AI

MDPI

*Article*

# Adapting the Parameters of RBF Networks Using Grammatical Evolution

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

Department of Informatics and Telecommunications, University of Ioannina, 45110 Ioannina, Greece; tzallas@uoi.gr (A.T.); ekarvounis@uoi.gr (E.K.)
* Correspondence: itsoulos@uoi.gr

**Abstract:** Radial basis function networks are widely used in a multitude of applications in various scientific areas in both classification and data fitting problems. These networks deal with the above problems by adjusting their parameters through various optimization techniques. However, an important issue to address is the need to locate a satisfactory interval for the parameters of a network before adjusting these parameters. This paper proposes a two-stage method. In the first stage, via the incorporation of grammatical evolution, rules are generated to create the optimal value interval of the network parameters. During the second stage of the technique, the mentioned parameters are fine-tuned with a genetic algorithm. The current work was tested on a number of datasets from the recent literature and found to reduce the classification or data fitting error by over 40% on most datasets. In addition, the proposed method appears in the experiments to be robust, as the fluctuation of the number of network parameters does not significantly affect its performance.

**Keywords:** neural networks; genetic algorithms; genetic programming; grammatical evolution

check for updates

**Citation:** Tsoulos, I.G.; Tzallas, A.; Karvounis, E. Adapting the Parameters of RBF Networks Using Grammatical Evolution. *AI* **2023**, *4*, 1059–1078. https://doi.org/10.3390/ai4040054

Academic Editors: Demos T. Tsahalis and Arslan Munir

Received: 20 October 2023
Revised: 18 November 2023
Accepted: 7 December 2023
Published: 11 December 2023



## 1. Introduction

Many practical problems of the modern world can be thought of as data fitting problems, such as, for example, problems that appear in physics [1,2], problems related to chemistry [3,4], economic problems [5,6], medicine problems [7,8], etc. Radial basis function (RBF) networks are commonly used machine learning tools to handle problems of this nature [9,10]. Usually, an RBF network is expressed using the following equation:

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

where the symbols in the equation are defined as follows:

1. The element $\overrightarrow{x}$ represents the input pattern from the dataset describing the problem. For the rest of this paper, the notation $d$ will be used to represent the number of elements in $\overrightarrow{x}$.
2. The parameter $k$ denotes the number of weights used to train the RBF network, and the associated vector of weights is denoted as $\overrightarrow{w}$.
3. The vectors $\overrightarrow{c_i}$, $i = 1, \ldots, k$ stand for the centers of the model.
4. The value $y(\overrightarrow{x})$ represents the value of the network for the given pattern $\overrightarrow{x}$.

The $\phi(x)$ function, in most cases, represents the Gaussian function given by:

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

The main advantages of RBF networks are as follows:

1. They have a simpler structure than other models used in machine learning, such as multilayer perceptron neural networks (MLPs) [11], since they have only one processing layer and therefore have faster training techniques, as well as faster response times.
2. They can be used to efficiently approximate any continuous function [12].

RBF networks have been applied in a variety of problems, such as physics problems [13–16], solving differential equations [17–19], robotics problems [20,21], face recognition [22], digital communications [23,24], chemistry problems [25,26], economic problems [27–29], and network security problems [30,31]. Recently, a variety of papers have appeared proposing novel initialization techniques for these networks' parameters [32–34]. Benoudjit et al. [35] discussed the effect of kernel widths on RBF networks. Moreover, Neruda et al. [36] presented a comparison of some learning methods for RBF networks. Additionally, a variety of pruning techniques [37–39] have been suggested in the literature for decreasing the number of parameters. Due to the widespread usage of RBF networks and because considerable computing time is often required for their effective training, in recent years, a series of techniques have been published [40,41] for the exploitation of parallel computing units to adjust their parameters.

In the same direction of research, other machine learning models have been proposed, such as support vector machines (SVM) [42,43] and decision trees [44,45]. Also, Wang et al. suggested an auto-encoder reduction method applied on a series of large datasets [46]. Various methods have been proposed in the same direction, such as the work of Agarwal and Bhanot [47], which proposed the use of the ABC algorithm [48] and the incorporation of the Firefly algorithm [49] to adapt an RBF network's parameters. Furthermore, Gyamfi et al. [50] recently proposed a differential RBF network that incorporated partial differential equations, aiming to make the network more robust in the presence of noisy data. Also, Li et al. [51] proposed a multivariate ensemble-based hierarchical linkage strategy (ME-HL) for the evaluation of the system reliability of aeroengine cooling blades.

The parameters of an RBF network can be modified in order to minimize the following loss function, which is called the training error of the network:

$$E(y(x,g)) = \sum_{i=1}^{m} \left(y(\overrightarrow{x}_i, \overrightarrow{g}) - t_i\right)^2 \tag{3}$$

where the parameter $m$ denotes the number of patterns and $t_i$ represent the expected output for pattern $\overrightarrow{x}_i$. The vector $\overrightarrow{g}$ represents the parameter set of the network.

A common method of calculating the parameters in these neural networks uses a technique to calculate the centers of the functions $\phi(x)$, and then the vector of weights $\overrightarrow{w}$ is calculated as a solution of a linear system of equations. Typically, the method used to calculate the centers is the well-known k-means method [52]. In many cases, this way of estimating the parameters leads to over-fitting of the model so that it cannot generalize satisfactorily to unknown data. Furthermore, since there is no range of values for the parameters, there is the possibility that they will take extremely large or extremely small values, with the result that any generalizability of the model is lost. This work suggests a two-phase method to minimize the error of Equation (3). During the first phase, an attempt is made to bound the parameter values to intervals at which the training error is likely to be significantly reduced. The identification of the most promising intervals for the parameters is performed using a technique that utilizes grammatical evolution [53], which collects information from the training data. During the second phase, the parameters can be trained inside the best located range of the first phase using a global optimization method [54,55]. In the proposed approach, the widely used method of genetic algorithm [56–58] was used for the second phase of the process. The main contributions of the suggested approach are as follows:

1. The first phase of the procedure seeks to locate a range of values for the parameters while also reducing the error of the network on the training dataset.
2. The rules grammatical evolution uses in the first phase are simple and can be generalized to any dataset for data classification or fitting.
3. The determination of the value interval is conducted in such a way that it is faster and more efficient to train the parameters with an optimization method during the second phase.
4. After identifying a promising value interval from the first phase, any global optimization method can be used on that value interval to effectively minimize the network training error.

The rest of this paper is divided into the following sections. In Section 2, the proposed method is fully described. Section 3 presents the used datasets and the conducted experiments. Finally, in Section 4, we provide a discussion on the conducted experiments.

## 2. Method Description

This section starts with an extended presentation of the grammatical evolution technique and the grammar that is used to generate partition rules for the parameter set of RBFs. Afterwards, the first phase of the proposed methodology is extensively analyzed, and then the second phase is presented, where a genetic algorithm is applied to the outcome of the first phase.

### *2.1. Grammatical Evolution*

Grammatical evolution is a genetic algorithm, where the chromosomes are integer numbers. Genetic algorithms were initially proposed by John Holland [59] and are biologicaly inspired algorithms. The algorithm starts by forming a population of potential solutions to an optimization problem. These solutions are called chromosomes, and they are gradually altered using the genetic operators of selection, crossover, and mutation [60]. The chromosomes in the grammatical evolution method stand for a series of production rules of any given BNF (Backus–Naur form) grammar [61]. Grammatical evolution has been applied with success in a variety of cases, such as function approximation [62,63], solving equations related to trigonometry [64], the automatic composition of music [65], the construction of neural networks [66,67], producing numeric constraints [68], video games [69,70], the estimation of energy demand [71], combinatorial optimization [72], and cryptography [73]. The BNF grammar can be used to describe the syntax of programming languages, and usually, it is defined as $G = (N, T, S, P)$, where:

- $N$ is a set of the non-terminal symbols. A series of production rules is associated with every non-terminal symbol. The application of these production rules produces series of terminal symbols.
- $T$ stands for the set of terminal symbols.
- $S$ denotes the start symbol of the grammar and $S \in N$.
- $P$ defines the set of production rules. These are rules that follow the following notations: $A \to a$ or $A \to aB$, $A, B \in N$, $a \in T$.

The algorithm begins using the symbol $S$ and gradually creates series of terminal symbols with the assistance of the production rules. The production rules are selected through the following procedure:

- Denote with V the next element form of the current chromosome.
- The next production rule is calculated as: Rule = V mod R. The number R stands for the total number of production rules for the non-terminal symbol that is currently under processing.

Algorithm 1 shows the BNF grammar used by the proposed method. Each non-terminal symbol of the grammar is enclosed in a <> symbol. The numbers that are enclosed in parentheses represent the sequence numbers of production rules for every non-terminal symbol. Every RBF network with $k$ weights is constructed by the following series of parameters:

1. A series of vectors $\overrightarrow{c_i}$, $i = 1, \ldots, k$ that stand for the centers of the model.
2. For every Gaussian unit, an additional parameter $\sigma_i$ is required.
3. The output weight vector $\overrightarrow{w}$.

The number n is the total number of parameters of the problem. In the case of this paper, it is the total number of parameters of the RBF network. For the current work, the number $n$ can be computed using the following formula:

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

**Algorithm 1** The BNF grammar used in the proposed method to produce intervals for the RBF parameters. By using this grammar in the first phase of the current work, the optimal interval of values for the parameters can be identified.

```
S::=<expr>   (0)
<expr> ::=  (<xlist> , <digit>,<digit>)  (0)
           |<expr>,<expr>                (1)
<xlist>::=x1    (0)
           | x2 (1)
           .........
           | xn (n)
<digit>  ::= 0 (0)
           | 1 (1)
```

The number $n$ in the corresponding grammar is computed as follows:

1. For each center $\overrightarrow{c_i}$, $i = 1, \ldots, k$, there are $d$ variables. As a consequence, every center requires $d \times k$ parameters.
2. Every Gaussian unit requires an additional parameter: $\sigma_i$, $i = 1, \ldots, k$, which means $k$ more parameters.
3. The weight vector $\overrightarrow{w}$ used in the output has $k$ parameters.

As an example of production, the chromosome $x = [9, 8, 6, 4, 15, 9, 16, 23, 8]$ is considered, where $d = 2$, $k = 2$, $n = 8$. The steps to produce the final program $p_{\text{test}} = (x7, 0, 1), (x1, 1, 0)$ are outlined in Table 1. Every partition program consists of a series of partition rules. Each partition rule contains three elements:

1. The variable for which its original interval will be partitioned, for example, $x_7$.
2. An integer number with values 0 and 1 at the left margin of the interval. If this value is 1, then the left margin of the corresponding variable's value field will be divided by two; otherwise, no change will be made.
3. An integer number with values 0 and 1 at the right end of the range of values of the variable. If this value is 1, then the right end of the corresponding variable's value field will be divided by two; otherwise, no change will be made.

Hence, for the example program $p_{\text{test}}$, the two partition rules will divide the right end of the variable $x_7$ and the left end of the variable $x_1$.

**Table 1.** The series of steps used to compute a valid expression from the BNF grammar for a given chromosome.

| Expression | Chromosome | Operation |
|---|---|---|
| | 9, 8, 6, 4, 15, 9, 16, 23, 8 | 9 mod 2 = 1 |
| <expr>,<expr> | 8, 6, 4, 15, 9, 16, 23, 8 | 8 mod 2 = 0 |
| (<xlist>,<digit>,<digit>),<expr> | 6, 4, 15, 9, 16, 23, 8 | 6 mod 8 = 6 |
| (x7,<digit>,<digit>),<expr> | 4, 15, 9, 16, 23, 8 | 4 mod 2 = 0 |

**Table 1.** *Cont.*

| Expression | Chromosome | Operation |
|---|---|---|
| (x7,0,<digit>),<expr> | 15, 9, 16, 23, 8 | 15 mod 2 = 1 |
| (x7,0,1),<expr> | 9, 16, 23, 8 | 9 mod 2 = 1 |
| (x7,0,1),(<xlist>,<digit>,<digit>) | 16, 23, 8 | 16 mod 8 = 0 |
| (x7,0,1),(x1,<digit>,<digit>) | 23, 8 | 23 mod 2 = 1 |
| (x7,0,1),(x1,1,<digit>) | 8 | 8 mod 2 = 0 |
| (x7,0,1),(x1,1,0) | | |

### *2.2. The First Phase of the Proposed Algorithm*

The purpose of this phase is to initialize the bounds of the RBF model and discover a promising interval for the corresponding values. For this initialization, the k-means algorithm [52] is used, which is also used for the traditional RBF network training technique. A description of this algorithm in a series of steps is shown in Algorithm 2.

**Algorithm 2** The k-means algorithm.

1. **Repeat**
   (a) **Define** $S_j = \{\}$, $j = 1 \ldots k$
   (b) **For** every pattern $x_i$, $i = 1, \ldots, m$ **do**
      i. **Compute** $j^* = \min_{i=1}^{k} \{D(x_i, c_j)\}$.
      ii. **Compute** $S_{j^*} = S_{j^*} \cup \{x_i\}$.
   (c) **EndFor**
   (d) **For** every center $c_j$, $j = 1 \ldots k$ **do**
      i. **Denote** as $M_j$ the number of points in set $S_j$
      ii. **Compute** $c_j$ as

$$c_j = \frac{1}{M_j} \sum_{i=1}^{M_j} x_i$$

   (e) **EndFor**
2. **Compute** the quantities $s_j$ as

$$\sigma_j^2 = \frac{\sum_{i=1}^{M_j} (x_i - c_j)^2}{M_j}$$

3. **Stop** the algorithm if centers $c_j$ do not change anymore.

Having calculated the centers $c_i$ and the corresponding variances $\sigma_i$, the algorithm continues to compute the vectors $\overrightarrow{L}$, $\overrightarrow{R}$ with dimension $n$, which are used as the initial bounds of the parameters. The above vectors are calculated through the procedure of Algorithm 3.

**Algorithm 3** The proposed algorithm used to locate the vectors $\overrightarrow{L}$, $\overrightarrow{R}$

1. **Set** m = 0
2. **Define** $F > 1$, the scaling factor.
3. **Define** $B > 0$, the initial upper bound for the weight vector $\overrightarrow{w}$.
4. **For** $i = 1 \ldots k$ **do**
   (a) **For** $j = 1 \ldots d$ **do**
      i. **Compute** $L_m$=$-F \times c_{ij}$, $R_m$=$F \times c_{ij}$
      ii. **Compute** $m = m + 1$
   (b) **EndFor**
   (c) **Compute** $L_m = -F \times \sigma_i$, $R_m = F \times \sigma_i$
   (d) **Compute** $m = m + 1$
5. **EndFor**
6. **For** $j = 1, \ldots, k$ **do**
   (a) **Compute** $L_m = -B$, $R_m = B$
   (b) **Compute** $m = m + 1$
7. **EndFor**

The range of values for the first $(d + 1) \times k$ parameters is estimated by multiplying the parameter $F$ by the values already estimated by the k-means algorithm. The bounds of the weight vector $\overrightarrow{w}$ are initialized using the value $B$. Subsequently, the genetic algorithm described here is performed to estimate the most promising range $\left[\overrightarrow{L}, \overrightarrow{R}\right]$ for the RBF parameters:

1. **Define** as $N_c$ the number of chromosomes that will participate in the the grammatical evolution procedure.
2. **Define** as $k$ the number of processing nodes of the used RBF model.
3. **Define** as $N_g$ the number of allowed generations.
4. **Define** as $p_s$ the used selection rate, with $p_s \leq 1$.
5. **Define** as $p_m$ the used mutation rate, with $p_m \leq 1$.
6. **Define** as $N_s$ the total number of RBF networks that will be created randomly in every fitness calculation.
7. **Initialize** $N_c$ chromosomes as sets of random numbers.
8. **Set** $f^* = [\infty, \infty]$ as the fitness of the best chromosome. The fitness function $f_g$ of any provided chromosome $g$ is considered an interval $f_g = \left[f_{g,\textbf{low}}, f_{g,\textbf{upper}}\right]$
9. **Set** iter = 0.
10. **For** $i = 1, \ldots, N_c$ **do**
    (a) **Produce** the partition program $p_i$ using the grammar of Figure 1 for the chromosome $i$.
    (b) **Produce** the bounds $\left[\overrightarrow{L_{p_i}}, \overrightarrow{R_{p_i}}\right]$ for the partition program $p_i$.
    (c) **Set** $E_{\min} = \infty$, $E_{\max} = -\infty$
    (d) **For** $j = 1, \ldots, N_S$ **do**
       i. **Create** randomly a set of parameters $\overrightarrow{g_j} \in \left[\overrightarrow{L_{p_i}}, \overrightarrow{R_{p_i}}\right]$
       ii. **Calculate** the error $E_{\overrightarrow{g_j}} = \sum_{k=1}^{M} \left(y\left(\overrightarrow{x_k}, \overrightarrow{g_j}\right) - t_k\right)^2$
       iii. **If** $E_{\overrightarrow{g_j}} \leq E_{\min}$, **then** $E_{\min} = E_{\overrightarrow{g_j}}$
       iv. **If** $E_{\overrightarrow{g_j}} \geq E_{\max}$, **then** $E_{\max} = E_{\overrightarrow{g_j}}$
    (e) **EndFor**
    (f) **Set** the fitness $f_i = [E_{\min}, E_{\max}]$
11. **EndFor**

12. **Perform** the procedure of selection. Initially, the chromosomes of the population are sorted according to their fitness values. Since the fitness values are intervals, the $L^*$ operator is defined as

$$L^*(f_a, f_b) = \begin{cases} \text{TRUE}, & a_1 < b_1, \text{OR } (a_1 = b_1 \text{ AND } a_2 < b_2) \\ \text{FALSE}, & \text{OTHERWISE} \end{cases} \quad (5)$$

As a consequence, the fitness value $f_a$ is considered smaller than $f_b$ if $L^*(f_a, f_b) =$ TRUE. The first $(1 - p_s) \times N_c$ chromosomes with smaller fitness values are copied without changes to the next generation of the algorithm. The rest of the chromosomes are replaced by chromosomes created in the crossover procedure.

13. **Perform** the crossover procedure. The crossover procedure will create new $p_s \times N_c$ chromosomes. For every pair of created offspring, two parents $(z, w)$ are selected from the current population using the tournament selection method. These parent will produce the offspring $\tilde{z}$ and $\tilde{w}$ using the one-point crossover method shown in Figure 1.
14. **Perform** the mutation procedure. In this process, a random number $r \in [0, 1]$ is drawn for every element of each chromosome. The corresponding element is changed randomly if $r \leq p_m$.
15. **Set** iter = iter + 1
16. **If** iter $\leq N_g$ , go to step 10.

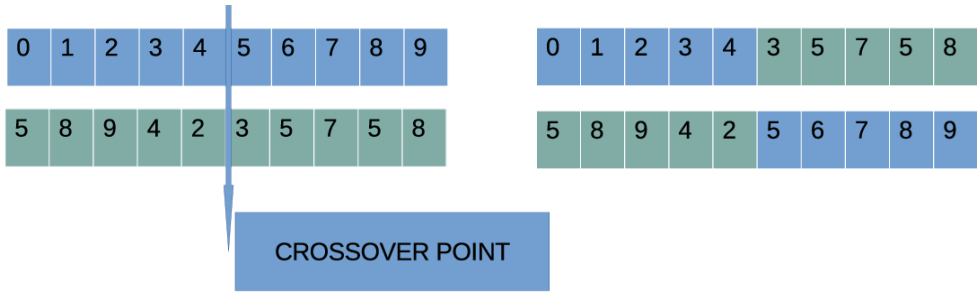


**Figure 1.** An example of the one-point crossover procedure, as used in grammatical evolution.

### *2.3. The Second Phase of the Proposed Algorithm*

The second phase utilizes a genetic algorithm to optimize the parameters of the RBF network. The optimization of the parameters uses as bounds the best interval produced in the first phase of the method. The layout of each chromosome is shown in Figure 2.

| $c_{11}$ | $c_{12}$ | ... | $c_{1d}$ | $\sigma_1$ | $c_{21}$ | $c_{22}$ | ... | $c_{2d}$ | $\sigma_2$ | ... | $c_{k1}$ | $c_{k2}$ | ... | $c_{kd}$ | $\sigma_k$ | $w_1$ | $w_2$ | ... | $w_k$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**Figure 2.** The layout of chromosomes in the second phase of the proposed algorithm.

1. **Initialization Step**
   (a) **Define** as $N_c$ the number of chromosomes.
   (b) **Define** as $N_g$ the total number of generations.
   (c) **Define** as $k$ the number of processing nodes of the used RBF model.
   (d) **Define** as $S = [L_{\text{best}}, R_{\text{best}}]$ the best-located interval of the first stage of the algorithm of Section 2.2.
   (e) **Produce** $N_C$ random chromosomes in $S$.
   (f) **Define** as $p_s$ the used selection rate, with $p_s \leq 1$.
   (g) **Define** as $p_m$ the used mutation rate, with $p_m \leq 1$.
   (h) **Set** iter = 0.
2. **Fitness calculation step**

(a) **For** $i = 1, \ldots, N_g$, **do**

i. **Compute** the fitness $f_i$ of each chromosome $g_i$ as $f_i = \sum_{j=1}^{m} \left( y(\overrightarrow{x_j}, \overrightarrow{g_i}) - t_j \right)^2$

(b) **EndFor**

3. **Genetic operations step**

(a) **Selection procedure.** Initially, the population is sorted according to the fitness values. The first $(1 - p_s) \times N_c$ chromosomes with the lowest fitness values remain intact. The rest of the chromosomes are replaced by offspring that will be produced during the crossover procedure.

(b) **Crossover procedure**: For every two new offspring $(\tilde{z}, \tilde{w})$, there are two parents $(z, w)$ that are selected from the current population with the selection procedure of tournament selection. The offspring are produced through the following process:

$$\begin{aligned} \tilde{z}_i &= a_i z_i + (1 - a_i) w_i \\ \tilde{w}_i &= a_i w_i + (1 - a_i) z_i \end{aligned} \tag{6}$$

The value $a_i$ is a random number, where $a_i \in [-0.5, 1.5]$ [74].

(c) **Perform** the mutation procedure. In this process, a random number $r \in [0, 1]$ is drawn for every element of each chromosome. The corresponding element is changed randomly if $r \leq p_m$.

4. **Termination Check Step**

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

(b) **If** iter $\leq N_g$ , go to step 2.

The steps of the current algorithm are also outlined graphically in Figure 3 using a flowchart.

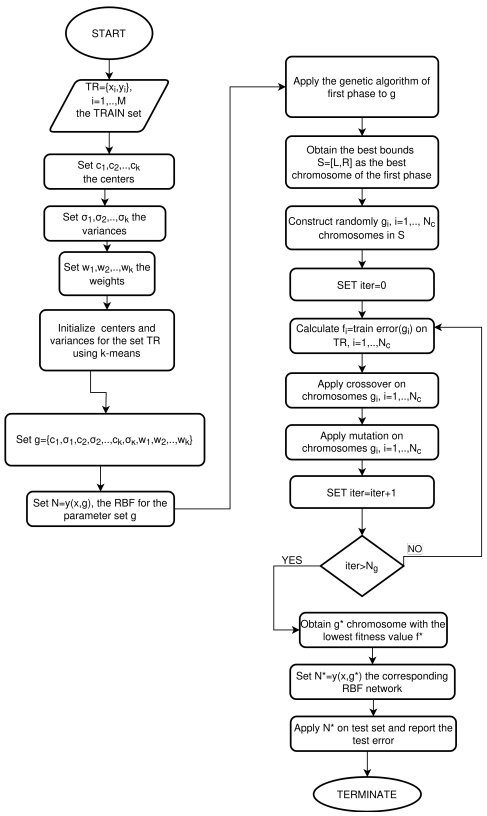


**Figure 3.** The flowchart of the proposed algorithm (g* denotes the best chromosome in the population and N* denotes the corresponding RBF network for g*).

## 3. Experiments

### *3.1. Experimental Datasets*

The proposed method was tested on a wide set of classification and regression problems found in the relevant literature. The method was compared against some other well-known machine learning models. The following databases were used to obtain the datasets:

1. The UCI dataset repository, https://archive.ics.uci.edu/ml/index.php (accessed on 5 December 2023);
2. The Keel repository, https://sci2s.ugr.es/keel/datasets.php (accessed on 5 December 2023) [75];
3. The Statlib URL http://lib.stat.cmu.edu/datasets/ (accessed on 5 December 2023).

The classification datasets are listed in Table 2, and the regression datasets are listed in Table 3.

**Table 2.** The classification datasets used in the experiments. The column DATASET denotes the number of the dataset, the column CLASSES stands for the number of classes in each dataset, and the column REFERENCE points to the bibliography where the use of the particular dataset is presented.

| Dataset | Classes | Reference |
|---|---|---|
| APPENDICITIS | 2 | [76] |
| AUSTRALIAN | 2 | [77] |
| BALANCE | 3 | [78] |
| CLEVELAND | 5 | [79,80] |
| DERMATOLOGY | 6 | [81] |
| HAYES ROTH | 3 | [82] |
| HEART | 2 | [83] |
| HOUSEVOTES | 2 | [84] |
| IONOSPHERE | 2 | [85,86] |
| LIVERDISORDER | 2 | [87] |
| MAMMOGRAPHIC | 2 | [88] |
| PARKINSONS | 2 | [89] |
| PIMA | 2 | [90] |
| POPFAILURES | 2 | [91] |
| SPIRAL | 2 | [92] |
| REGIONS2 | 5 | [93] |
| SAHEART | 2 | [94] |
| SEGMENT | 7 | [95] |
| WDBC | 2 | [96] |
| WINE | 3 | [97,98] |
| Z_F_S | 3 | [99] |
| ZO_NF_S | 3 | [99] |
| ZONF_S | 2 | [99] |
| ZOO | 7 | [100] |

**Table 3.** The regression datasets used in the experiments. The column DATASET denotes the number of the dataset, and the column REFERENCE points to the bibliography or URL (KEEL or STATLIB) where the use of the particular dataset is presented.

| Dataset | Reference |
|---|---|
| ABALONE | [101] |
| AIRFOIL | [102] |
| BASEBALL | STATLIB |
| BK | [103] |
| BL | STATLIB |
| CONCRETE | [104] |
| DEE | KEEL |
| DIABETES | KEEL |
| FA | STATLIB |
| HOUSING | [105] |
| MB | [103] |
| MORTGAGE | KEEL |
| NT | [106] |
| PY | [107] |
| QUAKE | [108] |
| TREASURY | KEEL |
| WANKARA | KEEL |

### *3.2. Experimental Results*

The RBF model used in the experiments was implemented in ANSI C++ with the assistance of the open-source Armadillo library [109]. The optimization methods used were also freely available from the OPTIMUS software, available from https://github.com/itsoulos/OPTIMUS/(accessed on 5 December 2023). For validation purposes, the 10-fold validation technique was used for all datasets and for all methods that participated in the experiments. Also, all the experiments were conducted 30 times, and the seed number of the random generator was different in each execution. The average classification error is reported for the classification datasets, and the average mean test error is reported for the regression datasets. The machine used in the experiments was an AMD Ryzen 5950X with 128 GB of RAM, and the operating system was Debian Linux. The values of the parameters used in the experiments are shown in Table 4. The experimental results for the classification datasets are outlined in Table 5, and the results for the regression datasets are listed in Table 6. For the tables with the experimental results, the following applies:

1. The column RPROP represents an artificial neural network [110,111]. This neural network has 10 processing nodes and was trained using the Rprop method [112].
2. The column denoted as ADAM indicates the application of the Adam optimizer [113,114] to train an artificial neural network with 10 hidden nodes.
3. The column NEAT (NeuroEvolution of Augmenting Topologies) [115] denotes the application of the NEAT method for neural network training.
4. The RBF-KMEANS column denotes the original two-phase training method for RBF networks.
5. The column GENRBF stands for the RBF training method introduced in [116].
6. The column PROPOSED stands for the results obtained using the proposed method.
7. In the experimental tables, an additional row was added with the title AVERAGE. This row contains the average classification or regression error for all datasets.

**Table 4.** The values used for the experimental parameters.

| Parameter | Value |
|---|---|
| $N_c$ | 200 |
| $N_g$ | 100 |
| $N_s$ | 50 |
| $F$ | 10.0 |
| $B$ | 100.0 |
| $k$ | 10 |
| $p_s$ | 0.90 |
| $p_m$ | 0.05 |

**Table 5.** The first column denotes the name of the classification dataset, and the numbers in the cells represent the classification error for every method used in the experiments. The last row stands for the average classification error for all datasets.

| Dataset | Rprop | Adam | Neat | Rbf-Kmeans | Genrbf | Proposed |
|---|---|---|---|---|---|---|
| Appendicitis | 16.30% | 16.50% | 17.20% | 12.23% | 16.83% | 15.77% |
| Australian | 36.12% | 35.65% | 31.98% | 34.89% | 41.79% | 22.40% |
| Balance | 8.81% | 7.87% | 23.14% | 33.42% | 38.02% | 15.62% |
| Cleveland | 61.41% | 67.55% | 53.44% | 67.10% | 67.47% | 50.37% |
| Dermatology | 15.12% | 26.14% | 32.43% | 62.34% | 61.46% | 35.73% |
| Hayes Roth | 37.46% | 59.70% | 50.15% | 64.36% | 63.46% | 35.33% |
| Heart | 30.51% | 38.53% | 39.27% | 31.20% | 28.44% | 15.91% |
| HouseVotes | 6.04% | 7.48% | 10.89% | 6.13% | 11.99% | 3.33% |
| Ionosphere | 13.65% | 16.64% | 19.67% | 16.22% | 19.83% | 9.30% |
| Liverdisorder | 40.26% | 41.53% | 30.67% | 30.84% | 36.97% | 28.44% |
| Mammographic | 18.46% | 46.25% | 22.85% | 21.38% | 30.41% | 17.72% |
| Parkinsons | 22.28% | 24.06% | 18.56% | 17.41% | 33.81% | 14.53% |
| Pima | 34.27% | 34.85% | 34.51% | 25.78% | 27.83% | 23.33% |
| Popfailures | 4.81% | 5.18% | 7.05% | 7.04% | 7.08% | 4.68% |
| Regions2 | 27.53% | 29.85% | 33.23% | 38.29% | 39.98% | 25.18% |
| Saheart | 34.90% | 34.04% | 34.51% | 32.19% | 33.90% | 29.46% |
| Segment | 52.14% | 49.75% | 66.72% | 59.68% | 54.25% | 49.22% |
| Spiral | 46.59% | 48.90% | 50.22% | 44.87% | 50.02% | 23.58% |
| Wdbc | 21.57% | 35.35% | 12.88% | 7.27% | 8.82% | 5.20% |
| Wine | 30.73% | 29.40% | 25.43% | 31.41% | 31.47% | 5.63% |
| Z_F_S | 29.28% | 47.81% | 38.41% | 13.16% | 23.37% | 3.90% |
| ZO_NF_S | 6.43% | 47.43% | 43.75% | 9.02% | 22.18% | 3.99% |
| ZONF_S | 27.27% | 11.99% | 5.44% | 4.03% | 17.41% | 1.67% |
| ZOO | 15.47% | 14.13% | 20.27% | 21.93% | 33.50% | 9.33% |
| **AVERAGE** | **26.56%** | **32.36%** | **30.11%** | **28.84%** | **33.35%** | **18.73%** |

**Table 6.** The first column denotes the name of the regression dataset, and the the numbers in the cells represent the regression error for every method used in the experiments. The last row stands for the average regression error for all datasets.

| Dataset | Rprop | Adam | Neat | Rbf-Kmeans | Genrbf | Proposed |
|---|---|---|---|---|---|---|
| ABALONE | 4.55 | 4.30 | 9.88 | 7.37 | 9.98 | 5.16 |
| AIRFOIL | 0.002 | 0.005 | 0.067 | 0.27 | 0.121 | 0.004 |
| BASEBALL | 92.05 | 77.90 | 100.39 | 93.02 | 98.91 | 81.26 |
| BK | 1.60 | 0.03 | 0.15 | 0.02 | 0.023 | 0.025 |
| BL | 4.38 | 0.28 | 0.05 | 0.013 | 0.005 | 0.0004 |
| CONCRETE | 0.009 | 0.078 | 0.081 | 0.011 | 0.015 | 0.006 |
| DEE | 0.608 | 0.630 | 1.512 | 0.17 | 0.25 | 0.16 |
| DIABETES | 1.11 | 3.03 | 4.25 | 0.49 | 2.92 | 1.74 |
| HOUSING | 74.38 | 80.20 | 56.49 | 57.68 | 95.69 | 21.11 |
| FA | 0.14 | 0.11 | 0.19 | 0.015 | 0.15 | 0.033 |
| MB | 0.55 | 0.06 | 0.061 | 2.16 | 0.41 | 0.19 |
| MORTGAGE | 9.19 | 9.24 | 14.11 | 1.45 | 1.92 | 0.014 |
| NT | 0.04 | 0.12 | 0.33 | 8.14 | 0.02 | 0.007 |
| PY | 0.039 | 0.09 | 0.075 | 0.012 | 0.029 | 0.019 |
| QUAKE | 0.041 | 0.06 | 0.298 | 0.07 | 0.79 | 0.034 |
| TREASURY | 10.88 | 11.16 | 15.52 | 2.02 | 1.89 | 0.098 |
| WANKARA | 0.0003 | 0.02 | 0.005 | 0.001 | 0.002 | 0.003 |
| **AVERAGE** | **11.71** | **11.02** | **11.97** | **10.17** | **12.54** | **6.46** |

On average, the current work appears to be 30–40% more accurate than the next best method. In many cases, this percentage exceeds 70%. Moreover, in the vast majority of problems, the proposed technique significantly outperforms the next best available method in terms of test error. In order to validate the results, an additional experiment was executed on the classification datasets, where the number of nodes increased from 5 to 20, and the results are graphically outlined in Figure 4. From this experiment, one can draw two conclusions: firstly, the proposed technique has a significant advantage over the others to a large extent in terms of average classification error, and secondly, the proposed method is shown to be robust and not significantly dependent on an increase in the processing nodes, since 5–10 processing nodes are enough to achieve low classification errors.

However, the proposed technique consists of two stages, and in each of them, a genetic algorithm should be executed. This means that it is significantly slower in computing time compared to the rest of the techniques, and, of course, it needs more computing resources. This is graphically shown in Figure 5, where the average execution time for the ADAM method and the proposed method is shown for the classification datasets when the number of processing nodes increases from 5 to 20. As expected, the current work requires significantly more time than a simple optimization technique such as ADAM, since it consists of two sequential genetic algorithms.

Of course, since we are talking about genetic algorithms, the training time required could be significantly reduced by using parallel techniques that take advantage of modern parallel computing structures, such as the MPI interface [117] or the OpenMP library [118]. The superiority of the proposed technique is also reinforced by the statistical tests carried out on the experimental results and outlined in Figure 6.

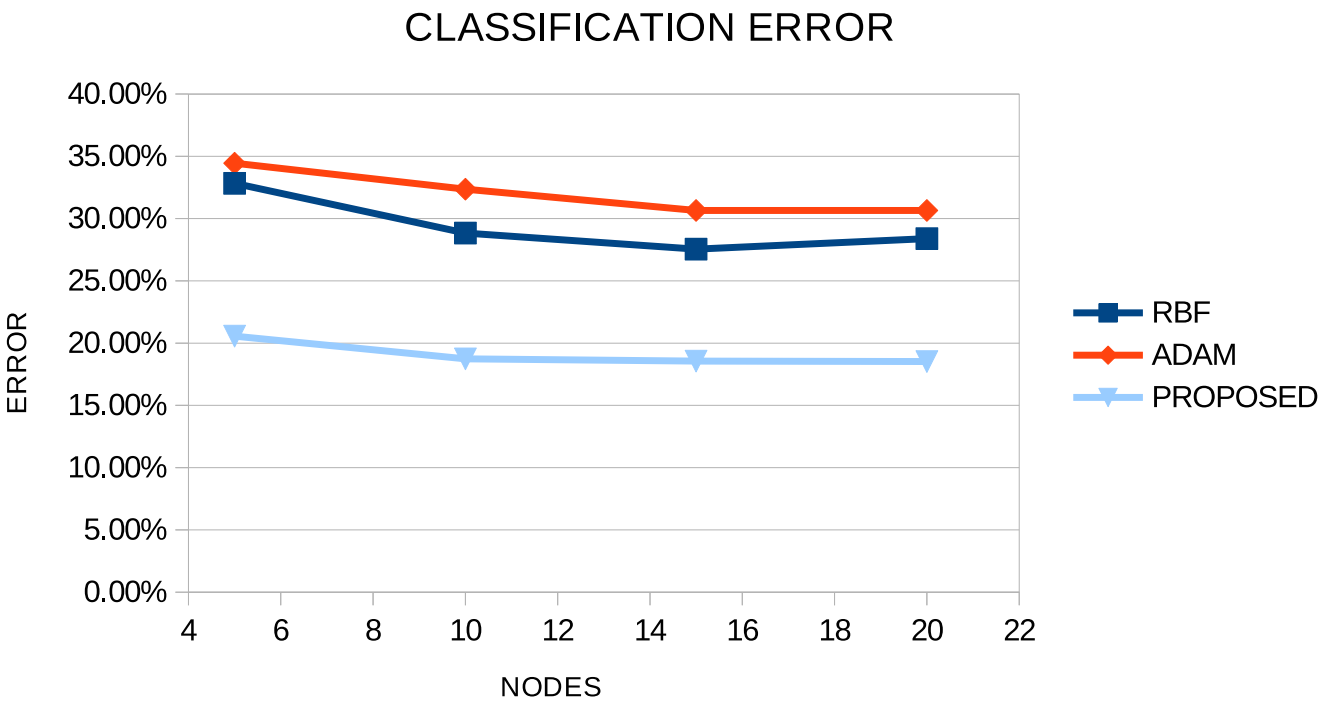


**Figure 4.** Average classification error for all classification datasets. The number of nodes increases from 5 to 20, and three models were used: the ADAM optimizer to optimize a neural network, the original RBF training method of two phases, and the proposed method.

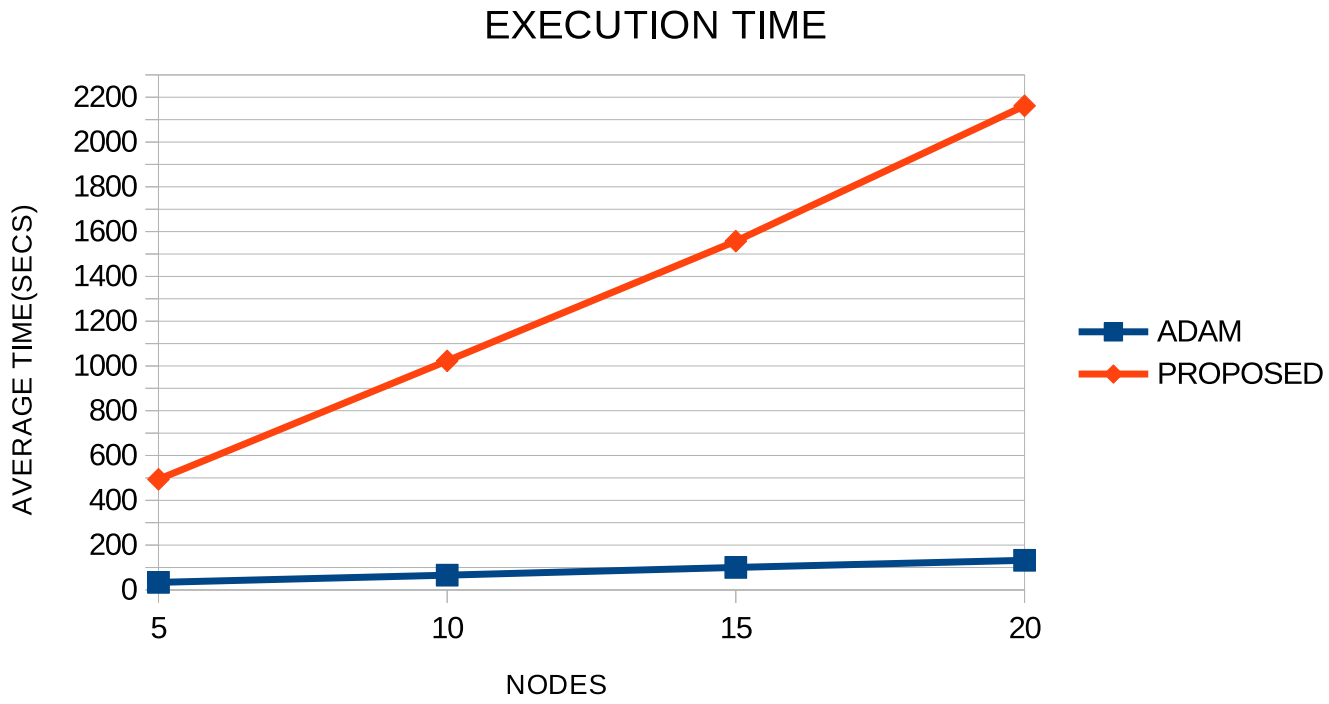


**Figure 5.** Average execution time for the ADAM optimizer used to train a neural network and the proposed technique.

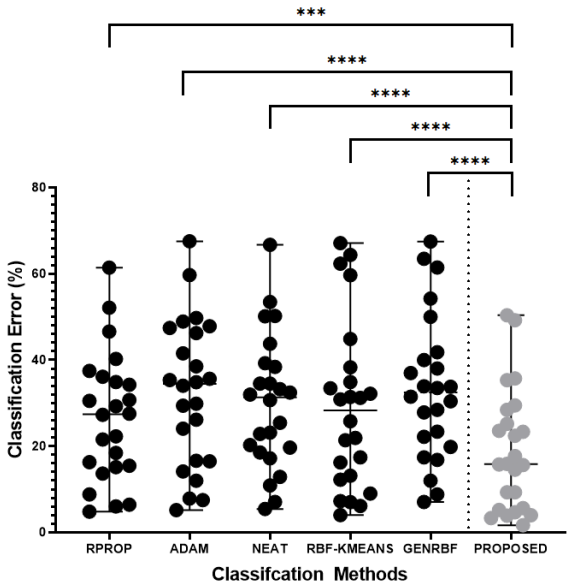


**Figure 6.** Scatter plot representation and the two-sample paired (Wilcoxon) signed-rank test results of the comparison for each of the five (5) classification methods (RPROP, ADAM, NEAT, RBF-KMEANS, and GENRBF) against the proposed method regarding the error on the twenty-four (24) classification datasets. The stars only intend to flag significance levels for the two most-used groups. A $p$-value of less than 0.001 is flagged with three stars (***). A $p$-value of less than 0.0001 is flagged with four stars (****).

In addition, an additional set of experiments was executed on the classification data. In this set of experiments, the critical parameter $F$ took the values 3, 5 and 10. The aim of this set of experiments was to establish the sensitivity of the proposed method to changes in its parameters. The experimental results are presented in Table 7, and a statistical test of the results is presented in Figure 7. The results and the statistics test indicate that there is no significant difference in the efficiency of the method for different values of the critical parameter $F$.

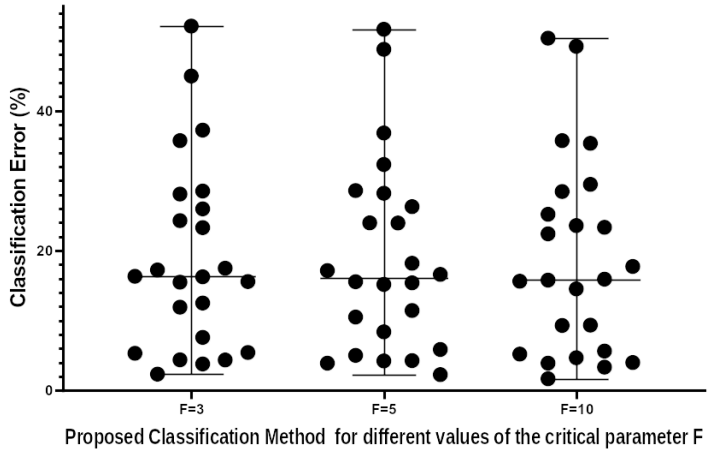


**Figure 7.** A Friedman test was conducted to find out whether different values of the critical parameter F had a difference or not in the classification error of the proposed method in twenty-four (24) other publicly available classification datasets. The analysis results for three different values of the critical parameter F (F = 3, F = 5, F = 10) indicated no significant difference.

**Table 7.** The following table presents experimental results from the use of the proposed technique in classification problems and for different values of the critical parameter $F$.

| Dataset | $F = 3$ | $F = 5$ | $F = 10$ |
|---|---|---|---|
| Appendicitis | 15.57% | 16.60% | 15.77% |
| Australian | 24.29% | 23.94% | 22.40% |
| Balance | 17.22% | 15.39% | 15.62% |
| Cleveland | 52.09% | 51.65% | 50.37% |
| Dermatology | 37.23% | 36.81% | 35.73% |
| Hayes Roth | 35.72% | 32.31% | 35.33% |
| Heart | 16.32% | 15.54% | 15.91% |
| HouseVotes | 4.35% | 3.90% | 3.33% |
| Ionosphere | 12.50% | 11.44% | 9.30% |
| Liverdisorder | 28.08% | 28.19% | 28.44% |
| Mammographic | 17.49% | 17.15% | 17.72% |
| Parkinsons | 16.25% | 15.17% | 14.53% |
| Pima | 23.29% | 23.97% | 23.33% |
| Popfailures | 5.31% | 5.86% | 4.68% |
| Regions2 | 25.97% | 26.29% | 25.18% |
| Saheart | 28.52% | 28.59% | 29.46% |
| Segment | 44.95% | 48.77% | 49.22% |
| Spiral | 15.49% | 18.19% | 23.58% |
| Wdbc | 5.43% | 5.01% | 5.20% |
| Wine | 7.59% | 8.39% | 5.63% |
| Z_F_S | 4.37% | 4.26% | 3.90% |
| ZO_NF_S | 3.79% | 4.21% | 3.99% |
| ZONF_S | 2.34% | 2.26% | 1.67% |
| ZOO | 11.90% | 10.50% | 9.33% |
| **AVERAGE** | **19.03%** | **18.93%** | **18.73%** |

## 4. Conclusions

In the current work, an innovative two-stage technique was proposed to efficiently train RBF artificial neural networks. In the first stage of the application, using grammatical evolution, the interval of values of the neural network parameters is partitioned so as to find a promising range that may contain low values of the training error. In the second stage, the neural network parameters are trained within the best range of values found in the first stage. The training of the parameters of the second phase is carried out using a genetic algorithm. The proposed method was applied on a wide series of well-known datasets from the relevant literature and was tested against a series of machine learning models. The new training technique was compared with the traditional method of training RBF networks, as well as with other machine learning models. From the experimental results, its superiority is evident in percentages that exceed 40%. However, since the proposed technique includes two genetic algorithms that are executed sequentially, the execution time required is longer compared to other techniques, especially for datasets with many patterns. An immediate solution to increase the speed of the method would be the use of parallel computing techniques, since genetic algorithms can, by nature, be directly parallelized.

Future improvements to the proposed method may include the following:

1. The proposed method could be applied to other variants of artificial neural networks.
2. Intelligent learning techniques could be used in place of the k-means technique to initialize the neural network parameters.
3. Techniques could be used to dynamically determine the number of necessary parameters for the neural network. For the time being, the number of parameters is considered constant, but this has the consequence of resulting in over-training phenomena being observed in various datasets.
4. Crossover and mutation techniques that focus more on the existing interval construction technique for model parameters could be implemented.
5. Efficient termination techniques for genetic algorithms could be used to obtain the most efficient termination of techniques without wasting computing time on unnecessary iterations.
6. Techniques that are based on parallel programming could be used to increase the speed of the method.

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

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Data is contained within the article.

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

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Mjahed, M. The use of clustering techniques for the classification of high energy physics data. *Nucl. Instrum. Methods Phys. Res. Sect. A Accel. Spectrometers Detect. Assoc. Equip.* **2006**, *559*, 199–202. [CrossRef]
2. Andrews, M.; Paulini, M.; Gleyzer, S.; Poczos, B. End-to-End Event Classification of High-Energy Physics Data. *J. Phys. Conf. Ser.* **2018**, *1085*, 042022. [CrossRef]
3. He, P.; Xu, C.J.; Liang, Y.Z.; Fang, K.T. Improving the classification accuracy in chemistry via boosting technique. *Chemom. Intell. Lab. Syst.* **2004**, *70*, 39–46. [CrossRef]
4. Aguiar, J.A.; Gong, M.L.; Tasdizen, T. Crystallographic prediction from diffraction and chemistry data for higher throughput classification using machine learning. *Comput. Mater. Sci.* **2020**, *173*, 109409.
5. Kaastra, I.; Boyd, M. Designing a neural network for forecasting financial and economic time series. *Neurocomputing* **1996**, *10*, 215–236. [CrossRef]
6. Hafezi, R.; Shahrabi, J.; Hadavandi, E. A bat-neural network multi-agent system (BNNMAS) for stock price prediction: Case study of DAX stock price. *Appl. Soft Comput.* **2015**, *29*, 196–210. [CrossRef]
7. Yadav, S.S.; Jadhav, S.M. Deep convolutional neural network based medical image classification for disease diagnosis. *J. Big Data* **2019**, *6*, 113. [CrossRef]
8. Qing, L.; Linhong, W.; Xuehai, D. A Novel Neural Network-Based Method for Medical Text Classification. *Future Internet* **2019**, *11*, 255. [CrossRef]
9. Park, J.; Sandberg, I.W. Universal Approximation Using Radial-Basis-Function Networks. *Neural Comput.* **1991**, *3*, 246–257. [CrossRef]
10. Montazer, G.A.; Giveki, D.; Karami, M.; Rastegar, H. Radial basis function neural networks: A review. *Comput. Rev. J.* **2018**, *1*, 52–74.
11. Abiodun, O.I.; Jantan, A.; Omolara, A.E.; Dada, K.V.; Mohamed, N.A.; Arshad, H. State-of-the-art in artificial neural network applications: A survey. *Heliyon* **2018**, *4*, e00938. [CrossRef] [PubMed]

12. Liao, Y.; Fang, S.C.; Nuttle, H.L.W. Relaxed conditions for radial-basis function networks to be universal approximators. *Neural Netw.* **2003**, *16*, 1019–1028. [CrossRef] [PubMed]
13. Teng, P. Machine-learning quantum mechanics: Solving quantum mechanics problems using radial basis function networks. *Phys. Rev. E* **2018**, *98*, 033305. [CrossRef]
14. Jovanović, R.; Sretenovic, A. Ensemble of radial basis neural networks with K-means clustering for heating energy consumption prediction. *FME Trans.* **2017**, *45*, 51–57. [CrossRef]
15. Gorbachenko, V.I.; Zhukov, M.V. Solving boundary value problems of mathematical physics using radial basis function networks. *Comput. Math. Math. Phys.* **2017**, *57*, 145–155. [CrossRef]
16. Määttä, J.; Bazaliy, V.; Kimari, J.; Djurabekova, F.; Nordlund, K.; Roos, T. Gradient-based training and pruning of radial basis function networks with an application in materials physics. *Neural Netw.* **2021**, *133*, 123–131. [CrossRef] [PubMed]
17. Nam, M.-D.; Thanh, T.-C. Numerical solution of differential equations using multiquadric radial basis function networks. *Neural Netw.* **2001**, *14*, 185–199.
18. Mai-Duy, N. Solving high order ordinary differential equations with radial basis function networks. *Int. J. Numer. Meth. Eng.* **2005**, *62*, 824–852. [CrossRef]
19. Sarra, S.A. Adaptive radial basis function methods for time dependent partial differential equations. *Appl. Numer.* **2005**, *54*, 79–94. [CrossRef]
20. Lian, R.-J. Adaptive Self-Organizing Fuzzy Sliding-Mode Radial Basis-Function Neural-Network Controller for Robotic Systems. *IEEE Trans. Ind. Electron.* **2014**, *61*, 1493–1503. [CrossRef]
21. Vijay, M.; Jena, D. Backstepping terminal sliding mode control of robot manipulator using radial basis functional neural networks. *Comput. Electr. Eng.* **2018**, *67*, 690–707. [CrossRef]
22. Er, M.J.; Wu, S.; Lu, J.; Toh, H.L. Face recognition with radial basis function (RBF) neural networks. *IEEE Trans. Neural Netw.* **2002**, *13*, 697–710. [PubMed]
23. Laoudias, C.; Kemppi, P.; Panayiotou, C.G. Localization Using Radial Basis Function Networks and Signal Strength Fingerprints in WLAN. In Proceedings of the GLOBECOM 2009—2009 IEEE Global Telecommunications Conference, Honolulu, HI, USA, 30 November–4 December 2009; pp. 1–6.
24. Azarbad, M.; Hakimi, S.; Ebrahimzadeh, A. Automatic recognition of digital communication signal. *Int. J. Energy Inf. Commun.* **2012**, *3*, 21–33.
25. Yu, D.L.; Gomm, J.B.; Williams, D. Sensor fault diagnosis in a chemical process via RBF neural networks. *Control Eng. Pract.* **1999**, *7*, 49–55. [CrossRef]
26. Shankar, V.; Wright, G.B.; Fogelson, A.L.; Kirby, R.M. A radial basis function (RBF) finite difference method for the simulation of reaction–diffusion equations on stationary platelets within the augmented forcing method. *Int. J. Numer. Meth. Fluids* **2014**, *75*, 1–22. [CrossRef]
27. Shen, W.; Guo, X.; Wu, C.; Wu, D. Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm. *Knowl.-Based Syst.* **2011**, *24*, 378–385. [CrossRef]
28. Momoh, J.A.; Reddy, S.S. Combined Economic and Emission Dispatch using Radial Basis Function. In Proceedings of the 2014 IEEE PES General Meeting | Conference & Exposition, National Harbor, MD, USA, 27–31 July 2014; pp. 1–5.
29. Sohrabi, P.; Shokri, B.J.; Dehghani, H. Predicting coal price using time series methods and combination of radial basis function (RBF) neural network with time series. *Miner. Econ.* **2021**, *36*, 207–216. [CrossRef]
30. Ravale, U.; Marathe, N.; Padiya, P. Feature Selection Based Hybrid Anomaly Intrusion Detection System Using K Means and RBF Kernel Function. *Procedia Comput. Sci.* **2015**, *45*, 428–435. [CrossRef]
31. Lopez-Martin, M.; Sanchez-Esguevillas, A.; Arribas, J.I.; Carro, B. Network Intrusion Detection Based on Extended RBF Neural Network With Offline Reinforcement Learning. *IEEE Access* **2021**, *9*, 153153–153170. [CrossRef]
32. Kuncheva, L.I. Initializing of an RBF network by a genetic algorithm. *Neurocomputing* **1997**, *14*, 273–288. [CrossRef]
33. Ros, F.; Pintore, M.; Deman, A.; Chrétien, J.R. Automatical initialization of RBF neural networks. *Chemom. Intell. Lab. Syst.* **2007**, *87*, 26–32. [CrossRef]
34. Wang, D.; Zeng, X.J.; Keane, J.A. A clustering algorithm for radial basis function neural network initialization. *Neurocomputing* **2012**, *77*, 144–155. [CrossRef]
35. Benoudjit, N.; Verleysen, M. On the Kernel Widths in Radial-Basis Function Networks. *Neural Process. Lett.* **2003**, *18*, 139–154. [CrossRef]
36. Neruda, R.; Kudova, P. Learning methods for radial basis function networks. *Future Gener. Comput. Syst.* **2005**, *21*, 1131–1142. [CrossRef]
37. Ricci, E.; Perfetti, R. Improved pruning strategy for radial basis function networks with dynamic decay adjustment. *Neurocomputing* **2006**, *69*, 1728–1732. [CrossRef]
38. Huang, G.-B.; Saratchandran, P.; Sundararajan, N. A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. *IEEE Trans. Neural Netw.* **2005**, *16*, 57–67. [CrossRef] [PubMed]
39. Bortman, M.; Aladjem, M. A Growing and Pruning Method for Radial Basis Function Networks. *IEEE Trans. Neural Netw.* **2009**, *20*, 1039–1045. [CrossRef]
40. Yokota, R.; Barba, L.A.; Knepley, M.G. PetRBF—A parallel O(N) algorithm for radial basis function interpolation with Gaussians. *Comput. Methods Appl. Mech. Eng.* **2010**, *199*, 1793–1804. [CrossRef]

41. Lu, C.; Ma, N.; Wang, Z. Fault detection for hydraulic pump based on chaotic parallel RBF network. *EURASIP J. Adv. Signal Process.* **2011**, *2011*, 49. [CrossRef]
42. Iranmehr, A.; Masnadi-Shirazi, H.; Vasconcelos, N. Cost-sensitive support vector machines. *Neurocomputing* **2019**, *343*, 50–64. [CrossRef]
43. Cervantes, J.; Lamont, F.G.; Mazahua, L.R.; Lopez, A. A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing* **2020**, *408*, 189–215. [CrossRef]
44. Kotsiantis, S.B. Decision trees: A recent overview. *Artif. Intell. Rev.* **2013**, *39*, 261–283. [CrossRef]
45. Bertsimas, D.; Dunn, J. Optimal classification trees. *Mach. Learn.* **2017**, *106*, 1039–1082. [CrossRef]
46. Wang, Y.; Yao, H.; Zhao, S. Auto-encoder based dimensionality reduction. *Neurocomputing* **2016**, *184*, 232–242. [CrossRef]
47. Agarwal, V.; Bhanot, S. Radial basis function neural network-based face recognition using firefly algorithm. *Neural Comput. Applic.* **2018**, *30*, 2643–2660. [CrossRef]
48. Jiang, S.; Lu, C.; Zhang, S.; Lu, X.; Tsai, S.B.; Wang, C.K.; Gao, Y.; Shi, Y.; Lee, C.H. Prediction of Ecological Pressure on Resource-Based Cities Based on an RBF Neural Network Optimized by an Improved ABC Algorithm. *IEEE Access* **2019**, *7*, 47423–47436. [CrossRef]
49. Khan, I.U.; Aslam, N.; Alshehri, R.; Alzahrani, S.; Alghamdi, M.; Almalki, A.; Balabeed, M. Cervical Cancer Diagnosis Model Using Extreme Gradient Boosting and Bioinspired Firefly Optimization. *Sci. Program.* **2021**, *2021*, 5540024. [CrossRef]
50. Gyamfi, K.S.; Brusey, J.; Gaura, E. Differential radial basis function network for sequence modelling. *Expert Syst. Appl.* **2022**, *189*, 115982. [CrossRef]
51. Li, X.Q.; Song, L.K.; Choy, Y.S.; Bai, G.C. Multivariate ensembles-based hierarchical linkage strategy for system reliability evaluation of aeroengine cooling blades. *Aerosp. Sci. Technol.* **2023**, *138*, 108325. [CrossRef]
52. MacQueen, J. Some methods for classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Los Angeles, CA, USA, 21 June 21–18 July 1967; pp. 281–297.
53. O'Neill, M.; Ryan, C. Grammatical evolution. *IEEE Trans. Evol. Comput.* **2001**, *5*, 349–358. [CrossRef]
54. Wang, H.Q.; Huang, D.S.; Wang, B. Optimisation of radial basis function classifiers using simulated annealing algorithm for cancer classification. *Electron. Lett.* **2005**, *41*, 630–632. [CrossRef]
55. Fathi, V.; Montazer, G.A. An improvement in RBF learning algorithm based on PSO for real time applications. *Neurocomputing* **2013**, *111*, 169–176. [CrossRef]
56. Goldberg, D. *Genetic Algorithms in Search, Optimization and Machine Learning*; Addison-Wesley Publishing Company: Reading, MA, USA, 1989.
57. Michaelewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs*; Springer-Verlag: Berlin/Heidelberg, Germany, 1996.
58. Grady, S.A.; Hussaini, M.Y.; Abdullah, M.M. Placement of wind turbines using genetic algorithms. *Renew. Energy* **2005**, *30*, 259–270. [CrossRef]
59. Holland, J.H. Genetic algorithms. *Sci. Am.* **1992**, *267*, 66–73. [CrossRef]
60. Stender, J. *Parallel Genetic Algorithms: Theory & Applications*; IOS Press: Amsterdam, The Netherlands, 1993.
61. Backus, J.W. The Syntax and Semantics of the Proposed International Algebraic Language of the Zurich ACM-GAMM Conference. In Proceedings of the International Conference on Information Processing, UNESCO, Paris, France, 15–20 June 1959; pp. 125–132.
62. Ryan, C.; Collins, J.; O'Neill, M. Grammatical evolution: Evolving programs for an arbitrary language. In *Genetic Programming. EuroGP 1998*; Banzhaf, W., Poli, R., Schoenauer, M., Fogarty, T.C., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1998; Volume 1391.
63. O'Neill, M.; Ryan, M.C. Evolving Multi-line Compilable C Programs. In *Genetic Programming*; Poli, R., Nordin, P., Langdon, W.B., Fogarty, T.C., Eds.; EuroGP 1999. Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1999; Volume 1598.
64. Ryan, C.; O'Neill, M.; Collins, J.J. Grammatical evolution: Solving trigonometric identities. In *Proceedings of Mendel*; Technical University of Brno, Faculty of Mechanical Engineering: Brno, Czech Republic, 1998; Volume 98.
65. Puente, A.O.; Alfonso, R.S.; Moreno, M.A. Automatic composition of music by means of grammatical evolution. In Proceedings of the APL '02: 2002 Conference on APL: Array Processing Languages: Lore, Problems, and Applications, Madrid, Spain, 22–25 July 2002; pp. 148–155.
66. De Campos, L.M.L.; de Oliveira, R.C.L.; Roisenberg, M. Optimization of neural networks through grammatical evolution and a genetic algorithm. *Expert Syst. Appl.* **2016**, *56*, 368–384. [CrossRef]
67. Soltanian, K.; Ebnenasir, A.; Afsharchi, M. Modular Grammatical Evolution for the Generation of Artificial Neural Networks. *Evol. Comput.* **2022**, *30*, 291–327. [CrossRef] [PubMed]
68. Dempsey, I.; Neill, M.O.; Brabazon, A. Constant creation in grammatical evolution. *Int. J. Innov. Appl.* **2007**, *1*, 23–38. [CrossRef]
69. Galván-López, E.; Swafford, J.M.; O'Neill, M.; Brabazon, A. Evolving a Ms. PacMan Controller Using Grammatical Evolution. In *Applications of Evolutionary Computation*; EvoApplications 2010. Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6024.
70. Shaker, N.; Nicolau, M.; Yannakakis, G.N.; Togelius, J.; O'Neill, M. Evolving levels for Super Mario Bros using grammatical evolution. In Proceedings of the 2012 IEEE Conference on Computational Intelligence and Games (CIG), Granada, Spain, 11–14 September 2012; pp. 304–331.
71. Martínez-Rodríguez, D.; Colmenar, J.M.; Hidalgo, J.I.; Micó, R.J.V.; Salcedo-Sanz, S. Particle swarm grammatical evolution for energy demand estimation. *Energy Sci. Eng.* **2020**, *8*, 1068–1079. [CrossRef]

72. Sabar, N.R.; Ayob, M.; Kendall, G.; Qu, R. Grammatical Evolution Hyper-Heuristic for Combinatorial Optimization Problems. *IEEE Trans. Evol. Comput.* **2013**, *17*, 840–861. [CrossRef]
73. Ryan, C.; Kshirsagar, M.; Vaidya, G.; Cunningham, A.; Sivaraman, R. Design of a cryptographically secure pseudo random number generator with grammatical evolution. *Sci. Rep.* **2022**, *12*, 8602. [CrossRef]
74. Kaelo, P.; Ali, M.M. Integrated crossover rules in real coded genetic algorithms. *Eur. J. Oper. Res.* **2007**, *176*, 60–76. [CrossRef]
75. Alcalá-Fdez, J.; Fernandez, A.; Luengo, J.; Derrac, J.; García, S.; Sánchez, L.; Herrera, F. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. *J. Mult.-Valued Log. Soft Comput.* **2011**, *17*, 255–287.
76. Weiss, S.M.; Kulikowski, C.A. *Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*; Morgan Kaufmann Publishers Inc.: Burlington, MA, USA, 1991.
77. Quinlan, J.R. Simplifying Decision Trees. *Int. Man-Mach. Stud.* **1987**, *27*, 221–234. [CrossRef]
78. Shultz, T.; Mareschal, D.; Schmidt, W. Modeling Cognitive Development on Balance Scale Phenomena. *Mach. Learn.* **1994**, *16*, 59–88. [CrossRef]
79. Zhou, Z.H.; Jiang, Y. NeC4.5: Neural ensemble based C4.5. *IEEE Trans. Knowl. Data Eng.* **2004**, *16*, 770–773. [CrossRef]
80. Setiono, R.; Leow, W.K. FERNN: An Algorithm for Fast Extraction of Rules from Neural Networks. *Appl. Intell.* **2000**, *12*, 15–25. [CrossRef]
81. Demiroz, G.; Govenir, H.A.; Ilter, N. Learning Differential Diagnosis of Eryhemato-Squamous Diseases using Voting Feature Intervals. *Artif. Intell. Med.* **1998**, *13*, 147–165.
82. Hayes-Roth, B.; Hayes-Roth, B.F. Concept learning and the recognition and classification of exemplars. *J. Verbal Learn. Verbal Behav.* **1977**, *16*, 321–338. [CrossRef]
83. Kononenko, I.; Šimec, E.; Robnik-Šikonja, M. Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF. *Appl. Intell.* **1997**, *7*, 39–55. [CrossRef]
84. French, R.M.; Chater, N. Using noise to compute error surfaces in connectionist networks: A novel means of reducing catastrophic forgetting. *Neural Comput.* **2002**, *14*, 1755–1769. [CrossRef]
85. Dy, J.G.; Brodley, C.E. Feature Selection for Unsupervised Learning. *J. Mach. Learn. Res.* **2004**, *5*, 845–889.
86. Perantonis, S.J.; Virvilis, V. Input Feature Extraction for Multilayered Perceptrons Using Supervised Principal Component Analysis. *Neural Process. Lett.* **1999**, *10*, 243–252. [CrossRef]
87. Garcke, J.; Griebel, M. Classification with sparse grids using simplicial basis functions. *Intell. Data Anal.* **2002**, *6*, 483–502. [CrossRef]
88. Elter, M.; Schulz-Wendtland, R.; Wittenberg, T. The prediction of breast cancer biopsy outcomes using two CAD approaches that both emphasize an intelligible decision process. *Med. Phys.* **2007**, *34*, 4164–4172. [CrossRef] [PubMed]
89. Little, M.A.; McSharry, P.E.; Hunter, E.J.; Spielman, J.; Ramig, L.O. Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. *IEEE Trans. Biomed. Eng.* **2009**, *56*, 1015. [CrossRef] [PubMed]
90. Smith, J.W.; Everhart, J.E.; Dickson, W.C.; Knowler, W.C.; Johannes, R.S. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In Proceedings of the Symposium on Computer Applications and Medical Care, Washington, DC, USA, 6–9 November 1988; IEEE Computer Society Press: Washington, DC, USA, 1988; pp. 261–265.
91. Lucas, D.D.; Klein, R.; Tannahill, J.; Ivanova, D.; Brandon, S.; Domyancic, D.; Zhang, Y. Failure analysis of parameter-induced simulation crashes in climate models. *Geosci. Model Dev.* **2013**, *6*, 1157–1171. [CrossRef]
92. Gavrilis, D.; Tsoulos, I.G.; Dermatas, E. Selecting and constructing features using grammatical evolution. *Pattern Recognit. Lett.* **2008**, *29*, 1358–1365. [CrossRef]
93. Giannakeas, N.; Tsipouras, M.G.; Tzallas, A.T.; Kyriakidi, K.; Tsianou, Z.E.; Manousou, P.; Hall, A.; Karvounis, E.C.; Tsianos, V.; Tsianos, E. A clustering based method for collagen proportional area extraction in liver biopsy images. In Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS, Milan, Italy, 25–29 August 2015; pp. 3097–3100.
94. Hastie, T.; Tibshirani, R. Non-parametric logistic and proportional odds regression. *JRSS-C (Appl. Stat.)* **1987**, *36*, 260–276. [CrossRef]
95. Dash, M.; Liu, H.; Scheuermann, P.; Tan, K.L. Fast hierarchical clustering and its validation. *Data Knowl. Eng.* **2003**, *44*, 109–138. [CrossRef]
96. Wolberg, W.H.; Mangasarian, O.L. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proc. Natl. Acad. Sci. USA* **1990**, *87*, 9193–9196. [CrossRef]
97. Raymer, M.; Doom, T.E.; Kuhn, L.A.; Punch, W.F. Knowledge discovery in medical and biological datasets using a hybrid Bayes classifier/evolutionary algorithm. *IEEE Trans. Syst. Man, And Cybern. Part B Cybern.* **2003**, *33*, 802–813. [CrossRef] [PubMed]
98. Zhong, P.; Fukushima, M. Regularized nonsmooth Newton method for multi-class support vector machines. *Optim. Methods Softw.* **2007**, *22*, 225–236. [CrossRef]
99. Andrzejak, R.G.; Lehnertz, K.; Mormann, F.; Rieke, C.; David, P.; Elger, C.E. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Phys. Rev. E* **2001**, *64*, 1–8. [CrossRef]
100. Koivisto, M.; Sood, K. Exact Bayesian Structure Discovery in Bayesian Networks. *J. Mach. Learn. Res.* **2004**, *5*, 549–573.

101. 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; Marine Research Laboratories, Department of Primary Industries and Fisheries: Hobart, Australia, 1994.
102. Brooks, T.F.; Pope, D.S.; Marcolini, A.M. *Airfoil Self-Noise and Prediction*; Technical Report, NASA RP-1218; NASA: Washington, DC, USA, 1989.
103. Simonoff, J.S. *Smooting Methods in Statistics*; Springer: Berlin/Heidelberg, Germany, 1996.
104. Cheng Yeh, I. Modeling of strength of high performance concrete using artificial neural networks. *Cem. Concr. Res.* **1998**, *28*, 1797–1808.
105. Harrison, D.; Rubinfeld, D.L. Hedonic prices and the demand for clean ai. *J. Environ. Econ. Manag.* **1978**, *5*, 81–102. [CrossRef]
106. 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. [CrossRef]
107. King, R.D.; Muggleton, S.; Lewis, R.; Sternberg, M.J.E. Drug design by machine learning: The use of inductive logic programming to model the structure-activity relationships of trimethoprim analogues binding to dihydrofolate reductase. *Proc. Natl. Acad. Sci. USA* **1992**, *89*, 11322–11326. [CrossRef]
108. Sikora, M.; Wrobel, L. Application of rule induction algorithms for analysis of data collected by seismic hazard monitoring systems in coal mines. *Arch. Min. Sci.* **2010**, *55*, 91–114.
109. Sanderson, C.; Curtin, R. Armadillo: A template-based C++ library for linear algebra. *J. Open Source Softw.* **2016**, *1*, 26. [CrossRef]
110. Bishop, C. *Neural Networks for Pattern Recognition*; Oxford University Press: Oxford, UK, 1995.
111. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Syst.* **1989**, *2*, 303–314. [CrossRef]
112. Riedmiller, M.; Braun, H. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP algorithm. In Proceedings of the International Conference on Neural Networks (ICNN'88), San Francisco, CA, USA, 28 March–1 April 1993; pp. 586–591.
113. Kingma, D.P.; Ba, J.L. ADAM: A method for stochastic optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), San Diego, CA, USA, 7–9 May 2015; pp. 1–15.
114. Xue, Y.; Tong, Y.; Neri, F. An ensemble of differential evolution and Adam for training feed-forward neural networks. *Inf. Sci.* **2022**, *608*, 453–471. [CrossRef]
115. Stanley, K.O.; Miikkulainen, R. Evolving Neural Networks through Augmenting Topologies. *Evol. Comput.* **2002**, *10*, 99–127. [CrossRef]
116. Ding, S.; Xu, L.; Su, C.; Jin, F. An optimizing method of RBF neural network based on genetic algorithm. *Neural Comput. Appl.* **2012**, *21*, 333–336. [CrossRef]
117. 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. [CrossRef]
118. Chandra, R.; Dagum, L.; Kohr, D.; Maydan, D.; McDonald, J.; Menon, R. *Parallel Programming in OpenMP*; Morgan Kaufmann Publishers Inc.: Burlington, MA, USA, 2001.