitive correlation, IEEE Trans. Inf & Syst. **88**, pp. 2399-2409, 2005.
46. N.K. Treadgold, T.D. Gedeon, Simulated annealing and weight decay in adaptive learning: the SARPROP algorithm, IEEE Trans. on Neural Networks **9**, pp. 662-668, 1998.
47. C.S. Leung, K.W. Wong, P.F. Sum, L.W. Chan, A pruning method for the recursive least squared algorithm, Neural networks **14**, pp. 147-174, 2001.
48. M. O'Neill, C. Ryan, Grammatical evolution, IEEE Trans. Evol. Comput. **5,**pp. 349–358, 2001.
49. 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.
50. 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.
51. 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.
52. C. Ryan, M. O'Neill, J.J. Collins, Grammatical evolution: Solving trigonometric identities, proceedings of Mendel. Vol. 98. 1998.
53. 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.
54. Lídio Mauro Limade Campo, R. Célio Limã Oliveira,Mauro Roisenberg, Optimization of neural networks through grammatical evolution and a genetic algorithm, Expert Systems with Applications **56**, pp. 368-384, 2016.
55. K. Soltanian, A. Ebnenasir, M. Afsharchi, Modular Grammatical Evolution for the Generation of Artificial Neural Networks, Evolutionary Computation **30**, pp 291–327, 2022.
56. I. Dempsey, M.O' Neill, A. Brabazon, Constant creation in grammatical evolution, International Journal of Innovative Computing and Applications **1** , pp 23–38, 2007.
57. 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.
58. 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.
59. 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.
60. 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.
61. 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.
62. 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.
63. F. Castejón, E.J. Carmona, Automatic design of analog electronic circuits using grammatical evolution, Applied Soft Computing **62**, pp. 1003-1018, 2018.
64. P. Kaelo, M.M. Ali, Integrated crossover rules in real coded genetic algorithms, European Journal of Operational Research **176**, pp. 60-76, 2007.
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 20 September 2023).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2023 doi:10.20944/preprints202310.0274.v1

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. I. Kononenko, E. Šimec, M. Robnik-Šikonja, Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF, Applied Intelligence **7**, pp. 39–55, 1997
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. 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.
77. J.G. Dy , C.E. Brodley, Feature Selection for Unsupervised Learning, The Journal of Machine Learning Research **5**, pp 845–889, 2004.
78. S. J. Perantonis, V. Virvilis, Input Feature Extraction for Multilayered Perceptrons Using Supervised Principal Component Analysis, Neural Processing Letters **10**, pp 243–252, 1999.
79. J. Garcke, M. Griebel, Classification with sparse grids using simplicial basis functions, Intell. Data Anal. **6**, pp. 483-502, 2002.
80. 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.
81. 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.
82. 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.
83. 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.
84. 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.
85. T. Hastie, R. Tibshirani, Non-parametric logistic and proportional odds regression, JRSS-C (Applied Statistics) **36**, pp. 260–276, 1987.
86. M. Dash, H. Liu, P. Scheuermann, K. L. Tan, Fast hierarchical clustering and its validation, Data & Knowledge Engineering **44**, pp 109–138, 2003.
87. 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.
88. 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.
89. P. Zhong, M. Fukushima, Regularized nonsmooth Newton method for multi-class support vector machines, Optimization Methods and Software **22**, pp. 225-236, 2007.
90. 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.
91. M. Koivisto, K. Sood, Exact Bayesian Structure Discovery in Bayesian Networks, The Journal of Machine Learning Research **5**, pp. 549–573, 2004.
92. 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.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2023 doi:10.20944/preprints202310.0274.v1

93. W. J Nash, T.L. Sellers, S.R. Talbot, A.J. Cawthor, W.B. Ford, 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 No. 48 (ISSN 1034-3288), 1994.
94. T.F. Brooks, D.S. Pope, A.M. Marcolini, Airfoil self-noise and prediction. Technical report, NASA RP-1218, July 1989.
95. J.S. Simonoff, Smooting Methods in Statistics, Springer - Verlag, 1996.
96. I.Cheng Yeh, Modeling of strength of high performance concrete using artificial neural networks, Cement and Concrete Research. **28**, pp. 1797-1808, 1998.
97. D. Harrison and D.L. Rubinfeld, Hedonic prices and the demand for clean ai, J. Environ. Economics & Management **5**, pp. 81-102, 1978.
98. J.S. Simonoff, Smooting Methods in Statistics, Springer - Verlag, 1996.
99. R.D. King, S. Muggleton, R. Lewis, M.J.E. Sternberg, Proc. Nat. Acad. Sci. USA **89**, pp. 11322–11326, 1992.
100. K. O. Stanley, R. Miikkulainen, Evolving Neural Networks through Augmenting Topologies, Evolutionary Computation **10**, pp. 99-127, 2002.
101. E. Cantu-Paz, D.E. Goldberg, Efficient parallel genetic algorithms: theory and practice, Computer methods in applied mechanics and engineering **186**, pp. 221-238, 2000.
102. T. Harada, E. Alba, Parallel genetic algorithms: a useful survey, ACM Computing Surveys (CSUR) **53**, pp. 1-39, 2022.
103. W. Gropp, E. Lusk, N. Doss, A. Skjellum, A high-performance, portable implementation of the MPI message passing interface standard, Parallel Computing **22**, pp. 789-828, 1996.
104. R. Chandra, L. Dagum, D. Kohr, D. Maydan,J. McDonald and R. Menon, Parallel Programming in OpenMP, Morgan Kaufmann Publishers Inc., 2001.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.