MDPI

Article

Utilizing a bounding procedure based on Simulated Annealing to effectively locate the bounds for the parameters of Radial Basis Function networks

Ioannis G. Tsoulos[1,*], Vasileios Charilogis[2], Dimitrios Tsalikakis[3]

[1] Department of Informatics and Telecommunications, University of Ioannina, Greece; itsoulos@uoi.gr
[2] Department of Informatics and Telecommunications, University of Ioannina, Greece; v.charilog@uoi.gr
[3] Department of Engineering Informatics and Telecommunications, University of Western Macedonia, 50100 Kozani, Greece; tsalikakis@gmail.com
* Correspondence: itsoulos@uoi.gr

Abstract: The Radial Basis Function (RBF) networks are an established parametric machine learning tool, which has been extensively utilized in data classification and data fitting problems. These specific machine learning tools have been applied in various scientific areas, such as problems in physics, chemistry, and medicine, with excellent results. A two-step technique is usually used to adjust the parameters of these models, which is in most cases extremely effective. However, it does not effectively explore the value space of the network parameters and often results in parameter stability problems. In this paper, the use of a bounding technique that explores the value space of the parameters of these networks using intervals generated by a procedure based on the Simulated Annealing method is recommended. After finding a promising range of values for the network parameters, a genetic algorithm is applied within this range of values to more effectively adjust its parameters. The new method was applied on a wide range of classification and regression datasets from the relevant literature and the results are reported in the current manuscript.

Keywords: Radial Basis Function networks; Simulated Annealing; Stochastic techniques; Evolutionary Computation

Citation: Tsoulos, I.G.; Charilogis, V.; Tsalikakis D. Utilizing a bounding procedure based on Simulated Annealing to effectively locate the bounds for the parameters of Radial Basis Function networks. *Journal Not Specified* **2023**, *1*, 0. https://doi.org/

Received:
Revised:
Accepted:
Published:

1. Introduction

A wide range of practical problems can be considered as classification or regression problems. Such problems occur in areas such as physics [1,2], astronomy [3,4], chemistry [5,6], medicine [7,8], economics [9,10] etc. A commonly used machine learning tool used to tackle such problems is the Radial Basis Function (RBF) network, expressed as the following function:

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

In this equation the following definitions are used for the symbols:

1. The vector $\overrightarrow{x}$ represents the input pattern with d number of features.
2. The parameter k stands for the number of weights of the network. These weights are represented by the values $w_i,\ i = 1, \ldots, k$.
3. The vectors $\overrightarrow{c_i},\ i = 1, .., k$ represent the the so - called centers of the network.
4. The function $\phi(x)$ commonly is a Gaussian function with the following definition:

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

A typical plot of the Gaussian function for $c = 0,\ \sigma = 1$ is depicted in Figure 1. From this graph it is deduced that the Gaussian RBF decreases regarding the distance from the c. The

parameter c and σ can be estimated using the K-Means algorithm introduced by MacQueen [11]. The training error of an RBF network is defined as:

$$E\left(R\left(\overrightarrow{x}\right)\right) = \sum_{i=1}^{M}\left(R\left(\overrightarrow{x_i}\right) - y_i\right)^2 \tag{3}$$

where $\left(\overrightarrow{x_i}, y_i\right),\ i = 1, ..., M$ represents the training set of the objective problem and the values y_i are the expected outputs for every pattern $\overrightarrow{x_i}$.

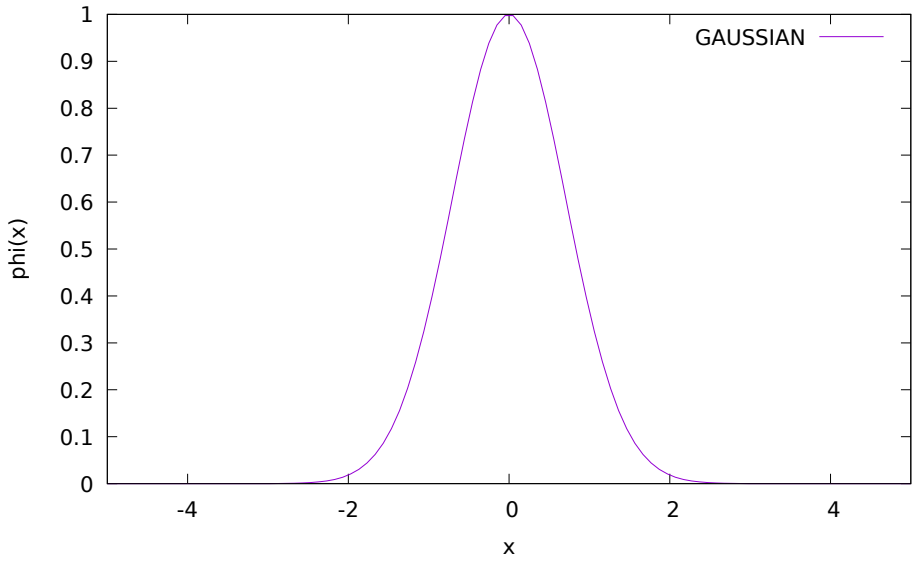

Figure 1. A typical plot for the Gaussian function.

RBF networks have been incorporated in various cases, such as face recognition [12], solutions of differential equations [13,14], stock prediction [15], robotics [16,17], network security [18,19] etc.

Due to the widespread use of these networks, several papers have been presented in recent years that study their basic characteristics. For example, Benoudjit et al [20] presented a discussion on kernel widths on RBF networks. Similarly, Oyang et al [21] presented a novel method for the estimation of the kernel density. Moreover, Ros et al [22] proposed an automatic method for the initialization of the parameters of RBF networks. Furthermore, a variety of pruning techniques have also been proposed [23,24] used to reduce efficiently the number of processing units of RBF networks in order to avoid the overfitting problem. Also, for the effective training of RBF networks a variety of methods has been proposed in the recent literature, such as the incorporation of Genetic Algorithms [25,26], usage of Particle Swam Optimization (PSO) method [27,28], the usage of methods based on Differential Evolution [29] etc. Furthermore, since there has been an extremely large spread of parallel programming techniques in recent decades, publications have also appeared that exploit such techniques for the efficient and fast training of the above networks [30,31].

In most cases, RBF networks are trained using a two-stage technique. In the first stage, the centers and variances are estimated using the K-means technique. In the second stage, a linear system is solved to find the weights of the Gaussian units. Although the above procedure is extremely fast, it cannot effectively explore the value space for the network parameters and in many cases numerical stability problems occur when solving the linear system. This paper proposes a three-stage method for the efficient training of RBF networks. In the first stage, an initial estimate of the centers and fluctuations of the RBF network is made using the K-means technique. Subsequently, an interval generation method based on Simulated Annealing [32] is used to efficiently identify the optimal interval of network

parameter values. The creation of the value interval is done within a value interval that is based on the initial estimate of the network parameters made in the first phase of the method. In the last phase of the proposed technique, a genetic algorithm is used to train the parameters of the machine learning model within the optimal value interval resulting from the second stage of the method.

The rest of this article is divided as follows: in section 2 the proposed method is presented in detail, in section 3 the used datasets as well as the conducted experiments are presented and finally, in section 4 some conclusions are discussed as well as some guidelines for improvements of the current work.

2. Method description

In this section, a detailed presentation of the three phases of the proposed technique is made using appropriate algorithms.

2.1. The first phase of the proposed method

A typical diagram of an RBF network is depicted in Figure 2. Every RBF network with k weights has the following sets of parameters:

1. A series of vectors $\overrightarrow{c_i}$, $i = 1, .., k$ that represent the centers of the network. Each center has d values.
2. Each Gaussian unit has an an additional parameter σ_i, that corresponds to the variance of this unit.
3. The set of output weights $\overrightarrow{w}$ with k values.

Provided that the input vector $\overrightarrow{x}$ has d values, the total number of parameters of an RBF network is calculated as:

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

For the determination of initial values for the centers and variances the K-means algorithm is utilized here and it is presented in Algorithm 1.

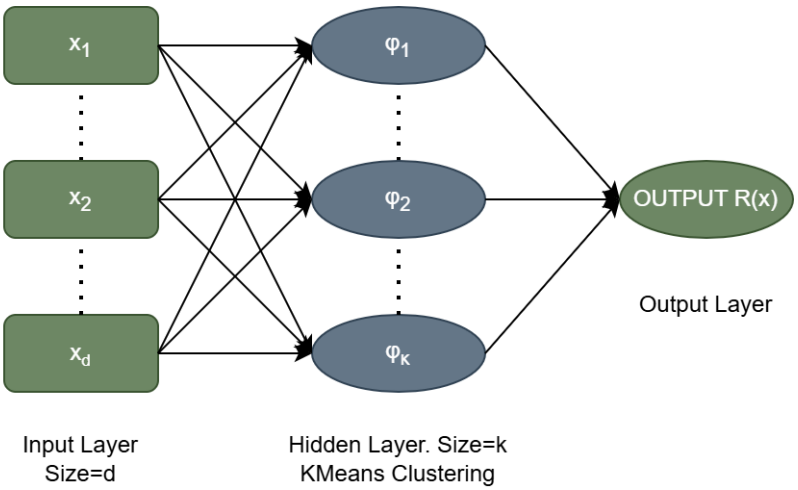

Figure 2. A typical diagram of an RBF network.

Algorithm 1 Description of the K-means algorithm.

1. **Initialization**
   (a) The input of the algorithm is the points $x_i,\ i = 1, \ldots, M$ belonging to the train set of the objective problem,
   (b) **Define** as k the number of centers.
   (c) **Set** $S_j = \{\ \},\ j = 1, \ldots, k$.
2. Repeat
   (a) **For** each point $x_i,\ i = 1, ..., M$ **do**
      i. **Set** $j^* = \text{argmin}_{m=1}^{k}\{D(x_i, c_m)\}$. The index j^* denotes the nearest center for pointx_i. The function $D(x, y)$ represents the Euclidean distance.
      ii. **Set** $S_{j^*} = S_{j^*} \cup \{x_i\}$.
      iii. **End For**
   (b) **For** every center $c_j,\ j = 1..k$ **do**
      i. **Set** M_j as the number of points in S_j
      ii. **Update** the center c_j with the following equation:

$$c_j = \frac{1}{M_j} \sum_{x_i \in S_j} x_i$$

   (c) **End For**
3. The algorithm terminate when c_j no longer changes
4. The output of the algorithm are the $\overrightarrow{c_i},\ i = 1, .., k$ centers and the corresponding variances $\sigma_i,\ i = 1, \ldots, k$

When this process is finished, the vector $\overrightarrow{z}$ with the calculated parameters is formed with the following procedure:

1. **Set** $l = 1$
2. **For** $i = 1, \ldots, k$ **do**
   (a) **For** $j = 1, \ldots, d$ **do**
      i. **Set** $z_l = c_{i,j}$
      ii. **Set** $l = l + 1$
   (b) **End For**
3. **End For**
4. **For** $i = 1, \ldots, k$ do
   (a) **Set** $z_l = \sigma_i$
   (b) **Set** $l = l + 1$
5. **End For**
6. **For** $i = 1, \ldots, k$ do
   (a) **Set** $z_l = w_0$, where w_0 a positive double precision number.
   (b) **Set** $l = l + 1$
7. **End For**

The layout of the vector $\overrightarrow{z}$ is outlined in Figure 3. In this layout, the initial values of the centers are entered at the beginning of the vector, then the initial values for the variances are entered, and finally the initial values for the network weights.

Figure 3. The scheme of the particles in the proposed method.

c_{11}	c_{12}	...	c_{1d}	c_{21}	c_{22}	...	c_{2d}	...	c_{k1}	c_{k2}	...	c_{kd}	σ_1	σ_2	...	σ_k	w_1	w_2	...	w_k

2.2. The second phase of the proposed method

In the second phase of the algorithm, the value interval for the network parameters is constructed using a technique based on Simulated Annealing. The Simulated Annealing is an optimization procedure used in a variety of cases, such as resource allocation [33], portfolio problems [34], energy problems [35], biology [36] etc. The algorithm starts from large values of a factor called temperature, which is gradually decreased. For large values of temperature the algorithm performs a wide exploration of the search space and at low values it focuses around some minimum of the objective function. The Simulated Annealing variant used here minimizes interval functions denoted as:

$$f(x) = [f_{\min}(x), f_{\max}(x)] \tag{5}$$

Also, in order to compare two intervals $a = [a_1, a_2]$ and $b = [b_1, b_2]$ the operator $D(a, b)$ is incorporated with the following definition:

$$D(a,b) = \begin{cases} \text{TRUE}, & a_1 < b_1 \text{ OR } (a_1 = b_1 \text{ AND } a_2 < b_2) \\ \text{FALSE}, & \text{otherwise} \end{cases} \tag{6}$$

The method used below assumes that there are value intervals for the network parameters. The calculation of the functional values for such value intervals is performed using Algorithm 2. The main steps of the method used in the second phase have as follows:

1. **Initialization step**.
   (a) **Construct** the bound vectors L^*, R^* using the vector $\overrightarrow{z}$ of the first phase as follows:
   $$\begin{aligned} L_i^* &= -F \times z_i \quad , \quad i = 1, \ldots, n \\ R_i^* &= F \times z_i \quad , \quad i = 1, \ldots, n \end{aligned}$$
   (b) **Set** $k = 0$, $T_0 > 0$, $\epsilon > 0$,$r_T > 0$, $r_T < 1$. The value T_0 is used to represent the temperature of the algorithm.
   (c) **Set** $N_{\text{eps}} > 0$, which is a positive integer and it is used as the number of samples that will be created in each iteration of the algorithm.
   (d) **Set** $a > 0$ a positive double precision number with the property $a < 1$, that will be used a the perturbation factor for the production of new samples.
   (e) **Set** $N_s > 0$ a positive integer number.
   (f) **Set** $x_b = [L^*, R^*], y_b = \text{fitness}(L^*, R^*, N_s)$.
   (g) **Set** $x_c = x_b, y_c = y_b$
2. **Main step**.
   (a) **For** $i = 1, \ldots, N_{\text{eps}}$ **do**
      i. **Produce** a random sample $x_t = (L_t, R_t)$ using the procedure of Algorithm 3 using as inputs to the function the interval $x_c = [L_c, R_c]$ and the double precision number a.
      ii. **Set** $y_t = \text{fitness}(L_t, R_t, N_s)$.
      iii. **If** $D(y_t, y_c) = \text{TRUE}$ **then Set** $x_c = x_t, y_c = y_t$
      iv. **Else Set** $x_c = x_t$ with probability $\min\left\{1, \exp\left(-\frac{f_t - f_c}{T_k}\right)\right\}$
      v. **End if**
   (b) **If** $D(x_c, x_b) = \text{TRUE}$ **then Set** $x_b = x_c, y_b = y_c$
   (c) **End For**
3. **Update temperature step**.
   (a) **Set** $T_{k+1} = T_k r_T$
   (b) **Set** $k = k + 1$.
   (c) **If** $T_k \leq \epsilon$ **stop** and **return** $x_b = [L_b, R_b]$ as the best located interval of bounds.
   (d) **Goto Main Step**.

Algorithm 2 The algorithm used to calculate the function value for intervals of parameters.

function fitness(L, R, N_s)

1. **Create** the set $T = \{r_1, r_2, \ldots, r_{N_s}\}$ with N_s random samples in $[L, R]$.
2. **Set** $E_{\min} = \infty$, $E_{\max} = -\infty$
3. **For** $i = 1, \ldots, N_s$ **do**
   (a) **Create** the RBF network $R_i(\overrightarrow{x})$ using as parameter set the corresponding sample $r_i \in T$.
   (b) **Calculate** the training error $f_{R_i} = \sum_{j=1}^{M} \left(R_i(\overrightarrow{x_j}) - y_j\right)^2$ for the training set of the objective problem.
   (c) **If** $f_{R_i} \leq E_{\min}$ **then** $E_{\min} = f_{R_i}$
   (d) **If** $f_{R_i} \geq E_{\max}$ **then** $E_{\max} = f_{R_i}$
4. **End For**
5. **Return** the interval $\left[E_{\min}, E_{\max}\right]$.

End function

Algorithm 3 The sampling function used in the Simulated Annealing algorithm.

Function sample(L, R, a)

1. **For** $i = 1, \ldots, n$ **do**
   (a) **Set** $m = L_i + \frac{R_i - L_i}{2}$
   (b) **Set** $L_i^x = L_i + ar_1(m - L_i)$, where $r_1 \in [0, 1]$ a random value.
   (c) Set $R_i^x = R_i - ar_2(R_i - m)$, where $r_2 \in [0, 1]$ a random value.
2. **End For**
3. **Return** $x = [L^x, R^x]$ the located interval.

End function

The algorithm starts from an initial value interval, which is based on the result of the first phase of the overall process. At each iteration it generates random value intervals that are close to the current value interval. If a value interval with a smaller functional value than the current one is presented it is accepted, otherwise it will be accepted with a probability that is high for large temperature values and decreases significantly as the temperature value drops. This means that the algorithm makes a wide exploration of the search space for high temperature values and centers around some minimum as the temperature value decreases.

2.3. The final phase of the proposed method

In the last phase of the proposed procedure, a genetic algorithm is executed to train the RBF network. The training is performed within the optimal value interval identified in the previous phase of the proposed procedure. The main steps of the proposed genetic algorithm have as follows:

1. **Initialization step**.
   (a) **Set** as N_g the number of maximum allowed generations and as N_c the number of chromosomes for the genetic population.
   (b) **Set** as p_s the selection rate with $p_s \leq 1$.
   (c) **Set** as p_m the mutation rate with $p_m \leq 1$.
   (d) **Initialize** the chromosomes g_i, $i = 1, \ldots, N_c$ as random vectors inside the bounds $[L_b, R_b]$ of the second phase of the suggested algorithm.
   (e) **Set** $k = 0$
2. **Fitness calculation step**.
   (a) **For** $i = 1, \ldots, N_c$ **do**

i. **Create** the RBF network $R_i(\vec{x})$ using as parameters the chromosome g_i.
ii. **Calculate** the corresponding fitness value $f_i = \sum_{j=1}^{M} \left(R_i(\vec{x_j}) - y_j\right)^2$

(b) **End For**

3. **Genetic operations step**.
   (a) **Selection procedure**. The $p_s \times N_c$ best chromosomes with the lowest fitness values are copied intact to the next generation. The remaining chromosomes will be replaced by new chromosomes produced in crossover and mutation procedure.
   (b) **Crossover procedure**. During the application of crossover, pairs of chromosomes are chosen from the current population with tournament selection. For every pair (z, w) of parents two new chromosomes $\tilde{z}$ and $\tilde{w}$ are will be produced using the following equations:

$$\begin{aligned} \tilde{z}_i &= a_i z_i + (1 - a_i) w_i \\ \tilde{w}_i &= a_i w_i + (1 - a_i) z_i \end{aligned} \tag{7}$$

   where $i = 1, \ldots, n$ and a_i are randomly selected values with $a_i \in [-0.5, 1.5]$ [37].
   (c) **Mutation procedure**. For every element of each chromosome a random number $r \in [0, 1]$ is produced. This element will be altered randomly when $r \leq p_m$.
4. **Termination check step**.
   (a) **Set** $k = k + 1$
   (b) **If** $k < N_g$ **then** go to Fitness calculation step.
5. **Testing step**.
   (a) **Obtain** the best chromosome g_b from the genetic population, with the lowest fitness value.
   (b) **Create** the corresponding RBF network $R_b(\vec{x})$
   (c) **Apply** $R_b(\vec{x})$ to the corresponding test set and measure the associated test error.

3. Experiments

A series of classification and regression datasets from various websites was incorporate to test the proposed method and to measure its reliability. The datasets used in the conducted experiments were downloaded from the following online databases:

1. The UCI database https://archive.ics.uci.edu/(accessed on 26 March 2025)[38]
2. The Keel website, https://sci2s.ugr.es/keel/datasets.php(accessed on 26 March 2025)[39]
3. The Statlib URL ftp://lib.stat.cmu.edu/datasets/index.html(accessed on 26 March 2025).

3.1. Experimental datasets

In the conducted experiments the following classification datasets were used:

1. The Alcohol dataset, used in various experiments regarding alcohol consumption [40].
2. The Appendicitis dataset [41].
3. The Australian dataset, used in various bank transactions [42].
4. The Balance dataset, used in a series of psychological experiments [43].
5. The Circular dataset, which is an artificial dataset.
6. The Cleveland dataset, which is a medical dataset [44,45].
7. The Dermatology dataset, obtained from various dermatology problems [46].
8. The Hayes roth dataset [47].
9. The Heart dataset, which is a medical dataset regarding heart diseases [48].

10. The HeartAttack dataset, used in the prediction of heart attacks.
11. The HouseVotes dataset, that contains data from Congressional voting in USA [49].
12. The Ionosphere dataset, obtained from various experiments in the ionosphere [50,51].
13. The Liverdisorder dataset, which is a medical dataset [52,53].
14. The Lymography dataset [54].
15. The Mammographic dataset, which is a medical dataset [55].
16. The Parkinsons dataset, used for the the detection of Parkinson's disease [56,57].
17. The Phoneme datasets, obtained from various sound experiments.
18. The Pima dataset, a medical dataset used for in the diabetes's disease [58].
19. The Popfailures dataset, related to measurements from climate model simulations [59].
20. The Regions2 dataset, which is a medical dataset [60].
21. The Saheart dataset, which is a medical dataset about heart diseases [61].
22. The Segment dataset, related to issues regarding image processing [62].
23. The Sonar dataset, related to sonar signals [63].
24. The Spiral dataset, which is an artificial dataset.
25. The StatHeart dataset, a medical dataset related to heart diseases.
26. The Student dataset, derived from various experiments in schools [64].
27. The Transfusion dataset [65].
28. The WDBC dataset, which is a medical dataset about the detection of cancer [66].
29. The Wine dataset, related to the detection of the quality of wines [67,68].
30. The EEG dataset, which is obtained from various EEG measurements [69,70]. The cases Z_F_S, Z_O_N_F_S, ZO_NF_S and ZONF_S were used from this dataset.
31. The ZOO dataset, used for animal classification [71].

Moreover, the following regression datasets were obtained for the conducted experiments:

1. The Abalone dataset, that contains data related to the age of abalones [72].
2. The Airfoil dataset, obtained from NASA [73].
3. The Auto dataset, used to estimate the fuel consumption in cars.
4. The Baseball dataset, related to the estimation of salary of baseball players.
5. The BK dataset, used for the prediction of points in basketball games [74].
6. The BL dataset, used in some some electricity experiments.
7. The Concrete dataset, derived from civil engineering [75].
8. The Dee dataset, used for the estimation of the price of electricity.
9. The Housing dataset, used to estimate the price of houses [76].
10. The Friedman datase[77].
11. The FA dataset, which is related to fat measurements.
12. The FY dataset, that contains data regarding the fruit flies.
13. The HO dataset, that was derived from the STATLIB repository.
14. The Laser dataset, used in various laser experiments.
15. The MB dataset, derived from the Smoothing Methods in Statistics.
16. The Mortgage dataset, that contains economic measurements.
17. The NT dataset [78].
18. The Plastic dataset, that contains measurements form experiments conducted for the pressure in plastics.
19. The PY dataset [79].
20. The PL dataset, downloaded from the STATLIB repository.
21. The Quake dataset, related to the estimation of the strength of earthquakes.
22. The SN dataset, that contains measurements about trellising and pruning.
23. The Stock dataset, which is an economic dataset about the prediction of the price of stocks.
24. The Treasury dataset, which contains economic measurements.

3.2. Experimental results

The code used in the experiments was implemented in the C++ programming language and a machine equipped with 128GB RAM running Debian Linux was utilized in the conducted experiments. Each experiment was executed 30 times and the average classification error was measured for the classification datasets and the average regression error was measured for the regression datasets. The classification error is computed using the following equation:

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

where the set $T = \{x_i, y_i\}$, $i = 1, \ldots, K$ stands for the test set of the current problem and $R(x)$ is the RBF model. The regression error is calculated through the following equation:

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

The values for the parameters of the proposed method are mentioned in Table 1. In the following tables that describe the experimental results the following notation is used:

1. The column DATASET represents the name of the objective problem.
2. The column BFGS denotes the application of the BFGS optimization method [80] in the training of a neural network [81,82] with 10 processing nodes.
3. The column ADAM stands for the incorporation of the ADAM optimizer [83] to train an artificial neural network with 10 processing nodes.
4. The column NEAT represents the usage of the NEAT method (NeuroEvolution of Augmenting Topologies) [84].
5. The column RBF-KMEANS stands for the usage of the original two - phase method to train an RBF network with 10 processing nodes.
6. The column GENRBF represents the incorporation of the method proposed in [85] to train an RBF network with 10 processing nodes.
7. The column PROPOSED denotes the usage of the proposed method to train an RBF network with 10 processing nodes.
8. The row AVERAGE represents the average classification or regression error.

Table 1. The values for the experimental parameters.

PARAMETER	MEANING	VALUE
w_0	Initial values for the weights	100.0
F	Scale factor used in the initialization	10.0
T_0	Initial temperature	10^6
ϵ	Small value used in comparisons	10^{-6}
a	Perturbation factor	0.001
N_s	Number of samples used in the fitness calculation	100
N_{eps}	Number of samples taken in Simulated Annealing	100
N_c	Number of chromosomes	500
N_g	Maximum number of allowed generations	200
p_s	Selection rate	0.1
p_m	Mutation rate	0.05

The results from the application of the previously mentioned machine learning methods to the classification datasets are depicted in Table 2 and for regression datasets the results are presented in Table 3.

Table 2. Experimental results for the classification datasets using the series of machine learning methods adopted here. The numbers in cells represent average classification error as measured on the corresponding test set.

DATASET	BFGS	ADAM	NEAT	RBF-KMEANS	GENRBF	PROPOSED
Alcohol	41.50%	57.78%	66.80%	49.38%	52.45%	31.28%
Appendicitis	18.00%	16.50%	17.20%	12.23%	16.83%	15.27%
Australian	38.13%	35.65%	31.98%	34.89%	41.79%	21.00%
Balance	8.64%	7.87%	23.14%	33.42%	38.02%	12.95%
Cleveland	77.55%	67.55%	53.44%	67.10%	67.47%	50.82%
Circular	6.08%	19.95%	35.18%	5.98%	21.43%	4.19%
Dermatology	52.92%	26.14%	32.43%	62.34%	61.46%	36.13%
Hayes Roth	37.33%	59.70%	50.15%	64.36%	63.46%	33.54%
Heart	39.44%	38.53%	39.27%	31.20%	28.44%	15.33%
HeartAttack	46.67%	45.55%	32.34%	29.00%	40.48%	18.52%
HouseVotes	7.13%	7.48%	10.89%	6.13%	11.99%	3.74%
Ionosphere	15.29%	16.64%	19.67%	16.22%	19.83%	7.39%
Liverdisorder	42.59%	41.53%	30.67%	30.84%	36.97%	27.92%
Lymography	35.43%	29.26%	33.70%	25.50%	29.33%	20.64%
Mammographic	17.24%	46.25%	22.85%	21.38%	30.41%	17.21%
Parkinsons	27.58%	24.06%	18.56%	17.41%	33.81%	15.35%
Phoneme	15.58%	29.43%	22.34%	23.32%	26.29%	16.62%
Pima	35.59%	34.85%	34.51%	25.78%	27.83%	23.59%
Popfailures	5.24%	5.18%	7.05%	7.04%	7.08%	4.80%
Regions2	36.28%	29.85%	33.23%	38.29%	39.98%	25.54%
Saheart	37.48%	34.04%	34.51%	32.19%	33.90%	29.64%
Segment	68.97%	49.75%	66.72%	59.68%	54.25%	40.83%
Sonar	25.85%	30.33%	34.10%	27.90%	37.13%	18.25%
Spiral	47.99%	48.90%	50.22%	44.87%	50.02%	22.52%
Statheart	39.65%	44.04%	44.36%	31.36%	42.94%	19.52%
Student	7.14%	5.13%	10.20%	5.49%	33.26%	5.11%
Transfusion	25.84%	25.68%	24.87%	26.41%	25.67%	24.59%
Wdbc	29.91%	35.35%	12.88%	7.27%	8.82%	5.00%
Wine	59.71%	29.40%	25.43%	31.41%	31.47%	7.71%
Z_F_S	39.37%	47.81%	38.41%	13.16%	23.37%	3.16%
Z_O_N_F_S	65.67%	78.79%	77.08%	48.70%	68.40%	46.77%
ZO_NF_S	43.04%	47.43%	43.75%	9.02%	22.18%	3.63%
ZONF_S	15.62%	11.99%	5.44%	4.03%	17.41%	1.79%
ZOO	10.70%	14.13%	20.27%	21.93%	33.50%	4.50%
AVERAGE	**32.98%**	**33.60%**	**32.46%**	**28.39%**	**34.64%**	**18.67%**

The table 2 presents the error rates of various machine learning models (BFGS, ADAM, NEAT, RBF-KMEANS, GENRBF, PROPOSED) on different classification datasets. Each row corresponds to a dataset, while each column represents the error rate of a specific model. These values indicate the percentage of incorrect predictions, with lower values reflecting better performance. The last row of the table includes the average error rates for each model. Statistical analysis of the data reveals significant insights. The PROPOSED model exhibits the lowest average error rate (18.67%) compared to the other models, establishing it as the optimal choice based on the table. Conversely, the other models demonstrate higher average error rates, with GENRBF showing the highest average (34.64%). Additionally, significant variations in error rates across datasets are observed. For instance, in the "Circular" and "ZONF_S" datasets, the PROPOSED model outperforms others with very low error rates (4.19% and 1.79%, respectively). Conversely, in datasets like "Cleveland," the NEAT model shows a lower error rate (53.44%) compared to the PROPOSED model (50.82%). Notably, in certain datasets, the performance of the PROPOSED model is significantly inferior to

other models. For example, in the "Alcohol" and "Z_F_S" datasets, the PROPOSED model exhibits much higher error rates compared to other models. This indicates that while the PROPOSED model generally has the lowest average error rate, its performance may not be consistent across all datasets. In conclusion, the PROPOSED model emerges as the best general choice for minimizing error rates, though its evaluation depends on the characteristics of each dataset. The performance differences among models highlight the need for careful model selection depending on the application.s

Table 3. Experimental results for regression datasets. Numbers in cells represent average regression error as calculated on the corresponding test set.

DATASET	BFGS	ADAM	NEAT	RBF-KMEANS	GENRBF	PROPOSED
Abalone	5.69	4.30	9.88	7.37	9.98	5.10
Airfoil	0.003	0.005	0.067	0.27	0.121	0.004
Auto	60.97	70.84	56.06	17.87	16.78	9.68
Baseball	119.63	77.90	100.39	93.02	98.91	86.19
BK	0.28	0.03	0.15	0.02	0.023	0.153
BL	2.55	0.28	0.05	0.013	0.005	0.0002
Concrete	0.066	0.078	0.081	0.011	0.015	0.006
Dee	2.36	0.630	1.512	0.17	0.25	0.16
Housing	97.38	80.20	56.49	57.68	95.69	18.70
Friedman	1.26	22.90	19.35	7.23	16.24	1.45
FA	0.426	0.11	0.19	0.015	0.15	0.019
FY	0.22	0.038	0.08	0.041	0.041	0.077
HO	0.62	0.035	0.169	0.03	0.076	0.01
Laser	0.015	0.03	0.084	0.03	0.075	0.003
MB	0.129	0.06	0.061	2.16	0.41	5.49
Mortgage	8.23	9.24	14.11	1.45	1.92	0.14
NT	0.129	0.12	0.33	8.14	0.02	0.007
PL	0.29	0.117	0.098	2.12	0.155	0.023
Plastic	20.32	11.71	20.77	8.62	25.91	2.29
PY	0.578	0.09	0.075	0.012	0.029	0.019
Quake	0.42	0.06	0.298	0.07	0.79	0.036
SN	0.40	0.026	0.174	0.027	0.027	0.024
Stock	302.43	180.89	12.23	12.23	25.18	1.53
Treasury	9.91	11.16	15.52	2.02	1.89	0.51
AVERAGE	**26.43**	**19.62**	**12.84**	**9.19**	**12.28**	**5.48**

The table 3 displays the absolute error values resulting from the application of various machine learning models (BFGS, ADAM, NEAT, RBF-KMEANS, GENRBF, PROPOSED) on regression datasets. Each row corresponds to a dataset, while each column shows the error of a specific model. The last row records the average error for each model. Lower error values indicate better model performance. Analysis shows that the PROPOSED model has the lowest average error (5.48), making it the most efficient choice among the available models. The second-best model is RBF-KMEANS, with an average error of 9.19, while other models, such as BFGS (26.43) and ADAM (19.62), exhibit significantly higher error values. The performance of the PROPOSED model is particularly impressive in datasets such as BL, where its error is nearly negligible (0.0002), and Mortgage, where it has a very low error (0.14) compared to other models. In datasets like Stock and Plastic, where errors are high across all models, the PROPOSED model still outperforms, with error values of 1.53 and 2.29, respectively, compared to the other models. However, there are instances where the performance difference of the PROPOSED model relative to others is small or even unfavorable. For example, in the Laser dataset, the ADAM model has an error of 0.03, slightly higher than the PROPOSED model's 0.003, while in the HO dataset, the PROPOSED model performs better (0.01), but the RBF-KMEANS model is comparably

close (0.03). In summary, the PROPOSED model achieves the lowest average error and the most consistent performance across most datasets, making it an ideal choice for regression problems. Nonetheless, certain models, such as RBF-KMEANS, may demonstrate competitive performance in specific cases, suggesting that model selection depends on the unique characteristics of each dataset.

The analysis of significance levels for the classification datasets, as illustrated in Figure 4, reveals that the proposed model statistically significantly outperforms all other models in every comparison pair. Specifically, the p-values indicate strong statistical differences: PROPOSED vs BFGS $\left(p = 10^{-8}\right)$, PROPOSED vs ADAM $\left(p = 6.2 \times 10^{-8}\right)$, PROPOSED vs NEAT $\left(p = 2.9 \times 10^{-9}\right)$, PROPOSED vs RBF-KMEANS $\left(p = 1.2 \times 10^{-6}\right)$ and PROPOSED vs GENRBF $\left(p = 1.2 \times 10^{-10}\right)$. These values suggest that the proposed model is significantly better than the others with high reliability.

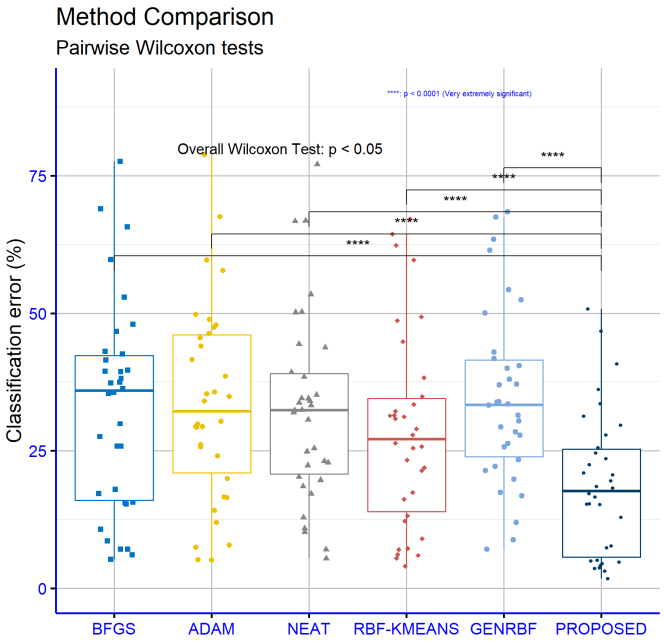

Figure 4. Statistical comparison of the experimental results for the classification datasets.

In Figure 5, which concerns the regression datasets, a similar pattern is observed, though the p-values are generally higher compared to the classification datasets. The proposed model demonstrates statistically significant superiority over the other models in all comparison pairs: PROPOSED vs BFGS ($p = 0.00011$), PROPOSED vs ADAM ($p = 0.015$), PROPOSED vs NEAT ($p = 0.00016$), PROPOSED vs RBF-KMEANS ($p = 0.0016$), and PROPOSED vs GENRBF ($p = 0.00049$). Although the significance is not as strong as in the classification datasets, the proposed model's superiority remains clear.

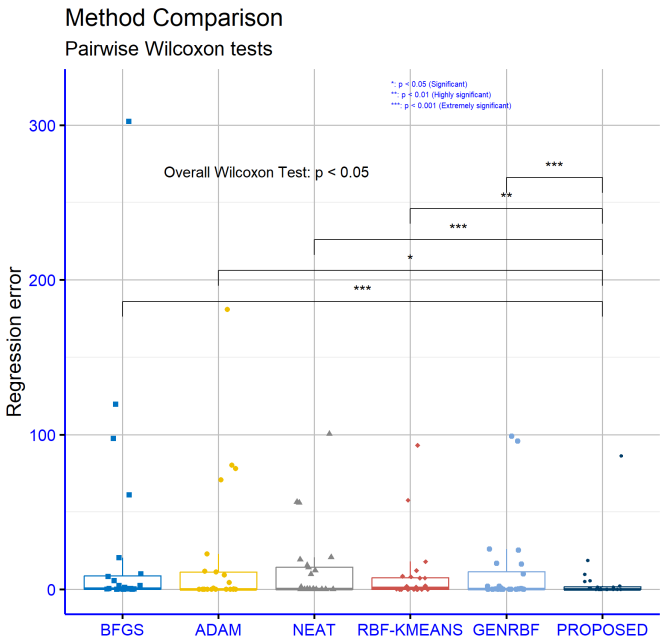

Figure 5. Statistical comparison of the obtained experimental results for the regression datasets.

3.3. Experiments with the perturbation factor a

In order to determine the stability of the proposed technique, another experiment was performed in which the perturbation factor a presented in the second stage of the proposed technique took a series of different values. The table 4 presents the error rates of the proposed machine learning model for three different values of the perturbation factor a (0.001, 0.005, 0.01) across various classification datasets. Each row represents a dataset, and the values indicate the model's error rate for each value of a. The last row includes the average error rates for each value of a. Analysis of the data shows that the smallest value of a (0.001) achieves the lowest average error rate (18.67%), while the largest value (0.01) results in the highest average (19.06%). This suggests that the model generally performs better with smaller values of a although the difference in averages is minimal. At the dataset level, there are cases where the model's performance is significantly affected by changes in the parameter. For instance, in the "Lymography" dataset, increasing a from 0.001 to 0.01 leads to a significant increase in the error rate from 20.64% to 30.33%. A similar trend is observed in the "ZOO" dataset, where the error rate rises from 4.50% to 6.87% for $a = 0.005$, but decreases again to 4.60% for $a = 0.01$. On the other hand, in datasets like "ZO_NF_S," the error remains unchanged at 3.63%, regardless of changes in a. Datasets such as "Z_F_S" and "ZONF_S" exhibit nonlinear behavior. In "Z_F_S," the error rate significantly decreases from 3.16% to 2.79% as a increases from 0.001 to 0.01, while in "ZONF_S," a similar decrease is observed from 1.79% to 1.74%. In conclusion, the analysis indicates that the perturbation factor a has a notable impact on the performance of the proposed model. Smaller values of a are generally associated with better performance; however, the optimal value may depend on the characteristics of each dataset. Instances where error rates increase or decrease nonlinearly with changes in "a" suggest the need for further investigation into the tuning of "a" for specific applications.

Table 4. Experimental results for the classification datasets using a series of values for perturbation factor a.

DATASET	$a = 0.001$	$a = 0.005$	$a = 0.01$
Alcohol	31.28%	30.88%	31.77%
Appendicitis	15.27%	15.77%	15.57%
Australian	21.00%	21.03%	20.77%
Balance	12.95%	13.13%	13.29%
Cleveland	50.82%	50.64%	51.11%
Circular	4.19%	4.01%	4.08%
Dermatology	36.13%	36.81%	36.67%
Hayes Roth	33.54%	33.44%	34.10%
Heart	15.33%	15.15%	15.22%
HeartAttack	18.52%	18.80%	18.88%
HouseVotes	3.74%	3.60%	4.12%
Ionosphere	7.39%	7.38%	7.29%
Liverdisorder	27.92%	28.27%	28.49%
Lymography	20.64%	20.57%	30.33%
Mammographic	17.21%	17.15%	17.14%
Parkinsons	15.35%	14.35%	15.23%
Phoneme	16.62%	16.74%	16.00%
Pima	23.59%	24.09%	23.99%
Popfailures	4.80%	4.82%	4.86%
Regions2	25.54%	25.62%	25.75%
Saheart	29.64%	29.93%	29.22%
Segment	40.83%	41.41%	41.96%
Sonar	18.25%	17.80%	17.83%
Spiral	22.52%	22.00%	22.27%
Statheart	19.52%	19.35%	19.58%
Student	5.11%	5.03%	5.25%
Transfusion	24.59%	24.70%	24.64%
Wdbc	5.00%	5.05%	5.07%
Wine	7.71%	7.88%	7.90%
Z_F_S	3.16%	3.66%	2.79%
Z_O_N_F_S	46.77%	46.83%	47.06%
ZO_NF_S	3.63%	3.63%	3.63%
ZONF_S	1.79%	1.81%	1.74%
ZOO	4.50%	6.87%	4.60%
AVERAGE	**18.67%**	**18.77%**	**19.06%**

The table 5 displays the absolute error values of the proposed machine learning model across various regression datasets for three different values of the perturbation factor a (0.001, 0.005, 0.01). Data analysis reveals that the parameter $a = 0.005$ yields the lowest average error (4.92), while the values $a = 0.001$ and $a = 0.01$ result in slightly higher averages (5.48 and 5.13, respectively). This difference indicates that 0.005 is generally the most suitable value for the model, ensuring better performance in most cases. At the dataset level, the impact of a varies. Some datasets, such as "Airfoil," "Concrete," "Dee," "HO," "Laser," "NT," "PL," "Plastic," and "Quake," show no change in error with variations in a as error values remain constant. In contrast, other datasets exhibit significant variations. For example, in the "Baseball" dataset, the error decreases from 86.19 for $a = 0.001$ to 77.46 for $a = 0.005$ then increases again to 81.97 for $a = 0.01$. Similarly, in the "MB" dataset, the error drastically decreases from 5.49 for $a = 0.001$ to 0.56 for $a = 0.005$ and further to 0.48 for $a = 0.01$. In datasets like "FA" and "FY," the error increases as a changes from 0.001 to 0.005, then decreases again for $a = 0.01$. In the "Treasury" dataset, the error shows a slight decline as a increases. In conclusion, the parameter a has a significant impact on

the model's performance in certain datasets, while in others, its effect is negligible. The lowest average error observed for $a = 0.005$ suggests that this value is generally optimal for the model, though further tuning may be required for specific datasets. Cases with high variability in errors highlight the need for deeper analysis and optimization of the a parameter based on the characteristics of each dataset.

Table 5. Experimental results for regression datasets using different values for parameter a.

DATASET	$a = 0.001$	$a = 0.005$	$a = 0.01$
Abalone	5.10	5.12	5.10
Airfoil	0.004	0.004	0.004
Auto	9.68	9.80	9.89
Baseball	86.19	77.46	81.97
BK	0.153	0.043	0.11
BL	0.0002	0.0002	0.0003
Concrete	0.006	0.006	0.006
Dee	0.16	0.16	0.16
Housing	18.70	18.73	19.20
Friedman	1.45	1.44	1.45
FA	0.019	0.09	0.07
FY	0.077	0.12	0.076
HO	0.01	0.01	0.01
Laser	0.003	0.003	0.003
MB	5.49	0.56	0.48
Mortgage	0.14	0.12	0.13
NT	0.007	0.007	0.007
PL	0.023	0.023	0.023
Plastic	2.29	2.29	2.29
PY	0.019	0.019	0.017
Quake	0.036	0.036	0.036
SN	0.024	0.025	0.024
Stock	1.53	1.52	1.52
Treasury	0.51	0.54	0.47
AVERAGE	**5.48**	**4.92**	**5.13**

Figure 6 compares different values of parameter a for the classification datasets. The p-values for the comparisons $a = 0.001$ vs $a = 0.005$ ($p = 0.36$), $a = 0.001$ vs $a = 0.01$ ($p = 0.071$), and $a = 0.005$ vs $a = 0.01$ ($p = 0.17$) indicate that the differences between the parameter values are not statistically significant. This suggests that varying parameter a within this range does not substantially affect the model's performance on these datasets.

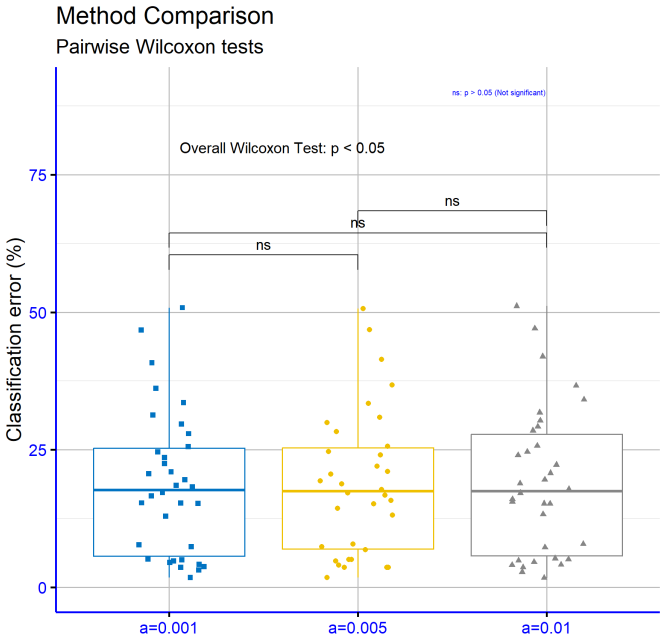

Figure 6. Statistical comparison of the obtained results from the application of the current method in the classification datasets using different values of perturbation factor a.

Figure 7 presents corresponding comparisons for the regression datasets, where a similar result is observed. The p-values for the comparisons $a = 0.001$ vs $a = 0.005$ ($p = 0.75$), $a = 0.001$ vs $a = 0.01$ ($p = 0.41$), and $a = 0.005$ vs $a = 0.01$ ($p = 0.94$) indicate the absence of statistically significant differences. This shows that the choice of parameter a does not significantly influence the model's performance on the regression datasets.

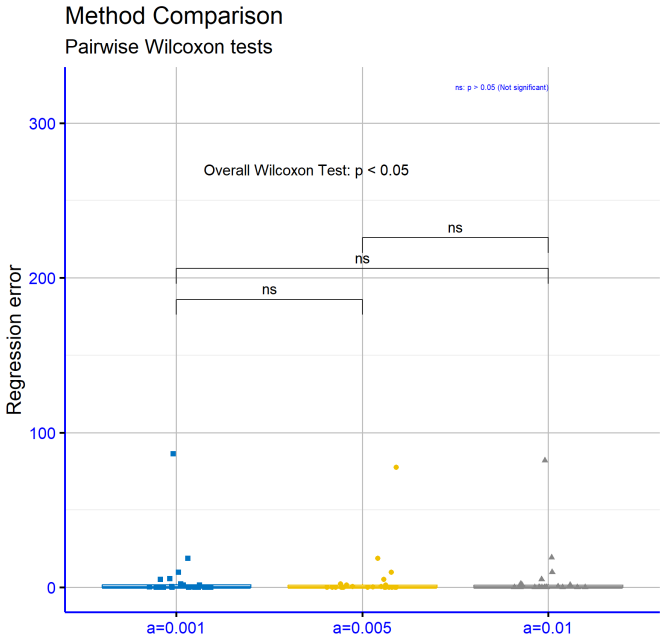

Figure 7. Statistical comparison of the obtained experiments results from the application of the proposed method to the regression datasets, using different values of the perturbation factor a.

3.4. Experiments with the parameter F

Another experiment was conducted using the initialization factor F. Table 6 presents the percentage error rates of the proposed machine learning model across various classification datasets for four different values of the parameter F (1.5, 3.0, 5.0, 10.0). Analyzing the data reveals that the parameter F influences the model's performance, but this effect varies by dataset. The lowest average error rate is observed for $F = 5.0$ (18.58%), indicating that this value is generally optimal. For the other values, slightly higher average error rates are noted: 18.88% for $F = 3.0$ 18.67% for $F = 10.0$ and the highest rate, 20.32%, for $F = 1.5$. Examining individual datasets, it is evident that for many of them, increasing F improves performance, as reflected in reduced error rates. Examples include the "Ionosphere," "Wine," and "ZONF_S" datasets, where error rates decrease as F increases. In "Ionosphere," the error rate drops from 12.92% for $F = 1.5$ to 7.39% for $F = 10.0$. In "Wine," the error rate decreases from 10.90% for $F = 1.5$ to 7.71% for $F = 10.0$. Similarly, in "ZONF_S," the error rate steadily decreases from 2.59% for $F = 1.5$ to 1.79% for $F = 10.0$. However, there are cases where increasing F does not lead to improvement or results in higher error rates. For example, in the "Segment" dataset, the error rate rises from 35.81% for $F = 1.5$ to 40.83% for $F = 10.0$. In the "Spiral" dataset, the error rate consistently increases from 13.28% for $F = 1.5$ to 22.52% for $F = 10.0$. A similar trend is observed in the "Z_O_N_F_S" dataset, where the error rate rises from 46.00% for $F = 1.5$ to 46.77% for $F = 10.0$. Overall, the parameter F significantly affects the model's performance, and the optimal value appears to be $F = 5.0$ as evidenced by the lowest average error rate. However, the exact impact depends on the characteristics of each dataset, emphasizing the need to fine-tune the parameter value for specific datasets to achieve optimal performance.

Table 6. Experimental results for the classification datasets using a series of values for parameter F.

DATASET	$F = 1.5$	$F = 3.0$	$F = 5.0$	$F = 10.0$
Alcohol	25.66%	29.16%	26.14%	31.28%
Appendicitis	16.30%	14.57%	15.50%	15.27%
Australian	23.53%	22.27%	20.81%	21.00%
Balance	15.12%	13.32%	12.68%	12.95%
Cleveland	51.81%	51.41%	50.70%	50.82%
Circular	4.75%	4.15%	4.52%	4.19%
Dermatology	36.69%	36.48%	36.39%	36.13%
Hayes Roth	46.18%	35.54%	34.18%	33.54%
Heart	16.68%	15.93%	15.68%	15.33%
HeartAttack	27.39%	20.38%	19.03%	18.52%
HouseVotes	3.80%	3.35%	3.85%	3.74%
Ionosphere	12.92%	8.27%	7.41%	7.39%
Liverdisorder	30.48%	29.27%	28.48%	27.92%
Lymography	29.89%	22.41%	21.93%	20.64%
Mammographic	18.00%	17.17%	16.96%	17.21%
Parkinsons	18.25%	17.18%	15.90%	15.35%
Phoneme	17.27%	15.88%	15.90%	16.62%
Pima	24.54%	24.17%	24.05%	23.59%
Popfailures	7.07%	5.35%	5.01%	4.80%
Regions2	26.07%	26.02%	25.78%	25.54%
Saheart	29.75%	28.91%	29.42%	29.64%
Segment	35.81%	36.84%	38.93%	40.83%
Sonar	24.68%	19.25%	16.98%	18.25%
Spiral	13.28%	15.25%	17.88%	22.52%
Statheart	19.98%	19.58%	19.63%	19.52%
Student	6.14%	6.30%	5.92%	5.11%
Transfusion	25.45%	25.23%	25.19%	24.59%
Wdbc	4.94%	4.92%	4.90%	5.00%
Wine	10.90%	9.37%	8.51%	7.71%
Z_F_S	4.13%	3.73%	3.67%	3.16%
Z_O_N_F_S	46.00%	45.61%	46.57%	46.77%
ZO_NF_S	3.67%	4.19%	3.16%	3.63%
ZONF_S	2.59%	2.37%	2.06%	1.79%
ZOO	11.17%	8.10%	8.00%	4.50%
AVERAGE	**20.32%**	**18.88%**	**18.58%**	**18.67%**

Table 7 provides the absolute error values of the proposed machine learning model across various regression datasets for four different values of the parameter F (1.5, 3.0, 5.0, 10.0). The data analysis shows that the parameter F affects the model's performance differently depending on the dataset. The average errors indicate that $F = 5.0$ yields the lowest overall error (5.22), followed by $F = 3.0$ with an average of 5.25. Higher averages are observed for $F = 1.5$ (5.52) and $F = 10.0$ (5.48), suggesting that deviating from $F = 5.0$ tends to increase error in some cases. Examining the datasets, it is evident that in several cases, increasing F improves performance, reducing error rates. For example, in the "Abalone" dataset, the error decreases from 6.39 for $F = 1.5$ to 5.10 for $F = 10.0$. Similarly, in the "Friedman" dataset, the error significantly decreases from 6.59 for $F = 1.5$ to 1.45 for $F = 10.0$. In the "Laser" dataset, the error decreases progressively from 0.022 for $F = 1.5$ to 0.003 for $F = 10.0$. Conversely, there are datasets where the effect of F is nonlinear or increases the error rate. For instance, in the "Housing" dataset, the error rises from 16.75 for $F = 1.5$ to 18.70 for $F = 10.0$. In the "MB" dataset, there is a sharp increase in error from 0.116 for $F = 1.5$ to 5.49 for $F = 10.0$, indicating that F significantly impacts model performance for this dataset. In summary, the parameter F has varying effects on

the model's performance across different datasets. While the average indicates that $F = 5.0$ is the optimal choice, precise optimization of the parameter should be dataset-specific. Additionally, extreme parameter values may lead to significant performance degradation in certain datasets, as seen in examples like "MB" and "Housing."

Table 7. Experimental results for regression datasets using different values for parameter F.

DATASET	$F = 1.5$	$F = 3.0$	$F = 5.0$	$F = 10.0$
Abalone	6.39	5.79	5.57	5.10
Airfoil	0.004	0.004	0.004	0.004
Auto	9.83	9.69	9.67	9.68
Baseball	84.27	83.01	84.57	86.19
BK	0.275	0.048	0.071	0.153
BL	0.41	0.0005	0.0003	0.0002
Concrete	0.006	0.006	0.006	0.006
Dee	0.16	0.16	0.16	0.16
Housing	16.75	17.82	18.07	18.70
Friedman	6.59	3.85	1.67	1.45
FA	0.054	0.03	0.053	0.019
FY	0.216	0.246	0.332	0.077
HO	0.01	0.01	0.01	0.01
Laser	0.022	0.011	0.005	0.003
MB	0.116	0.135	0.307	5.49
Mortgage	0.56	0.59	0.36	0.14
NT	0.007	0.007	0.007	0.007
PL	0.024	0.023	0.023	0.023
Plastic	2.37	2.33	2.31	2.29
PY	2.33	0.049	0.022	0.019
Quake	0.036	0.036	0.036	0.036
SN	0.028	0.04	0.025	0.024
Stock	1.45	1.51	1.49	1.53
Treasury	0.50	0.59	0.43	0.51
AVERAGE	**5.52**	**5.25**	**5.22**	**5.48**

In Figure 8, which compares different values of parameter F for the classification datasets, several statistically significant differences are observed. The p-values for the comparisons $F = 1.5$ vs $F = 3.0$ ($p = 0.00019$), $F = 1.5$ vs $F = 5.0$ ($p = 0.00012$), and $F = 1.5$ vs $F = 10.0$ ($p = 0.00069$) indicate a strong difference in the model's performance. In contrast, the values for the comparisons $F = 3.0$ vs $F = 5.0$ ($p = 0.027$), $F = 3.0$ vs $F = 10.0$ ($p = 0.062$), and $F = 5.0$ vs $F = 10.0$ ($p = 0.23$) show that the differences between larger values of parameter F are less significant.

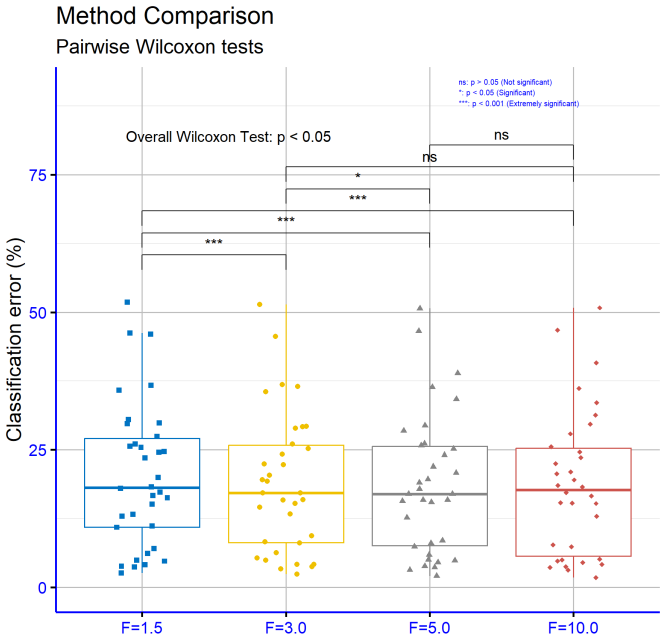

Figure 8. Statistical comparison for the experimental results by the application of the proposed method with different values of parameter F. The method was applied on the classification datasets.

Figure 9 examines comparisons of parameter F for the regression datasets and shows no statistically significant differences. The p-values for the comparisons $F = 1.5$ vs $F = 3.0$ ($p = 0.18$), $F = 1.5$ vs $F = 5.0$ ($p = 0.15$), $F = 1.5$ vs $F = 10.0$ ($p = 0.21$), $F = 3.0$ vs $F = 5.0$ ($p = 0.7$), $F = 3.0$ vs $F = 10.0$ ($p = 0.54$), and $F = 5.0$ vs $F = 10.0$ ($p = 0.89$) indicate that variations in the value of parameter F do not significantly affect the model's performance on the regression datasets. This may suggest greater stability of the model to changes in this parameter compared to the classification datasets.

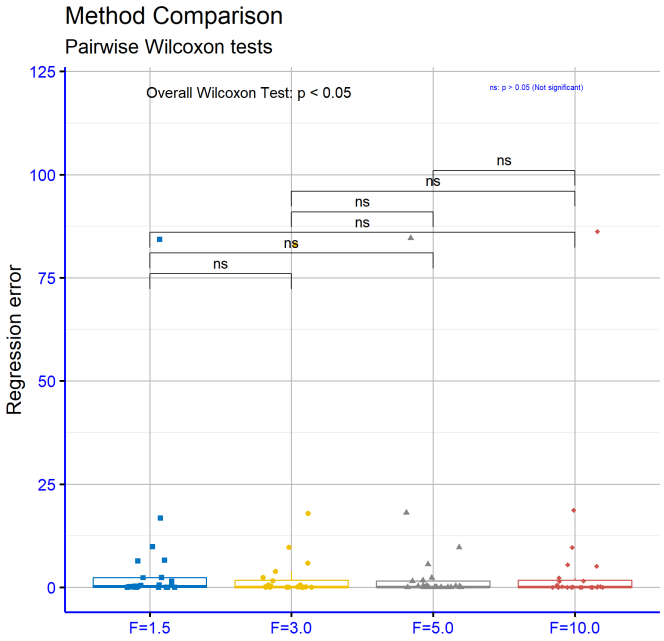

Figure 9. Statistical comparison for the obtained results by the application of the proposed method to the regression datasets. For this experiment different values of parameter F were used.

4. Conclusions

The article focuses on optimizing the parameter tuning process in Radial Basis Function networks through a multidimensional and innovative approach that combines techniques such as Simulated Annealing and genetic algorithms. The proposed method surpasses traditional two-stage approaches, where parameters are typically determined using fixed processes like K-means clustering followed by a split between training and validation phases. Instead, the article introduces a three-phase process. In the first phase, initial parameter estimation is performed using K-means, ensuring a stable starting point. In the second phase, the application of Simulated Annealing provides an advanced mechanism for exploring the parameter space, avoiding local minima and examining a broader range of potential values. Finally, the third phase integrates genetic algorithms, enabling the optimization of parameters based on the model's actual performance. This process ensures a comprehensive and adaptive approach, reducing the likelihood of overfitting and numerical instability. The novelty of the method lies not only in the three-phase process but also in the model's ability to adapt to datasets with diverse characteristics, such as those involving complex nonlinear relationships or multidimensional dependencies. Experimental results demonstrate the method's superiority compared to traditional techniques, such as BFGS, ADAM, NEAT, and RBF-KMEANS, achieving improvements in terms of average error rates across both classification and regression datasets.

The article's conclusions clearly highlight the superiority of the proposed method. In classification datasets, the method achieves lower average error rates, particularly when parameters a and F are optimally configured. For instance, in classification datasets, the parameter F significantly influences performance, with $F = 5.0$ proving to be the most effective value in many cases. In datasets such as "Ionosphere" and "Wine," dramatic reductions in error rates are observed as the value of F increases, emphasizing the importance of selecting this parameter correctly. Similarly, in regression datasets, the proposed model demonstrates exceptional performance, with notable examples including datasets like "Abalone" and "Friedman," where the error decreases significantly compared to traditional techniques. The "MB" dataset is particularly interesting, as the use of Simulated Annealing contributed to a substantial performance improvement, avoiding errors commonly encountered in traditional approaches. Despite the generally positive results, there are cases where the method's performance is suboptimal, such as in the "Segment" dataset, indicating that the method is not universally generalizable without adjustment to the characteristics of each dataset. The need for further research on the parameters a and F becomes evident, as these parameters critically impact performance across numerous datasets.

For the future, the article proposes numerous directions for further exploration and development. Initially, applying the method to more diverse data categories, such as time series, image data, or even genetic data, could broaden its application scope. Additionally, the dynamic adjustment of parameters a and F during training, using reinforcement learning techniques, could further enhance performance by eliminating the need for manual tuning. Furthermore, integrating the method into deep learning systems, such as Convolutional or Recurrent Neural Networks, could lead to hybrid approaches that combine the flexibility of RBF with the computational power of deep neural networks. Moreover, investigating the robustness of the method in environments with dynamic or imbalanced data could provide additional insights into its generalizability. Finally, analyzing the method's performance on big data and integrating it with technologies like distributed processing or cloud computing could open new avenues, enabling its scalability to larger-scale problems. The methodology proposed in the article not only sets new standards for the performance of RBF but also paves the way for further innovation in the field of machine learning.

Author Contributions: V.C. and I.G.T. conducted the experiments, employing several datasets and provided the comparative experiments. D.T. and V.C. performed the statistical analysis and prepared the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Informed Consent Statement: Not applicable.

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

Conflicts of Interest: The authors declare no conflict of interest.
Not applicable.

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. Viquar, M., Basak, S., Dasgupta, A., Agrawal, S., & Saha, S. (2019). Machine learning in astronomy: A case study in quasar-star classification. Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2018, Volume 3, 827-836.
4. Luo, S., Leung, A. P., Hui, C. Y., & Li, K. L. (2020). An investigation on the factors affecting machine learning classifications in gamma-ray astronomy. Monthly Notices of the Royal Astronomical Society, 492(4), 5377-5390.
5. 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.
6. 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.
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. I. Kaastra, M. Boyd, Designing a neural network for forecasting financial and economic time series, Neurocomputing **10**, pp. 215-236, 1996.
10. 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.
11. 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.
12. M.J. Er, S. Wu, J. Lu, H.L. Toh, Face recognition with radial basis function (RBF) neural networks, IEEE Transactions on Neural Networks **13**, pp. 697-710, 2002.
13. Nam Mai-Duy, Thanh Tran-Cong, Numerical solution of differential equations using multiquadric radial basis function networks, Neural Networks 14, pp. 185-199, 2001.
14. N. Mai-Duy, Solving high order ordinary differential equations with radial basis function networks. Int. J. Numer. Meth. Engng. **62**, pp. 824-852, 2005.
15. Shen, W., Guo, X., Wu, C., & Wu, D. (2011). Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm. Knowledge-Based Systems, 24(3), 378-385.
16. 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.
17. 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.
18. 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.
19. 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.
20. N. Benoudjit, M. Verleysen, On the Kernel Widths in Radial-Basis Function Networks, Neural Processing Letters **18**, pp. 139–154, 2003.
21. Oyang, Y. J., Hwang, S. C., Ou, Y. Y., Chen, C. Y., & Chen, Z. W. (2005). Data classification with radial basis function networks based on a novel kernel density estimation algorithm. IEEE transactions on neural networks, 16(1), 225-236.
22. Ros, F., Pintore, M., Deman, A., & Chrétien, J. R. (2007). Automatical initialization of RBF neural networks. Chemometrics and intelligent laboratory systems, 87(1), 26-32.
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. Harpham, C., Dawson, C. W., & Brown, M. R. (2004). A review of genetic algorithms applied to training radial basis function networks. Neural Computing & Applications, 13, 193-201.
26. Sarimveis, H., Alexandridis, A., Mazarakis, S., & Bafas, G. (2004). A new algorithm for developing dynamic radial basis function neural network models based on genetic algorithms. Computers & chemical engineering, 28(1-2), 209-217.
27. Rani R, H. J., & Victoire T, A. A. (2018). Training radial basis function networks for wind speed prediction using PSO enhanced differential search optimizer. PloS one, 13(5), e0196871.
28. Zhang, W., & Wei, D. (2018). Prediction for network traffic of radial basis function neural network model based on improved particle swarm optimization algorithm. Neural Computing and Applications, 29(4), 1143-1152.
29. Qasem, S. N., Shamsuddin, S. M., & Zain, A. M. (2012). Multi-objective hybrid evolutionary algorithms for radial basis function neural network design. Knowledge-Based Systems, 27, 475-497.
30. 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.
31. 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.
32. L. Ingber, Very fast simulated re-annealing, Mathematical and Computer Modelling **12**, pp. 967-973, 1989.
33. Aerts, J. C., & Heuvelink, G. B. (2002). Using simulated annealing for resource allocation. International Journal of Geographical Information Science, 16(6), 571-587.
34. K. Ganesh, M. Punniyamoorthy, Optimization of continuous-time production planning using hybrid genetic algorithms-simulated annealing, Int J Adv Manuf Technol **26**, pp. 148–154, 2005.
35. El-Naggar, K. M., AlRashidi, M. R., AlHajri, M. F., & Al-Othman, A. K. (2012). Simulated annealing algorithm for photovoltaic parameters identification. Solar Energy, 86(1), 266-274.
36. Dupanloup, I., Schneider, S., & Excoffier, L. (2002). A simulated annealing approach to define the genetic structure of populations. Molecular ecology, 11(12), 2571-2581.
37. P. Kaelo, M.M. Ali, Integrated crossover rules in real coded genetic algorithms, European Journal of Operational Research **176**, pp. 60-76, 2007.
38. M. Kelly, R. Longjohn, K. Nottingham, The UCI Machine Learning Repository. 2023. Available online: https://archive.ics.uci.edu (accessed on 20 September 2023).
39. 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.
40. 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.
41. 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.
42. J.R. Quinlan, Simplifying Decision Trees. International Journal of Man-Machine Studies **27**, pp. 221-234, 1987.
43. T. Shultz, D. Mareschal, W. Schmidt, Modeling Cognitive Development on Balance Scale Phenomena, Machine Learning **16**, pp. 59-88, 1994.
44. 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.
45. R. Setiono , W.K. Leow, FERNN: An Algorithm for Fast Extraction of Rules from Neural Networks, Applied Intelligence **12**, pp. 15-25, 2000.
46. 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.
47. 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.
48. I. Kononenko, E. Šimec, M. Robnik-Šikonja, Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF, Applied Intelligence **7**, pp. 39–55, 1997.
49. 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.
50. J.G. Dy , C.E. Brodley, Feature Selection for Unsupervised Learning, The Journal of Machine Learning Research **5**, pp 845–889, 2004.
51. S. J. Perantonis, V. Virvilis, Input Feature Extraction for Multilayered Perceptrons Using Supervised Principal Component Analysis, Neural Processing Letters **10**, pp 243–252, 1999.
52. J. Garcke, M. Griebel, Classification with sparse grids using simplicial basis functions, Intell. Data Anal. **6**, pp. 483-502, 2002.
53. J. Mcdermott, R.S. Forsyth, Diagnosing a disorder in a classification benchmark, Pattern Recognition Letters **73**, pp. 41-43, 2016.
54. 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.

55. 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.
56. 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.
57. 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.
58. 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.
59. 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.
60. 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.
61. T. Hastie, R. Tibshirani, Non-parametric logistic and proportional odds regression, JRSS-C (Applied Statistics) **36**, pp. 260–276, 1987.
62. M. Dash, H. Liu, P. Scheuermann, K. L. Tan, Fast hierarchical clustering and its validation, Data & Knowledge Engineering **44**, pp 109–138, 2003.
63. Gorman, R.P.; Sejnowski, T.J. Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets. Neural Netw. 1988, 1, 75–89.
64. 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.
65. 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.
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. 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.
68. P. Zhong, M. Fukushima, Regularized nonsmooth Newton method for multi-class support vector machines, Optimization Methods and Software **22**, pp. 225-236, 2007.
69. 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.
70. 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.
71. M. Koivisto, K. Sood, Exact Bayesian Structure Discovery in Bayesian Networks, The Journal of Machine Learning Research **5**, pp. 549–573, 2004.
72. 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
73. 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 5 March 2025).
74. J.S. Simonoff, Smooting Methods in Statistics, Springer - Verlag, 1996.
75. I.Cheng Yeh, Modeling of strength of high performance concrete using artificial neural networks, Cement and Concrete Research. **28**, pp. 1797-1808, 1998.
76. D. Harrison and D.L. Rubinfeld, Hedonic prices and the demand for clean ai, J. Environ. Economics & Management **5**, pp. 81-102, 1978.
77. Friedman, J. (1991): Multivariate Adaptative Regression Splines. Annals of Statistics, 19:1, 1--141.
78. Mackowiak, P.A., Wasserman, S.S., Levine, M.M., 1992. 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. Amer. Med. Assoc. 268, 1578–1580
79. R.D. King, S. Muggleton, R. Lewis, M.J.E. Sternberg, Proc. Nat. Acad. Sci. USA **89**, pp. 11322–11326, 1992.
80. M.J.D Powell, A Tolerant Algorithm for Linearly Constrained Optimization Calculations, Mathematical Programming **45**, pp. 547-566, 1989.
81. C. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.
82. G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of Control Signals and Systems **2**, pp. 303-314, 1989.

83. 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.
84. K. O. Stanley, R. Miikkulainen, Evolving Neural Networks through Augmenting Topologies, Evolutionary Computation **10**, pp. 99-127, 2002.
85. S. Ding, L. Xu, C. Su et al, An optimizing method of RBF neural network based on genetic algorithm. Neural Comput & Applic 21, pp. 333–336, 2012.