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, applied on a wide series of classification and regression problems that was proposed in various research topics 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 lowest value 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 produced software was tested for its effectiveness on a wide series of datasets. 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

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]. Commonly these neural networks are formed using the following mathematical expression:

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

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 so - called centers of the network.
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 output value $\phi(x)$ uses only the distance of vectors x and c.

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:

RBF neural networks have been extensively used and studied in the recent 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 classification 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].

Furthermore, a discussion on the kernel widths of RBF networks is provided in the work of Benoudjit et al [29]. Paetz proposed a method [30] that can reduce the number of neurons in RBF networks with dynamic decay adjustment, in order to increase the generalization abilities of these networks. Yu et al. in their work [31] proposed an incremental design technique for RBF networks in order to estimate the optimal architecture of these networks. Moreover, Alexandridis et al. [32] proposed the incorporation of the Particle Swarm Optimization method to effectively estimate the weights of RBF networks. Also, Neruda et al [33] provided a detailed comparison of methods used to estimate the parameters of RBF networks. Additionally, due to the widespread usage of parallel programming techniques, a series of techniques that exploit parallel computing units have been suggested in recent years for RBF network training [34,35].

In most cases, the set of the parameters of the model is estimated 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 [36]. In the second phase, the set of weights $\vec{w}$ 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 calculate these parameters with the classical calculation method, it is necessary to solve a system of equations, but in many cases the solution of such a system leads to numerical problems as determinants appear with a value quite close to zero. 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 [37]. 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 contribution of the proposed technique and the implemented software can be summarized as follows:

1. The method can efficiently construct the structure of RBF networks and achieve optimal adjustment of their parameters.
2. The resulting networks do not have the numerical problems caused by the traditional training technique of RBF networks.
3. The created software provides an easy user interface for performing experiments and can be installed on almost any operating system.

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 incorporated in the conducted experiments are listed followed by the experimental results, in section 4 a discussion on the experimental results is provided and finally in section 5 a series of conclusions is presented.

2. Materials and Methods

The original training method of RBF networks is presented here followed by the detail presentation of the Grammatical Evolution procedure. Afterwards, the proposed method is fully described accompanied by a full working example of the proposed software.

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 calculate 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 [38] with the assistance of a genetic algorithm that utilizes integer chromosomes. These chromosomes represent rule numbers from the underlying grammar and the method has been used successfully in a series of cases, such as: trigonometric problems [39], production of music [40], construction of neural networks [41,42], video games [44,45], credit classification [46], network security [47] etc. Furthermore, many researchers have developed and published a series of software regarding Grammatical Evolution, such as the GEVA [48] which suggests a GUI environment, a

statistical software written in R named gramEvol [50], a Matlab toolbox named GeLab [51], the GenClass software used in classification problems [52], the QFc software [53] used to construct artificial features etc.

The proposed procedure constructs the structure of an RBF neural network using Grammatical Evolution and a BNF grammar and an appropriately modified Genetic Algorithm [54,55] that guides the course of the procedure. The grammars used in the Grammatical Evolution procedure are BNF grammars expressed as sets $G = (N, T, S, P)$ where

- The symbol N stands for the set of non-terminal symbols.
- The symbol T represents the set of terminal symbols.
- S is a non - terminal symbol, used as the start symbol of the grammar.
- P is the set of production rules that are used to produce terminal symbols from non - terminal symbols.

For each chromosome, Grammatical Evolution initiates from the symbol S and through a series of production steps it creates programs with terminal symbols by substituting non-terminal symbols with the right hand of the selected production rule. The selection of the production rule is accomplished in two steps:

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

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

Figure 1. The BNF grammar that is used by the proposed method to create RBF networks. Every non - terminal symbol is associated with a series of production rules that will produce terminal symbols. The numbers in the parentheses denote the sequential number of each production rule, used by the Grammatical Evolution technique during the creation of RBF networks.

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

1. **Initialization Step**.
   (a) **Set** the number of chromosomes N_c and the 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** the chromosomes as sets of positive random integers.
(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. **Compute** the fitness f_i for the train set as:

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

      The set $(\overrightarrow{x_j}, t_j)$, $j = 1, ..., M$ represents 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 production of the new chromosomes is conducted using the one -point crossover procedure. An example of this process is graphically outlined 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 replaced by a new randomly produced element.
   (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:

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

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

The steps of the proposed method are also outlined as a flowchart in Figure 3.

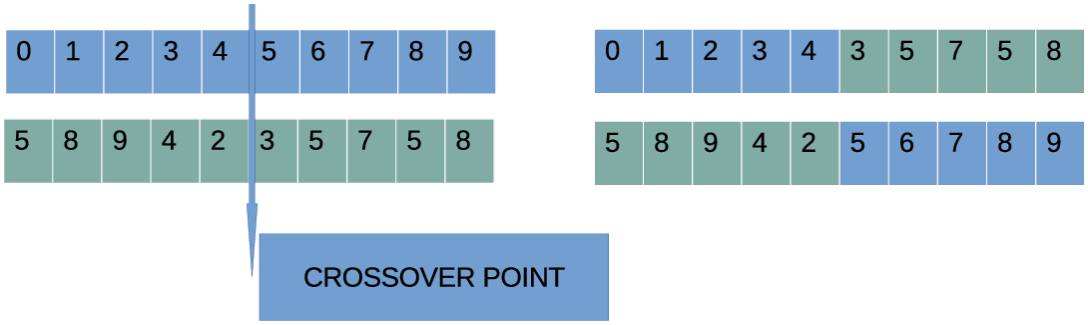

Figure 2. A graphical example of the one - point crossover. This procedure is used in the Grammatical Evolution method to produce new chromosomes.

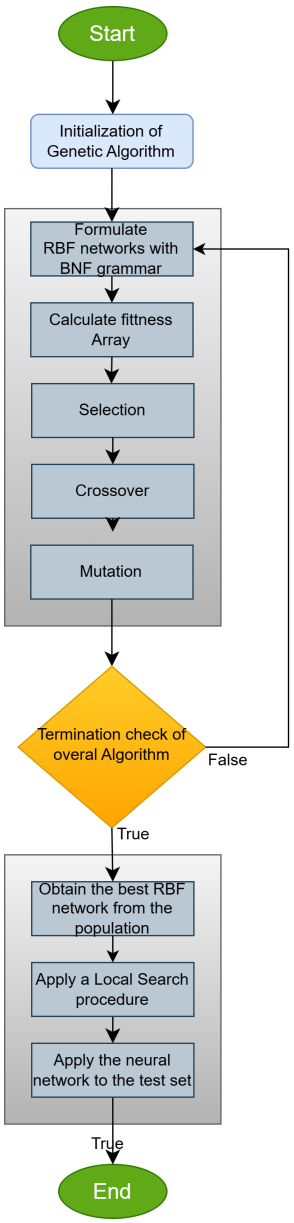

Figure 3. The steps of the proposed method as a flowchart.

2.3. Installation procedure

The software was coded entirely in ANSI C++ and the software is available freely from the following URL: https://github.com/itsoulos/RbfCon (accessed on 14 November 2024). The user should install the QT Library, which is accessible 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 RBFCON.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 4. The first number denoted as D in this figure determines the dimension of the input dataset and the second number denoted as M represents 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> This integer parameter denotes the number of chromosomes in the genetic algorithm (parameter N_c of the algorithm). 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 (parameter N_g of the current algorithm). The original value 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 [43], inverse eigenvalue problems [56], seismic problems [57], training of deep neural networks [58] etc. Also, the a series of modifications have been proposed that take advantage of modern parallel computing systems [59–61].

(b) *bfgs*. The BFGS variant of Powell was used here [63], when this option is enabled.
(c) *adam*. This option denotes the application of the Adam optimizer [64] as the local optimization algorithm.
(d) *gradient*. This option represents the usage of the Gradient Descent method [65] 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.

D
M

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

Figure 4. The format for the used datasets.

2.5. Example of execution

As a full working example, consider the Wdbc dataset [66] 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.

The basic elements that the software requires when initializing the process are essentially the file containing the training data as well as the file containing the test data. Both of these files must have the proposed structure but also the same number of patterns. The user

can change other elements of the algorithm, such as the number of chromosomes, through a series of parameters, but this is not required. At the end, the software will display the average training error as well as the average error in the test set as well as the average classification error.

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 are used from the following websites:

1. The UCI repository, https://archive.ics.uci.edu/(accessed on 14 November 2024)[67]
2. The Keel repository, https://sci2s.ugr.es/keel/datasets.php(accessed on 14 November 2024)[68].
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 contains the used classification datasets for the the conducted experiments:

1. **Appendicitis**, a medical dataset that was studied in [69,70]. This problem has 2 distinct classes.
2. **Alcohol**, a dataset related to alcohol consumption [71]. This dataset contains 4 distinct classes.
3. **Australian**, an economic dataset [72]. The number of classes for this dataset is 2.
4. **Bands**, related to problems occurred in printing [73] and it has two classes.
5. **Cleveland**, a medical dataset proposed in a series of research papers [74,75]. This dataset has 5 distinct classes.
6. **Dermatology**, which is also a medical dataset originated in [76] and it has 6 classes.
7. **Ecoli**, that is related to problems regarding proteins [77] 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 [78]. This dataset has 3 classes.
11. **Heart**, a medical dataset about heart diseases [79]. This dataset has 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 [80].
15. **Ionosphere**, used for measurements from the ionosphere [81,82] and it has two classes.
16. **Liverdisorder**, which is a medical dataset [83,84] with two classes.
17. **Lymography** [85], that has 4 classes.
18. **Magic**, this dataset contains generated data to simulate registration of high energy gamma particles [86] and it has two classes.
19. **Mammographic**, a medical dataset related to the presence of breast cancer [87].
20. **Parkinsons**, a dataset used for the detection of Parkinson's disease [88,89].
21. **Pima**, a medical dataset that was useful for the detection of diabetes [90].
22. **Popfailures**, a dataset contains climate measurements [91] and it has two classes.
23. **Regions2**, a medical dataset that contains measurements from liver biopsy images [92] and it has five classes.
24. **Ring**, which is a problem with 20 dimensions and two classes, related to a series of multivariate normal distributions.
25. **Saheart**, a medical dataset related to heart diseases with 2 classes [93].
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 [94].
30. **Tae**, this dataset consist of evaluations of teaching performance and it has 3 classes.
31. **Transfusion**, which is a medical dataset [95].
32. **Wdbc**, a medical dataset used to detect the presence of breast cancer[96,97].
33. **Wine**, a dataset contains measurements about the quality of wines [98,99], with 3 distinct classes.
34. **EEG** dataset, which is a medical dataset about EEG measurements studied in a series of papers [100,101]. 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 [102] and it contains seven classes.

The next list provided the regression datasets that was incorporated during 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 [104] 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 [105].
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 [106] 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 [107] 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 [108].
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 denotes the train 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 technique of ten - fold cross validation was incorporated for the validation of the experimental results. The experiments were executed on an AMD Ryzen 5950X with 128GB of RAM, running as the operating system the Debian Linux. The values for the parameters of the algorithm are presented in Table 1.

Table 1. The values for the experimental parameters.

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

This particular set of parameters has been used in a multitude of scientific publications and can be considered as a compromise between the efficiency of an evolutionary process such as the one proposed in this work and the speed necessary to extract the results of its application within a reasonable time.

The experimental results from the application of various method on the classification datasets are shown in Table 2 and for the regression datasets in 3. For all tables the following notation was used:

1. The column MLP(ADAM) represents the usage of the ADAM optimization method [64] to train an artificial neural network with $k = 10$ processing nodes.
2. The column MLP(BFGS) stands for the usage of the BFGS optimization method [63] in the training process of an artificial neural network with $k = 10$ processing nodes.
3. The column RBF denotes the usage 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.
4. The column CRBF stands for the utilization of the current software using the parameters of Table 1 and as local search procedure the *none* option was selected.
5. The column CRBF(LBFGS) denotes the application of the current method using as local search procedure the L-Bfgs method.
6. The column CRBF(BFGS) stands for the application of the current work using as local search procedure the Bfgs method.
7. The row denoted as AVERAGE, outlines the average classification or regression error for all datasets in the corresponding table.
8. The row denoted as STDEV depicts the standard deviation for all datasets in the corresponding table.

Table 2. Experimental results for the classification datasets using the mentioned machine learning models.

DATASET	MLP(ADAM)	MLP(BFGS)	RBF	CRBF	CRBF(LBFGS)	CRBF(BFGS)
APPENDICITIS	16.50%	18.00%	12.23%	13.60%	14.10%	13.60%
ALCOHOL	57.78%	41.50%	49.38%	51.24%	49.32%	45.11%
AUSTRALIAN	35.65%	38.13%	34.89%	14.14%	14.26%	14.23%
BANDS	36.92%	36.67%	37.17%	35.75%	36.58%	36.03%
CLEVELAND	67.55%	77.55%	67.10%	49.14%	49.52%	50.28%
DERMATOLOGY	26.14%	52.92%	62.34%	45.20%	38.43%	36.66%
ECOLI	64.43%	69.52%	59.48%	54.18%	54.39%	53.03%
FERT	23.98%	23.20%	15.00%	15.20%	15.50%	15.90%
HABERMAN	29.00%	29.34%	25.10%	26.27%	27.07%	26.40%
HAYES-ROTH	59.70%	37.33%	64.36%	34.54%	39.00%	36.76%
HEART	38.53%	39.44%	31.20%	17.22%	17.44%	16.96%
HEARTATTACK	45.55%	46.67%	29.00%	22.60%	21.83%	21.53%
HEPATITIS	68.13%	72.47%	64.63%	54.25%	47.50%	48.75%
HOUSEVOTES	7.48%	7.13%	6.13%	3.05%	3.65%	3.05%
IONOSPHERE	16.64%	15.29%	16.22%	14.32%	13.40%	12.37%
LIVERDISORDER	41.53%	42.59%	30.84%	32.21%	31.71%	31.35%
LYMOGRAPHY	39.79%	35.43%	25.50%	26.36%	25.71%	21.49%
MAGIC	40.55%	17.30%	21.28%	22.18%	19.62%	20.35%
MAMMOGRAPHIC	46.25%	17.24%	21.38%	17.67%	19.02%	17.30%
PARKINSONS	24.06%	27.58%	17.41%	12.79%	13.37%	12.47%
PIMA	34.85%	35.59%	25.78%	24.13%	24.90%	24.15%
POPFAILURES	5.18%	5.24%	7.04%	6.98%	6.94%	6.80%
REGIONS2	29.85%	36.28%	38.29%	26.34%	26.81%	26.55%
RING	28.80%	29.24%	21.67%	11.08%	10.13%	10.33%
SAHEART	34.04%	37.48%	32.19%	29.52%	29.28%	29.19%
SPAMBASE	48.05%	18.16%	29.35%	16.95%	15.53%	15.60%
SPIRAL	47.67%	47.99%	44.87%	42.14%	43.88%	43.05%
STATHEART	44.04%	39.65%	31.36%	19.22%	19.15%	18.19%
STUDENT	5.13%	7.14%	5.49%	7.63%	6.33%	4.32%
TAE	60.20%	51.58%	60.02%	56.73%	56.33%	56.40%
TRANSFUSION	25.68%	25.84%	26.41%	25.15%	24.70%	24.36%
WDBC	35.35%	29.91%	7.27%	6.77%	6.52%	6.36%
WINE	29.40%	59.71%	31.41%	11.00%	11.65%	10.71%
Z_F_S	47.81%	39.37%	13.16%	11.13%	11.47%	10.57%
Z_O_N_F_S	78.79%	65.67%	48.70%	52.34%	49.32%	46.22%
ZO_NF_S	47.43%	43.04%	9.02%	11.08%	11.18%	8.90%
ZONF_S	11.99%	15.62%	4.03%	4.14%	3.94%	3.58%
ZOO	14.13%	10.70%	21.93%	11.60%	9.00%	10.90%
AVERAGE	**37.23%**	**35.36%**	**30.23%**	**24.63%**	**24.17%**	**23.42%**
STDEV	**18.25%**	**18.36%**	**18.41%**	**15.92%**	**15.46%**	**15.25%**

Table 3. Experimental results for the regression datasets using the series of mentioned machine learning models.

DATASET	MLP(ADAM)	MLP(BFGS)	RBF	CRBF	CRBF(LBFGS)	CRBF(BFGS)
ABALONE	4.30	5.69	7.37	6.14	5.35	5.32
AIRFOIL	0.005	0.003	0.27	0.004	0.004	0.002
AUTO	70.84	60.97	17.87	11.03	10.03	9.45
BK	0.025	0.28	0.02	0.02	0.02	0.03
BL	0.62	2.55	0.013	0.04	0.024	0.01
BASEBALL	77.90	119.63	93.02	66.74	65.80	64.52
CONCRETE	0.078	0.066	0.011	0.012	0.010	0.009
DEE	0.63	2.36	0.17	0.23	0.22	0.20
FA	0.048	0.43	0.015	0.013	0.014	0.011
HO	0.035	0.62	0.03	0.013	0.015	0.012
HOUSING	81.00	97.38	57.68	20.60	19.02	18.40
LASER	0.03	0.015	0.03	0.06	0.05	0.03
LW	0.028	2.98	0.03	0.011	0.011	0.011
MB	0.06	0.129	5.43	0.055	0.06	0.12
MORTGAGE	9.24	8.23	1.45	0.165	0.18	0.074
NT	0.006	0.129	13.97	0.006	0.006	0.006
PLASTIC	11.71	20.32	8.62	3.58	2.86	2.43
PL	0.32	0.58	2.118	0.064	0.026	0.024
QUAKE	0.117	0.29	0.07	0.036	0.036	0.036
SN	0.026	0.4	0.027	0.025	0.026	0.025
STOCK	180.89	302.43	12.23	8.31	6.90	6.25
TREASURY	11.16	9.91	2.02	0.0027	0.12	0.10
TZ	0.43	0.22	0.036	0.036	0.036	0.035
AVERAGE	**19.54**	**27.64**	**9.67**	**5.10**	**4.82**	**4.66**
STDEV	**43.67**	**68.11**	**22.01**	**14.34**	**14.05**	**13.76**

4. Discussion

In Table 2, for the APPENDICITIS dataset, the low error rates of the RBF and CRBF models (12.23% and 13.60%, respectively) indicate that Radial Basis Function-based models are well-suited for this problem. This performance may be linked to their ability to handle non-linear relationships in the data. The higher error rates of the MLP models (above 16%) suggest they might require more tuning or larger datasets for optimal performance. In the ALCOHOL dataset, error rates range from 41.50% (MLP(BFGS)) to 57.78% (MLP(ADAM)). This wide variation implies that the dataset's features are particularly challenging for most models, likely due to noise or high dimensionality. The intermediate error rates of the CRBF model (49.38%) show that while RBF-based models can handle some complexity, further adjustments are needed to make them competitive. In the AUSTRALIAN dataset, CRBF and its variations achieve exceptionally low error rates (14%), highlighting their strong adaptation to the dataset's characteristics. In contrast, MLP(ADAM) and MLP(BFGS) exhibit significantly higher errors (35%-38%), demonstrating their struggles to perform effectively on this problem. The CLEVELAND dataset, on the other hand, exemplifies complexity. MLP models show error rates as high as 77.55%, while CRBF models reduce the error to about 49%. This gap underscores the superiority of RBF-based approaches for data with complex, non-linear structures. Overall, the analysis reveals that CRBF models have a clear advantage in datasets with non-linear relationships and lower dimensionality. While MLP models are more flexible, they seem to require extensive parameter tuning to achieve comparable performance. Employing an RBF model with appropriate customization may often be the optimal choice for many of the examined problems.

In Table 3, for the ABALONE dataset, MLP models show higher values, such as 4.3 for MLP(ADAM) and 5.69 for MLP(BFGS), while CRBF models perform better. The best result comes from CRBF(BFGS), with a value of 5.32, indicating that CRBF models adapt

better to this dataset. In the AIRFOIL dataset, CRBF-based models exhibit exceptional performance, with CRBF(BFGS) achieving the best value of 0.002. In contrast, MLP models show higher values, such as 0.005 for MLP(ADAM), demonstrating that RBF-based models are more accurate in this case. In the AUTO dataset, there is a clear superiority of CRBF models. CRBF(BFGS) achieves the best value of 9.45, while MLP models record significantly higher values, such as 70.84 for MLP(ADAM). This indicates that MLP struggles to adapt effectively to this problem. In the BASEBALL dataset, CRBF models achieve lower error values, with the best result being 64.52 for CRBF(BFGS). In comparison, MLP(ADAM) shows a much higher value of 77.9, highlighting the advantage of CRBF models in this dataset. In the HOUSING dataset, the differences are even more pronounced. CRBF(BFGS) achieves the best performance with a value of 18.4, while MLP(ADAM) records 81, and MLP(BFGS) reaches 97.38. This stark contrast emphasizes the inability of MLP models to perform effectively on this dataset. In the PLASTIC dataset, CRBF models also deliver excellent performance. CRBF(BFGS) achieves the best result at 2.43, while MLP models show much higher values, such as 11.71 for MLP(ADAM). Overall, CRBF-based models demonstrate significantly lower error values across most datasets, indicating their ability to adapt to various data types. MLP models, although flexible, struggle in many cases, possibly due to the need for extensive parameter tuning or limitations in generalization. The overall analysis suggests that CRBF models are more reliable for regression problems, especially when the datasets involve non-linear relationships or high complexity.

Also, in Figures 5, 6 a statistical comparison between all method is depicted.

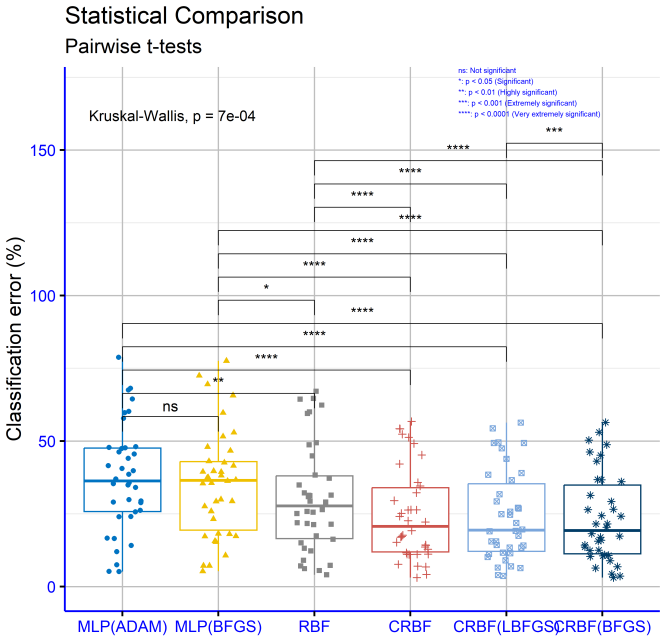

Figure 5. Statistical comparison for the used methods and the classification datasets.

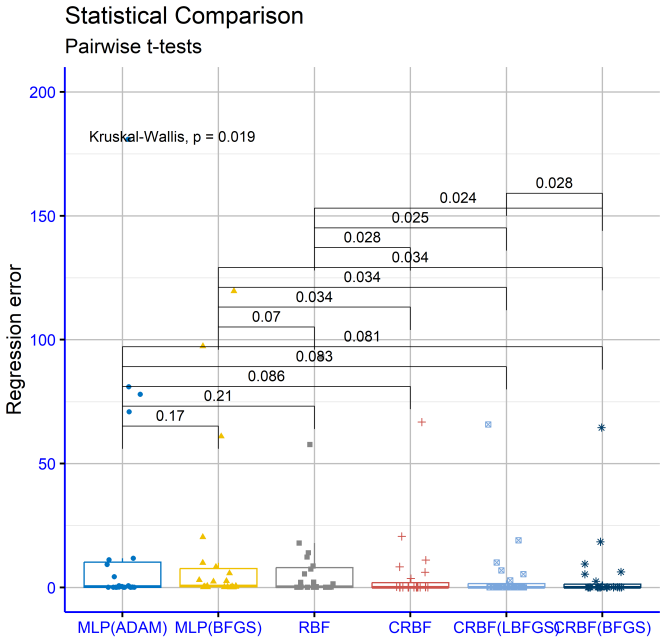

Figure 6. Statistical test for the regression datasets and the used methods.

In Figure 5, the Kruskal-Wallis test revealed statistically significant differences among the methods overall, with a general p-value of 0.0007, indicating that at least one method differs significantly from the others in terms of classification error rates. Pairwise comparisons provide further insights into the differences between specific methods. The comparison between MLP(ADAM) and MLP(BFGS) showed no statistically significant difference (p = 0.34), suggesting that the two methods perform similarly. In contrast, the comparison between MLP(ADAM) and RBF revealed a statistically significant difference (p = 0.0026), indicating that RBF achieves distinct error rates compared to MLP(ADAM). The comparison of MLP(ADAM) with CRBF showed a highly significant difference (p = 1.8e-07), highlighting the superiority of CRBF over MLP(ADAM). Similarly, the comparison between MLP(ADAM) and CRBF(LBFGS) demonstrated even greater statistical significance, with p = 4.6e-08, confirming the advantage of CRBF(LBFGS). Additionally, the comparison of MLP(ADAM) with CRBF(BFGS) exhibited extremely high statistical significance (p = 1.8e-08), favoring CRBF(BFGS). The subsequent comparisons of MLP(BFGS) with other methods also revealed statistically significant differences. The comparison between MLP(BFGS) and RBF yielded p = 0.011, indicating a notable difference, while the comparison of MLP(BFGS) with CRBF had p = 3.8e-06, clearly demonstrating the superiority of CRBF. Comparisons of MLP(BFGS) with CRBF(LBFGS) and CRBF(BFGS) resulted in p-values of 1.9e-06 and 5.3e-07, respectively, confirming the statistical significance of the differences. Furthermore, comparisons of RBF with CRBF, CRBF(LBFGS), and CRBF(BFGS) all showed statistically significant differences, with p-values of 9.8e-05, 3.9e-05, and 4.3e-06, respectively. Finally, the comparison between CRBF(LBFGS) and CRBF(BFGS) also revealed a statistically significant difference, with p = 0.0008. Overall, the results demonstrate clear performance differences among the methods, with CRBF consistently achieving superior results.

In Figure 6, The Kruskal-Wallis test revealed statistically significant differences among the methods on regression datasets, with an overall Kruskal-Wallis p-value of 0.0019. This indicates that at least one method performs significantly differently from the others. However, pairwise comparisons show mixed results. The comparison between MLP(ADAM) and MLP(BFGS) did not show a statistically significant difference (p = 0.17), suggesting similar performance between these two methods. Similarly, the comparison between MLP(ADAM) and RBF was not significant (p = 0.21). Comparisons of MLP(ADAM) with CRBF, CRBF(LBFGS), and CRBF(BFGS) yielded p-values of 0.086, 0.083, and 0.081, re-

spectively, indicating trends toward differences but without statistical significance. The comparisons involving MLP(BFGS) against other methods yielded more pronounced results. Specifically, differences with CRBF, CRBF(LBFGS), and CRBF(BFGS) were statistically significant, with p = 0.034 for all three cases, indicating the superiority of the CRBF methods over MLP(BFGS). For the comparisons between RBF and the CRBF methods, statistically significant differences were observed. The comparison between RBF and CRBF had a p-value of 0.028, while comparisons with CRBF(LBFGS) and CRBF(BFGS) had p-values of 0.025 and 0.024, respectively, confirming the superior performance of the CRBF methods. Finally, the comparison between CRBF(LBFGS) and CRBF(BFGS) revealed a statistically significant difference, with p = 0.028, highlighting a notable distinction in the performance of these two methods. Overall, the results suggest that while MLP methods perform similarly, CRBF methods consistently achieve better results, with statistically significant differences observed in most comparisons against the other methods.

Summarizing the above results, one can see that the proposed technique for building and training RBF networks improves the performance of these networks by more than 20% on average on classification data and by more than 50% on regression data. In all datasets, the proposed method brought about a significant reduction in classification error in 80% of cases and in many datasets the reduction in classification or regression error was extremely significant.

5. Conclusions

A novel method that constructs the architecture of RBF neural networks with assistance was described in the present manuscript, 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 improving the parameters of the RBF network with the application of some local optimization procedures. The experimental results 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 utilizes Genetic Algorithms, parallel optimization procedures, such as MPI [109] or OpenMP [110] 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. N. Benoudjit, M. Verleysen, On the Kernel Widths in Radial-Basis Function Networks, Neural Processing Letters **18**, pp. 139–154, 2003.
30. J. Paetz, Reducing the number of neurons in radial basis function networks with dynamic decay adjustment, Neurocomputing **62**, pp. 79-91, 2004.
31. H. Yu, P. D. Reiner, T. Xie, T. Bartczak, B. M. Wilamowski, An Incremental Design of Radial Basis Function Networks, IEEE Transactions on Neural Networks and Learning Systems **25**, pp. 1793-1803, 2014.
32. A. Alexandridis, E. Chondrodima, H. Sarimveis, Cooperative learning for radial basis function networks using particle swarm optimization, Applied Soft Computing **49**, pp. 485-497, 2016.
33. R. Neruda, P. Kudova, Learning methods for radial basis function networks, Future Generation Computer Systems **21**, pp. 1131–1142, 2005.
34. R. Yokota, L.A. Barba, M. G. Knepley, PetRBF — A parallel O(N) algorithm for radial basis function interpolation with Gaussians, Computer Methods in Applied Mechanics and Engineering **199**, pp. 1793-1804, 2010.
35. C. Lu, N. Ma, Z. Wang, Fault detection for hydraulic pump based on chaotic parallel RBF network, EURASIP J. Adv. Signal Process. **2011**, 49, 2011.
36. 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.
37. M. O'Neill, C. Ryan, Grammatical evolution, IEEE Trans. Evol. Comput. **5,**pp. 349–358, 2001.
38. 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.
39. C. Ryan, M. O'Neill, J.J. Collins, Grammatical evolution: Solving trigonometric identities, proceedings of Mendel. Vol. 98. 1998.
40. 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.
41. 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.
42. K. Soltanian, A. Ebnenasir, M. Afsharchi, Modular Grammatical Evolution for the Generation of Artificial Neural Networks, Evolutionary Computation **30**, pp 291–327, 2022.
43. 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.
44. 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.
45. 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.
46. A. Brabazon, M. O'Neill, Credit classification using grammatical evolution, Informatica **30.3**, 2006.
47. 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.
48. 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.
49. 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.
50. F. Noorian, A.M. de Silva, P.H.W. Leong, gramEvol: Grammatical Evolution in R, Journal of Statistical Software **71**, pp. 1–26, 2016.
51. 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
52. N.Anastasopoulos, I.G. Tsoulos, A. Tzallas, GenClass: A parallel tool for data classification based on Grammatical Evolution, SoftwareX **16**, 100830, 2021.
53. I.G. Tsoulos, QFC: A Parallel Software Tool for Feature Construction, Based on Grammatical Evolution, Algorithms **15**, 295, 2022.
54. D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Publishing Company, Reading, Massachussets, 1989.

55. Z. Michaelewicz, Genetic Algorithms + Data Structures = Evolution Programs. Springer - Verlag, Berlin, 1996.
56. 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.
57. Y. Rao, Y. Wang, Seismic waveform tomography with shot-encoding using a restarted L-BFGS algorithm, Scientific Reports **7**, pp. 1-9, 2017.
58. 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
59. Y. Fei, G. Rong, B. Wang, W. Wang, Parallel L-BFGS-B algorithm on GPU, Computers & Graphics **40**, pp. 1-9, 2014.
60. 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.
61. M.M. Najafabadi, T.M. Khoshgoftaar, F. Villanustre et al, Large-scale distributed L-BFGS, J Big Data **4**, 22, 2017.
62. J.L. Morales, A numerical study of limited memory BFGS methods, Applied Mathematics Letters **15**, pp. 481-487, 2002.
63. M.J.D Powell, A Tolerant Algorithm for Linearly Constrained Optimization Calculations, Mathematical Programming **45**, pp. 547-566, 1989.
64. 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.
65. Amari, S. I. (1993). Backpropagation and stochastic gradient descent method. Neurocomputing, 5(4-5), 185-196.
66. 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.
67. M. Kelly, R. Longjohn, K. Nottingham, The UCI Machine Learning Repository, https://archive.ics.uci.edu.
68. 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.
69. 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.
70. M. Wang, Y.Y. Zhang, F. Min, Active learning through multi-standard optimization, IEEE Access **7**, pp. 56772–56784, 2019.
71. 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.
72. J.R. Quinlan, Simplifying Decision Trees. International Journal of Man-Machine Studies **27**, pp. 221-234, 1987.
73. B. Evans, D. Fisher, Overcoming process delays with decision tree induction. IEEE Expert **9**, pp. 60-66, 1994.
74. 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.
75. R. Setiono , W.K. Leow, FERNN: An Algorithm for Fast Extraction of Rules from Neural Networks, Applied Intelligence **12**, pp. 15-25, 2000.
76. 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.
77. 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.
78. 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.
79. I. Kononenko, E. Šimec, M. Robnik-Šikonja, Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF, Applied Intelligence **7**, pp. 39–55, 1997
80. 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.
81. J.G. Dy , C.E. Brodley, Feature Selection for Unsupervised Learning, The Journal of Machine Learning Research **5**, pp 845–889, 2004.
82. S. J. Perantonis, V. Virvilis, Input Feature Extraction for Multilayered Perceptrons Using Supervised Principal Component Analysis, Neural Processing Letters **10**, pp 243–252, 1999.
83. J. Garcke, M. Griebel, Classification with sparse grids using simplicial basis functions, Intell. Data Anal. **6**, pp. 483-502, 2002.
84. J. Mcdermott, R.S. Forsyth, Diagnosing a disorder in a classification benchmark, Pattern Recognition Letters **73**, pp. 41-43, 2016.
85. 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.
86. Heck, D., Knapp, J., Capdevielle, J. N., Schatz, G., & Thouw, T. (1998). CORSIKA: A Monte Carlo code to simulate extensive air showers.
87. 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.
88. 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.

89. 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.
90. 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.
91. 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.
92. 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.
93. T. Hastie, R. Tibshirani, Non-parametric logistic and proportional odds regression, JRSS-C (Applied Statistics) **36**, pp. 260–276, 1987.
94. 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.
95. 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.
96. 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.
97. 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.
98. 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.
99. P. Zhong, M. Fukushima, Regularized nonsmooth Newton method for multi-class support vector machines, Optimization Methods and Software **22**, pp. 225-236, 2007.
100. 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.
101. 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
102. M. Koivisto, K. Sood, Exact Bayesian Structure Discovery in Bayesian Networks, The Journal of Machine Learning Research **5**, pp. 549–573, 2004.
103. 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
104. 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).
105. I.Cheng Yeh, Modeling of strength of high performance concrete using artificial neural networks, Cement and Concrete Research. **28**, pp. 1797-1808, 1998.
106. D. Harrison and D.L. Rubinfeld, Hedonic prices and the demand for clean ai, J. Environ. Economics & Management **5**, pp. 81-102, 1978.
107. Simonoff, J.S. Smooting Methods in Statistics; Springer: Berlin/Heidelberg, Germany, 1996.
108. 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.
109. 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.
110. Chandra, R. Parallel Programming in OpenMP; Morgan Kaufmann: Cambridge, MA, USA, 2001.