MDPI

Article

RbfCon: Construct RBF neural networks with Grammatical Evolution

Ioannis G. Tsoulos[1,*], Ioannis Varvaras[2] and Vasileios Charilogis[3]

[1] Department of Informatics and Telecommunications, University of Ioannina, Greece; itsoulos@uoi.gr
[2] Department of Informatics and Telecommunications, University of Ioannina, Greece; i_varvaras@hotmail.com
[3] Department of Informatics and Telecommunications, University of Ioannina, Greece; v.charilog@uoi.gr
* Correspondence: itsoulos@uoi.gr

Abstract: Radial Basis Function networks considered as a machine learning tool, used in classification and regression problems derived from various areas of the modern world. However, in many cases, the initial training method used to fit the parameters of these models can produce poor results either due to unstable numerical operations or its inability to effectively locate the global minimum of the error function. The current work proposes a novel method that constructs the architecture of this model and estimates the values for each parameter of the model with the incorporation of Grammatical Evolution. The proposed method was coded in ANSI C++ and the suggested software was tested on a wide series of problems from various areas. The experimental results certify the adequacy of the new method to solve difficult problems and in the vast majority of cases the error in classification or approximation of functions is significantly lower than the case where the original training method was applied.

Keywords: Neural networks; Genetic programming; Grammatical Evolution; Evolutionary algorithms

Citation: Tsoulos, I.G.; Varvaras, I.; Charilogis, V. . RbfCon: Construct RBF neural networks with Grammatical Evolution. *Journal Not Specified* **2024**, *1*, 0. https://doi.org/

Received:
Revised:
Accepted:
Published:

1. Introduction

A variety of real - world problems can be considered as classification and regression problems, handled by machine learning models that have been thoroughly studied in the recent bibliography. Such problems arise in physics [1,2], chemistry [3,4], economics [5,6], medicine [7,8] etc. One common machine learning model, that is applied in many areas, is the Radial Basis Function (RBF) neural network [9,10]. Typically, an RBF network is defined using the following equation:

$$y(\overrightarrow{x}) = \sum_{i=1}^{k} w_i \phi(\| \overrightarrow{x} - \overrightarrow{c_i} \|) \tag{1}$$

The following notation is used in Equation 1:

1. The vector $\overrightarrow{x}$ represents the input pattern with dimension d.
2. The parameter k stands for the number of weights of the model. These weights are represented by vector $\overrightarrow{w}$
3. The vectors $\overrightarrow{c_i}$, $i = 1, .., k$ represent the centers of this model.
4. The final output of the model for the input pattern $\overrightarrow{x}$ is denoted by $y(\overrightarrow{x})$
5. The function $\phi(x)$ in most cases is represented by the Gaussian function defined as:

$$\phi(x) = \exp\left(-\frac{(x-c)^2}{\sigma^2} \right) \tag{2}$$

This function is selected as the output function, since the value $\phi(x)$ depends only on the distance of vectors x and c.

RBF neural networks have been widely used and studied in the relevant literature and, among others, one can find applications in problems derived from physics [12,13], robotics [14,15], security problems [16,17], image processing [18,19] etc. Due to the widespread use of these machine learning models and their use in both function learning and data fitting problems, a number of techniques have been presented in recent years that seek to more accurately identify their parameters. Such methods include techniques used to initialize the parameters of these models [20–22], or pruning methods for the optimal adaptation of the architecture of these models [23–25]. Also, global optimization methods have been proposed to estimate the optimal set of parameters of RBF networks in a series of research papers [26–28].

In most cases, the parameters of an RBF neural network are calculated using a two phase method: during the first phase, the set of centers and variances of equation 1 are calculated using the K-Means algorithm [29]. In the second phase, the set of weights $\overrightarrow{w}$ is obtained by solving a linear system of equations. Although the previous procedure is capable of estimating the optimal set of parameters in a very short time, it nevertheless has a number of problems such as numerical stability, accurate determination of the number of weights, over - fitting problems, etc. To tackle such problems, a novel method is proposed here that constructs the optimal architecture of RBF networks using a method that incorporates the Grammatical Evolution procedure [30]. The proposed technique is able to estimate both the optimal architecture of the neural network and the numerical values of its parameters, and in this paper both the algorithm used and the software created for this process, which is freely available from the internet, will be presented in detail.

The software proposed in this work is fully implemented in the programming language ANSI C++ and is an executable that has a series of command line parameters with which the user can efficiently process the data sets at his disposal.

The rest of this article is divided in the following sections: in section 2 the proposed methodology and the used software are thoroughly discussed, in section 3 the datasets used in the experiments are presented followed by the experimental results and finally in section 4 some conclusions are discussed.

2. Materials and Methods

This section starts with a description of the original training procedure for RBF neural networks and afterwards it provides a presentation of the Grammatical Evolution procedure and continues with a detail description of the proposed method. Finally, the proposed software is presented and a full working example of execution is shown.

2.1. The original training procedure of RBF neural networks

The typical training procedure of an RBF network $y(x) = \sum_{i=1}^{k} w_i \phi(\|x - c_i\|)$ involves two major steps: in first step the centers $\overrightarrow{c_i}$ and the variances are calculated using the K-means procedure, which is outlined in Algorithm 1. During the second step a system of equations should be solved in order to obtain the values for weights w_i, $i = 1, \ldots, k$ as follows:

1. **Set** $W = w_{kj}$ the matrix of k weights, $\Phi = \phi_j(x_i)$ and $T = \{t_i\}$, where t_i, $i = 1, ..., M$ are the expected values for input patterns x_i.
2. **Solve:**

$$\Phi^T \left(T - \Phi W^T \right) = 0 \tag{3}$$

$$W^T = \left(\Phi^T \Phi \right)^{-1} \Phi^T T = \Phi^\dagger T \tag{4}$$

The matrix $\Phi^\dagger = \left(\Phi^T \Phi \right)^{-1} \Phi^T$ denotes the the pseudo-inverse of Φ, with

$$\Phi^\dagger \Phi = I \tag{5}$$

Algorithm 1 The used K-means Algorithm

1. **Initialization**
   (a) **Set** k the number of centers.
   (b) **Read** the patterns of the training dataset $x_i,\ i = 1, \ldots, M$
   (c) **Set** $S_j = \emptyset$, from $j = 1, \ldots, k$.
2. **For** every pattern $x_i,\ i = 1, ..., M$ **do**
   (a) **Set** $j^* = \text{argmin}_{m=1}^{k}\{D(x_i, c_m)\}$ where the variable j^* denotes the nearest center from x_i
   (b) **Set** $S_{j^*} = S_{j^*} \cup \{x_i\}$.
3. **End For**
4. **For** each center $c_j,\ j = 1..k$ **do**
   (a) **Calculate and denote** as M_j the number of samples in S_j
   (b) **Update** the center c_j as
   $$c_j = \frac{1}{M_j} \sum_{x_i \in S_j} x_i$$
5. **End For**
6. **If** the centers c_j did not change then terminate else goto step 2

2.2. The used construction procedure

The Grammatical Evolution procedure is able to construct programs in the provided Backus Naur Form (BNF) grammar [31] using a genetic algorithm with integer chromosomes. These chromosomes represent rule numbers from the underlying grammar and the method has been used successfully in a variety of problems, such as: trigonometric problems [32], production of music [33], construction of neural networks [34,35], video games [37,38], credit classification [39], network security [40] etc. Furthermore, many researchers have developed and published a series of software regarding Grammatical Evolution, such as the GEVA [41] which suggests a GUI environment, a statistical software written in R named gramEvol [43], a Matalab toolbox named GeLab [44], the GenClass software used in classification problems [45], the QFc software [46] used to construct artificial features etc.

The proposed procedure constructs the structure of an RBF neural network using Grammatical Evolution and an appropriately modified Genetic Algorithm [47,48] that guides the course of the procedure. The required BNF grammar for the current algorithm is outlined in Figure 1. The terminal symbol exp represents the function $\exp(x)$ and the terminal symbol pow stands for the power operator x^y.

```
S:=<rbfexpr>
<rbfexpr>::=<Node>
           | <Node> + <rbfexpr>
<Node>::=<number>*exp(<sum>)+<number>
<sum>::= ( -<dist>/(pow(<number>,2)))
           |    <sum>+<sum>
<dist>::= pow(<xxlist>-<number>,2)
<xxlist>::= x1
             |    x2
             ..............
             |    xd
<number>::= (<digitlist>.<digitlist>)
             |    (-<digitlist>.<digitlist>)
<digitlist>::= <digit>
             | <digit><digitlist>
<digit>::= 0
             |  1
             ...........
             |  9
```

Figure 1. The BNF grammar of the proposed method.

The steps of the method used to create RBF neural networks are listed below:

1. **Initialization Step**.
   (a) **Set** the number of chromosomes N_c and the maximum number of allowed generations N_g.
   (b) **Set** the selection rate of the genetic algorithm, denoted as p_c where $p_c \leq 1$.
   (c) **Set** the mutation rate of the genetic algorithm, denoted as p_m, with $p_m \leq 1$.
   (d) **Initialize** randomly each element of each chromosome as a positive random integer number.
   (e) **Set** k=0, the generation number.
2. **Main loop step**.
   (a) Evaluate fitness.
      i. For every chromosome C_i, $i = 1, \ldots, N_c$ **create** the corresponding neural network using the grammar of Figure 1. Denote this network as $y_i(x)$.
      ii. **Calculate** the fitness f_i for the the train set as:

$$f_i = \sum_{j=1}^{M} \left(y_i(\overrightarrow{x}_j) - t_j\right)^2 \tag{6}$$

      where the set $(\overrightarrow{x_j}, t_j)$, $j = 1, ..., M$ denotes the train dataset of the objective problem.
   (b) Apply the selection operator. A sorting of the chromosomes is performed according to their fitness values. The best $(1 - p_s) \times N_c$ of them are copied without changes to the next generation. The rest of chromosomes will be replaced by new chromosomes produced during the crossover procedure.
   (c) Apply the crossover operator. During this procedure a series of $p_s \times N_c$ offsprings will be produced. For each pair of offsprings denoted as $\tilde{z}$ and $\tilde{w}$, two chromosomes (z, w) should be selected from the original population using tournament selection. The new chromosomes are produced from (z, w) using the one - point crossover procedure. An example of the one - point crossover graphically presented in Figure 2.

(d) Perform mutation. For each element of every chromosome, draw randomly a number $r \in [0,1]$. If $r \leq p_m$ then the corresponding element is altered randomly.
(e) **Set** k=k+1

3. **Check for termination step**. If $k \leq N_g$ then **goto** step 2.
4. **Local Optimization step**. Obtain the chromosome C^* and produce the corresponding RBF neural network $y^*(x)$. The parameters of this neural network are obtained by minimizing with some local search procedure the training error defined as:

$$E(y^*(x)) = \sum_{j=1}^{M} \left(y^*(\overrightarrow{x}_j) - t_j\right)^2 \quad (7)$$

5. **Application in test set**. Apply the final model $y^*(x)$ to the test dataset and report the test error.

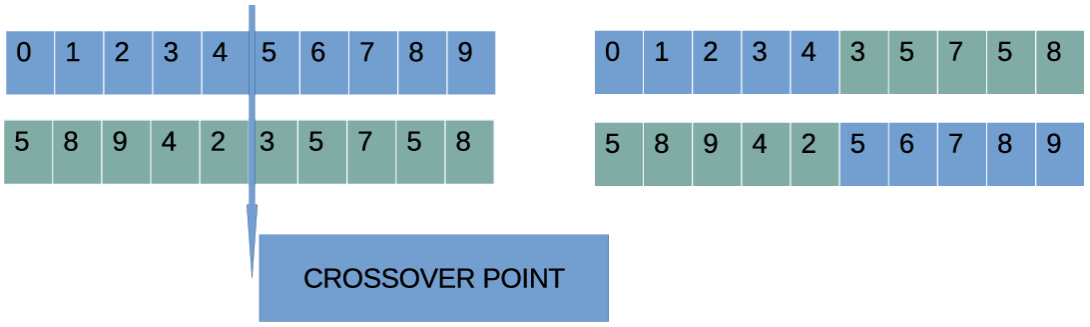

Figure 2. An example for the procedure of one - point crossover, as used in the Grammatical Evolution procedure.

2.3. Installation procedure

The software was coded entirelly in ANSI C++ and the software can be downloaded freely from the following URL: https://github.com/itsoulos/RbfCon (accessed on 14 November 2024). The user should install the QT Library, which is freely available from https://qt.io (accessed on 14 November 2024) in order to compile the software. After downloading the RbfCon-master.zip file, the user should execute the following commands, under any Unix - based system:

1. unzip RbfCon-master.zip.
2. cd RbfCon-master
3. qmake (or qmake-qt5 in some Linux installations).
4. make

For windows systems the user alternatively could install the software using the RBF-CON.msi located in the distribution, which is an installation file that installs the software without the need for the QT Library.

2.4. The main executable RbfCon

The final outcome of the compilation has a series of command line parameters, that the user may adjust. The list of this parameters has as follows:

1. ––trainfile=<filename> The string parameter *filename* determines the name of the file containing the training dataset for the software. The user is required to provide this parameter to start the process. The format for this file is shown in Figure 3. The first number denoted as D in this figure determines the dimension of the input dataset and the second number denoted as M stands for the number of input patterns. Every line in the file contains a pattern and the required desired output.

2. −−testfile=<filename> The string parameter *filename* stands for the name of the test dataset used by the software. The user should provide at least the parameters *trainfile* and *testfile* in order to initiate the method. The format of the test dataset is the same as the training dataset.
3. −−chromosome_count=<count> The integer parameter *count* represents the number of chromosomes that the genetic algorithm will contain. The default value for this parameter is 500.
4. −−chromosome_size=<size> The integer parameter *size* determines the size of each chromosome for the Grammatical Evolution process. The default value for this parameter is 100.
5. −−selection_rate=<rate> The double precision parameter *rate* stands for the selection rate of the Grammatical Evolution process. The default value for this parameter is 0.10 (10%).
6. −−mutation_rate=<rate> The double precision parameter *rate* represents the mutation rate for the Grammatical Evolution process. The default value for this parameter is 0.05 (5%).
7. −−generations=<gens> The integer parameter *gens* stands for the maximum number of allowed generations for the Grammatical Evolution procedure. The default value for this parameter is 500.
8. −−local_method=<method> The used local optimization method, that will be applied to the parameters of the RBF model when the Grammatical Evolution procedure finished. The available local optimization methods are:
   (a) *lbfgs*. The method L-BFGS can be considered as a variation of the Broyden–Fletcher–Goldfarb–Shanno (BFGS) optimization technique [11], that utilizes a minimum amount of memory. This optimization method has been used in many cases, such as image reconstruction [36], inverse eigenvalue problems [49], seismic problems [50], training of deep neural networks [51] etc. Also, the a series of modifications have been proposed that take advantage of modern parallel computing systems [52–54].
   (b) *bfgs*. The BFGS variant of Powell was used here [56], when this option is enabled.
   (c) *adam*. This option denotes the application of the Adam optimizer [57] as the local optimization algorithm.
   (d) *gradient*. This option represents the usage of the Gradient Descent method [58] as the local optimization algorithm.
   (e) *none*. With this option no local optimization method will be used after the Genetic Algorithm is terminated.
9. −−iterations=<iters> This integer parameter determines the number of consecutive applications of the proposed technique to the original data set. The default value is 30.

Figure 3. The format for the used datasets.

D
M

$$\begin{array}{ccccc} x_{11} & x_{12} & \dots & x_{1D} & y_1 \\ x_{21} & x_{22} & \dots & x_{2D} & y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{M1} & x_{M2} & \dots & x_{MD} & y_M \end{array}$$

2.5. Example of execution

As an example of execution consider the Wdbc dataset [59] which is located under EXAMPLES subdirectory. This dataset is used for the detection of breast tumors. An example of execution could be the following:

```
./RbfCon --trainfile=EXAMPLES/wdbc.train --testfile=EXAMPLES/wdbc.test --iterations=2
```

The output of this command is shown below:

```
Iteration:     1 TRAIN ERROR:          20.40039584
Iteration:     1 TEST  ERROR:        0.07733519754
Iteration:     1 CLASS ERROR:                9.12%
Iteration:     1 SOLUTION:     ((6.711)*(exp((-((pow(x13-(-97.73),2))+
  ((pow(x23-(43.7),2))+((pow(x23-(84.7),2))+
  (pow(x23-(91.3),2))))))/(2*pow((60.1),2))))+
  (-0.0099969))+((-0.39)*(exp((-((pow(x25-(9.336),2))+
  ((pow(x18-(6.8),2))+(pow(x4-(6.52),2))))))/(2*pow((711.5),2))))+
  (0.00185452))+((3.7)*(exp((-(pow(x3-(-6.8),2)))/(2*pow((2.3),2))))+
  (-0.001))+((-5.8)*(exp((-(pow(x2-(-99969.10),2)))/
  (2*pow((6342.3747),2))))+(-0.002))
Iteration:     2 TRAIN ERROR:          16.06705417
Iteration:     2 TEST  ERROR:        0.05654528154
Iteration:     2 CLASS ERROR:                4.56%
Iteration:     2 SOLUTION:     ((4.4)*(exp((-((pow(x27-(3.6),2))+
  (pow(x23-(8.9893),2))))/(2*pow((49.9),2))))+(-0.004))+
  ((-2.09)*(exp((-((pow(x23-(-9.1),2))+
  ((pow(x30-(-40.4),2))+((pow(x4-(98.5),2))+
  (pow(x22-(-4.4),2))))))/(2*pow((189.9),2))))+(0.007))+
  ((-9.8)*(exp((-((pow(x2-(24.4),2))+((pow(x28-(2.04),2))+
  (pow(x27-(9.11),2))))))/(2*pow((3.1),2))))+(0.009))+
  ((-67.49)*(exp((-(pow(x10-(-2629940.4),2)))/(2*pow((098.5),2))))+(0.0094))
Average Train Error:          18.23372501
Average Test  Error:        0.06694023954
Average Class Error:                6.84%
```

The program prints the train error, the test error and the classification for every run as well as the averages for these values. Also, in each execution the main program shows the constructed RBF network as a string value.

3. Results

The new technique can be used to construct RBF networks for both classification and data fitting problems. For this reason, datasets covering many scientific areas and freely available from the following websites were used:

1. The UCI repository, https://archive.ics.uci.edu/(accessed on 14 November 2024)[60]
2. The Keel repository, https://sci2s.ugr.es/keel/datasets.php(accessed on 14 November 2024)[61].
3. The Statlib URL **ftp://lib.stat.cmu.edu/datasets/index.html**(accessed on 14 November 2024).

3.1. The used datasets

The following list of classification datasets was used in the conducted experiments:

1. **Appendicitis**, a medical dataset that was studied in [62,63]. This problem has 2 distinct classes.
2. **Alcohol**, a dataset related to alcohol consumption [64]. This dataset contains 4 distinct classes.
3. **Australian**, an economic dataset [65], related to the risk in bank transactions. The number of classes for this dataset is 2.
4. **Bands**, related to problems occurred in printing [66] and it has two classes.
5. **Cleveland**, a medical dataset proposed in a series of research papers [67,68]. This dataset has 5 distinct classes.
6. **Dermatology**, which is also a medical dataset originated in [69] and it has 6 classes.
7. **Ecoli**, that is related to problems regarding proteins [70] and it has 8 classes.
8. **Fert**, used in demographics and it has two distinct classes.
9. **Haberman**, a medical dataset related to the detection of breast cancer. This dataset has two classes.
10. **Hayes-roth** dataset [71]. This dataset has 3 classes.

11. **Heart**, which is a medical dataset related to heart diseases [72] with two classes.
12. **HeartAttack**, which is a medical dataset related to heart diseases having two distinct classes.
13. **Hepatitis**, a dataset used for the detection of hepatitis.
14. **Housevotes**, used for the Congressional voting in USA [73].
15. **Ionosphere**, used for measurements from the ionosphere [74,75] and it has two classes.
16. **Liverdisorder**, which is a medical dataset [76,77] with two classes.
17. **Lymography** [78], that has 4 classes.
18. **Magic**, this dataset contains generated data to simulate registration of high energy gamma particles [79] and it has two classes.
19. **Mammographic**, a medical dataset used for the prediction of breast cancer [80].
20. **Parkinsons**, a dataset used for the detection of Parkinson's disease [81,82].
21. **Pima**, a medical dataset that was useful for the detection of diabetes [83].
22. **Popfailures**, a dataset contains climate measurements [84] and it has two classes.
23. **Regions2**, a medical dataset that contains measurements from liver biopsy images [85] and it has five classes.
24. **Ring**, which is a problem with 20 dimensions and two classes. Each class is drawn from a multivariate normal distribution.
25. **Saheart**, a medical dataset related to heart diseases with 2 classes [86].
26. **Statheart**, which is a also a medical dataset related to heart diseases.
27. **Spambase**, a dataset used to detect spam emails from a large database. The dataset has two distinct classes.
28. **Spiral**, an artificial dataset with two classes.
29. **Student**, a dataset contains measurements from various experiments in schools [87].
30. **Tae**, this dataset consist of evaluations of teaching performance and it has 3 classes.
31. **Transfusion**, which is a medical dataset [88].
32. **Wdbc**, a medical dataset used to detect the presence of breast cancer that studied in a series of research papers [89,90].
33. **Wine**, a dataset contains measurements about the quality of wines [91,92], with 3 distinct classes.
34. **EEG** dataset, which is a medical dataset about EEG measurements studied in a series of papers [93,94]. The following cases were obtained from this dataset: Z_F_S, ZO_NF_S, Z_O_N_F_S and ZONF_S.
35. **Zoo**, that used to detect the class of some animals [95] and it contains seven classes.

Also the following list of regression datasets was used in the conducted experiments:

1. **Abalone**, a dataset that used to predict the age of abalones with 8 features.
2. **Airfoil**, a dataset provided by NASA [97] with 5 features.
3. **Auto**, a dataset used to predict the fuel consumption with 7 features.
4. **BK**, that contains measurements from a series of basketball games and it has 4 features.
5. **BL**, a dataset used to record electricity experiments and it contains 7 features.
6. **Baseball**, a dataset with 16 features used to estimate the income of baseball players.
7. **Concrete**, a dataset with 8 features used in civil engineering [98].
8. **DEE**, a dataset with 6 featured that was to predict the electricity cost.
9. **FA**, that contains measurements about the body fat and it has 18 features.
10. **HO**, a dataset originated in the STATLIB repository with 13 features.
11. **Housing**, used to predict the price of houses [99] and it has 13 features.
12. **Laser**, which is a dataset with 4 features and it has been used in various laser experiments.
13. **LW**, a dataset with 9 features used to record the weight of babies.
14. **MB**, a dataset provide by from Smoothing Methods in Statistics [100] with 2 features.
15. **Mortgage**, an economic dataset from USA with 15 features.
16. **NT**, a dataset with two features used to record body temperatures [101].
17. **Plastic**, a dataset with 2 features used to detect the pressure on plastics.
18. **PL**, a dataset with 2 features provided by the STATLIB repository.

19. **Quake**, a dataset used to measure the strength of earthquakes with 3 features.
20. **SN**, a dataset that provides experimental measurements related to trellising and pruning. This dataset has 11 features.
21. **Stock**, a dataset with 9 features used to approximate the prices of various stocks.
22. **Treasury**, an economic dataset from USA that contains 15 features.
23. **TZ**, which is a dataset originated in the STATLIB repository and it has 60 features.

3.2. Experimental results

The implementation of the machine learning methods that participated in the conducted experiments was made using ANSI C++. The experiments were conducted 30 times and the average classification or regression error as measured on the test was recorded. The classification error is measured as:

$$E_C(M(x)) = 100 \times \frac{\sum_{i=1}^{N}(\text{class}(M(x_i)) - y_i)}{N} \tag{8}$$

where $M(x)$ denotes the used machine learning model and set T is defined as $T = (x_i, y_i),\ i = 1, \ldots, N$, that denotes the used testing dataset. On the other hand, the regression error is calculated as:

$$E_R(M(x)) = \frac{\sum_{i=1}^{N}(M(x_i) - y_i)^2}{N} \tag{9}$$

The well - known method of ten - fold cross validation was used to validate the experimental results. All the experiments were carried out on an AMD Ryzen 5950X with 128GB of RAM. The used operating system was Debian Linux. The values for the parameters of the proposed method are listed in Table 1.

Table 1. The values for the parameters, as used in the conducted experiments.

PARAMETER	VALUE
chromosome_count	500
chromosome_size	100
selection_rate	0.1
mutation_rate	0.05
generations	500

The experimental results for the classification datasets are shown in Table 2 and for the regression datasets in 3The following notation was used in these tables:

1. The column RBF denotes the application of the original two - phase method for the training of an RBF neural network with $k = 10$ weights. This method was described previously in subsection 2.1.
2. The column CONRBF denotes the application of the current software using the parameters of Table 1 and as local search procedure the *none* option was selected.
3. The column CONRBF(LBFGS) denotes the application of the current method using as local search procedure the L-Bfgs method.
4. The column CONRBF(BFGS) stands for the application of the current work using as local search procedure the Bfgs method.
5. The row AVERAGE represents the average classification or regression error for all datasets in the corresponding table.

Table 2. Experimental results using the original training method and the proposed one for the classification datasets.

DATASET	RBF	CONRBF	CONRBF(LBFGS)	CONRBF(BFGS)
APPENDICITIS	12.23%	13.60%	14.10%	13.60%
ALCOHOL	49.38%	51.24%	49.32%	45.11%
AUSTRALIAN	34.89%	14.14%	14.26%	14.23%
BANDS	37.17%	35.75%	36.58%	36.03%
CLEVELAND	67.10%	49.14%	49.52%	50.28%
DERMATOLOGY	62.34%	45.20%	38.43%	36.66%
ECOLI	59.48%	54.18%	54.39%	53.03%
FERT	15.00%	15.20%	15.50%	15.90%
HABERMAN	25.10%	26.27%	27.07%	26.40%
HAYES-ROTH	64.36%	34.54%	39.00%	36.76%
HEART	31.20%	17.22%	17.44%	16.96%
HEARTATTACK	29.00%	22.60%	21.83%	21.53%
HEPATITIS	64.63%	54.25%	47.50%	48.75%
HOUSEVOTES	6.13%	3.05%	3.65%	3.05%
IONOSPHERE	16.22%	14.32%	13.40%	12.37%
LIVERDISORDER	30.84%	32.21%	31.71%	31.35%
LYMOGRAPHY	25.50%	26.36%	25.71%	21.49%
MAGIC	21.28%	22.18%	19.62%	20.35%
MAMMOGRAPHIC	21.38%	17.67%	19.02%	17.30%
PARKINSONS	17.41%	12.79%	13.37%	12.47%
PIMA	25.78%	24.13%	24.90%	24.15%
POPFAILURES	7.04%	6.98%	6.94%	6.80%
REGIONS2	38.29%	26.34%	26.81%	26.55%
RING	21.67%	11.08%	10.13%	10.33%
SAHEART	32.19%	29.52%	29.28%	29.19%
SPAMBASE	29.35%	16.95%	15.53%	15.60%
SPIRAL	44.87%	42.14%	43.88%	43.05%
STATHEART	31.36%	19.22%	19.15%	18.19%
STUDENT	5.49%	7.63%	6.33%	4.32%
TAE	60.02%	56.73%	56.33%	56.40%
TRANSFUSION	26.41%	25.15%	24.70%	24.36%
WDBC	7.27%	6.77%	6.52%	6.36%
WINE	31.41%	11.00%	11.65%	10.71%
Z_F_S	13.16%	11.13%	11.47%	10.57%
Z_O_N_F_S	48.70%	52.34%	49.32%	46.22%
ZO_NF_S	9.02%	11.08%	11.18%	8.90%
ZONF_S	4.03%	4.14%	3.94%	3.58%
ZOO	21.93%	11.60%	9.00%	10.90%
AVERAGE	**30.23%**	**24.63%**	**24.17%**	**23.42%**

Table 3. Experimental results using the original train method and the proposed one for a series of regression datasets.

DATASET	RBF	CONRBF	CONRBF(LBFGS)	CONRBF(BFGS)
ABALONE	7.37	6.14	5.35	5.32
AIRFOIL	0.27	0.004	0.004	0.002
AUTO	17.87	11.03	10.03	9.45
BK	0.02	0.02	0.02	0.03
BL	0.013	0.04	0.024	0.01
BASEBALL	93.02	66.74	65.80	64.52
CONCRETE	0.011	0.012	0.010	0.009
DEE	0.17	0.23	0.22	0.20
FA	0.015	0.013	0.014	0.011
HO	0.03	0.013	0.015	0.012
HOUSING	57.68	20.60	19.02	18.40
LASER	0.03	0.06	0.05	0.03
LW	0.03	0.011	0.011	0.011
MB	5.43	0.055	0.06	0.12
MORTGAGE	1.45	0.165	0.18	0.074
NT	13.97	0.006	0.006	0.006
PLASTIC	8.62	3.58	2.86	2.43
PL	2.118	0.064	0.026	0.024
QUAKE	0.07	0.036	0.036	0.036
SN	0.027	0.025	0.026	0.025
STOCK	12.23	8.31	6.90	6.25
TREASURY	2.02	0.0027	0.12	0.10
TZ	0.036	0.036	0.036	0.035
AVERAGE	**9.67**	**5.10**	**4.82**	**4.66**

In table 2, the four models were evaluated based on their error rates across 40 different datasets. It was observed that the error rate varied significantly between datasets, reflecting differences in each model's performance depending on the dataset. In general, the models CONRBF, CONRBF(LBFGS), and CONRBF(BFGS) showed lower error rates compared to RBF in many datasets, suggesting that the parameterized versions of CONRBF have better adaptability. In several datasets, CONRBF outperformed the original RBF in terms of error rate. For example, in datasets such as AUSTRALIAN and SPAMBASE, CONRBF shows a markedly lower error rate compared to RBF. The models CONRBF(LBFGS) and CONRBF(BFGS) frequently demonstrate improved error rates relative to the original CONRBF, although these differences are not always significant. In datasets such as HEPATITIS, TAE, and WINE, both CONRBF(LBFGS) and CONRBF(BFGS) outperform the simple CONRBF model. Despite the general trend of improved performance in the parameterized versions of CONRBF, exceptions do exist. For instance, in the dataset Z_O_N_F_S, RBF performs better than the other models, while in ZO_NF_S and ZONF_S, the differences between models are minimal. CONRBF with LBFGS and CONRBF with BFGS exhibit similar performance, although one model occasionally outperforms the other in certain datasets. In DERMATOLOGY and ZOO, CONRBF(LBFGS) achieves a lower error rate than CONRBF(BFGS), whereas in datasets like HEARTATTACK, the differences are very small. In conclusion, the comparison of models reveals that the parameterized versions of CONRBF, especially with the LBFGS and BFGS optimization methods, generally outperform the simple RBF model in many datasets.

The results shown in Table 3 include the error values for various datasets, where different training methods were applied. The values in the tables represent errors (not percentages) obtained by applying each model to each dataset. The CONRBF method shows lower error rates compared to the original RBF in almost all datasets, indicating that the proposed method provides improvements over the original. For example, in the

HOUSING dataset, the error is reduced from 57.68 (RBF) to 20.6 (CONRBF), while in the BASEBALL dataset, the error decreases from 93.02 (RBF) to 66.74 (CONRBF). The LBFGS and BFGS variations of the CONRBF method further reduce errors compared to the original CONRBF in certain datasets. For instance, in the AUTO dataset, the error decreases from 11.03 (CONRBF) to 10.03 (CONRBF with LBFGS) and 9.45 (CONRBF with BFGS). Similarly, in the STOCK dataset, the error drops from 8.31 (CONRBF) to 6.9 (CONRBF with LBFGS) and 6.25 (CONRBF with BFGS), demonstrating that the addition of these optimization methods can further enhance the model's performance. In summary, the results indicate that the proposed CONRBF method, along with its LBFGS and BFGS optimized variants, has the potential to reduce error rates compared to the original RBF method.

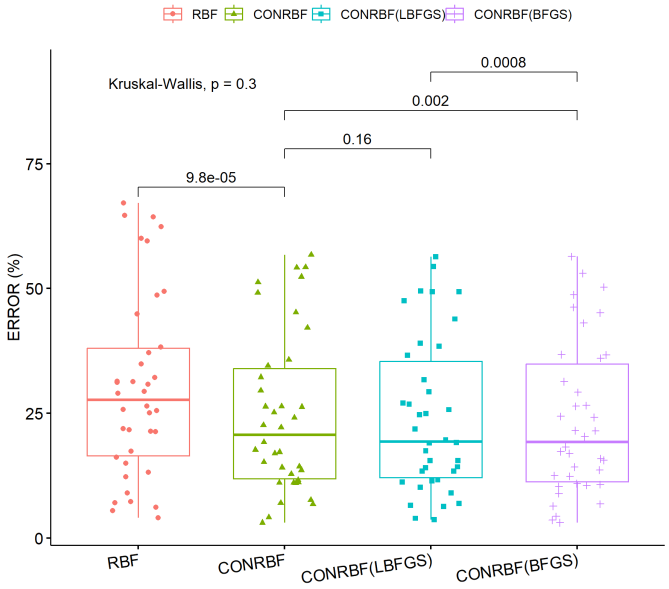

Figure 4. Statistical comparison for the used methods and the classification datasets.

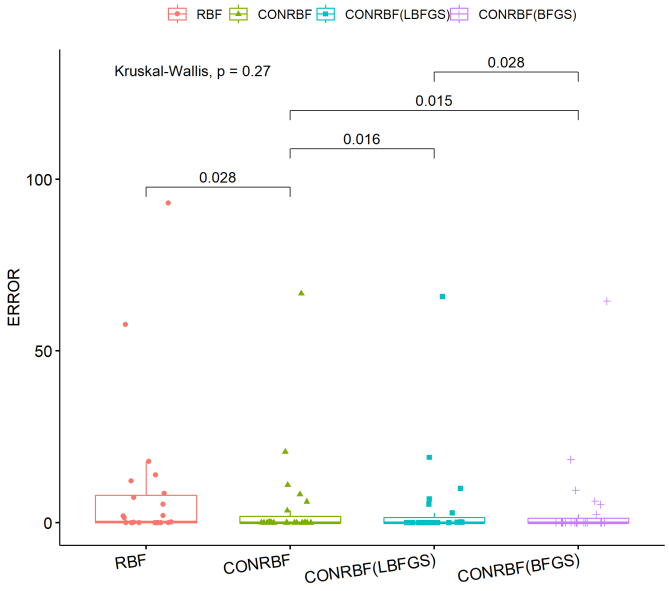

Figure 5. Statistical test for the regression datasets and the used methods.

In Figure 4, paired comparisons were subsequently performed between individual pairs of models using the paired t-test. For RBF vs. CONRBF: $p = 9.8 \times 10^{-5}$ – there is a statistically significant difference between the two models ($p < 0.05$), indicating that the CONRBF model exhibits a significantly different error rate compared to RBF. For CONRBF vs. CONRBF(LBFGS): $p = 0.16$ – there is no statistically significant difference between these two models at the 0.05 significance level, suggesting that the two versions of CONRBF have similar error rates. For CONRBF vs. CONRBF(BFGS): $p = 0.002$ – there is a statistically significant difference ($p < 0.05$) between these two variants, indicating that the BFGS method yields a different error rate compared to the original CONRBF. For CONRBF(LBFGS) vs. CONRBF(BFGS): $p = 0.0008$ – the difference between these two methods is also statistically significant ($p < 0.05$), showing that CONRBF(BFGS) has a different performance from CONRBF(LBFGS) in terms of error rate. In summary, the results of the paired comparisons indicate that the differences between some models are statistically significant, suggesting that the choice of model variant can substantially impact the error rate. However, the overall analysis using the Kruskal-Wallis test did not find a statistically significant difference between the groups as a whole.

The statistical analysis conducted in Figure 5 to compare the different machine learning models shows distinct patterns in their error rates. Paired comparisons were performed between each model using a paired t-test, and the significance levels for each pair are as follows: for RBF vs. CONRBF, p=0.028 for CONRBF vs. CONRBF(LBFGS), p=0.016 for CONRBF vs. CONRBF(BFGS), p=0.015 and for CONRBF(LBFGS) vs. CONRBF(BFGS), p=0.028. All comparisons resulted in $p<0.05$, indicating that there are statistically significant differences between each pair of models. This suggests that each variant of the CONRBF model, including the LBFGS and BFGS optimizations, provides a unique effect on the error rate, significantly differentiating it from the others in performance. However, when analyzing the groups as a whole using the Kruskal-Wallis test, the result was p=0.27, which is greater than the standard significance level of 0.05. This indicates that, while there are significant differences between individual pairs, there is no overall statistically significant difference across all four models collectively. This result implies that, while certain optimizations improve performance in pairwise comparisons, these differences are not strong enough to conclude a significant variation across the entire set of models. Therefore, the choice of model variant can impact error rates significantly in specific

comparisons, but these effects are not uniform across all datasets when considered in aggregate.

4. Conclusions

A novel method that constructs the architecture of RBF neural networks with assistance was described in the current paper, accompanied by the relevant software. The method can construct the architecture of RBF networks using a process that incorporates the Grammatical Evolution procedure. Also, the used method can estimate the parameters of the network, which can be further modified by the application of a local optimization procedure. The proposed method can be applied without modifications to classification and regression datasets. The new method was compared against the traditional method used to train RBF networks and the experimental results validated the efficiency of the proposed work in a wide series of classification and regression datasets found in the recent bibliography.

The used software was coded in ANSI C++ with the assistance of the freely available library of QT, in order to be portable on the majority of operating systems. The user can control the process by a series of simple command line options, such as the number of needed chromosomes for the genetic population or the selection rate used. Also, the software offers the possibility of enhancing the parameters of the RBF network with the application of some local optimization procedures. The experimental results clearly indicated that the application of the local optimization algorithm can further reduce the test error in a series of datasets.

Future improvements to the software may include the periodic application of the local search procedure during the execution of the genetic algorithm or even the incorporation of global optimization procedures, in order to reduce even further the test error of the current work. Also, since the method is based on Genetic Algorithms, parallel optimization procedures, such as MPI [102] or OpenMP [103] can be used to reduce the execution time of the method. Finally, the software could possibly be extended to allow the user to input other grammars as parameters, or even to be able to use other output functions besides Gaussian.

Author Contributions: The proposed software was conceived and implemented by all members of the team. The experiments were conducted using various classification and regression datasets, and the comparative results were presented by I.G.T and I.V. The statistical analysis was carried out by V.C. All authors have reviewed and approved the final published version.

Funding: This research received no external funding.

Institutional Review Board 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 conflicts of interest.

References

1. M. Mjahed, The use of clustering techniques for the classification of high energy physics data, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **559**, pp. 199-202, 2006.
2. M Andrews, M Paulini, S Gleyzer, B Poczos, End-to-End Event Classification of High-Energy Physics Data, Journal of Physics: Conference Series **1085**, 2018.
3. P. He, C.J. Xu, Y.Z. Liang, K.T. Fang, Improving the classification accuracy in chemistry via boosting technique, Chemometrics and Intelligent Laboratory Systems 70, pp. 39-46, 2004.
4. J.A. Aguiar, M.L. Gong, T.Tasdizen, Crystallographic prediction from diffraction and chemistry data for higher throughput classification using machine learning, Computational Materials Science **173**, 109409, 2020.

5. I. Kaastra, M. Boyd, Designing a neural network for forecasting financial and economic time series, Neurocomputing **10**, pp. 215-236, 1996.
6. R. Hafezi, J. Shahrabi, E. Hadavandi, A bat-neural network multi-agent system (BNNMAS) for stock price prediction: Case study of DAX stock price, Applied Soft Computing **29**, pp. 196-210, 2015.
7. S.S. Yadav, S.M. Jadhav, Deep convolutional neural network based medical image classification for disease diagnosis. J Big Data **6**, 113, 2019.
8. L. Qing, W. Linhong , D. Xuehai, A Novel Neural Network-Based Method for Medical Text Classification, Future Internet **11**, 255, 2019.
9. J. Park and I. W. Sandberg, Universal Approximation Using Radial-Basis-Function Networks, Neural Computation **3**, pp. 246-257, 1991.
10. G.A. Montazer, D. Giveki, M. Karami, H. Rastegar, Radial basis function neural networks: A review. Comput. Rev. J **1**, pp. 52-74, 2018.
11. R. Fletcher, A new approach to variable metric algorithms, Computer Journal 13, pp. 317-322, 1970.
12. V.I. Gorbachenko, M.V. Zhukov, Solving boundary value problems of mathematical physics using radial basis function networks. Comput. Math. and Math. Phys. **57**, pp. 145–155, 2017.
13. J. Määttä, V. Bazaliy, J. Kimari, F. Djurabekova, K. Nordlund, T. Roos, Gradient-based training and pruning of radial basis function networks with an application in materials physics, Neural Networks **133**, pp. 123-131, 2021.
14. R. -J. Lian, Adaptive Self-Organizing Fuzzy Sliding-Mode Radial Basis-Function Neural-Network Controller for Robotic Systems, IEEE Transactions on Industrial Electronics **61**, pp. 1493-1503, 2014.
15. M. Vijay, D. Jena, Backstepping terminal sliding mode control of robot manipulator using radial basis functional neural networks. Computers & Electrical Engineering **67**, pp. 690-707, 2018.
16. U. Ravale, N. Marathe, P. Padiya, Feature Selection Based Hybrid Anomaly Intrusion Detection System Using K Means and RBF Kernel Function, Procedia Computer Science **45**, pp. 428-435, 2015.
17. M. Lopez-Martin, A. Sanchez-Esguevillas, J. I. Arribas, B. Carro, Network Intrusion Detection Based on Extended RBF Neural Network With Offline Reinforcement Learning, IEEE Access **9**, pp. 153153-153170, 2021.
18. Foody, G. M. (2004). Supervised image classification by MLP and RBF neural networks with and without an exhaustively defined set of classes. International Journal of Remote Sensing, 25(15), 3091-3104.
19. Er, M. J., Wu, S., Lu, J., & Toh, H. L. (2002). Face recognition with radial basis function (RBF) neural networks. IEEE transactions on neural networks, 13(3), 697-710.
20. L.I. Kuncheva, Initializing of an RBF network by a genetic algorithm, Neurocomputing **14**, pp. 273-288, 1997.
21. F. Ros, M. Pintore, A. Deman, J.R. Chrétien, Automatical initialization of RBF neural networks, Chemometrics and Intelligent Laboratory Systems **87**, pp. 26-32, 2007.
22. D. Wang, X.J. Zeng, J.A. Keane, A clustering algorithm for radial basis function neural network initialization, Neurocomputing **77**, pp. 144-155, 2012.
23. E. Ricci, R. Perfetti, Improved pruning strategy for radial basis function networks with dynamic decay adjustment, Neurocomputing **69**, pp. 1728-1732, 2006.
24. Guang-Bin Huang, P. Saratchandran and N. Sundararajan, A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation, IEEE Transactions on Neural Networks **16**, pp. 57-67, 2005.
25. M. Bortman and M. Aladjem, A Growing and Pruning Method for Radial Basis Function Networks, IEEE Transactions on Neural Networks **20**, pp. 1039-1045, 2009.
26. Chen, J.Y.; Qin, Z.; Jia, J. A PSO-Based Subtractive Clustering Technique for Designing RBF Neural Networks. In Proceedings of the 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), Hong Kong, 1–6 June 2008; pp. 2047–2052. [Google Scholar]
27. Esmaeili, A.; Mozayani, N. Adjusting the parameters of radial basis function networks using Particle Swarm Optimization. In Proceedings of the 2009 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, Hong Kong, 11–13 May 2009; pp. 179–181. [Google Scholar]
28. O'Hora, B.; Perera, J.; Brabazon, A. Designing Radial Basis Function Networks for Classification Using Differential Evolution. In Proceedings of the 2006 IEEE International Joint Conference on Neural Network Proceedings, Vancouver, BC, USA, 16–21 July 2006; pp. 2932–2937.
29. MacQueen, J.: Some methods for classification and analysis of multivariate observations, in: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Vol. 1, No. 14, pp. 281-297, 1967.
30. M. O'Neill, C. Ryan, Grammatical evolution, IEEE Trans. Evol. Comput. **5,**pp. 349–358, 2001.
31. 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.
32. C. Ryan, M. O'Neill, J.J. Collins, Grammatical evolution: Solving trigonometric identities, proceedings of Mendel. Vol. 98. 1998.
33. 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.
34. 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.

35. K. Soltanian, A. Ebnenasir, M. Afsharchi, Modular Grammatical Evolution for the Generation of Artificial Neural Networks, Evolutionary Computation **30**, pp 291–327, 2022.
36. H. Wang, H. Gemmeke, T. Hopp, J. Hesser, Accelerating image reconstruction in ultrasound transmission tomography using L-BFGS algorithm, In: Medical Imaging 2019: Ultrasonic Imaging and Tomography; 109550B (2019) https://doi.org/10.1117/12.2512654 Event: SPIE Medical Imaging, 2019, San Diego, California, United States.
37. 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.
38. 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.
39. A. Brabazon, M. O'Neill, Credit classification using grammatical evolution, Informatica **30.3**, 2006.
40. S. Şen, J.A. Clark. A grammatical evolution approach to intrusion detection on mobile ad hoc networks, In: Proceedings of the second ACM conference on Wireless network security, 2009.
41. M. O'Neill, E. Hemberg, C. Gilligan, E. Bartley, J. McDermott, A. Brabazon, GEVA: grammatical evolution in Java. ACM SIGEVOlution **3**, pp. 17-22, 2008.
42. A. Ortega, M. de la Cruz, M. Alfonseca, Christiansen Grammar Evolution: Grammatical Evolution With Semantics, IEEE Transactions on Evolutionary Computation **11**, pp. 77-90, Feb. 2007.
43. F. Noorian, A.M. de Silva, P.H.W. Leong, gramEvol: Grammatical Evolution in R, Journal of Statistical Software **71**, pp. 1–26, 2016.
44. M.A. Raja, C. Ryan, GELAB - A Matlab Toolbox for Grammatical Evolution. In: Yin, H., Camacho, D., Novais, P., Tallón-Ballesteros, A. (eds) Intelligent Data Engineering and Automated Learning – IDEAL 2018. IDEAL 2018. Lecture Notes in Computer Science(), vol 11315, 2018. Springer, Cham. https://doi.org/10.1007/978-3-030-03496-2_22
45. N.Anastasopoulos, I.G. Tsoulos, A. Tzallas, GenClass: A parallel tool for data classification based on Grammatical Evolution, SoftwareX **16**, 100830, 2021.
46. I.G. Tsoulos, QFC: A Parallel Software Tool for Feature Construction, Based on Grammatical Evolution, Algorithms **15**, 295, 2022.
47. D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Publishing Company, Reading, Massachussets, 1989.
48. Z. Michaelewicz, Genetic Algorithms + Data Structures = Evolution Programs. Springer - Verlag, Berlin, 1996.
49. Z. Dalvand, M. Hajarian, Solving generalized inverse eigenvalue problems via L-BFGS-B method, Inverse Problems in Science and Engineering **28**, pp. 1719-1746, 2020.
50. Y. Rao, Y. Wang, Seismic waveform tomography with shot-encoding using a restarted L-BFGS algorithm, Scientific Reports **7**, pp. 1-9, 2017.
51. Yousefi, M., Martínez Calomardo, Á. (2022). A Stochastic Modified Limited Memory BFGS for Training Deep Neural Networks. In: Arai, K. (eds) Intelligent Computing. SAI 2022. Lecture Notes in Networks and Systems, vol 507. Springer, Cham. https://doi.org/10.1007/978-3-031-10464-0_2
52. Y. Fei, G. Rong, B. Wang, W. Wang, Parallel L-BFGS-B algorithm on GPU, Computers & Graphics **40**, pp. 1-9, 2014.
53. L. D'Amore, G. Laccetti, D. Romano, G. Scotti, A. Murli, Towards a parallel component in a GPU–CUDA environment: a case study with the L-BFGS Harwell routine, International Journal of Computer Mathematics **92**, pp. 59-76, 2015.
54. M.M. Najafabadi, T.M. Khoshgoftaar, F. Villanustre et al, Large-scale distributed L-BFGS, J Big Data **4**, 22, 2017.
55. J.L. Morales, A numerical study of limited memory BFGS methods, Applied Mathematics Letters **15**, pp. 481-487, 2002.
56. M.J.D Powell, A Tolerant Algorithm for Linearly Constrained Optimization Calculations, Mathematical Programming **45**, pp. 547-566, 1989.
57. 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.
58. Amari, S. I. (1993). Backpropagation and stochastic gradient descent method. Neurocomputing, 5(4-5), 185-196.
59. 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.
60. M. Kelly, R. Longjohn, K. Nottingham, The UCI Machine Learning Repository, https://archive.ics.uci.edu.
61. 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.
62. 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.
63. M. Wang, Y.Y. Zhang, F. Min, Active learning through multi-standard optimization, IEEE Access **7**, pp. 56772–56784, 2019.
64. Tzimourta, K.D.; Tsoulos, I.; Bilero, I.T.; Tzallas, A.T.; Tsipouras, M.G.; Giannakeas, N. Direct Assessment of Alcohol Consumption in Mental State Using Brain Computer Interfaces and Grammatical Evolution. Inventions 2018, 3, 51.
65. J.R. Quinlan, Simplifying Decision Trees. International Journal of Man-Machine Studies **27**, pp. 221-234, 1987.
66. B. Evans, D. Fisher, Overcoming process delays with decision tree induction. IEEE Expert **9**, pp. 60-66, 1994.
67. 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.

68. R. Setiono , W.K. Leow, FERNN: An Algorithm for Fast Extraction of Rules from Neural Networks, Applied Intelligence **12**, pp. 15-25, 2000.
69. 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.
70. P. Horton, K.Nakai, A Probabilistic Classification System for Predicting the Cellular Localization Sites of Proteins, In: Proceedings of International Conference on Intelligent Systems for Molecular Biology **4**, pp. 109-15, 1996.
71. 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.
72. I. Kononenko, E. Šimec, M. Robnik-Šikonja, Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF, Applied Intelligence **7**, pp. 39–55, 1997
73. 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.
74. J.G. Dy , C.E. Brodley, Feature Selection for Unsupervised Learning, The Journal of Machine Learning Research **5**, pp 845–889, 2004.
75. S. J. Perantonis, V. Virvilis, Input Feature Extraction for Multilayered Perceptrons Using Supervised Principal Component Analysis, Neural Processing Letters **10**, pp 243–252, 1999.
76. J. Garcke, M. Griebel, Classification with sparse grids using simplicial basis functions, Intell. Data Anal. **6**, pp. 483-502, 2002.
77. J. Mcdermott, R.S. Forsyth, Diagnosing a disorder in a classification benchmark, Pattern Recognition Letters **73**, pp. 41-43, 2016.
78. G. Cestnik, I. Konenenko, I. Bratko, Assistant-86: A Knowledge-Elicitation Tool for Sophisticated Users. In: Bratko, I. and Lavrac, N., Eds., Progress in Machine Learning, Sigma Press, Wilmslow, pp. 31-45, 1987.
79. Heck, D., Knapp, J., Capdevielle, J. N., Schatz, G., & Thouw, T. (1998). CORSIKA: A Monte Carlo code to simulate extensive air showers.
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, S.J Roberts et al, Exploiting Nonlinear Recurrence and Fractal Scaling Properties for Voice Disorder Detection. BioMed Eng OnLine **6**, 23, 2007.
82. M.A. Little, P.E. McSharry, E.J. Hunter, J. Spielman, L.O. Ramig, Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. IEEE Trans Biomed Eng. **56**, pp. 1015-1022, 2009.
83. J.W. Smith, J.E. Everhart, W.C. Dickson, W.C. Knowler, R.S. Johannes, Using the ADAP learning algorithm to forecast the onset of diabetes mellitus, In: Proceedings of the Symposium on Computer Applications and Medical Care IEEE Computer Society Press, pp.261-265, 1988.
84. D.D. Lucas, R. Klein, J. Tannahill, D. Ivanova, S. Brandon, D. Domyancic, Y. Zhang, Failure analysis of parameter-induced simulation crashes in climate models, Geoscientific Model Development **6**, pp. 1157-1171, 2013.
85. N. Giannakeas, M.G. Tsipouras, A.T. Tzallas, K. Kyriakidi, Z.E. Tsianou, P. Manousou, A. Hall, E.C. Karvounis, V. Tsianos, E. Tsianos, A clustering based method for collagen proportional area extraction in liver biopsy images (2015) Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS, 2015-November, art. no. 7319047, pp. 3097-3100.
86. T. Hastie, R. Tibshirani, Non-parametric logistic and proportional odds regression, JRSS-C (Applied Statistics) **36**, pp. 260–276, 1987.
87. P. Cortez, A. M. Gonçalves Silva, Using data mining to predict secondary school student performance, In Proceedings of 5th FUture BUsiness TEChnology Conference (FUBUTEC 2008) (pp. 5–12). EUROSIS-ETI, 2008.
88. I-Cheng Yeh, King-Jang Yang, Tao-Ming Ting, Knowledge discovery on RFM model using Bernoulli sequence, Expert Systems with Applications **36**, pp. 5866-5871, 2009.
89. Jeyasingh, S., & Veluchamy, M. (2017). Modified bat algorithm for feature selection with the wisconsin diagnosis breast cancer (WDBC) dataset. Asian Pacific journal of cancer prevention: APJCP, 18(5), 1257.
90. Alshayeji, M. H., Ellethy, H., & Gupta, R. (2022). Computer-aided detection of breast cancer on the Wisconsin dataset: An artificial neural networks approach. Biomedical signal processing and control, 71, 103141.
91. M. Raymer, T.E. Doom, L.A. Kuhn, W.F. Punch, Knowledge discovery in medical and biological datasets using a hybrid Bayes classifier/evolutionary algorithm. IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society, **33** , pp. 802-813, 2003.
92. P. Zhong, M. Fukushima, Regularized nonsmooth Newton method for multi-class support vector machines, Optimization Methods and Software **22**, pp. 225-236, 2007.
93. R. G. Andrzejak, K. Lehnertz, F.Mormann, C. Rieke, P. David, and C. E. Elger, "Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: dependence on recording region and brain state," Physical Review E, vol. 64, no. 6, Article ID 061907, 8 pages, 2001.
94. A. T. Tzallas, M. G. Tsipouras, and D. I. Fotiadis, "Automatic Seizure Detection Based on Time-Frequency Analysis and Artificial Neural Networks," Computational Intelligence and Neuroscience, vol. 2007, Article ID 80510, 13 pages, 2007. doi:10.1155/2007/80510

95. M. Koivisto, K. Sood, Exact Bayesian Structure Discovery in Bayesian Networks, The Journal of Machine Learning Research **5**, pp. 549–573, 2004.
96. Nash, W.J.; Sellers, T.L.; Talbot, S.R.; Cawthor, A.J.; Ford, W.B. The Population Biology of Abalone (_Haliotis_ species) in Tasmania. I. Blacklip Abalone (_H. rubra_) from the North Coast and Islands of Bass Strait, Sea Fisheries Division; Technical Report No. 48; Department of Primary Industry and Fisheries, Tasmania: Hobart, Australia, 1994; ISSN 1034-3288
97. Brooks, T.F.; Pope, D.S.; Marcolini, A.M. Airfoil Self-Noise and Prediction. Technical Report, NASA RP-1218. July 1989. Available online: https://ntrs.nasa.gov/citations/19890016302 (accessed on 14 November 2024).
98. I.Cheng Yeh, Modeling of strength of high performance concrete using artificial neural networks, Cement and Concrete Research. **28**, pp. 1797-1808, 1998.
99. D. Harrison and D.L. Rubinfeld, Hedonic prices and the demand for clean ai, J. Environ. Economics & Management **5**, pp. 81-102, 1978.
100. Simonoff, J.S. Smooting Methods in Statistics; Springer: Berlin/Heidelberg, Germany, 1996.
101. Mackowiak, P.A.; Wasserman, S.S.; Levine, M.M. A critical appraisal of 98.6 degrees f, the upper limit of the normal body temperature, and other legacies of Carl Reinhold August Wunderlich. J. Am. Med. Assoc. 1992, 268, 1578–1580.
102. Gropp, W.; Lusk, E.; Doss, N.; Skjellum, A. A high-performance, portable implementation of the MPI message passing interface standard. Parallel Comput. 1996, 22, 789–828.
103. Chandra, R. Parallel Programming in OpenMP; Morgan Kaufmann: Cambridge, MA, USA, 2001.