Article

Bound the parameters of neural networks using Particle Swarm Optimization

Ioannis G. Tsoulos[1,*], Alexandros Tzallas[1], Evangelos Karvounis[1], Dimitrios Tsalikakis[2]

[1] Department of Informatics and Telecommunications, University of Ioannina, Greece
[2] University of Western Macedonia, Department of Engineering Informatics and Telecommunications, Greece
* Correspondence: itsoulos@uoi.gr;

Abstract: Artificial Neural Networks are machine learning models widely used in many sciences as well as in practical applications. The basic element of these models is a vector of parameters, the values of which should be estimated using some computational method, and this process is called training. For effective training of the network, computational methods from the field of global minimization are often used. However, for global minimization techniques to be effective, the bounds of the objective function should also be clearly defined. In this paper, a two-stage global optimization technique is presented for efficient training of artificial neural networks. In the first stage, the bounds for the neural network parameters are estimated using Particle Swarm Optimization and, in the following phase, the parameters of the network are optimized within the bounds of the first phase using global optimization techniques. The suggested method was used on a series of well-known problems in the literature and the experimental results were more than encouraging.

Keywords: Global optimization; local optimization; stochastic methods; evolutionary techniques; termination rules.

Citation: Tsoulos, I.G.; Tzallas, A.l; Karvounis E; Tsalikakis D Bound the parameters of neural networks using Particle Swarm Optimization. *Journal Not Specified* **2022**, *1*, 0.
https://doi.org/

Received:
Accepted:
Published:

1. Introduction

Artificial neural networks (ANNs) are parametric machine learning models [1,2], which have been widely used during the last decades in a series of practical problems from scientific fields such as physics problems [3–5], chemistry problems [6–8], problems related to medicine [9,10], economic problems [11–13] etc. Also, recently ANNs have been applied to models solving Differential Equations [14,15], agricultural problems [16,17], facial expression recognition [18], wind speed prediction [19], the gas consumption problem [20], intrusion detection [21] etc. Usually, neural networks are defined as a function $N(\overrightarrow{x}, \overrightarrow{w})$, provided that the vector $\overrightarrow{x}$ is the input pattern to the network and the vector $\overrightarrow{w}$ stands for the weight vector. To estimate the weight vector, the so-called training error is minimized, which is defined as the sum:

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

In equation 1 the values t $\left(\overrightarrow{x_i}, y_i\right),\ i = 1, ..., M$ defined the training set for the neural network. The values y_i denote the expected output for the pattern $\overrightarrow{x_i}$.

To minimize the quantity in equation 1, several techniques have been proposed in the relevant literature such as: Back Propagation method [22,23], the RPROP method [24–26], Quasi Newton methods [28,29], Simulated Annealing [30,31], Genetic Algorithms [32,33], Particle Swarm Optimization [34,35],Differential Optimization methods [36], Evolutionary Computation [37], the Whale optimization algorithm [38], the Butterfly optimization algorithm [39], etc. In addition, many researchers have focused their attention on techniques for initializing the parameters of artificial neural networks, such as the usage of decision trees to initialize neural networks [40], a method based on Cachy's inequality [41], usage of genetic algorithms [42], initialization based on discriminant learning [43] etc. Also, many researchers were also concerned with the construction of artificial neural network architectures, such as the usage of Cross Validation to propose the architecture of neural networks [44], incorporation of the Grammatical Evolution technique [46] to construct the architecture of neural networks as well as to estimate the values of the weights [45], evolution of neural networks using a method based on cellular automata [47] etc. Also, since in recent years there has been a leap forward in the development of parallel architectures, a number of works have been presented that take advantage of such computational techniques [48,49].

However, in many cases, the training methods of artificial neural networks suffer from the problem of overfitting, i.e. although they succeed in significantly reducing the training error of equation 1, they do not perform similarly on unknown data that was not present during training. These unknown data sets are commonly called test sets. The overfitting problem is usually handled using a a variety of methods, such as weight sharing [50,51], pruning of parameters, i.e reducing the size of the network [52–54], the dropout technique [55,56], weight decaying [57,58], the Sarporp method [59], positive correlation methods [60] etc. The overfitting problem is thoroughly discussed in Geman et all [61] and in the article by Hawkins [62].

A key reason why the problem of overtraining in artificial neural networks is present, is that there is no well-defined interval of values in which the network parameters are initialized and trained by the optimization methods. This, in practice, means that the values of the parameters are changed indiscriminately in order to reduce the value of the equation 1. In this work it is proposed to use the Particle Swarm Optimization (PSO) technique [63] for the reliable calculation of the value interval of the parameters of an artificial neural network. The PSO method was chosen since it is a fairly fast global optimization method, easily adaptable to any optimization problem, and does not require many execution parameters to be defined by the user. The PSO method was applied with success to many difficult problems, such as problems that arise in physics [64,65], chemistry [66,67], medicine [68,69], economics [70] etc. In the proposed method, the PSO technique is used to minimize the equation 1 to which a penalty factor has been added, so as not to allow the parameters of artificial neural networks to vary uncontrollably. After the minimization of the modified function is done, the parameters of the neural network are initialized in an interval of values around the optimal value located by the PSO method, and then the original form of equation 1 is minimized without a penalty factor this time.

Other tasks in a similar direction include, for example, the work of Hwang and Ding [71] that suggest prediction intervals for the weights of neural networks, the work of Kasiviswanathan et al [72] that constructs prediction intervals for neural networks applied on rainfall runoff models, the work of Sodhi and Chandra [73] that propose an interval based method for weight initialization of neural networks etc. Also, a review for weight initialization strategies can be found in the paper of Narkhede et al [74].

The following sections are organized as follows: in section 2 the suggested technique is fully analyzed and discussed, in section 3 the experimental datasets as well as the experimental results are described and discussed and in section 4 the conclusions from the application of current work are discussed.

2. The proposed method

2.1. Preliminaries

Suppose a neural network with a hidden processing layer is available that uses the so - called sigmoid function as activation function. The sigmoid function is defined as:

$$\sigma(x) = \frac{1}{1+\exp(-x)} \tag{2}$$

The equation for every hidden node of the neural network has as:

$$o_i(x) = \sigma\left(p_i^T x + \theta_i\right), \tag{3}$$

The vector p_i represents the weight vector and the value θ_i denotes the bias the node i. Hence, the total equation for a neural network with H hidden has as follows:

$$N(x) = \sum_{i=1}^{H} v_i o_i(x), \tag{4}$$

The value v_i denotes the output weight for node i. Therefore writing the overall equation by using one vector to hold both the weights p_i, v_i and the biases θ_i of the networks and using the previous equations equations 3 and 4 has as follows:

$$N(\overrightarrow{x}, \overrightarrow{w}) = \sum_{i=1}^{H} w_{(d+2)i-(d+1)}\sigma\left(\sum_{j=1}^{d} x_j w_{(d+2)i-(d+1)+j} + w_{(d+2)i}\right) \tag{5}$$

The value d stands for the dimension of the input vector $\overrightarrow{x}$. Observing the equation 5 it is obvious that in many cases, the sigmoid function is driven to 1 or 0 and as a consequence the training error of the neural network can get trapped in local minima and finally, the neural network it will lose its generalization abilities. Therefore, a technique should be devised by which the values of the sigmoid will be restricted to some interval of values. In the present work, the limitation of the neural network parameters to a range of values is carried out using the Particle Swarm Optimization method.

2.2. The bounding algorithm

In the case of the sigmoid function of equation 2 , if the parameter x is large, then the function will very quickly tend to 1 and if it is very small, it will very quickly tend to 0. This means that the function will very quickly lose any generalizing abilities it has. Therefore, the parameter x should somehow be in some interval of values, so that there are no generalization problems. For this reason, the quantity $B(L)$ is estimated, where L is a limit for the absolute value of the parameter x of the sigmoid function. The steps for this calculation are shown in Algorithm 1. This function will eventually return the average of the overruns made for the x parameter of the sigmoid function. The higher this average is, the more likely it is that the artificial neural network will not be able to generalize satisfactorily.

Algorithm 1 A function in pseudocode to calculate the quantity $B(L)$ for a given parameter L. The parameter M represents the number of patterns for the neural network $N(x, w)$.

```
1.  Function $B(L)$
2.  Define $v = 0$
3.  For $i = 1..H$ Do
    (a)  For $j = 1..M$ Do
         i.  If $\left|\sum_{k=1}^{d}\left(w_{(d+2)i-(d+i)+k}x_{jk}\right) + w_{(d+2)i}\right| > L$ then $v = v + 1$
    (b)  EndFor
4.  EndFor
5.  Return $\frac{v}{H \star M}$
6.  End Function
```

2.3. The PSO algorithm

The Particle Swarm Optimization method is based on a swarm of vectors that are commonly called particles. These particles can also be considered potential values of the total minimum of the objective function. Each particle is associated with two vectors: the current position denoted as $\overrightarrow{p}$ and the corresponding speed $\overrightarrow{u}$ at which they are moving towards the global minimum. In addition, each particle maintains in the vector $p_{i,b}$ the best position in which it has been so far and the total population, maintains in the vector p_{best} the best position that any of the particles have been found in the past. The purpose of the method is to move the total population toward the global minimum through a series of iterations. In each iteration, the velocity of each particle is calculated based on its current position, its best position in the past and the best located position of the population.

In this work, the PSO technique is used to train artificial neural networks by minimizing the error function as defined in the equation 3 along with a penalty factor depending on the function $B(L)$ defined in subsection 2.2. Hence, the PSO technique will minimize the equation:

$$E_T(N(x,w),L) = \sum_{i=1}^{M} \left(N(\overrightarrow{x}_i, \overrightarrow{w}) - y_i\right)^2 \times (1 + \alpha B(L)) \tag{6}$$

where α is a penalty factor with $\alpha > 1$. Hence, the main steps of a PSO algorithm are shown in Algorithm 2.

Algorithm 2 The base PSO algorithm executed in one processing unit.

```
1. Initialization Step .
    (a) Set $k = 0$, as the iteration number.
    (b) Set $H$ the hidden nodes for the neural network.
    (c) Set $m$ as the total number of particles. Each particle corresponds to a randomly
        selected set of parameters for the neural network
    (d) Set $\text{k}_{\text{max}}$ as the maximum number of iterations allowed.
    (e) Initialize velocities $u_1, u_2, ..., u_m$ randomly.
    (f) For $i = 1..m$ do $p_{i,b} = p_i$. The vector $p_{i,b}$ corresponds to the best located
        position of particle $i$.
    (g) Set $p_{\text{best}} = \arg\min_{i \in 1..m} f(p_i)$
2. If $k \geq k_{\text{max}}$, then terminate.
3. For $i = 1..m$ Do
    (a) Compute the velocity $u_i$ using the vectors $u_i,\ p_{i.b}$ and $p_{\text{best}}$
    (b) Set the new position $p_i = p_i + u_i$
    (c) Calculate the $f(p_i)$ for particle $p_i$ using the equation 6 as $f(p_i) =$
        $E_T(N(x, p_i), L)$
    (d) If $f(p_i) \leq f(p_{i,b})$ then $p_{i,b} = x_i$
4. End For
5. Set $p_{\text{best}} = \arg\min_{i \in 1..m} f(p_i)$
6. Set $k = k + 1$.
7. Goto Step 2
```

The velocity of every particle p_i usually is calculated as:

$$u_i = \omega u_i + r_1 c_1 (p_i - x_i) + r_2 c_2 (p_{\text{best}} - x_i) \tag{7}$$

where

1. The variables $r_1,\ r_2$ are numbers defined randomly in $[0, 1]$.
2. The constants $c_1,\ c_2$ are defined in $[1, 2]$.
3. The variable ω is called inertia, proposed in [75].

For the proposed algorithm, the inertia calculation used in [76–78] was used and it is defined as:

$$\omega_k = \frac{k_{\max} - \mathrm{k}}{\mathrm{k}_{\max}} \left(\omega_{\max} - \omega_{\min} \right) + \omega_{\min} \tag{8}$$

where ω_{min} and $\omega_{\max}$ are the minimum and the maximum value for inertia respectively.

2.4. Application of local optimization

After the Particle Swarm Optimization is completed in the vector p_{best} is the optimal set of parameters for the artificial neural network. From this set, a local optimization method can be started in order to achieve an even lower value for the neural network error. In addition, the optimal set of weights can be used to calculate an interval for the parameters of the neural network. The error function of the equation 3 will be minimized inside this interval. The interval $[LW, RW]$ for the parameter vector w of the neural network is calculated through the next steps:

```
1. For $i = 1..n$ do
    (a) Set $LW_i = -F \times \left| p_{\text{best},i} \right|$
    (b) Set $RW_i = F \times \left| p_{\text{best},i} \right|$
2. EndFor
```

The value F will be called margin factor with $F > 1$. In the proposed algorithm, a BFGS version of Powell [79] was used as the local search procedure, which is a version of the BFGS method [80], that utilizes bounds for the objective function.

3. Experiments

The efficiency of the suggested method was measured using a set of well-known problems from the relevant literature. The experimental results from the application of the proposed method was compared with other artificial neural network training techniques. In addition, experiments were carried out to show the dependence of the method on its basic parameters. The classification datasets incorporated in the relevant experiments can be found at:

1. UCI dataset repository, https://archive.ics.uci.edu/ml/index.php
2. Keel repository, https://sci2s.ugr.es/keel/datasets.php[81].

The majority of regression datasets was found in the Statlib URL ftp://lib.stat.cmu.edu/datasets/index.html.

3.1. Experimental datasets

The dataset used as classification problems are the following:

1. **Appendictis** a medical dataset, found in [82].
2. **Australian** dataset [83]. It is a dataset related to bank applications.
3. **Balance** dataset [84], a cognitive dataset.
4. **Cleveland** dataset, a medical datasets found in a variety of papers[85,86].
5. **Bands** dataset, a dataset related to printing problems.
6. **Dermatology** dataset [87], a medical dataset.
7. **Hayes roth** dataset [89].
8. **Heart** dataset [88], a dataset about heart diseases.
9. **HouseVotes** dataset [90].
10. **Ionosphere** dataset, this dataset has been thoroughly studied in many papers [91,92].
11. **Liverdisorder** dataset [93], a medical dataset.
12. **Mammographic** dataset [94], a medical dataset.
13. **Page Blocks** dataset [95], related to documents.
14. **Parkinsons** dataset [96], a dataset related to Parkinson's decease.
15. **Pima** dataset [97], a medical dataset.

16. **Popfailures** dataset [98], a dataset related to climate.
17. **Regions2** dataset, a medical dataset used in liver biopsy images of patients with hepatitis C [99].
18. **Saheart** dataset [100], a medical dataset about heart disease.
19. **Segment** dataset [101].
20. **Wdbc** dataset [102], a dataset about breast tumors.
21. **Wine** dataset, a dataset about chemical analysis for wines wines [103,104].
22. **Eeg** datasets [17], it is medical datasets about EEG signals and three distinct cases used here named Z_F_S, ZO_NF_S and ZONF_S respectively.
23. **Zoo** dataset [105].

The regression datasets used in the conducted experiments have as follows:

1. **Abalone** dataset [107], for the predictioon of age of abalone.
2. **Airfoil** dataset, a dataset provided by NASA [108].
3. **Baseball** dataset, a dataset used to calculate the salary of baseball players.
4. **BK** dataset [109], used for prediction of points in a basketball game.
5. **BL** dataset, used in machine problems.
6. **Concrete** dataset [110], a civil engineering dataset.
7. **Dee** dataset, used to estimate the price of the electricity.
8. **Diabetes** dataset, a medical dataset.
9. **Housing** dataset [111].
10. **FA** dataset, used to fit body fat to other measurements.
11. **MB** dataset [112].
12. **MORTGAGE** dataset, holding economic data from USA.
13. **PY** dataset, (Pyrimidines problem)[113].
14. **Quake** dataset, used to approximate the strength of a earthquake.
15. **Treasure** dataset, which contains Economic data information of USA.
16. **Wankara** dataset, a weather dataset.

3.2. Experimental results

To make a reliable estimate of the efficiency of the method, the ten - fold validation method was used and 30 experiments were conducted. In every experiment, different random values were used. The neural network used in the experiments has one hidden layer with 10 neurons. The selected activation function is the sigmoid function. The average classification or regression error on the test set is reported in the experimental tables. The parameters used in the experiments are shown in Table 1. The proposed method is compared against some other methods from the relevant literature:

1. A simple genetic algorithm using m chromosomes, denotes by GENETIC in the experimental tables. Also, in order to achieve a better solution, the local optimization method BFGS is applied to the best chromosome of the population when the genetic algorithm terminates.
2. The Radial Basis Function (RBF) neural network [114], where the number of weights was set to 10.
3. The optimization method Adam [115] as provided by the OptimLib. This library can be downloaded freely from https://github.com/kthohr/optim (accessed on 4 April 2023).
4. The Rprop method [24]. This method was found in the freely available FCNN programming package [116].
5. The NEAT method (NeuroEvolution of Augmenting Topologies) [117]. The method is implemented in the EvolutionNet programming package downloaded from https://github.com/BiagioFesta/EvolutionNet(accessed on 4 April 2023).

The experimental parameters for the methods Adam, Rprop and NEAT are proposed in the corresponding software. The experimental results for the classification data are shown in the table 2 and the corresponding results for the regression datasets are shown in table

3. In both tables, the last row, denoted as AVERAGE, indicates the average classification or regression error for the associated datasets. All the experiments were conducted on AMD Epyc 7552 equipped with 32GB of RAM. The operating system was the Ubuntu 20.04 operating system. For the conducted experiments, the Optimus programming library available from https://github.com/itsoulos/OPTIMUS(accessed on 4 April 2023) was used.

Table 1. This table presents the values of the parameters used during the execution of the experiments.

PARAMETER	MEANING	VALUE
m	Number of particles or chromosomes	200
$k_{\max}$	Maximum number of iterations	200
$\omega_{\min}$	Minimum value for inertia	0.4
$\omega_{\max}$	Maximum value for inertia	0.9
H	Number of weights	10
α	Penalty factor	100.0
L	The limit for the function $B(L)$	10.0
F	The margin factor	5.0

Table 2. Average classification error for the classification datasets for all mentioned methods.

DATASET	GENETIC	RBF	ADAM	RPROP	NEAT	PROPOSED
Appendicitis	18.10%	12.23%	16.50%	16.30%	17.20%	16.97%
Australian	32.21%	34.89%	35.65%	36.12%	31.98%	26.96%
Balance	8.97%	33.42%	7.87%	8.81%	23.14%	7.52%
Bands	35.75%	37.22%	36.25%	36.32%	34.30%	35.77%
Cleveland	51.60%	67.10%	67.55%	61.41%	53.44%	48.40%
Dermatology	30.58%	62.34%	26.14%	15.12%	32.43%	14.30%
Hayes Roth	56.18%	64.36%	59.70%	37.46%	50.15%	36.33%
Heart	28.34%	31.20%	38.53%	30.51%	39.27%	18.99%
HouseVotes	6.62%	6.13%	7.48%	6.04%	10.89%	7.10%
Ionosphere	15.14%	16.22%	16.64%	13.65%	19.67%	13.15%
Liverdisorder	31.11%	30.84%	41.53%	40.26%	30.67%	32.07%
Lymography	23.26%	25.31%	29.26%	24.67%	33.70%	27.05%
Mammographic	19.88%	21.38%	46.25%	18.46%	22.85%	17.37%
PageBlocks	8.06%	10.09%	7.93%	7.82%	10.22%	6.47%
Parkinsons	18.05%	17.42%	24.06%	22.28%	18.56%	14.60%
Pima	32.19%	25.78%	34.85%	34.27%	34.51%	26.34%
Popfailures	5.94%	7.04%	5.18%	4.81%	7.05%	5.27%
Regions2	29.39%	38.29%	29.85%	27.53%	33.23%	26.29%
Saheart	34.86%	32.19%	34.04%	34.90%	34.51%	32.49%
Segment	57.72%	59.68%	49.75%	52.14%	66.72%	18.99%
Wdbc	8.56%	7.27%	35.35%	21.57%	12.88%	6.01%
Wine	19.20%	31.41%	29.40%	30.73%	25.43%	10.92%
Z_F_S	10.73%	13.16%	47.81%	29.28%	38.41%	8.55%
ZO_NF_S	8.41%	9.02%	47.43%	6.43%	43.75%	7.11%
ZONF_S	2.60%	4.03%	11.99%	27.27%	5.44%	2.61%
ZOO	16.67%	21.93%	14.13%	15.47%	20.27%	5.80%
AVERAGE	**23.47%**	**27.69%**	**30.81%**	**25.37%**	**28.87%**	**18.21%**

Table 3. Average regression error for all mentioned methods and regression datasets.

DATASET	GENETIC	RBF	ADAM	RPROP	NEAT	PROPOSED
ABALONE	7.17	7.37	4.30	4.55	9.88	4.34
AIRFOIL	0.003	0.27	0.005	0.002	0.067	0.002
BASEBALL	103.60	93.02	77.90	92.05	100.39	58.78
BK	0.027	0.02	0.03	1.599	0.15	0.03
BL	5.74	0.01	0.28	4.38	0.05	0.02
CONCRETE	0.0099	0.011	0.078	0.0086	0.081	0.003
DEE	1.013	0.17	0.63	0.608	1.512	0.23
DIABETES	19.86	0.49	3.03	1.11	4.25	0.65
HOUSING	43.26	57.68	80.20	74.38	56.49	21.85
FA	1.95	0.02	0.11	0.14	0.19	0.02
MB	3.39	2.16	0.06	0.055	0.061	0.051
MORTGAGE	2.41	1.45	9.24	9.19	14.11	0.31
PY	105.41	0.02	0.09	0.039	0.075	0.08
QUAKE	0.04	0.071	0.06	0.041	0.298	0.044
TREASURY	2.929	2.02	11.16	10.88	15.52	0.34
WANKARA	0.012	0.001	0.02	0.0003	0.005	0.0002
AVERAGE	**18.55**	**10.30**	**11.70**	**12.44**	**12.70**	**5.42**

Table 4. Comparison for precision and recall between the proposed method and the Genetic Algorithm for a series of classification datasets.

DATASET	PRECISION GENETIC	RECALL GENETIC	PRECISION PROPOSED	RECALL PROPOSED
PARKINSONS	0.77	0.68	0.82	0.77
WINE	0.75	0.79	0.90	0.89
HEART	0.73	0.72	0.81	0.80

The proposed two-phase technique is shown to outperform the others in both classification and regression problems in terms of accuracy on the test set. In many data sets, the difference in accuracy provided by the proposed technique can reach or even exceed 70% and this is more evident in regression dataset, where the average gain from the next best is 54%. In the case of the classification data, the most effective method before the proposed one appears to be the genetic algorithm and the difference in accuracy between them is of the order of 23%. On the other hand, for the regression datasets, the radial basis network is the next most effective after the proposed technique and the average difference in accuracy between them is of the order of 47%.Also, a comparison in terms of precision and recall between the proposed method and the genetic algorithm is shown in Table 4 for a series of select classification datasets. And in this table, the proposed technique shows higher accuracy than the genetic algorithm method.

Furthermore, scatter plots that indicate the performance of all mentioned methods are shown in Figures 1 and 2.

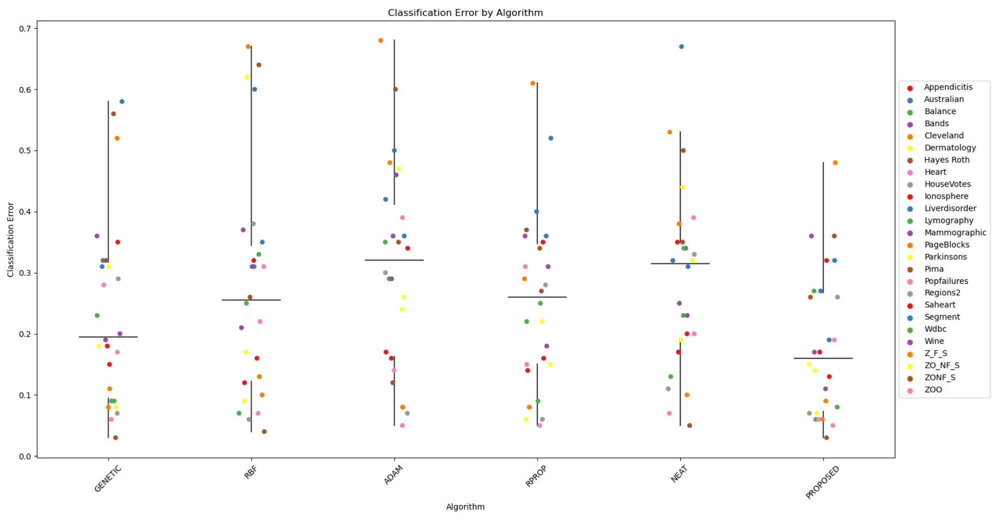

Figure 1. The scatter plot provides a clear overview of the performance of the proposed algorithm compared to the other six algorithms across the 26 datasets. It allows for visual identification of patterns, trends, and potential outliers in the classification errors. The plot serves as a concise visual summary of the comparative performance of the algorithms, providing insights into the effectiveness of the proposed algorithm in relation to the other algorithms in a variety of datasets.

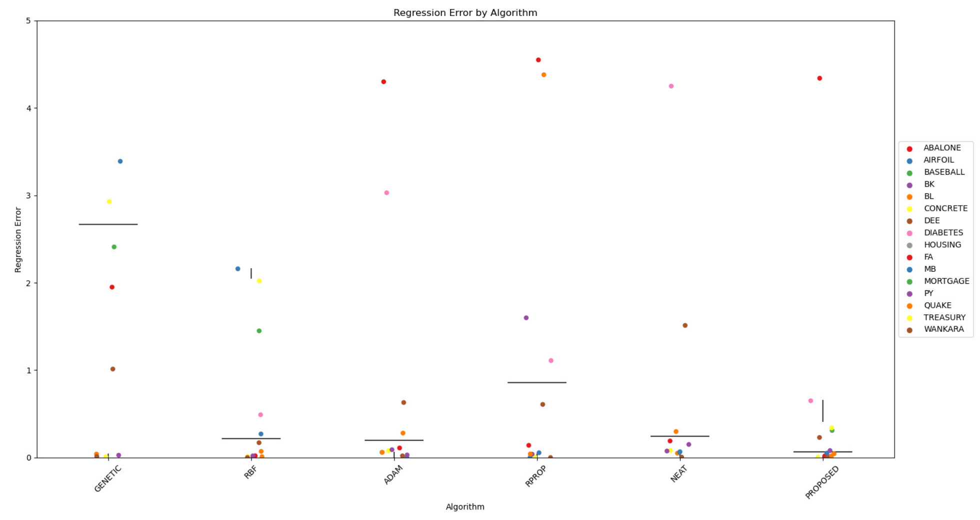

Figure 2. The scatter plot visually represents the performance of the regression classification algorithms in terms of classification errors. Each point on the plot represents the regression error of a particular algorithm in a specific dataset. The x-axis represents the regression algorithms, while the y-axis represents the regression error. Different datasets are denoted by different colors for clarity.

Also, the effectiveness of the usage of the BFGS local search method is measured in an additional experiment, where the local search method for the proposed technique is replaced by the ADAM optimizer. The results for the classification datasets are shown in Table 5 and the corresponding results for the regression datasets in Table 6. In both tables, the column PROPOSED_BFGS denotes the application of the proposed method using the BFGS local search as the procedure of the second phase and the column PROPOSED_ADAM denotes the incorporation of the ADAM optimizer during the second phase instead of the BFGS.

Table 5. Experimental results for the classification datasets using the BFGS and the Adam local optimization methods during the second phase.

DATASET	ADAM	PROPOSED_BFGS	PROPOSED_ADAM
Appendicitis	16.50%	16.97%	16.03%
Australian	35.65%	26.96%	32.45%
Balance	7.87%	7.52%	7.59%
Bands	36.25%	35.77%	35.72%
Cleveland	67.55%	48.40%	47.62%
Dermatology	26.14%	14.30%	19.78%
Hayes Roth	59.70%	36.33%	36.87%
Heart	38.53%	18.99%	18.86%
HouseVotes	7.48%	7.10%	4.20%
Ionosphere	16.64%	13.15%	9.32%
Liverdisorder	41.53%	32.07%	32.24%
Lymography	29.26%	27.05%	27.64%
Mammographic	46.25%	17.37%	21.38%
PageBlocks	7.93%	6.47%	7.33%
Parkinsons	24.06%	14.60%	16.77%
Pima	34.85%	26.34%	30.19%
Popfailures	5.18%	5.27%	4.56%
Regions2	29.85%	26.29%	26.14%
Saheart	34.04%	32.49%	32.55%
Segment	49.75%	18.99%	37.51%
Wdbc	35.35%	6.01%	7.40%
Wine	29.40%	10.92%	12.80%
Z_F_S	47.81%	8.55%	9.76%
ZO_NF_S	47.43%	7.11%	8.87%
ZONF_S	11.99%	2.61%	2.68%
ZOO	14.13%	5.80%	6.47%
AVERAGE	**30.81%**	**18.21%**	**19.72%**

Table 6. Experimental results for the regression datasets using the BFGS and the Adam local optimization methods during the second phase.

DATASET	ADAM	PROPOSED_BFGS	PROPOSED_ADAM
ABALONE	4.30	4.34	4.49
AIRFOIL	0.005	0.002	0.003
BASEBALL	77.90	58.78	72.43
BK	0.03	0.03	0.02
BL	0.28	0.02	0.01
CONCRETE	0.078	0.003	0.004
DEE	0.63	0.23	0.25
DIABETES	3.03	0.65	0.44
HOUSING	80.20	21.85	34.22
FA	0.11	0.02	0.02
MB	0.06	0.051	0.047
MORTGAGE	9.24	0.31	1.83
PY	0.09	0.08	0.02
QUAKE	0.06	0.044	0.039
TREASURY	11.16	0.34	2.36
WANKARA	0.02	0.0002	0.0002
AVERAGE	**11.70**	**5.42**	**7.26**

The experimental results demonstrate that the proposed methodology can give excellent results in learning categories or in learning functions even if the Adam method is used as a local minimization method. However, the Adam method requires a much longer execution time than the Bfgs method and this is shown graphically in Figure 3.

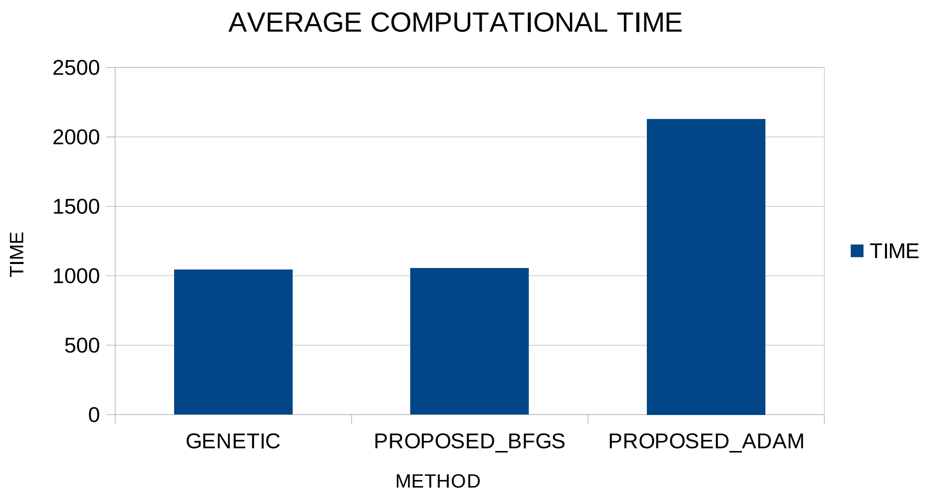

Figure 3. Average execution time for the Abalone dataset for three different methods: The simple Genetic method, the proposed method with the incorporation of BFGS and the proposed method with the usage of the Adam optimizer during the second phase of execution.

This graph shows the running time for the Abalone dataset, which is a time-consuming problem. The first column shows the execution time for the simple genetic algorithm, in the second column is the execution time for the proposed method using the Bfgs local optimization method in the second phase and in the third column is the execution time for the proposed method using the Adam method in the second stage. Although, as shown in the previous tables, the Adam method can have similar levels of success as the Bfgs method, it nevertheless requires significant computing time for its completion.

Also, the average regression error for the proposed method when the number of particles increases from 50 to 400 is graphically outlined in Figure 4.

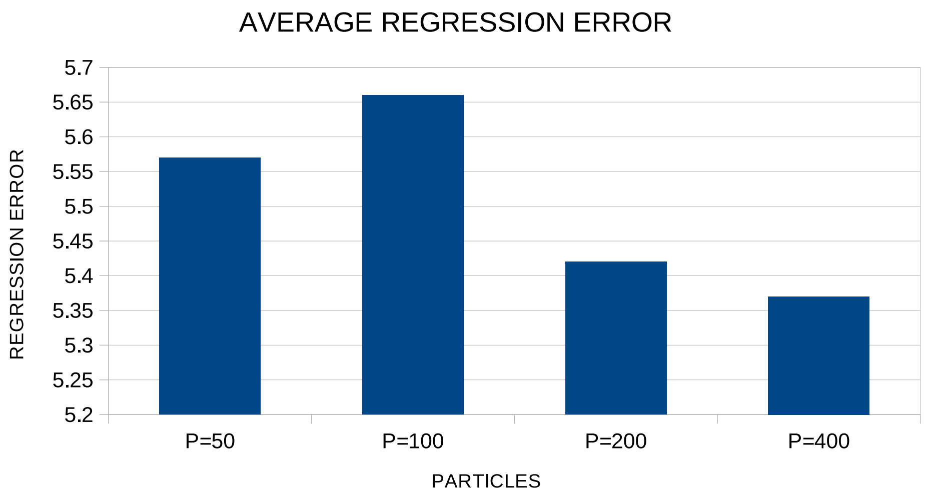

Figure 4. Average regression error for the proposed method using different numbers of particles in each experiment.

The proposed technique achieves the best results when the number of particles in the particle swarm optimization exceeds 100. However, the average error is about the same for 200 and 400 particles, and therefore, choosing 200 particles to run experiments appears to be more effective, since it will require less computing time.

In addition, in order to see if there is a dependence of the results on the critical parameters L and F of the proposed method, a series of additional experiments were carried out in which these parameters were varied. In the first phase, the proposed technique was applied to the regression datasets for different values of the coefficient L varying from 2.5 to 20.0 and the average regression error is graphically shown in Figure 5.

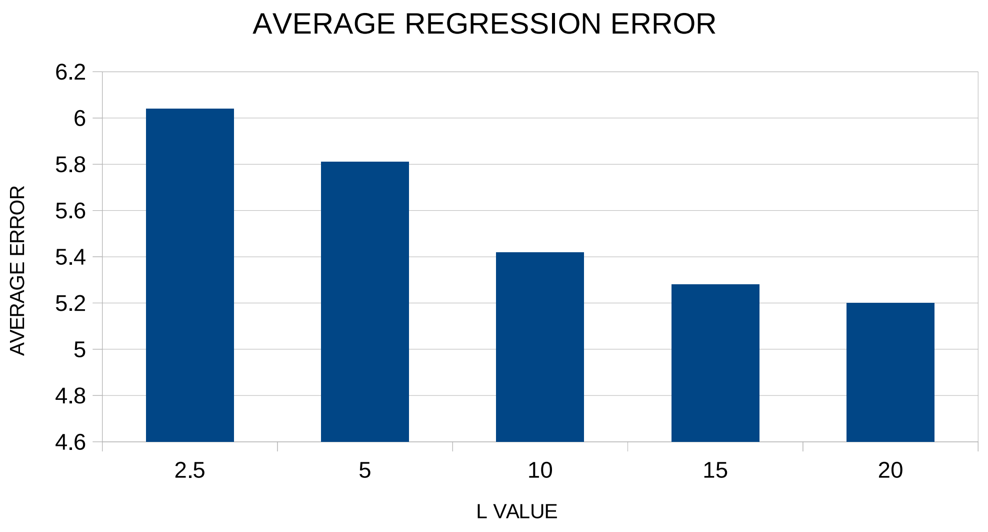

Figure 5. Experiments with the L parameter for the regression datasets.

If one judges from the experimental results, it is clear that the increase in the value of the coefficient λ positively affects the performance of the method, but after the value $\lambda = 10$ this effect decreases drastically. For low values of this coefficient, there is a limitation of the parameters of the artificial neural network to low values and therefore a lag in the accuracy of the neural network is expected.

Also, corresponding experiments were performed with the value of the coefficient F increasing from 2 to 15 and these are graphically illustrated in figure 6.

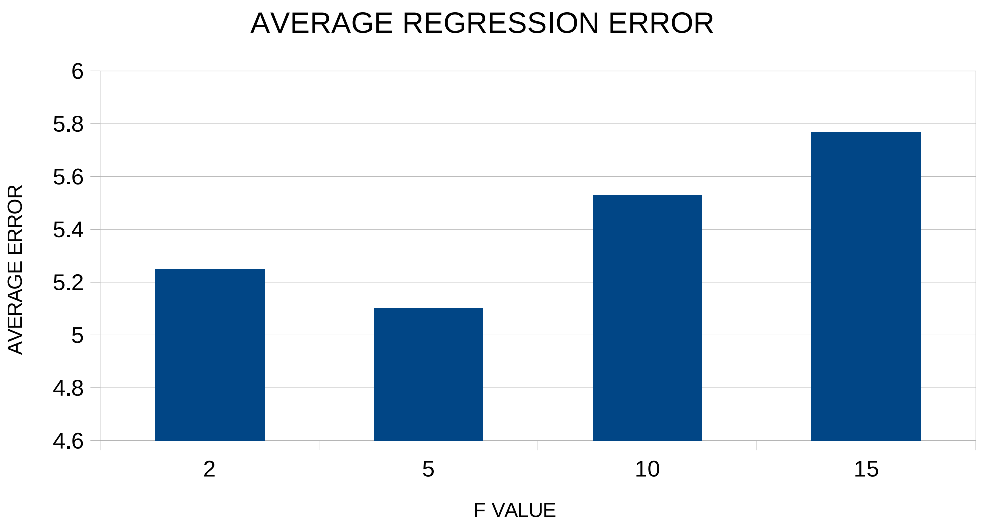

Figure 6. Experiments with the value of parameter F for the regression datasets.

For the experiments with the coefficient F one can see that the gain from any variation of this coefficient is limited, although the lowest values of the average error are achieved when the coefficient has a value close to 5.

4. Conclusions

In this paper, a two-stage technique for efficient training of artificial neural networks problems found in many scientific fields was presented. During the first phase, a widely used global optimization technique such as the Particle Swarm Optimization was used to minimize the training error of the artificial neural network to which a penalty factor had been added. This penalty factor is incorporated to maintain the effectiveness of the artificial neural network in generalizing to unknown data as well. The calculation of the penalty factor is based on the observation that the artificial neural network can lose its generalization abilities when the input values in the sigmoid activation function exceed some predetermined threshold. After the particle optimization technique is performed in the second phase, the best particle is used both as an initializer of a local optimization method and as a basis for calculating bounds on the parameters of the artificial neural network.

The suggested method was applied to a wide range of classification and regression problems found in the recent literature and the experimental results were more than encouraging. In addition, when comparing the proposed technique with other widely used methods from the relevant literature, it seems that the proposed technique significantly outperforms them, especially in the case of regression problems. In relevant experiments carried out regarding the sensitivity of the proposed technique on its critical parameters, it was found to be quite robust without large error fluctuations.

Future extensions of the technique may include its application to other network cases such as Radial Basis Function artificial neural networks (RBFs), as well as the use of global optimization methods in the second stage of the proposed technique or even the creation of appropriate termination techniques.

Author Contributions: A.T. and I.G.T. conducted the experiments, employing several datasets and provided the comparative experiments. D.T. and E.K. performed the statistical analysis and prepared the manuscript. 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.

Institutional Review Board Statement: Not applicable.

Acknowledgments: The experiments of this research work were performed at the high performance computing system established at Knowledge and Intelligent Computing Laboratory, Department of Informatics and Telecommunications, University of Ioannina, acquired with the project "Educational Laboratory equipment of TEI of Epirus" with MIS 5007094 funded by the Operational Programme "Epirus" 2014–2020, by ERDF and national funds.

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. Lin Shen, Jingheng Wu, and Weitao Yang, Multiscale Quantum Mechanics/Molecular Mechanics Simulations with Neural Networks, Journal of Chemical Theory and Computation **12**, pp. 4934-4946, 2016.
7. Sergei Manzhos, Richard Dawes, Tucker Carrington, Neural network-based approaches for building high dimensional and quantum dynamics-friendly potential energy surfaces, Int. J. Quantum Chem. **115**, pp. 1012-1020, 2015.
8. Jennifer N. Wei, David Duvenaud, and Alán Aspuru-Guzik, Neural Networks for the Prediction of Organic Chemistry Reactions, ACS Central Science **2**, pp. 725-732, 2016.
9. 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.
10. Ronadl Bartzatt, Prediction of Novel Anti-Ebola Virus Compounds Utilizing Artificial Neural Network (ANN), Chemistry Faculty Publications **49**, pp. 16-34, 2018.
11. Lukas Falat and Lucia Pancikova, Quantitative Modelling in Economics with Advanced Artificial Neural Networks, Procedia Economics and Finance **34**, pp. 194-201, 2015.
12. 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.
13. G. Tkacz, Neural network forecasting of Canadian GDP growth, International Journal of Forecasting **17**, pp. 57-69, 2001.
14. 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.
15. 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.
16. A. Topuz, Predicting moisture content of agricultural products using artificial neural networks, Advances in Engineering Software **41**, pp. 464-470, 2010.
17. 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.
18. H. Boughrara, M. Chtourou, C. Ben Amar et al, Facial expression recognition based on a mlp neural network using constructive training algorithm. Multimed Tools Appl **75**, pp. 709–731, 2016.
19. H. Liu, H.Q Tian, Y.F. Li, L. Zhang, Comparison of four Adaboost algorithm based artificial neural networks in wind speed predictions, Energy Conversion and Management **92**, pp. 67-81, 2015.
20. J. Szoplik, Forecasting of natural gas consumption with artificial neural networks, Energy **85**, pp. 208-220, 2015.
21. H. Bahram, N.J. Navimipour, Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm, ICT Express 5, pp. 56-59, 2019.
22. D.E. Rumelhart, G.E. Hinton and R.J. Williams, Learning representations by back-propagating errors, Nature **323**, pp. 533 - 536 , 1986.
23. T. Chen and S. Zhong, Privacy-Preserving Backpropagation Neural Network Learning, IEEE Transactions on Neural Networks **20**, , pp. 1554-1564, 2009.
24. 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.
25. 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.

26. 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.
27. Neural Networks, Procedia Computer Science **135**, pp. 35-42, 2018.
28. 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.
29. 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.
30. 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.
31. Y. Da, G. Xiurun, An improved PSO-based ANN with simulated annealing technique, Neurocomputing **63**, pp. 527-533, 2005.
32. 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
33. X. Yao, Evolving artificial neural networks, Proceedings of the IEEE, 87(9), pp. 1423-1447, 1999.
34. 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.
35. Jianbo Yu, Shijin Wang, Lifeng Xi, Evolving artificial neural networks using an improved PSO and DPSO **71**, pp. 1054-1060, 2008.
36. J. Ilonen, J.K. Kamarainen, J. Lampinen, Differential Evolution Training Algorithm for Feed-Forward Neural Networks, Neural Processing Letters **17**, pp. 93–105, 2003.
37. M. Rocha, P. Cortez, J. Neves, Evolution of neural networks for classification and regression, Neurocomputing **70**, pp. 2809-2816, 2007.
38. I. Aljarah, H. Faris, S. Mirjalili, Optimizing connection weights in neural networks using the whale optimization algorithm, Soft Comput **22**, pp. 1–15, 2018.
39. S.M.J. Jalali, S. Ahmadian, P.M. Kebria, A. Khosravi, C.P. Lim, S. Nahavandi, Evolving Artificial Neural Networks Using Butterfly Optimization Algorithm for Data Classification. In: Gedeon, T., Wong, K., Lee, M. (eds) Neural Information Processing. ICONIP 2019. Lecture Notes in Computer Science(), vol 11953. Springer, Cham, 2019.
40. I. Ivanova, M. Kubat, Initialization of neural networks by means of decision trees, Knowledge-Based Systems **8**, pp. 333-344, 1995.
41. 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.
42. F. Itano, M. A. de Abreu de Sousa, E. Del-Moral-Hernandez, Extending MLP ANN hyper-parameters Optimization by using Genetic Algorithm, In: 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 2018, pp. 1-8, 2018.
43. K. Chumachenko, A. Iosifidis, M. Gabbouj, Feedforward neural networks initialization based on discriminant learning, Neural Networks **146**, pp. 220-229, 2022.
44. R. Setiono, Feedforward Neural Network Construction Using Cross Validation,Neural Computation **13**, pp. 2865-2877, 2001.
45. I.G. Tsoulos, D. Gavrilis, E. Glavas, Neural network construction and training using grammatical evolution, Neurocomputing 72, pp. 269-277, 2008.
46. M. O'Neill, C. Ryan, Grammatical evolution, IEEE Trans. Evol. Comput. **5**, pp. 349–358, 2001.
47. K.J. Kim, S.B. Cho, Evolved neural networks based on cellular automata for sensory-motor controller, Neurocomputing **69**, pp. 2193-2207, 2006.
48. M. Martínez-Zarzuela, F.J. Díaz Pernas, J.F. Díez Higuera, M.A. Rodríguez, Fuzzy ART Neural Network Parallel Computing on the GPU. In: Sandoval, F., Prieto, A., Cabestany, J., Graña, M. (eds) Computational and Ambient Intelligence. IWANN 2007. Lecture Notes in Computer Science, vol 4507. Springer, Berlin, Heidelberg, 2007.
49. A.A. Huqqani, E. Schikuta, S. Ye Peng Chen, Multicore and GPU Parallelization of Neural Networks for Face Recognition, Procedia Computer Science **18**, pp. 349-358, 2013.
50. S.J. Nowlan and G.E. Hinton, Simplifying neural networks by soft weight sharing, Neural Computation 4, pp. 473-493, 1992.
51. J.K. Kim, M.Y. Lee, J.Y. Kim, B. J. Kim, J. H. Lee, An efficient pruning and weight sharing method for neural network, In: 2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), Seoul, Korea (South), pp. 1-2, 2016.
52. S.J. Hanson and L.Y. Pratt, Comparing biases for minimal network construction with back propagation, In D.S. Touretzky (Ed.), Advances in Neural Information Processing Systems, Volume 1, pp. 177-185, San Mateo, CA: Morgan Kaufmann, 1989.
53. M.C. Mozer and P. Smolensky, Skeletonization: a technique for trimming the fat from a network via relevance assesment. In D.S. Touretzky (Ed.), Advances in Neural Processing Systems, Volume 1, pp. 107-115, San Mateo CA: Morgan Kaufmann, 1989.
54. M. Augasta and T. Kathirvalavakumar, Pruning algorithms of neural networks — a comparative study, Central European Journal of Computer Science, 2003.
55. Nitish Srivastava, G E Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan R Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, Journal of Machine Learning Research **15**, pp. 1929-1958, 2014.
56. A. Iosifidis, A. Tefas, I. Pitas, DropELM: Fast neural network regularization with Dropout and DropConnect, Neurocomputing **162**, pp. 57-66, 2015.
57. A. Gupta, S.M. Lam, Weight decay backpropagation for noisy data, Neural Networks **11**, pp. 1127-1138, 1998.

58. M. Carvalho and T. B. Ludermir, Particle Swarm Optimization of Feed-Forward Neural Networks with Weight Decay, 2006 Sixth International Conference on Hybrid Intelligent Systems (HIS'06), Rio de Janeiro, Brazil, 2006, pp. 5-5.
59. 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.
60. M.D. Shahjahan, M. Kazuyuki, Neural network training algorithm with possitive correlation, IEEE Trans. Inf & Syst. **88**, pp. 2399-2409, 2005.
61. S. Geman, E. Bienenstock and R. Doursat, Neural networks and the bias/variance dilemma, Neural Computation 4 , pp. 1 - 58, 1992.
62. Douglas M. Hawkins, The Problem of Overfitting, J. Chem. Inf. Comput. Sci. **44**, pp. 1–12, 2004.
63. F. Marini, B. Walczak, Particle swarm optimization (PSO). A tutorial, Chemometrics and Intelligent Laboratory Systems **149**, pp. 153-165, 2015.
64. Anderson Alvarenga de Moura Meneses, Marcelo Dornellas, Machado Roberto Schirru, Particle Swarm Optimization applied to the nuclear reload problem of a Pressurized Water Reactor, Progress in Nuclear Energy **51**, pp. 319-326, 2009.
65. Ranjit Shaw, Shalivahan Srivastava, Particle swarm optimization: A new tool to invert geophysical data, Geophysics **72**, 2007.
66. C. O. Ourique, E.C. Biscaia, J.C. Pinto, The use of particle swarm optimization for dynamical analysis in chemical processes, Computers & Chemical Engineering **26**, pp. 1783-1793, 2002.
67. H. Fang, J. Zhou, Z. Wang et al, Hybrid method integrating machine learning and particle swarm optimization for smart chemical process operations, Front. Chem. Sci. Eng. **16**, pp. 274–287, 2022.
68. M.P. Wachowiak, R. Smolikova, Yufeng Zheng, J.M. Zurada, A.S. Elmaghraby, An approach to multimodal biomedical image registration utilizing particle swarm optimization, IEEE Transactions on Evolutionary Computation **8**, pp. 289-301, 2004.
69. Yannis Marinakis. Magdalene Marinaki, Georgios Dounias, Particle swarm optimization for pap-smear diagnosis, Expert Systems with Applications **35**, pp. 1645-1656, 2008.
70. Jong-Bae Park, Yun-Won Jeong, Joong-Rin Shin, Kwang Y. Lee, An Improved Particle Swarm Optimization for Nonconvex Economic Dispatch Problems, IEEE Transactions on Power Systems **25**, pp. 156-16**216**6, 2010.
71. J.T.G. Hwang, A.A. Ding, Prediction Intervals for Artificial Neural Networks, Journal of the American Statistical Association **92**, pp. 748-757, 1997.
72. K.S. Kasiviswanathan, R. Cibin, K.P. Sudheer, I. Chaubey, Constructing prediction interval for artificial neural network rainfall runoff models based on ensemble simulations, Journal of Hydrology **499**, pp. 275-288, 2013.
73. S.S. Sodhi, P. Chandra, Interval based Weight Initialization Method for Sigmoidal Feedforward Artificial Neural Networks, AASRI Procedia **6**, pp. 19-25, 2014.
74. M.V. Narkhede, P.P. Bartakke, M.S. Sutaone, A review on weight initialization strategies for neural networks. Artif Intell Rev **55**, pp. 291–322, 2022.
75. J. Kennedy and R. Eberhart, "Particle swarm optimization," Proceedings of ICNN'95 - International Conference on Neural Networks, 1995, pp. 1942-1948 vol.4, doi: 10.1109/ICNN.1995.488968.
76. R.C. Eberhart, Y.H. Shi, Tracking and optimizing dynamic systems with particle swarms, in: Congress on Evolutionary Computation, Korea, 2001.
77. Y.H. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, in: Congress on Evolutionary Computation, Washington DC, USA, 1999.
78. Y.H. Shi, R.C. Eberhart, Experimental study of particle swarm optimization, in: SCI2000 Conference, Orlando, 2000.
79. M.J.D Powell, A Tolerant Algorithm for Linearly Constrained Optimization Calculations, Mathematical Programming **45**, pp. 547-566, 1989.
80. R. Fletcher, A new approach to variable metric algorithms, Computer Journal **13**, pp. 317-322, 1970.
81. 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.
82. 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.
83. J.R. Quinlan, Simplifying Decision Trees. International Journal of Man-Machine Studies **27**, pp. 221-234, 1987.
84. T. Shultz, D. Mareschal, W. Schmidt, Modeling Cognitive Development on Balance Scale Phenomena, Machine Learning **16**, pp. 59-88, 1994.
85. 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.
86. R. Setiono , W.K. Leow, FERNN: An Algorithm for Fast Extraction of Rules from Neural Networks, Applied Intelligence **12**, pp. 15-25, 2000.
87. 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.
88. I. Kononenko, E. Šimec, M. Robnik-Šikonja, Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF, Applied Intelligence **7**, pp. 39–55, 1997

89. 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.
90. 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.
91. J.G. Dy , C.E. Brodley, Feature Selection for Unsupervised Learning, The Journal of Machine Learning Research **5**, pp 845–889, 2004.
92. S. J. Perantonis, V. Virvilis, Input Feature Extraction for Multilayered Perceptrons Using Supervised Principal Component Analysis, Neural Processing Letters **10**, pp 243–252, 1999.
93. J. Garcke, M. Griebel, Classification with sparse grids using simplicial basis functions, Intell. Data Anal. **6**, pp. 483-502, 2002.
94. 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.
95. F. Esposito F., D. Malerba, G. Semeraro, Multistrategy Learning for Document Recognition, Applied Artificial Intelligence **8**, pp. 33-84, 1994.
96. 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.
97. 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.
98. 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.
99. 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.
100. T. Hastie, R. Tibshirani, Non-parametric logistic and proportional odds regression, JRSS-C (Applied Statistics) **36**, pp. 260–276, 1987.
101. M. Dash, H. Liu, P. Scheuermann, K. L. Tan, Fast hierarchical clustering and its validation, Data & Knowledge Engineering **44**, pp 109–138, 2003.
102. 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.
103. 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.
104. P. Zhong, M. Fukushima, Regularized nonsmooth Newton method for multi-class support vector machines, Optimization Methods and Software **22**, pp. 225-236, 2007.
105. M. Koivisto, K. Sood, Exact Bayesian Structure Discovery in Bayesian Networks, The Journal of Machine Learning Research **5**, pp. 549–573, 2004.
106. 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.
107. 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.
108. T.F. Brooks, D.S. Pope, A.M. Marcolini, Airfoil self-noise and prediction. Technical report, NASA RP-1218, July 1989.
109. J.S. Simonoff, Smooting Methods in Statistics, Springer - Verlag, 1996.
110. I.Cheng Yeh, Modeling of strength of high performance concrete using artificial neural networks, Cement and Concrete Research. **28**, pp. 1797-1808, 1998.
111. D. Harrison and D.L. Rubinfeld, Hedonic prices and the demand for clean ai, J. Environ. Economics & Management **5**, pp. 81-102, 1978.
112. J.S. Simonoff, Smooting Methods in Statistics, Springer - Verlag, 1996.
113. R.D. King, S. Muggleton, R. Lewis, M.J.E. Sternberg, Proc. Nat. Acad. Sci. USA **89**, pp. 11322–11326, 1992.
114. J. Park and I. W. Sandberg, Universal Approximation Using Radial-Basis-Function Networks, Neural Computation **3**, pp. 246-257, 1991.
115. 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.
116. Grzegorz Klima, Fast Compressed Neural Networks, available from http://fcnn.sourceforge.net/.
117. K. O. Stanley, R. Miikkulainen, Evolving Neural Networks through Augmenting Topologies, Evolutionary Computation **10**, pp. 99-127, 2002.