processes

MDPI

Article

Training Feedforward Neural Networks Using an Enhanced Marine Predators Algorithm

Jinzhong Zhang and Yubao Xu *

School of Electrical and Optoelectronic Engineering, West Anhui University, Lu'an 237012, China; zhangjinzhongz@126.com
* Correspondence: xuyubao@wxc.edu.cn

Abstract: The input layer, hidden layer, and output layer are three models of the neural processors that make up feedforward neural networks (FNNs). Evolutionary algorithms have been extensively employed in training FNNs, which can correctly actualize any finite training sample set. In this paper, an enhanced marine predators algorithm (MPA) based on the ranking-based mutation operator (EMPA) was presented to train FNNs, and the objective was to attain the minimum classification, prediction, and approximation errors by modifying the connection weight and deviation value. The ranking-based mutation operator not only determines the best search agent and elevates the exploitation ability, but it also delays premature convergence and accelerates the optimization process. The EMPA integrates exploration and exploitation to mitigate search stagnation, and it has sufficient stability and flexibility to acquire the finest solution. To assess the significance and stability of the EMPA, a series of experiments on seventeen distinct datasets from the machine learning repository of the University of California Irvine (UCI) were utilized. The experimental results demonstrated that the EMPA has a quicker convergence speed, greater calculation accuracy, higher classification rate, strong stability and robustness, which is productive and reliable for training FNNs.

Keywords: feedforward neural networks; marine predators algorithm; ranking-based mutation operator; experimental results

Citation: Zhang, J.; Xu, Y. Training Feedforward Neural Networks Using an Enhanced Marine Predators Algorithm. *Processes* **2023**, *11*, 924. https://doi.org/10.3390/pr11030924

Academic Editor: Jie Zhang

Received: 14 February 2023
Revised: 13 March 2023
Accepted: 16 March 2023
Published: 17 March 2023

1. Introduction

Artificial neural networks are a cross-disciplinary subject that involves neuroscience, brain science, artificial intelligence, computer science, and so on, and they mainly simulate the network structure of human brain neurons to process memory information [1–5]. As an effective and feasible mathematical model, the networks have been employed in various domains, such as pattern recognition, intelligent robots, intelligent control, biomedicine, and function approximation. Feedforward neural networks (FNNs) are one of the relatively prominent methods, which have some characteristics of simple network topology, fault-tolerant distributed storage, massively parallel computation, and strong self-organization and self-adaptability. The purpose of training is to obtain the minimal objective function with connection weights and deviation values in the network according to the collected dataset information, which effectively measure the discrepancy between the predicted values and real values. In recent years, various swarm intelligence methodologies have been used to train feedforward neural networks, such as ant lion optimization (ALO) [6], African vultures optimization algorithm (AVOA) [7], dingo optimization algorithm (DOA) [8], flower pollination algorithm (FPA) [9], moth flame optimization (MFO) [10], salp swarm algorithm (SSA) [11], and sperm swarm optimization (SSO) [12].

Zhang et al. designed an efficient grafting constructive algorithm to train FNNs. This algorithm had faster convergence accuracy and a smaller calculation error [13]. Fan et al., introduced a new backpropagation learning algorithm based on graph regularization to resolve FNNs; this algorithm obtained a better objective value and adjustment parameters [14]. Qu et al. applied a learnable anti-noise receiver algorithm to optimize FNNs; this

algorithm had a higher search efficiency and training accuracy [15]. Admon et al. utilized a novel search algorithm to train FNNs and resolve integer order differential equations; this algorithm had a certain practicability and reliability to obtain a high-precision solution [16]. Guo et al. invented an indicator correlation elimination algorithm to optimize FNNs; this algorithm had better training results [17]. Zhang et al. introduced a quantum genetic algorithm to train FNNs; this algorithm had superiority and robustness in obtaining the tuning parameters [18]. Venkatachalapathy et al. utilized FNNs to resolve nonlinear ordinary differential equations; this method had a simple network framework and high convergence accuracy [19]. Liao et al. created a novel deep learning algorithm based on FNNs to resolve the flows of dynamical systems; this algorithm's efficiency and accuracy were relatively excellent [20]. Shao et al. produced a genetic approach to train optical FNNs; this algorithm had strong integrity and flexibility to satisfy a high classification accuracy [21]. Wu et al. employed the swarm intelligence algorithm to optimize the welding sequence optimization and FNNs; this algorithm eliminated premature convergence and generated the best solution [22]. Raziani et al. combined a modified whale optimization algorithm based on a nonlinear function with FNNs to resolve medical classification problems; this method had faster operation efficiency and better evaluation indexes [23]. Dong et al. designed an efficient and reliable training algorithm to solve FNNs; this algorithm utilized flexibility and stability to obtain a better objective value [24]. Fontes et al. designed a modified constructive algorithm to configure FNNs; this algorithm effectively trained datasets to obtain a higher classification accuracy [25]. Zheng et al. employed the Tschauner–Hempel equation to optimize FNNs; this method had high analytical solutions and good training results [26]. Yılmaz et al. introduced a differential evolution method to train artificial neural networks; this method used a better network structure to obtain the best solution [27]. Luo et al. presented a spotted hyena optimization to optimize FNNs; this algorithm had certain superiority and robustness for obtaining relatively optimal parameters [28]. Askari et al. introduced a political optimization algorithm to train FNNs; the classification accuracy and optimization rate of this algorithm were better [29]. Duman et al. integrated a manta ray foraging optimization algorithm to train FNNs; this algorithm utilized an effective search mechanism to obtain the optimal parameters [30]. Pan et al. employed FNNs to optimize full wave nonlinear inverse scattering; this technique had a faster calculation rate [31]. Wu et al. described a beetle antennae search method to optimize neural networks; this approach had a strong robustness for determining the superior solution [32]. Mahmoud et al. designed a pseudoinverse learning algorithm to train side road convolution neural networks; this algorithm had strong dependability and reliability for achieving the best experimental results [33]. Jamshidi et al. introduced a hybrid echo state network for pattern recognition and classification; this method had faster processing speed and the best optimization results [34]. Khalaj et al. utilized hybrid machine learning techniques and computational mechanics to design oxide precipitation hardened alloys; the method had strong robustness and stability for obtaining a high accuracy [35]. Daneshfar et al. applied an octonion-based nonlinear echo state network for speech emotion recognition in the metaverse; this method had a strong stability to obtain a high calculation accuracy and better optimization performance [36]. Abd Elminaam et al. proposed the marine predators algorithm to resolve feature selection; and the algorithm had better accuracy, sensitivity, and specificity [37]. Zhang et al. presented a domain adaptation network for remaining useful life prediction; this proposed method had a strong stability to determine the best results [38]. Zhang et al. utilized an integrated multitasking intelligent bearing fault diagnosis scheme to realize detection, classification, and fault identification [39]. Zhang et al. proposed an integrated multi-head dual sparse self-attention network for remaining useful life prediction; this method had excellent superiority and robustness [40]. Zhang et al. designed a parallel hybrid neural network for remaining useful life prediction in prognostics; this method had better results [41]. To summarize, evolutionary algorithms have strong robustness, parallelism, and scalability to train FNNs. These algorithms have strong stability and feasibility to determine the objective function value.

The MPA is derived from the universal hunting and gathering mechanisms, particularly Lévy flight, Brownian motion, and the optimal encounter rate policy between the predator and prey [42]. To enhance the availability and practicability, the ranking-based mutation operator was added to the basic MPA, which accelerates the calculation speed and enhances the exploitation to improve the selection probability to mitigate premature convergence. The EMPA was utilized to train FNNs, and the objective was to attain the minimum classification, prediction, and approximation errors by training the FNNs and modifying the connection weight and deviation value. The EMPA has the properties of straightforward algorithm architecture, excellent control parameters, great traversal efficiency, strong stability, and easy implementation. The EMPA integrates exploration or exploitation to determine the best solution. The experimental results demonstrated that the EMPA has certain effectiveness and feasibility to achieve a quicker convergence speed and a greater calculation accuracy. Meanwhile, the EMPA has strong stability and robustness for achieving a higher classification rate.

The following sections make up the article. Section 2 covers the mathematical modeling of FNNs. Section 3 explains the MPA. Section 4 shows the EMPA. Section 5 depicts EMPA-based feedforward neural networks. The experimental results and analysis are exhibited in Section 6. Finally, conclusions and future research are illustrated in Section 7.

2. Mathematical Modeling of FNNs

The FNNs are also known as multilayer perception (MLP), which are among the most widely used and rapidly developed artificial neural networks. Each neuron is exclusively coupled to the neuron in the previous layer, and they are arranged in layers. There is no feedback between layers. Three-layer FNNs mainly include the input layer, hidden layer, and output layer, which is shown in Figure 1.

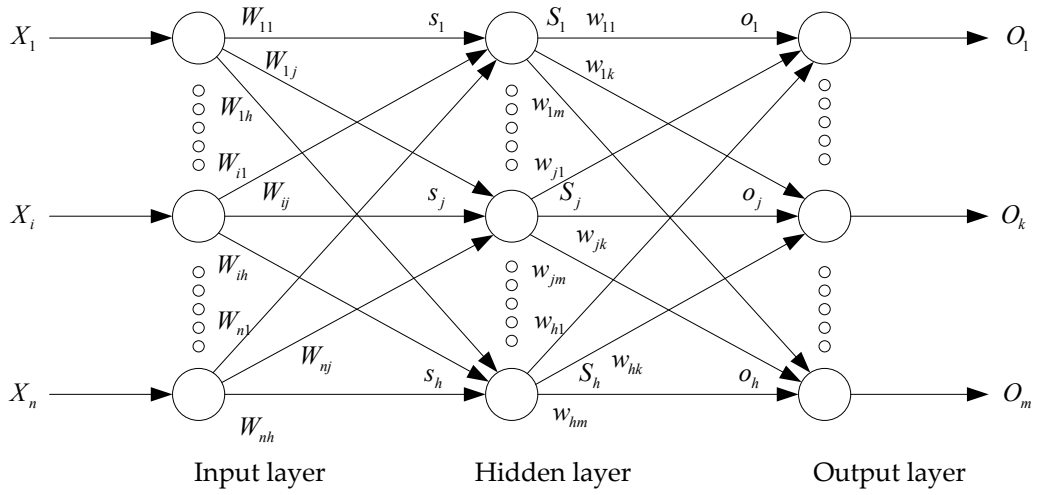

Figure 1. Three-layer feedforward neural networks.

The weighted sum of the input layer is computed as:

$$s_j = \sum_{i=1}^{n} (W_{i,j} \cdot X_i) - \theta_j,\ j = 1, 2, \ldots, h \tag{1}$$

where n denotes the input nodes, W_{ij} denotes the connection weight from the ith note of the input layer to the jth note of the hidden layer, X_i denotes the ith input node, and θ_j denotes the deviation value of jth hidden node.

The output value of each hidden layer is computed as:

$$S_j = sigmoid(s_j) = \frac{1}{(1 + exp(-s_j))},\ j = 1, 2, \ldots, h \tag{2}$$

The values of the output layer are computed as:

$$o_k = \sum_{j=1}^{h} (w_{j,k} \cdot S_j) - \theta'_k,\ k = 1, 2, \ldots, m \tag{3}$$

$$O_k = sigmoid(o_k) = \frac{1}{(1 + exp(-o_k))},\ k = 1, 2, \ldots, m \tag{4}$$

where h denotes the input nodes, w_{jk} denotes the connection weight from the jth note of the hidden layer to the kth note of the output layer, S_j denotes the jth input node, and θ'_k denotes the deviation value of the kth output node. The connection weight and deviation value are the most important component of FNNs, which determine the final output value.

3. MPA

The MPA utilizes the Lévy flight, Brownian motion, and the optimal encounter rate policy between the predator and prey in marine ecosystems to achieve the best value.

3.1. Initialization

The MPA utilizes a random positioning mechanism to initialize the population and to simulate marine predation. The position is computed as follows:

$$X_0 = X_{min} + rand(X_{max} - X_{min}) \tag{5}$$

where X_{max} and X_{min} denote the search space boundary, and $rand$ denotes a uniformly distributed randomized number in $[0, 1]$.

The *Elite* matrix is computed as follows:

$$Elite = \begin{bmatrix} X^I_{1,1} & X^I_{1,2} & \cdots & X^I_{1,D} \\ X^I_{2,1} & X^I_{2,2} & \cdots & X^I_{2,D} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ X^I_{N,1} & X^I_{N,2} & \cdots & X^I_{N,D} \end{bmatrix}_{N \times D} \tag{6}$$

where $\overrightarrow{X}^I$ denotes the top predator vector, N denotes the population size, and D denotes the spatial dimension.

The *Prey* matrix is computed as follows:

$$Prey = \begin{bmatrix} X_{1,1} & X_{1,2} & \cdots & X_{1,D} \\ X_{2,1} & X_{2,2} & \cdots & X_{2,D} \\ X_{3,1} & X_{3,2} & \vdots & X_{3,D} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ X_{N,1} & X_{N,2} & \cdots & X_{N,D} \end{bmatrix}_{N \times D} \tag{7}$$

where X_{ij} denotes the jth spatial position of the ith prey.

3.2. MPA Optimization Scenarios

According to the different velocity ratios of the predator and prey, the MPA is separated into three parts: high-velocity ratio, unit velocity ratio, and low-velocity ratio.

Phase 1: The high-velocity ratio. The predator moves more slowly than the prey. The best foraging strategy is to capture the prey by utilizing Brownian motion and retaining

the original position. In this phase, the MPA performs the exploration. The position is computed as follows:

$$Iter < \frac{1}{3} Max\_Iter \tag{8}$$

$$\overrightarrow{stepsize_i} = \vec{R}_B \otimes (\overrightarrow{Elite_i} - \vec{R}_B \otimes \overrightarrow{Prey_i}),\ i = 1, \ldots, N \tag{9}$$

$$\overrightarrow{Prey_i} = \overrightarrow{Prey_i} + P \cdot \vec{R} \otimes \overrightarrow{stepsize_i} \tag{10}$$

where $\overrightarrow{stepsize_i}$ denotes a motion step, $\vec{R}_B$ denotes a random walk vector with normal distribution, $\overrightarrow{Elite_i}$ denotes a top predator matrix, $\overrightarrow{Prey_i}$ denotes a prey matrix, $P = 0.5$ denotes a constant value, $\vec{R}$ denotes a randomized vector in $[0, 1]$, and $\otimes$ denotes entry-wise multiplication.

Phase 2: The unit velocity ratio. The moving velocity of the predator and prey is consistent. In this phase, the MPA gradually transits from exploration to exploitation. The prey is based on Lévy flight, and half of the population quantity is designed for exploitation. The predator is based on Brownian motion, and the other half of the population quantity is designed for exploration. The position is computed as follows:

$$\frac{1}{3} Max\_Iter < Iter < \frac{2}{3} Max\_Iter \tag{11}$$

For the first half of the population quality, we compute:

$$\overrightarrow{stepsize_i} = \vec{R}_L \otimes (\overrightarrow{Elite_i} - \vec{R}_L \otimes \overrightarrow{Prey_i}),\ i = 1, \ldots, N/2 \tag{12}$$

$$\overrightarrow{Prey_i} = \overrightarrow{Prey_i} + P \cdot \vec{R} \otimes \overrightarrow{stepsize_i} \tag{13}$$

where $\vec{R}_L$ denotes a randomized vector of Lévy flight.

For the second half of the population quality, we compute:

$$\overrightarrow{stepsize_i} = \vec{R}_B \otimes (\vec{R}_B \otimes \overrightarrow{Elite_i} - \overrightarrow{Prey_i}),\ i = N/2, \ldots, N \tag{14}$$

$$\overrightarrow{Prey_i} = \overrightarrow{Elite_i} + P \cdot CF \otimes \overrightarrow{stepsize_i} \tag{15}$$

$$CF = (1 - \frac{Iter}{Max\_Iter})^{(2\frac{Iter}{Max\_Iter})} \tag{16}$$

where CF denotes a flexible parameter.

Phase 3: The low-velocity ratio. The predator moves more swiftly than the prey. The predator is based on Lévy flight and utilizes exploitation to capture the prey. The position is computed as follows:

$$Iter > \frac{2}{3} Max\_Iter \tag{17}$$

$$\overrightarrow{stepsize_i} = \vec{R}_L \otimes (\vec{R}_L \otimes \overrightarrow{Elite_i} - \overrightarrow{Prey_i}),\ i = 1, \ldots, N \tag{18}$$

$$\overrightarrow{Prey_i} = \overrightarrow{Elite_i} + P \cdot CF \otimes \overrightarrow{stepsize_i} \tag{19}$$

3.3. Eddy Formation and FAD's Effect

The eddy formation and fish aggregating devices (*FADs*) have a profound impact on the feeding behavior of marine predators, which avoids premature convergence and search stagnation. The position is computed as follows:

$$\overrightarrow{Prey_i} = \begin{cases} \overrightarrow{Prey_i} + CF[\vec{X}_{min} + \vec{R} \otimes (\vec{X}_{max} - \vec{X}_{min})] \otimes \vec{U} & if\ r \leq FADs \\ \overrightarrow{Prey_i} + [FADs(1-r) + r](\overrightarrow{Prey_{r1}} - \overrightarrow{Prey_{r2}}) & if\ r > FADs \end{cases} \quad (20)$$

where $FADs = 0.2$ denotes a probability of the *FADs* effect, $\vec{U}$ denotes a binary vector with arrays containing zero and one, r denotes a randomized number in $[0, 1]$, and r_1 and r_2 denote randomized indexes of the prey matrix.

To precisely clarify the solution process, the pseudocode of the MPA is expressed in Algorithm 1.

Algorithm 1: MPA

Begin
Step 1. Initialize the marine predator population $X_i (i = 1, 2, \ldots, N)$ and control parameters
Step 2. Assess the fitness value of each predator
Discover the ideal predator
Step 3. while $(Iter < Max\_Iter)$ do
for each predator
Construct the Elite and prey matrices via Equations (6) and (7)
If $(Iter < \frac{1}{3} Max\_Iter)$
Renew prey via Equations (9) and (10)
Else if $\frac{1}{3} Max\_Iter < Iter < \frac{2}{3} Max\_Iter$
For the first half of the population quality $(i = 1, \ldots, N/2)$
Renew prey via Equations (12) and (13)
For the other half of the population quality $(i = N/2, \ldots, N)$
Renew prey via Equations (14) and (15)
Else if $Iter > \frac{2}{3} Max\_Iter$
Renew prey via Equations (18) and (19)
End if
Identify and amend any predator that travels beyond the search scope
Complete memory conserving and Elite Renew
Utilizing *FADs* effect and renewing prey via Equation (20)
$Iter = Iter + 1$
Return the best predator
End

4. EMPA

The ranking-based mutation operator filters out the best marine predator to avoid search stagnation and to enhance exploitation ability [43]. The ranking of fitness values from best to worst is computed as follows:

$$R_i = N - i, \quad i = 1, 2, \ldots, N \quad (21)$$

where N is the population size. The optimal predator has the best ranking, and the selection probability P_i of the *ith* predator is calculated as follows:

$$p_i = \frac{R_i}{N},\ i = 1, 2, \ldots, N \quad (22)$$

The ranking-based mutation operator "DE/rand/1" is given in Algorithm 2. The aquatic predator with the highest ranking is more likely to be allocated as a terminal vector or vector, and the genetic information is transmitted to the offspring. If both differential mutation vectors are from the higher-order vector, the operator's search step may drastically reduce and avoid premature convergence.

Algorithm 2: Ranking-based mutation operator of "DE/rand/1"

Begin
Sort the population, and assign the ranking and selection probability P_i for each predator
Randomly select $r_1 \in \{1, N\}$ {base vector index}
while $rand > p_{r_1}$ *or* $r_1 == i$
Randomly select $r_1 \in \{1, N\}$
end
Randomly select $r_2 \in \{1, N\}$ {terminal vector index}
while $rand > p_{r_2}$ *or* $r_2 == r_1$ *or* $r_2 == i$
Randomly select $r_2 \in \{1, N\}$
end
Randomly select $r_3 \in \{1, N\}$ {starting vector index}
while $r_3 == r_2$ *or* $r_3 == r_1$ *or* $r_3 == i$
Randomly select $r_3 \in \{1, N\}$
end
End

The EMPA can balance the exploration and exploitation to improve the convergence speed and calculation accuracy. The pseudocode of the EMPA is expressed in Algorithm 3.

Algorithm 3: EMPA

Begin
Step 1. Initialize the marine predator population $X_i (i = 1, 2, \ldots, N)$ and control parameters
Step 2. Assess the fitness value of each predator
Discover the ideal predator
Step 3. while ($Iter < Max\_Iter$) do
for each predator
Sort the population; assign the ranking and selection probability P_i for each predator
/*ranking-based mutation stage*/
Randomly select $r_1 \in \{1, N\}$ {base vector index}
while $rand > p_{r_1}$ *or* $r_1 == i$
Randomly select $r_1 \in \{1, N\}$
end
Randomly select $r_2 \in \{1, N\}$ {terminal vector index}
while $rand > p_{r_2}$ *or* $r_2 == r_1$ *or* $r_2 == i$
Randomly select $r_2 \in \{1, N\}$
end
Randomly select $r_3 \in \{1, N\}$ {starting vector index}
while $r_3 == r_2$ *or* $r_3 == r_1$ *or* $r_3 == i$
Randomly select $r_3 \in \{1, N\}$
end /*end of ranking-based mutation stage*/
Construct the Elite and prey matrices via Equations (6) and (7)
If $(Iter < \frac{1}{3} Max\_Iter)$
Renew prey via Equations (9) and (10)
Else if $\frac{1}{3} Max\_Iter < Iter < \frac{2}{3} Max\_Iter$
For the first half of the population quality $(i = 1, \ldots, N/2)$
Renew prey via Equations (12) and (13)
For the other half of the population quality $(i = N/2, \ldots, N)$
Renew prey via Equations (14) and (15)
Else if $Iter > \frac{2}{3} Max\_Iter$
Renew prey via Equations (18) and (19)
End if
Identify and amend any predator that travels beyond the search scope
Complete memory conserving and Elite Renew
Utilizing *FADs* effect and renewing prey via Equation (20)
$Iter = Iter + 1$
Return the best predator
End

5. EMPA-Based Feedforward Neural Networks

The intention of training the FNNs is not only to acquire the global optimal solution for the given input value, but also to identify the best combination of the connection weight and deviation value. The FNNs with vector mechanisms are computed as follows:

$$\vec{V} = \left\{ \vec{W}, \vec{\theta} \right\} = \{W_{1,1}, W_{1,2}, \ldots, W_{n,n}, h, \theta_1, \theta_2, \ldots, \theta_h\} \tag{23}$$

The mean squared error (*MSE*) is used as an evaluation index to estimate the expected output and actual output, which classifies and predicts all training samples in the datasets. The *MSE* is computed as follows:

$$MSE = \sum_{i=1}^{m} \left(o_i^k - d_i^k\right)^2 \tag{24}$$

where m denotes the output size, d_i^k denotes the expected value of the ith input unit according to the kth training sample, and o_i^k denotes the actual value of the ith input unit according to the kth training sample.

The data set contains numerous training samples, and each sample needs to be evaluated by FNNs. The average value of the *MSE* is computed as follows:

$$\overline{MSE} = \sum_{k=1}^{s} \frac{\sum_{i=1}^{m} \left(o_i^k - d_i^k\right)^2}{s} \tag{25}$$

where s denotes the training sample size.

The fitness value of training the FNNs is computed as follows:

$$Minimize : F(\overline{V}) = \overline{MSE} \tag{26}$$

The correlation between the issue scope and the EMPA scope is revealed in Table 1. The EMPA-based feedforward neural networks are expressed in Algorithm 4. The flowchart of the EMPA for FNNs is shown in Figure 2.

Table 1. Correlation between issue scope and EMPA scope.

Issue Scope	EMPA Scope
A set scheme $(P_1, P_2, \ldots, P_N)$ to tackle the FNNs	A marine predator population $(X_1, X_2, \ldots, X_N)$
The optimal scheme to obtain the best solution	The marine predator or search agent
The evaluation value of FNNs	The fitness value of EMPA

Algorithm 4: EMPA-based feedforward neural networks

Begin
Step 1. Initialize the marine predator population $X_i(i = 1, 2, \ldots, N)$, control parameters, and the structure of FNNs; each predator denotes the connection weight and deviation value
Step 2. Assess the fitness value of each predator via Equation (24); assign connection weight to the predator
Discover the ideal predator
Step 3. while $(Iter < Max\_Iter)$ do
for each predator
Sort the population; assign the ranking and selection probability P_i for each predator
/*ranking-based mutation stage*/
Randomly select $r_1 \in \{1, N\}$ {base vector index}
while $rand > p_{r_1}$ *or* $r_1 == i$
Randomly select $r_1 \in \{1, N\}$
end
Randomly select $r_2 \in \{1, N\}$ {terminal vector index}
while $rand > p_{r_2}$ *or* $r_2 == r_1$ *or* $r_2 == i$
Randomly select $r_2 \in \{1, N\}$
end
Randomly select $r_3 \in \{1, N\}$ {starting vector index}
while $r_3 == r_2$ *or* $r_3 == r_1$ *or* $r_3 == i$
Randomly select $r_3 \in \{1, N\}$
end /*end of ranking-based mutation stage*/
Construct the Elite and prey matrices via Equations (6) and (7)
If $(Iter < \frac{1}{3} Max\_Iter)$
Renew prey via Equations (9) and (10)
Else if $\frac{1}{3} Max\_Iter < Iter < \frac{2}{3} Max\_Iter$
For the first half of the population quality $(i = 1, \ldots, N/2)$
Renew prey via Equations (12) and (13)
For the other half of the population quality $(i = N/2, \ldots, N)$
Renew prey via Equations (14) and (15)
Else if $Iter > \frac{2}{3} Max\_Iter$
Renew prey via Equations (18) and (19)
End if
Identify and amend any predator that travels beyond the search scope
Complete memory conserving and Elite Renew
Utilizing *FADs* effect and renewing prey via Equation (20), and assessing the fitness value of each predator via Equation (24)
$Iter = Iter + 1$
Return the best predator
End

Complexity Analysis

In this section, both the time and space complexity of the EMPA-based feedforward neural networks are analyzed.

Time complexity: The EMPA-based feedforward neural networks mainly contain four steps: initialization, EMPA optimization scenarios (Phase 1—high-velocity ratio Phase 2—unit velocity ratio, Phase 3—low-velocity ratio), eddy formation and FAD's effect, and halting judgment. The population size is N, the maximum iteration is Max_Iter, and the problem dimension is D. The time complexity of initialization is $O(N * D)$. The time complexity of the EMPA optimization scenarios is $O(N * D * Max\_Iter)$. The time complexity of the eddy formation and FADs' effect is $O(Max\_Iter)$. The time complexity of the halting judgment is $O(1)$. Thus, the total time complexity of the EMPA-based feedforward neural networks is $O(N * D * Max\_Iter)$.

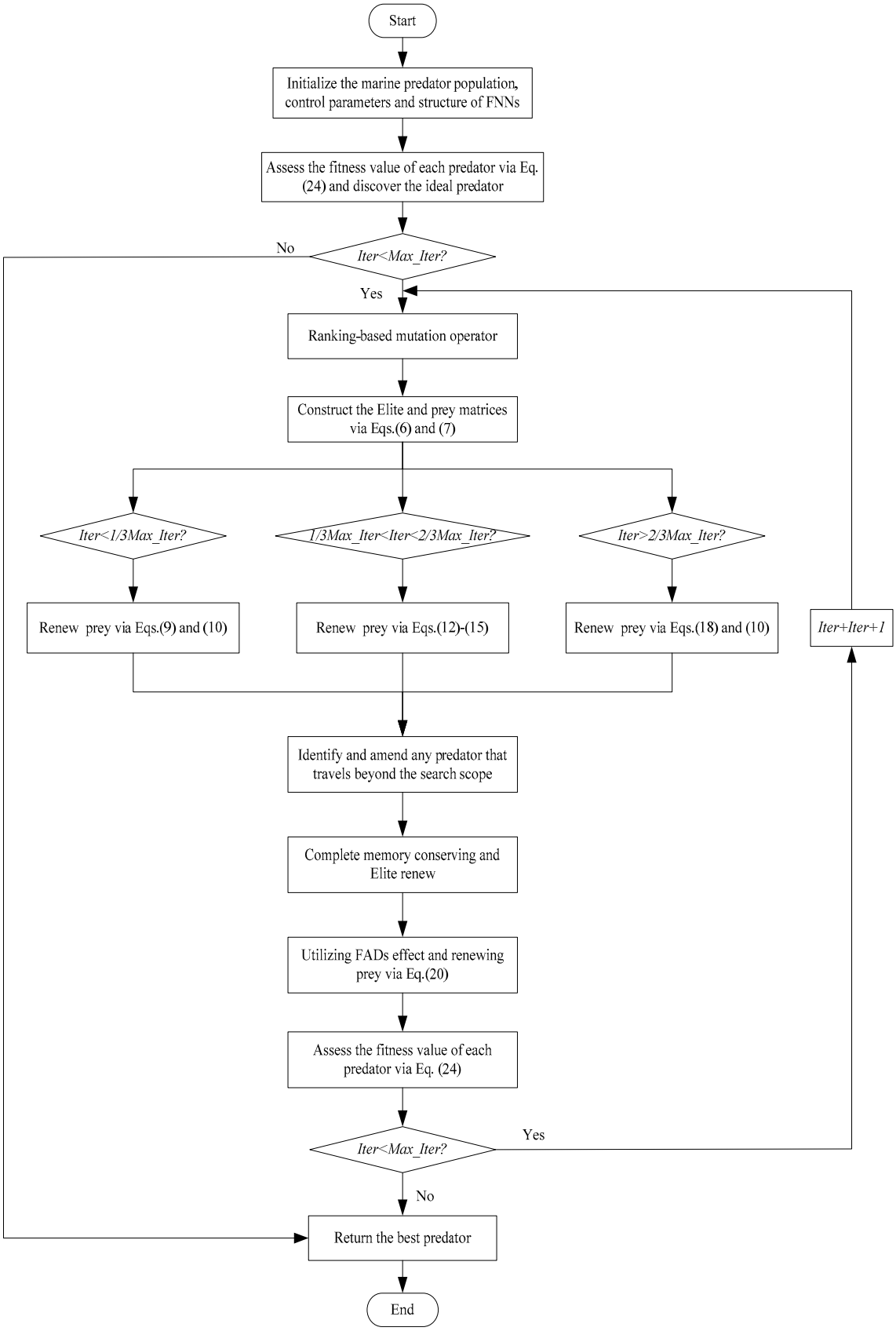

Figure 2. Flowchart of EMPA for FNNs.

Space complexity: the amount of extra storage in an algorithm is viewed as a measure of space complexity. The population size is N and the problem dimension is D. The EMPA utilizes N search agents to calculate the space complexity. Therefore, the total space complexity of the EMPA-based feedforward neural networks is $O(N * D)$, and the space efficiency of the EMPA is effective and stable.

6. Experimental Results and Analysis

6.1. Experimental Setup

The numerical experiment was implemented on a computer with an Intel Core i9-12900HX 2.30 GHz CPU, RTX 3080 Ti, and 64 GB memory with Windows 11 system. All algorithms were programmed in MATLAB R2018b.

6.2. Test Datasets

The test datasets are from the machine learning repository of the University of California Irvine (UCI), which were used to evaluate the stability and robustness of the MPA. The details of the datasets are revealed in Table 2.

Table 2. The details of the datasets.

Datasets	Attribute	Class	Training	Testing	Input	Hidden	Output
Blood	4	2	493	255	4	9	2
Scale	4	3	412	213	4	9	3
Survival	3	2	202	104	3	7	2
Liver	6	2	227	118	6	13	2
Seeds	7	3	139	71	7	15	3
Wine	13	3	117	61	13	27	3
Iris	4	3	99	51	4	9	3
Statlog	13	2	178	92	13	27	2
XOR	3	2	4	4	3	7	2
Balloon	4	2	10	10	4	9	2
Cancer	9	2	599	100	9	19	2
Diabetes	8	2	507	261	8	17	2
Gene	57	2	70	36	57	115	2
Parkinson	22	2	129	66	22	45	2
Splice	60	2	660	340	60	121	2
WDBC	30	2	394	165	30	61	2
Zoo	16	7	67	34	16	33	7

6.3. Parameter Setting

To establish viability and suitability, the MPA was contrasted with other algorithms that contained the ALO, AVOA, DOA, FPA, MFO, SSA, SSO, and MPA. The control parameters were indicative experimental values that were taken from the source publications. The initial parameters of all algorithms are revealed in Table 3.

Table 3. Initial parameters of all algorithms.

Algorithms	Parameters	Values
ALO	Unpredictable value *rand*	[0,1]
	Constant number w	5
AVOA	Randomized number L_1	[0,1]
	Randomized number L_2	[0,1]
	Randomized number z	$[-1,1]$
	Randomized number h	$[-2,2]$
	Randomized number *rand*	[0,1]
	Randomized number u	[0,1]
	Randomized number v	[0,1]
	Constant number β	1.5

Table 3. *Cont.*

Algorithms	Parameters	Values
DOA	Randomized vector a_1	[0,1]
	Randomized vector a_2	[0,1]
	Coefficient vector A	(1,0)
	Coefficient vector B	(1,1)
	Randomized number b	(0,3)
FPA	Switch probability ρ	0.8
	Step size λ	1.5
	Randomized number ε	[0,1]
MFO	Constant number b	1
	Randomized number t	$[-1,1]$
	Randomized number r	$[-2,-1]$
SSA	Randomized number c_2	[0,1]
	Randomized number c_3	[0,1]
SSO	Velocity damping factor D	[0,1]
	Randomized number ph_Rand_1	[7,14]
	Randomized number ph_Rand_2	[7,14]
	Randomized number ph_Rand_3	[7,14]
MPA	Uniform randomized number $rand$	[0,1]
	Uniform randomized number R	[0,1]
	Constant number P	0.5
	Probability of $FADs$ effect	0.2
	Binary vector U	[0,1]
	Randomized number r	[0,1]
MPA	Uniform randomized number $rand$	[0,1]
	Uniform randomized number R	[0,1]
	Constant number P	0.5
	Probability of $FADs$ effect	0.2
	Binary vector U	[0,1]
	Randomized number r	[0,1]
	Scaling factor F	0.7

6.4. Results and Analysis

For each algorithm, the population size was 30, the maximum iteration was 500, and the independent run was 20. Best, Worst, Mean and Std denote the optimal value, worst value, mean value, and standard deviation, respectively. Accuracy denotes the classification rate, and the ranking is based on accuracy. These evaluation indexes can comprehensively reflect the overall reliability and superiority of each algorithm.

The experimental results of multiple datasets are revealed in Table 4. Different algorithms utilized different datasets to train the feedforward neural networks, and the purpose was to minimize the gap between anticipated output and actual output by modifying the connection weight and deviation value. To verify the effectiveness and feasibility, the EMPA was compared with other algorithms by training massive datasets. For the blood and scale datasets, the optimal values, worst values, mean values and standard deviations of the EMPA were superior to those of the ALO, AVOA, DOA, FPA, SSA, SSO, and MPA. The classification rate and ranking of the EMPA were the highest, which indicates that the EMPA appropriately modifies the traversal mechanism to arrive at the overall optimum solution. For survival, liver, and statlog datasets, when compared with the ALO, AVOA, DOA, FPA, MFO, SSA, SSO, and MPA, the optimal values, worst values, and mean values of the EMPA were superior, and the standard deviations, classification rate, and ranking of

the EMPA were comparatively greater, which indicates that the EMPA provides superiority and feasibility to integrate the exploration and exploitation, as well as to obtain the best solution. For the XOR, balloon, splice, and zoo datasets, all evaluation indexes of the EMPA were superior to those of the ALO, AVOA, DOA, FPA, MFO, SSA, SSO, and MPA, which indicates that the EMPA utilizes the unique predatory mechanism and position update mechanism to avoid search stagnation and obtain the accurate solutions. For the seeds, wine, iris, cancer, diabetes, gene, parkinson, WDBC datasets, all evaluation indexes of the EMPA were better than those of the ALO, AVOA, DOA, FPA, MFO, SSA, SSO, and MPA, which indicates that the EMPA utilizes some advantages and characteristics to achieve parameter adjustment and traversal search. All classification rates and ranking of the EMPA were better compared to the ALO, AVOA, DOA, FPA, MFO, SSA, SSO, and MPA. These comparison algorithms achieve the balance between exploration and exploitation by adjusting control parameters to a certain extent, but they easily fall into local optimum and premature convergences to yield a slow convergence speed, low calculation accuracy, and worse classification rate. The EMPA, based on the marine predators foraging strategy, utilizes a distinctive optimization mechanism of Lévy flight, Brownian motion, and the optimal encounter rate policy to capture the prey in the marine ecosystem. The EMPA has the properties of straightforward algorithm architecture, excellent control parameters, great traversal efficiency, strong stability, and easy implementation. The EMPA not only successfully balances exploration and exploitation to eliminate search stagnation and slow convergence, but it also efficiently traverses the entire search space to modify parameters and identify the ideal solution. In summary, the EMPA has significant resilience and stability to efficiently train the feedforward neural networks.

Table 4. Experimental results of multiple datasets.

Datasets	Result	ALO	AVOA	DOA	FPA	MFO	SSA	SSO	MPA	EMPA
Blood	Best	3.07×10^{-1}	2.99×10^{-1}	3.18×10^{-1}	3.09×10^{-1}	2.99×10^{-1}	3.06×10^{-1}	3.39×10^{-1}	3.02×10^{-1}	2.96×10^{-1}
	Worst	3.21×10^{-1}	3.58×10^{-1}	3.66×10^{-1}	3.22×10^{-1}	3.08×10^{-1}	3.95×10^{-1}	3.64×10^{-1}	3.07×10^{-1}	3.06×10^{-1}
	Mean	3.15×10^{-1}	3.17×10^{-1}	3.47×10^{-1}	3.16×10^{-1}	3.03×10^{-1}	3.26×10^{-1}	3.50×10^{-1}	3.05×10^{-1}	3.02×10^{-1}
	Std	4.25×10^{-3}	1.55×10^{-2}	1.49×10^{-2}	2.99×10^{-3}	2.07×10^{-3}	2.21×10^{-2}	7.50×10^{-3}	1.53×10^{-3}	2.31×10^{-3}
	Accuracy	80.39	80.39	79.61	80	80	80	77.25	81.57	85.21
	Rank	3	3	5	4	4	4	6	2	1
Scale	Best	1.39×10^{-1}	1.21×10^{-1}	2.86×10^{-1}	1.56×10^{-1}	1.20×10^{-1}	1.28×10^{-1}	2.27×10^{-1}	1.05×10^{-1}	9.74×10^{-2}
	Worst	1.86×10^{-1}	2.99×10^{-1}	6.42×10^{-1}	1.85×10^{-1}	1.57×10^{-1}	2.13×10^{-1}	4.79×10^{-1}	1.79×10^{-1}	1.57×10^{-1}
	Mean	1.60×10^{-1}	1.80×10^{-1}	4.69×10^{-1}	1.68×10^{-1}	1.41×10^{-1}	1.59×10^{-1}	3.63×10^{-1}	1.33×10^{-1}	1.23×10^{-1}
	Std	1.18×10^{-2}	5.18×10^{-2}	9.35×10^{-2}	7.18×10^{-3}	9.75×10^{-3}	2.24×10^{-2}	7.96×10^{-2}	2.14×10^{-2}	1.68×10^{-2}
	Accuracy	89.20	86.85	79.34	88.03	87.32	90.14	85.92	90.61	92.02
	Rank	4	7	9	5	6	3	8	2	1
Survival	Best	3.60×10^{-1}	3.46×10^{-1}	3.87×10^{-1}	3.59×10^{-1}	3.11×10^{-1}	3.36×10^{-1}	4.08×10^{-1}	3.03×10^{-1}	2.96×10^{-1}
	Worst	3.86×10^{-1}	3.95×10^{-1}	4.26×10^{-1}	3.80×10^{-1}	3.62×10^{-1}	4.15×10^{-1}	4.39×10^{-1}	3.49×10^{-1}	3.38×10^{-1}
	Mean	3.75×10^{-1}	3.67×10^{-1}	4.09×10^{-1}	3.71×10^{-1}	3.33×10^{-1}	3.64×10^{-1}	4.19×10^{-1}	3.27×10^{-1}	3.21×10^{-1}
	Std	5.74×10^{-3}	1.21×10^{-2}	1.30×10^{-2}	6.20×10^{-3}	1.46×10^{-2}	2.56×10^{-2}	8.38×10^{-3}	1.08×10^{-2}	1.04×10^{-2}
	Accuracy	80.76	79.33	80.29	79.81	77.88	78.85	79.33	81.73	81.94
	Rank	3	6	4	5	8	7	6	2	1
Liver	Best	4.05×10^{-1}	3.70×10^{-1}	4.52×10^{-1}	4.25×10^{-1}	3.49×10^{-1}	3.60×10^{-1}	4.85×10^{-1}	3.39×10^{-1}	3.00×10^{-1}
	Worst	4.57×10^{-1}	4.59×10^{-1}	4.85×10^{-1}	4.64×10^{-1}	3.97×10^{-1}	5.66×10^{-1}	5.11×10^{-1}	3.89×10^{-1}	3.60×10^{-1}
	Mean	4.32×10^{-1}	4.12×10^{-1}	4.72×10^{-1}	4.45×10^{-1}	3.76×10^{-1}	4.42×10^{-1}	4.94×10^{-1}	3.59×10^{-1}	3.38×10^{-1}
	Std	1.54×10^{-2}	2.70×10^{-2}	1.04×10^{-2}	9.28×10^{-3}	1.38×10^{-2}	5.63×10^{-2}	7.52×10^{-3}	1.47×10^{-2}	1.55×10^{-2}
	Accuracy	69.49	71.19	56.78	60.17	72.88	60.59	56.78	74.58	75.43
	Rank	5	4	8	7	3	6	8	2	1
Seeds	Best	6.25×10^{-2}	4.32×10^{-2}	2.35×10^{-1}	8.39×10^{-2}	1.74×10^{-3}	4.29×10^{-2}	2.20×10^{-1}	1.29×10^{-2}	1.43×10^{-2}
	Worst	3.84×10^{-1}	1.66×10^{-1}	6.76×10^{-1}	1.19×10^{-1}	8.10×10^{-2}	3.57×10^{-1}	5.12×10^{-1}	9.35×10^{-2}	9.35×10^{-2}
	Mean	2.25×10^{-1}	8.63×10^{-2}	3.99×10^{-1}	9.85×10^{-2}	4.64×10^{-2}	1.04×10^{-1}	3.47×10^{-1}	5.77×10^{-2}	4.94×10^{-2}
	Std	1.48×10^{-1}	3.35×10^{-2}	1.38×10^{-1}	8.74×10^{-3}	2.05×10^{-2}	8.62×10^{-2}	7.81×10^{-2}	2.23×10^{-2}	2.18×10^{-2}
	Accuracy	73.38	90.14	70.42	92.96	92.96	88.73	78.87	94.37	94.43
	Rank	7	4	8	3	3	5	6	2	1
Wine	Best	4.68×10^{-2}	8.55×10^{-3}	3.93×10^{-1}	1.97×10^{-2}	8.68×10^{-6}	8.55×10^{-3}	2.39×10^{-1}	8.54×10^{-3}	0
	Worst	6.07×10^{-1}	3.32×10^{-1}	7.26×10^{-1}	7.55×10^{-2}	5.98×10^{-2}	4.27×10^{-1}	6.48×10^{-1}	1.03×10^{-1}	5.12×10^{-2}
	Mean	2.91×10^{-1}	8.49×10^{-2}	5.36×10^{-1}	4.23×10^{-2}	3.28×10^{-2}	1.34×10^{-1}	4.56×10^{-1}	4.49×10^{-2}	1.75×10^{-2}
	Std	1.62×10^{-1}	7.77×10^{-2}	8.74×10^{-2}	1.60×10^{-2}	1.56×10^{-2}	1.42×10^{-1}	1.05×10^{-1}	2.74×10^{-2}	1.47×10^{-2}
	Accuracy	86.89	91.80	54.10	88.52	86.88	90.16	77.05	93.44	95.08
	Rank	6	3	9	5	7	4	8	2	1

Table 4. *Cont.*

Datasets	Result	ALO	AVOA	DOA	FPA	MFO	SSA	SSO	MPA	EMPA
Iris	Best	3.56×10^{-2}	9.01×10^{-4}	1.95×10^{-1}	9.34×10^{-2}	6.57×10^{-4}	4.60×10^{-3}	2.12×10^{-1}	0	0
	Worst	2.30×10^{-1}	1.21×10^{-1}	6.02×10^{-1}	1.54×10^{-1}	4.56×10^{-2}	3.85×10^{-1}	4.58×10^{-1}	7.77×10^{-2}	7.07×10^{-2}
	Mean	1.55×10^{-1}	4.55×10^{-2}	3.99×10^{-1}	1.21×10^{-1}	2.23×10^{-2}	8.86×10^{-2}	3.35×10^{-1}	4.27×10^{-2}	3.64×10^{-2}
	Std	6.72×10^{-2}	3.16×10^{-2}	1.20×10^{-1}	1.98×10^{-2}	1.15×10^{-2}	1.28×10^{-1}	6.16×10^{-2}	2.20×10^{-2}	1.48×10^{-2}
	Accuracy	89.22	85.29	90.20	97.55	88.24	90.20	90.20	98.04	98.23
	Rank	5	7	4	3	6	4	4	2	1
Statlog	Best	1.85×10^{-1}	1.20×10^{-1}	3.17×10^{-1}	1.74×10^{-1}	1.51×10^{-1}	1.97×10^{-1}	3.22×10^{-1}	1.02×10^{-1}	8.25×10^{-2}
	Worst	5.57×10^{-1}	2.92×10^{-1}	4.81×10^{-1}	2.30×10^{-1}	2.51×10^{-1}	3.03×10^{-1}	4.93×10^{-1}	2.42×10^{-1}	1.74×10^{-1}
	Mean	2.64×10^{-1}	2.36×10^{-1}	3.92×10^{-1}	1.96×10^{-1}	1.79×10^{-1}	2.37×10^{-1}	4.09×10^{-1}	1.56×10^{-1}	1.23×10^{-1}
	Std	7.46×10^{-2}	4.60×10^{-2}	4.52×10^{-2}	1.57×10^{-2}	2.41×10^{-2}	2.90×10^{-2}	5.01×10^{-2}	4.32×10^{-2}	2.68×10^{-2}
	Accuracy	81.52	80.43	67.39	80.43	83.70	80.43	75	85.87	85.96
	Rank	4	5	7	5	3	5	6	2	1
XOR	Best	2.53×10^{-3}	0	0	2.21×10^{-2}	0	1.77×10^{-56}	3.04×10^{-3}	0	0
	Worst	8.96×10^{-2}	5.00×10^{-1}	5.00×10^{-1}	6.51×10^{-2}	4.58×10^{-11}	2.50×10^{-1}	3.27×10^{-1}	0	0
	Mean	1.90×10^{-2}	3.00×10^{-1}	2.24×10^{-1}	3.49×10^{-2}	2.30×10^{-12}	1.25×10^{-2}	1.87×10^{-1}	0	0
	Std	2.01×10^{-2}	2.51×10^{-1}	2.19×10^{-1}	1.28×10^{-2}	1.02×10^{-11}	5.59×10^{-2}	1.08×10^{-1}	0	0
	Accuracy	100	50	75	75	50	75	50	100	100
	Rank	1	3	2	2	3	2	3	1	1
Balloon	Best	1.37×10^{-9}	0	0	1.83×10^{-5}	0	0	1.82×10^{-4}	0	0
	Worst	9.06×10^{-5}	0	4.96×10^{-1}	9.38×10^{-4}	0	5.06×10^{-30}	2.13×10^{-1}	0	0
	Mean	2.36×10^{-5}	0	1.45×10^{-1}	2.23×10^{-4}	0	2.54×10^{-31}	6.22×10^{-2}	0	0
	Std	3.12×10^{-5}	0	1.56×10^{-1}	2.49×10^{-4}	0	1.13×10^{-30}	6.55×10^{-2}	0	0
	Accuracy	100	100	80	100	80	80	100	100	100
	Rank	1	1	2	1	2	2	1	1	1
Cancer	Best	3.59×10^{-2}	3.35×10^{-2}	5.92×10^{-2}	3.44×10^{-2}	2.43×10^{-2}	3.69×10^{-2}	7.07×10^{-2}	2.44×10^{-2}	2.15×10^{-2}
	Worst	2.55×10^{-1}	7.77×10^{-2}	2.91×10^{-1}	4.82×10^{-2}	4.84×10^{-2}	5.80×10^{-2}	1.58×10^{-1}	5.01×10^{-2}	4.50×10^{-2}
	Mean	7.66×10^{-2}	4.78×10^{-2}	2.22×10^{-1}	4.41×10^{-2}	3.70×10^{-2}	4.62×10^{-2}	1.15×10^{-1}	3.84×10^{-2}	3.45×10^{-2}
	Std	7.31×10^{-2}	9.00×10^{-3}	7.57×10^{-2}	3.59×10^{-3}	5.19×10^{-3}	5.51×10^{-3}	2.60×10^{-2}	6.08×10^{-3}	7.04×10^{-3}
	Accuracy	99	99	98	99	99	99	99	99	99
	Rank	1	1	2	1	1	1	1	1	1
Diabetes	Best	3.22×10^{-1}	3.03×10^{-1}	3.58×10^{-1}	3.17×10^{-1}	2.88×10^{-1}	3.15×10^{-1}	3.87×10^{-1}	2.93×10^{-1}	2.70×10^{-1}
	Worst	3.74×10^{-1}	4.13×10^{-1}	4.58×10^{-1}	3.63×10^{-1}	3.22×10^{-1}	5.41×10^{-1}	4.83×10^{-1}	3.15×10^{-1}	3.12×10^{-1}
	Mean	3.45×10^{-1}	3.32×10^{-1}	4.14×10^{-1}	3.38×10^{-1}	3.03×10^{-1}	3.90×10^{-1}	4.47×10^{-1}	3.03×10^{-1}	2.90×10^{-1}
	Std	1.53×10^{-2}	2.48×10^{-2}	2.64×10^{-2}	1.33×10^{-2}	7.50×10^{-3}	6.06×10^{-2}	2.76×10^{-2}	5.61×10^{-3}	9.99×10^{-3}
	Accuracy	78.16	77.01	74.33	80.08	77.78	78.93	67.82	80.84	80.93
	Rank	5	7	8	3	6	4	9	2	1
Gene	Best	2.43×10^{-1}	7.14×10^{-2}	2.83×10^{-1}	5.08×10^{-4}	7.14×10^{-2}	1.57×10^{-1}	3.77×10^{-1}	7.95×10^{-11}	0
	Worst	4.14×10^{-1}	3.57×10^{-1}	9.00×10^{-1}	2.45×10^{-1}	2.71×10^{-1}	3.57×10^{-1}	4.50×10^{-1}	1.43×10^{-1}	5.71×10^{-2}
	Mean	3.19×10^{-1}	2.18×10^{-1}	3.93×10^{-1}	1.08×10^{-1}	1.78×10^{-1}	2.79×10^{-1}	4.15×10^{-1}	5.22×10^{-2}	2.35×10^{-2}
	Std	5.53×10^{-2}	7.55×10^{-2}	1.57×10^{-1}	5.97×10^{-2}	5.22×10^{-2}	5.78×10^{-2}	1.86×10^{-2}	4.38×10^{-2}	1.75×10^{-2}
	Accuracy	5.56	19.44	8.33	30.56	25	8.33	2.78	33.33	40.33
	Rank	7	5	6	3	4	6	8	2	1
Parkinson	Best	7.38×10^{-2}	1.99×10^{-2}	1.44×10^{-1}	3.87×10^{-2}	4.60×10^{-3}	3.88×10^{-2}	1.47×10^{-1}	1.8×10^{-121}	0
	Worst	2.33×10^{-1}	2.56×10^{-1}	4.04×10^{-1}	9.76×10^{-2}	1.86×10^{-1}	2.33×10^{-1}	2.96×10^{-1}	9.30×10^{-2}	9.30×10^{-2}
	Mean	1.44×10^{-1}	8.66×10^{-2}	3.06×10^{-1}	7.40×10^{-2}	5.36×10^{-2}	1.33×10^{-1}	2.17×10^{-1}	4.35×10^{-2}	4.50×10^{-2}
	Std	5.42×10^{-2}	6.39×10^{-2}	7.75×10^{-2}	1.71×10^{-2}	4.33×10^{-2}	6.27×10^{-2}	3.69×10^{-2}	3.88×10^{-2}	3.11×10^{-2}
	Accuracy	71.21	72.73	68.18	72.73	71.21	72.73	69.70	75.76	75.76
	Rank	3	2	5	2	3	2	4	1	1
Splice	Best	5.42×10^{-1}	2.80×10^{-1}	4.67×10^{-1}	3.88×10^{-1}	3.36×10^{-1}	4.50×10^{-1}	6.63×10^{-1}	1.27×10^{-1}	9.81×10^{-2}
	Worst	6.59×10^{-1}	4.74×10^{-1}	4.99×10^{-1}	4.86×10^{-1}	6.68×10^{-1}	6.15×10^{-1}	8.53×10^{-1}	1.55×10^{-1}	4.31×10^{-1}
	Mean	5.94×10^{-1}	3.86×10^{-1}	4.86×10^{-1}	4.33×10^{-1}	4.29×10^{-1}	5.47×10^{-1}	7.76×10^{-1}	1.41×10^{-1}	1.93×10^{-1}
	Std	3.32×10^{-2}	6.09×10^{-2}	9.92×10^{-3}	2.15×10^{-2}	8.67×10^{-2}	4.54×10^{-2}	5.22×10^{-2}	8.04×10^{-3}	1.12×10^{-1}
	Accuracy	52.65	74.12	50.88	64.71	77.94	59.12	48.53	82.65	83.27
	Rank	7	4	8	5	3	6	9	2	1
WDBC	Best	4.55×10^{-2}	2.23×10^{-2}	1.40×10^{-1}	4.42×10^{-2}	2.76×10^{-2}	4.57×10^{-2}	1.72×10^{-1}	1.38×10^{-2}	8.24×10^{-3}
	Worst	1.17×10^{-1}	2.83×10^{-1}	5.12×10^{-1}	6.67×10^{-2}	4.67×10^{-2}	1.22×10^{-1}	3.58×10^{-1}	6.25×10^{-2}	4.56×10^{-2}
	Mean	7.96×10^{-2}	5.96×10^{-2}	3.32×10^{-1}	5.50×10^{-2}	3.41×10^{-2}	7.18×10^{-2}	2.67×10^{-1}	4.16×10^{-2}	2.61×10^{-2}
	Std	2.16×10^{-2}	5.54×10^{-2}	1.15×10^{-1}	6.83×10^{-3}	4.78×10^{-3}	2.04×10^{-2}	4.76×10^{-2}	1.22×10^{-2}	1.03×10^{-2}
	Accuracy	91.52	94.55	86.67	95.76	93.94	92.12	85.45	98.79	98.85
	Rank	7	4	8	3	5	6	9	2	1
Zoo	Best	3.58×10^{-1}	7.46×10^{-2}	5.04×10^{-1}	1.57×10^{-1}	1.49×10^{-2}	8.96×10^{-2}	4.18×10^{-1}	1.49×10^{-2}	4.13×10^{-43}
	Worst	7.76×10^{-1}	5.52×10^{-1}	7.78×10^{-1}	3.73×10^{-1}	2.54×10^{-1}	7.01×10^{-1}	1.6×10^{1}	1.49×10^{-1}	1.11×10^{-1}
	Mean	5.11×10^{-1}	3.07×10^{-1}	6.11×10^{-1}	2.82×10^{-1}	1.13×10^{-1}	3.01×10^{-1}	9.52×10^{-1}	6.33×10^{-2}	4.40×10^{-2}
	Std	1.14×10^{-1}	1.24×10^{-1}	7.18×10^{-2}	5.46×10^{-2}	6.36×10^{-2}	1.69×10^{-1}	3.37×10^{-1}	3.22×10^{-2}	3.22×10^{-2}
	Accuracy	52.94	52.94	41.18	67.65	76.47	64.71	50	82.35	87.27
	Rank	6	6	8	4	3	5	7	2	1

The Wilcoxon rank–sum test was actualized to distinguish the EMPA and other algorithms [44]. $p < 0.05$ indicates that the discrepancy is noteworthy, $p \geq 0.05$ indicates that the discrepancy is not noteworthy, and N/A indicates that a "not applicable" discrepancy. The results of the *p*-value Wilcoxon rank–sum test are revealed in Table 5. The experimental results indicate that the discrepancy between EMPA and other algorithms was noteworthy.

Table 5. Results of the *p*-value Wilcoxon rank-sum test.

Datasets	ALO	AVOA	DOA	FPA	MFO	SSA	SSO	MPA
Blood	6.80×10^{-8}	1.20×10^{-6}	6.80×10^{-8}	6.80×10^{-8}	2.85×10^{-2}	9.17×10^{-8}	6.80×10^{-8}	7.58×10^{-4}
Scale	5.22×10^{-7}	2.59×10^{-5}	6.79×10^{-8}	7.89×10^{-8}	1.48×10^{-3}	1.41×10^{-5}	6.79×10^{-8}	1.56×10^{-3}
Survival	6.80×10^{-8}	6.80×10^{-8}	6.80×10^{-8}	6.80×10^{-8}	1.14×10^{-2}	1.06×10^{-7}	6.80×10^{-8}	1.08×10^{-3}
Liver	6.80×10^{-8}	6.80×10^{-8}	6.80×10^{-8}	6.80×10^{-8}	5.23×10^{-7}	7.90×10^{-8}	6.80×10^{-8}	7.58×10^{-4}
Seeds	3.95×10^{-6}	1.98×10^{-4}	6.71×10^{-8}	2.19×10^{-7}	9.68×10^{-5}	2.73×10^{-4}	6.71×10^{-8}	2.55×10^{-2}
Wine	7.31×10^{-8}	1.72×10^{-5}	6.29×10^{-8}	3.54×10^{-5}	1.56×10^{-3}	3.04×10^{-6}	6.29×10^{-8}	7.64×10^{-4}
Iris	5.17×10^{-7}	4.40×10^{-2}	6.71×10^{-8}	6.71×10^{-8}	3.18×10^{-3}	5.07×10^{-4}	6.71×10^{-8}	3.78×10^{-2}
Statlog	6.69×10^{-8}	6.83×10^{-7}	6.69×10^{-8}	7.78×10^{-8}	2.33×10^{-6}	6.69×10^{-8}	6.69×10^{-8}	1.14×10^{-3}
XOR	8.01×10^{-9}	4.68×10^{-5}	2.55×10^{-5}	8.01×10^{-9}	1.05×10^{-7}	8.01×10^{-9}	8.01×10^{-9}	N/A
Balloon	8.01×10^{-9}	N/A	2.49×10^{-5}	8.01×10^{-9}	3.42×10^{-3}	2.99×10^{-8}	8.01×10^{-9}	N/A
Cancer	7.49×10^{-6}	6.59×10^{-6}	6.68×10^{-8}	4.12×10^{-5}	1.98×10^{-4}	1.58×10^{-5}	6.68×10^{-8}	8.54×10^{-4}
Diabetes	6.80×10^{-8}	1.66×10^{-7}	6.80×10^{-8}	6.80×10^{-8}	1.04×10^{-4}	6.80×10^{-8}	6.80×10^{-8}	5.90×10^{-5}
Gene	5.97×10^{-8}	6.07×10^{-8}	6.07×10^{-8}	2.47×10^{-6}	6.07×10^{-8}	5.93×10^{-8}	6.07×10^{-8}	1.59×10^{-2}
Parkinson	2.55×10^{-7}	3.84×10^{-2}	6.75×10^{-8}	2.56×10^{-3}	7.56×10^{-4}	2.93×10^{-6}	6.75×10^{-8}	N/A
Splice	6.80×10^{-8}	1.25×10^{-5}	6.80×10^{-8}	2.96×10^{-7}	1.58×10^{-6}	6.80×10^{-8}	6.80×10^{-8}	2.85×10^{-5}
WDBC	7.89×10^{-8}	7.40×10^{-5}	6.79×10^{-8}	7.89×10^{-8}	1.14×10^{-2}	6.79×10^{-8}	6.79×10^{-8}	2.33×10^{-4}
Zoo	6.49×10^{-8}	1.18×10^{-7}	6.52×10^{-8}	6.52×10^{-8}	3.20×10^{-4}	1.56×10^{-7}	6.52×10^{-8}	5.27×10^{-4}

The convergent curves of the EMPA and other algorithms under different datasets are shown in Figures 3–19. The convergent curve is an important method to measure the overall optimization and traversal search, which not only intuitively reflects the convergence rate and computation accuracy of the feedforward neural networks trained by the EMPA and other algorithms, but also objectively observes the iteration process, as well as the stability and feasibility of different algorithms. For the blood, scale, survival, liver, seeds, wine, iris, statlog, XOR, balloon, cancer, diabetes, gene, parkinson, splice, WDBC and zoo datasets, compared with the ALO, AVOA, DOA, FPA, MFO, SSA, SSO, and MPA, the evaluation indexes of the EMPA were relatively better in optimal values, worst values, mean values, and standard deviations. The classification rate and ranking of the EMPA were superior to those of other algorithms. The convergence rate and computation accuracy of the EMPA were superior to those of the ALO, AVOA, DOA, FPA, MFO, SSA, SSO, and MPA, which indicates that the EMPA has remarkable feasibility and resilience to eliminate search stagnation and acquire the connection weight and deviation value. The optimal values and convergence effect of the EMPA were superior to those of other algorithms under different datasets. The EMPA has the properties of straightforward algorithm architecture, excellent control parameters, great traversal efficiency, strong stability, and easy implementation. The EMPA integrates exploration and exploitation to renew the position information and arrive at the global ideal solution. The EMPA is a practical and efficient method for training feedforward neural networks.

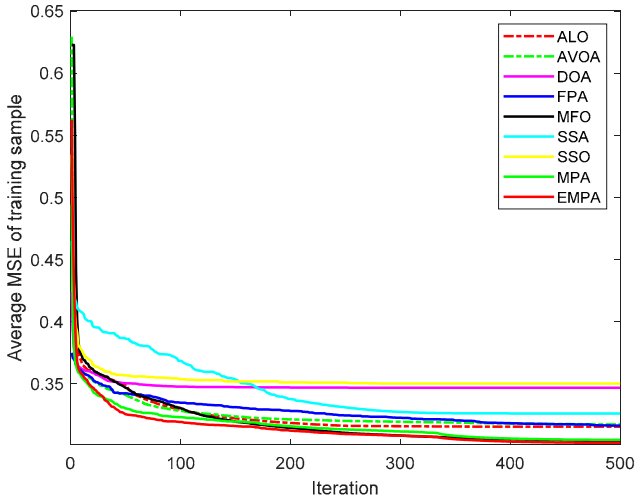

Figure 3. The convergent curves of Blood.

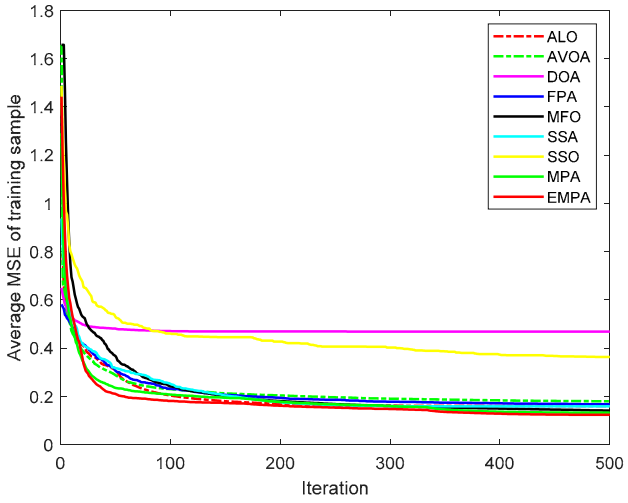

Figure 4. The convergent curves of Scale.

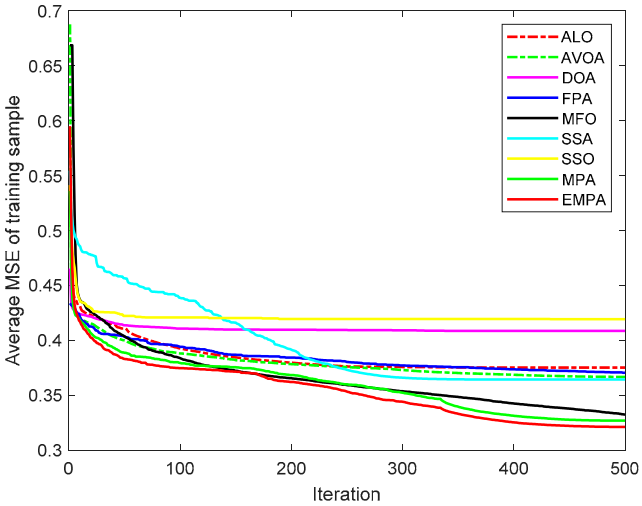

Figure 5. The convergent curves of Survival.

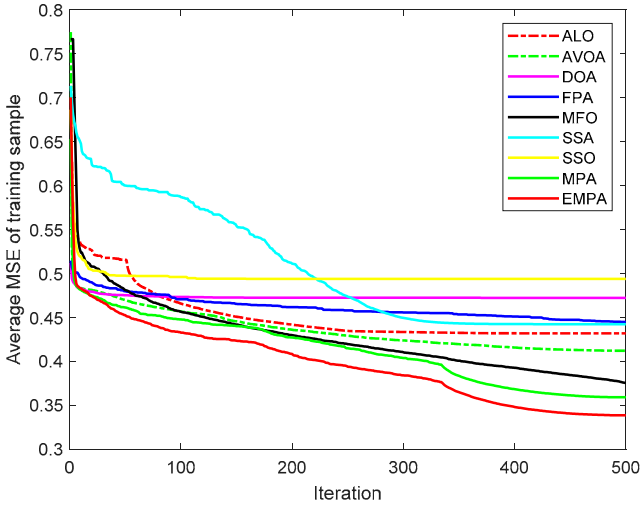

Figure 6. The convergent curves of Liver.

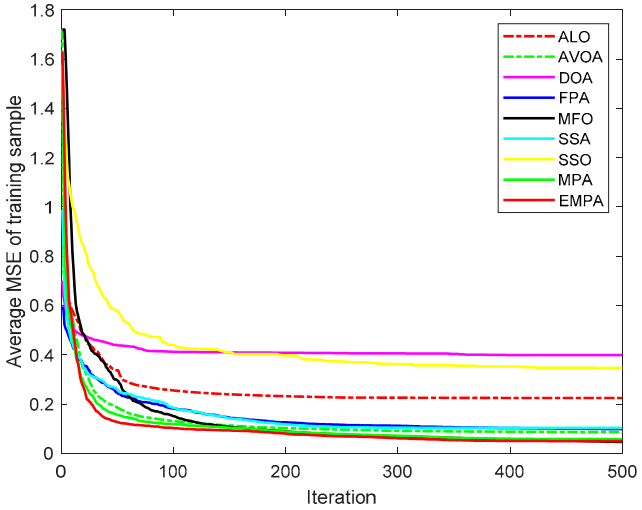

Figure 7. The convergent curves of Seeds.

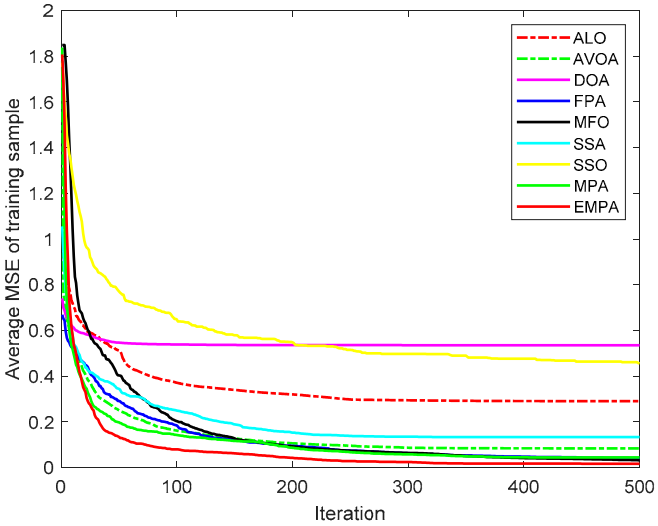

Figure 8. The convergent curves of Wine.

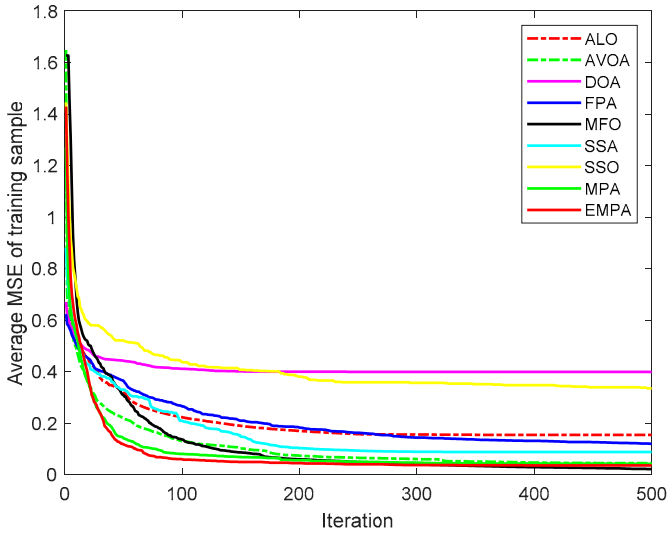

Figure 9. The convergent curves of Iris.

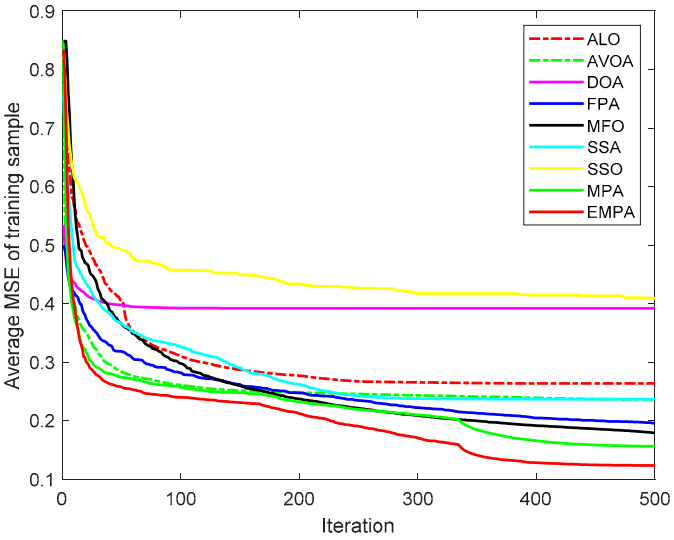

Figure 10. The convergent curves of Statlog.

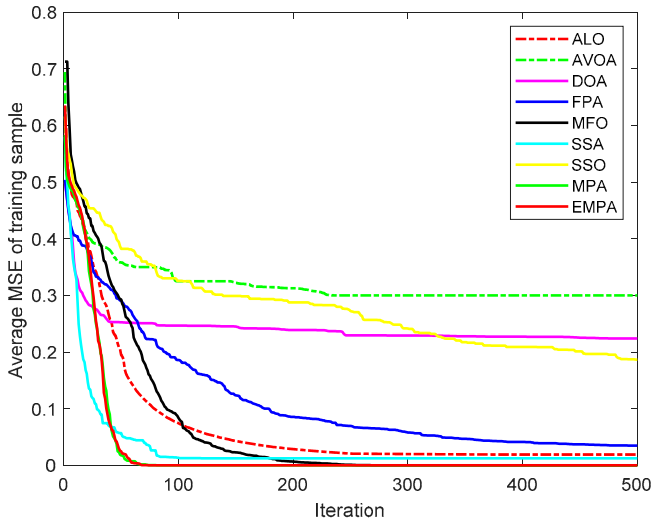

Figure 11. The convergent curves of XOR.

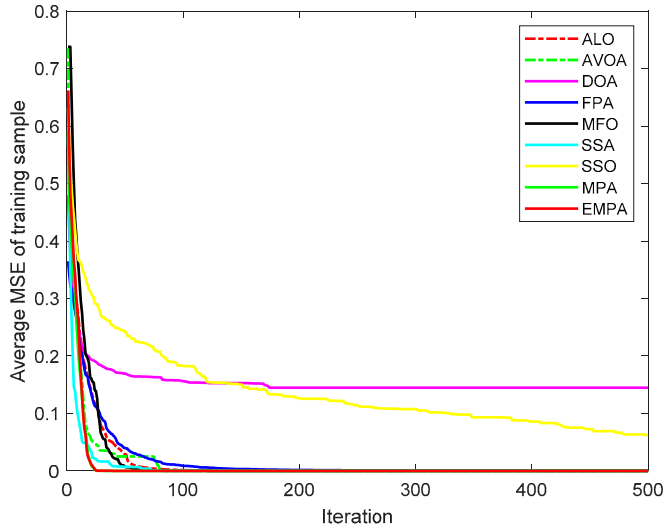

Figure 12. The convergent curves of Balloon.

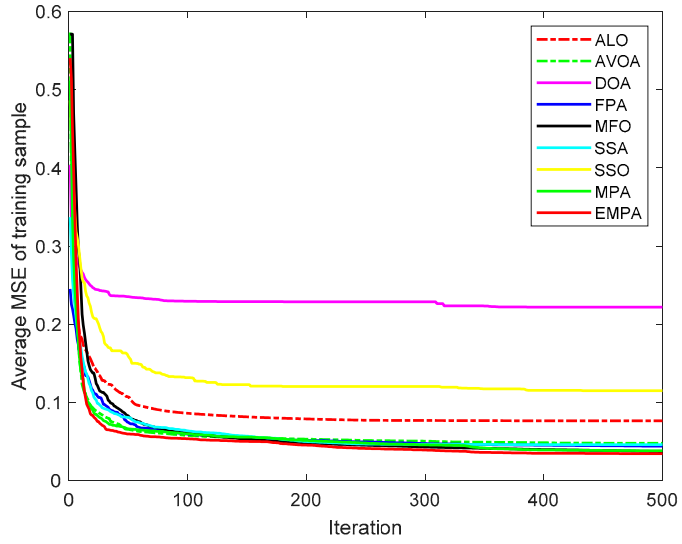

Figure 13. The convergent curves of Cancer.

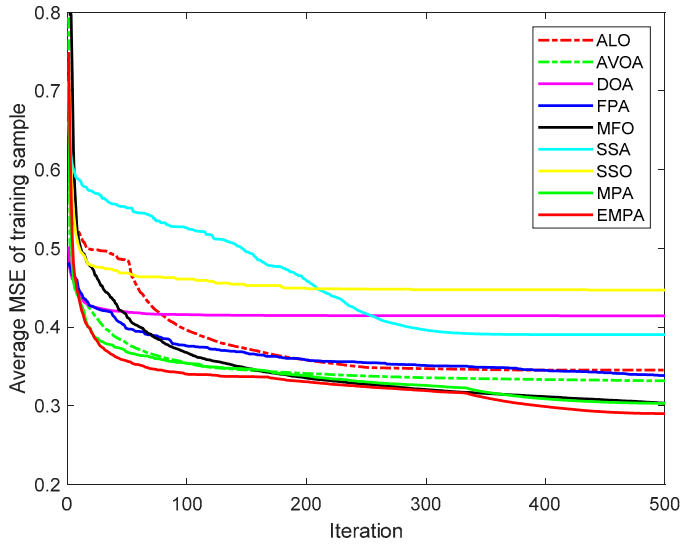

Figure 14. The convergent curves of Diabetes.

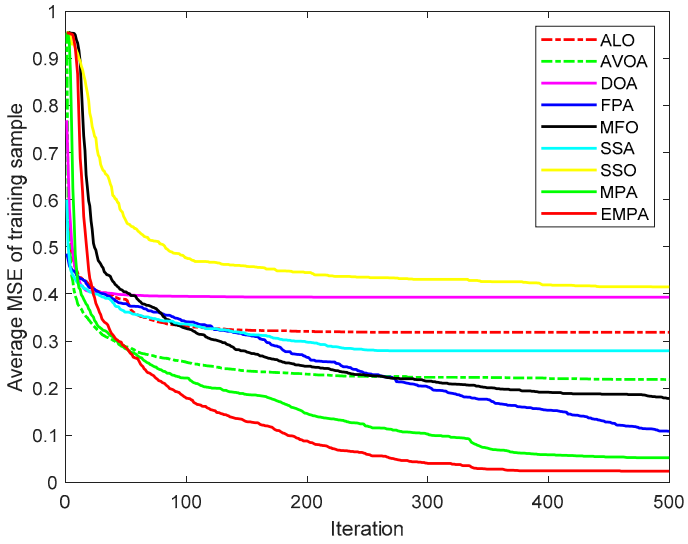

Figure 15. The convergent curves of Gene.

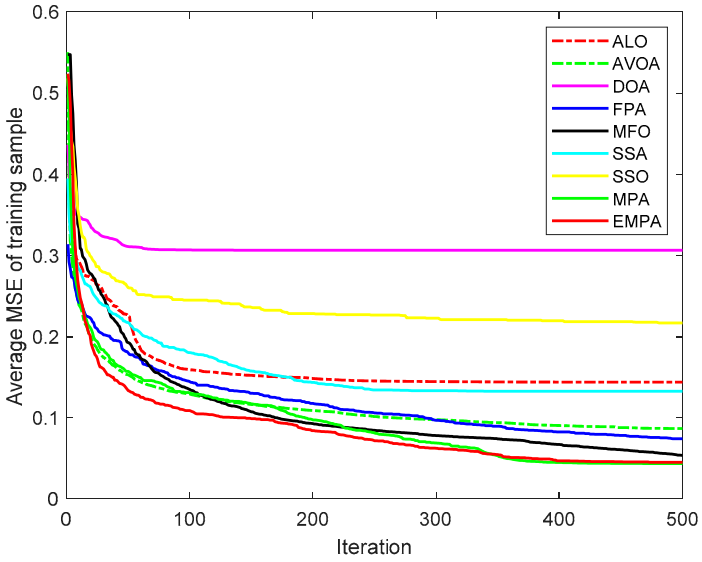

Figure 16. The convergent curves of Parkinson.

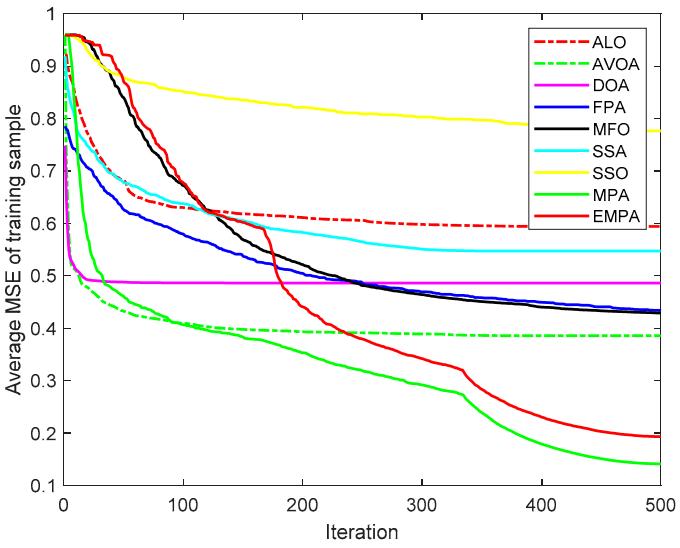

Figure 17. The convergent curves of Splice.

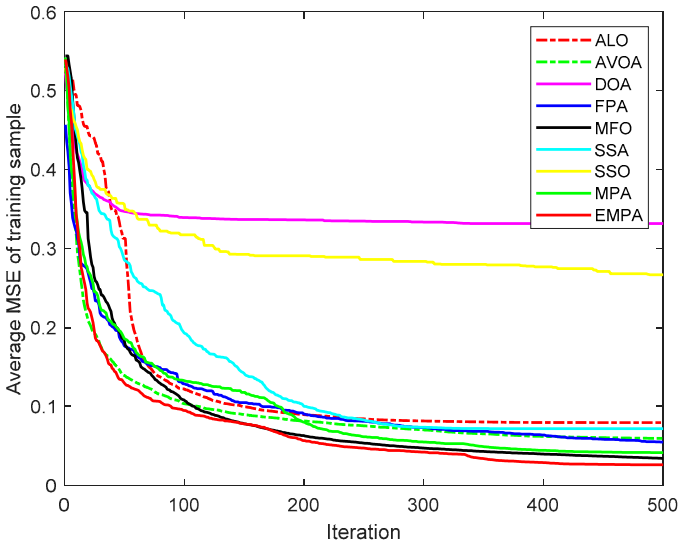

Figure 18. The convergent curves of WDBC.

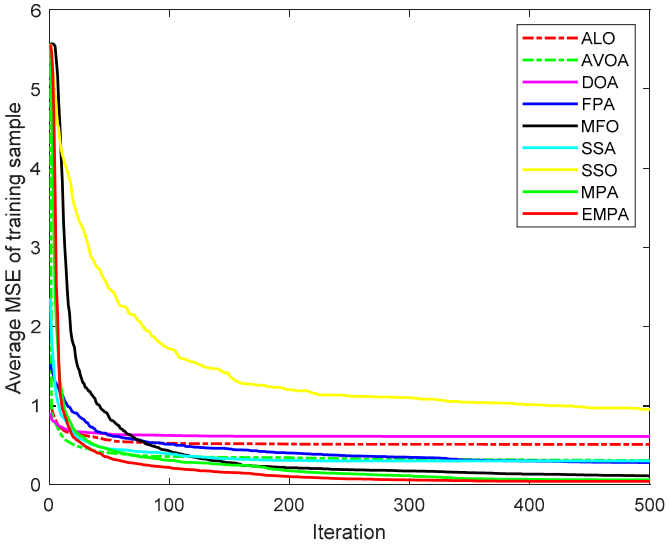

Figure 19. The convergent curves of Zoo.

The ANOVA tests of the EMPA and other algorithms under different datasets are shown in Figures 20–36. The standard deviation is an important method to measure the dispersion degree of data average values, which can accurately portray the stability and consistency of comparison algorithms in resolving the feedforward neural networks. The lower standard deviation showed that the algorithm has extensive exploration and exploitation to acquire more stable experimental data. For different datasets, the standard deviations of the EMPA were lower than those of the ALO, AVOA, DOA, FPA, MFO, SSA, SSO, and MPA, which indicates that the EMPA has exceptional stability and durability. The EMPA had greater computational efficiency and stronger dependability to attain a more stable standard deviation. The optimal values, worst values, mean values, classification rate and ranking of the EMPA were relatively better compared to ALO, AVOA, DOA, FPA, MFO, SSA, SSO, and MPA. The EMPA, based on the marine predators foraging strategy, utilizes a distinctive optimization mechanism of Lévy flight, Brownian motion, and the optimal encounter rate policy to determine the global optimal solution. The EMPA has strong global and local search abilities to avoid search stagnation and premature convergence, which enhances the convergence effect and optimization ability. The EMPA has strong stability and robustness to train the feedforward neural networks. Meanwhile, The EMPA

has a certain superiority and significance for receiving the better connection weight and deviation value.

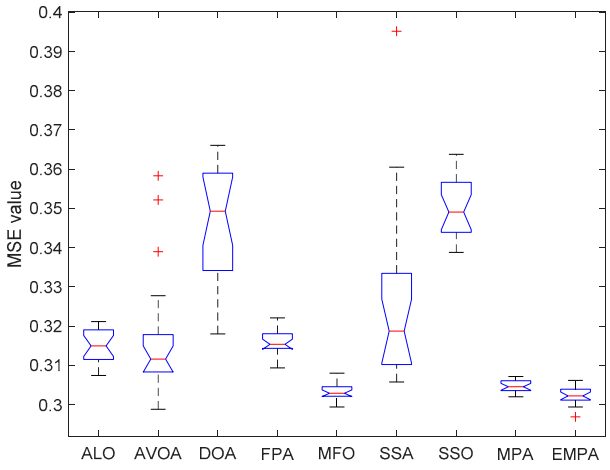

Figure 20. The ANOVA test of Blood.

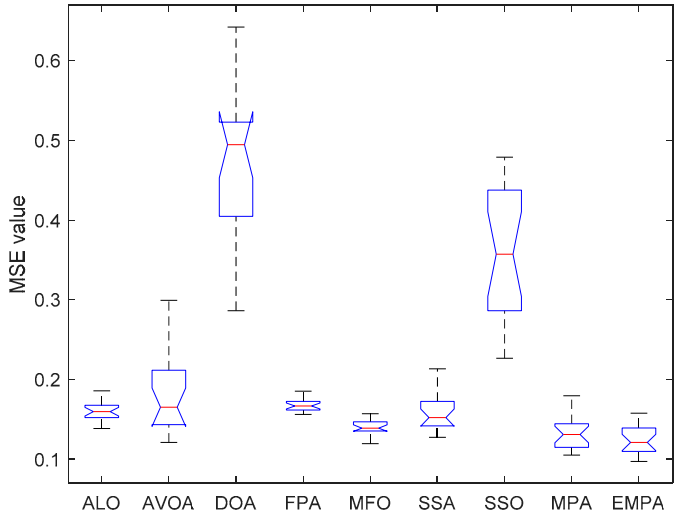

Figure 21. The ANOVA test of Scale.

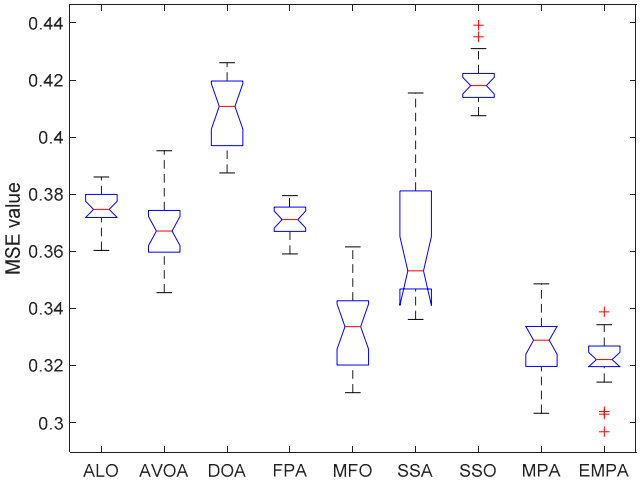

Figure 22. The ANOVA test of Survival.

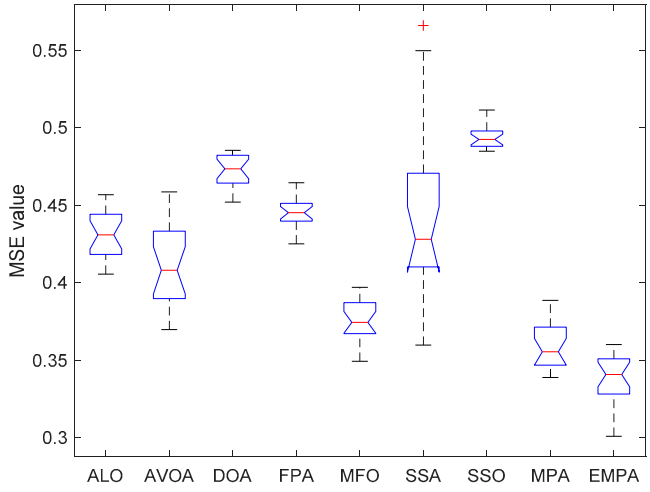

Figure 23. The ANOVA test of Liver.

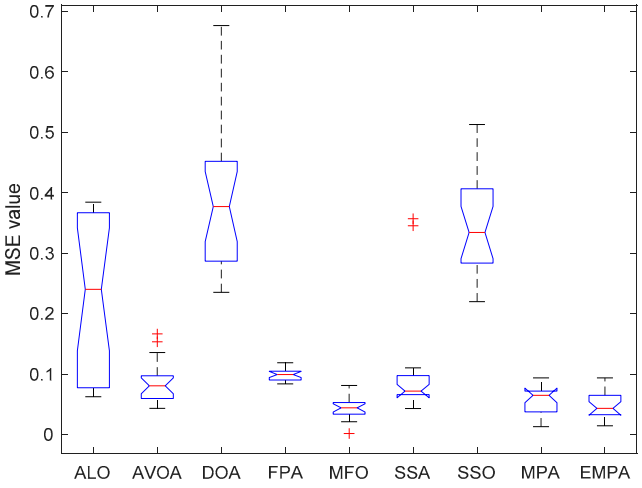

Figure 24. The ANOVA test of Seeds.

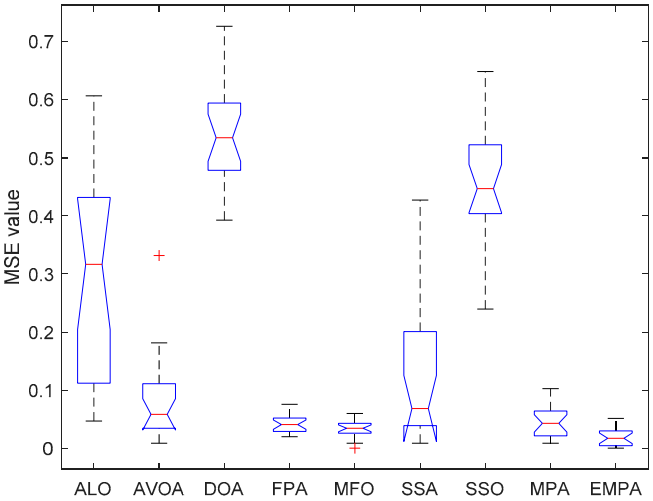

Figure 25. The ANOVA test of Wine.

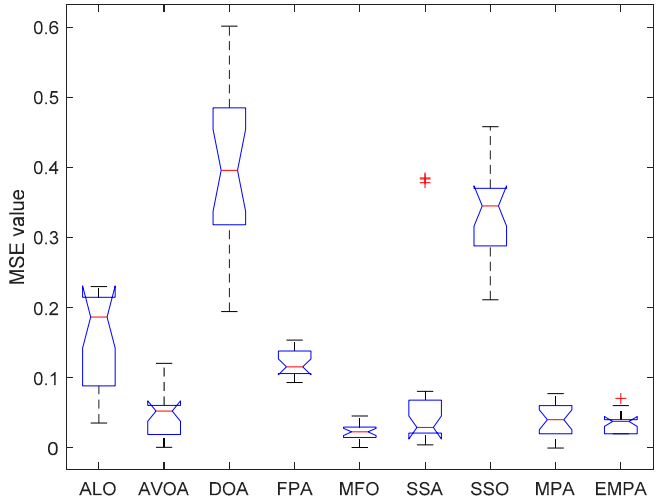

Figure 26. The ANOVA test of Iris.

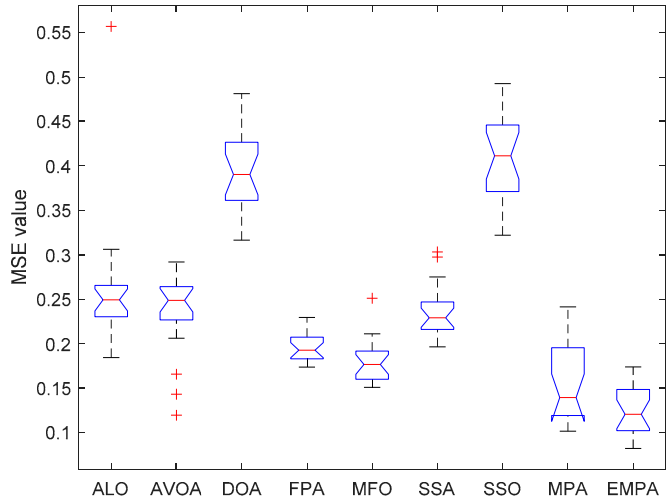

Figure 27. The ANOVA test of Statlog.

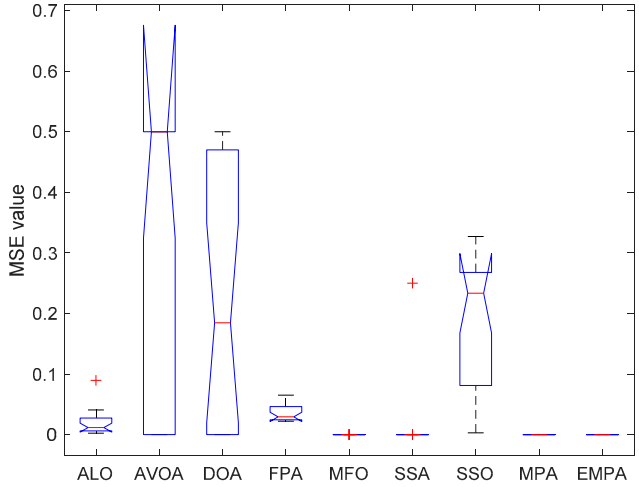

Figure 28. The ANOVA test of XOR.

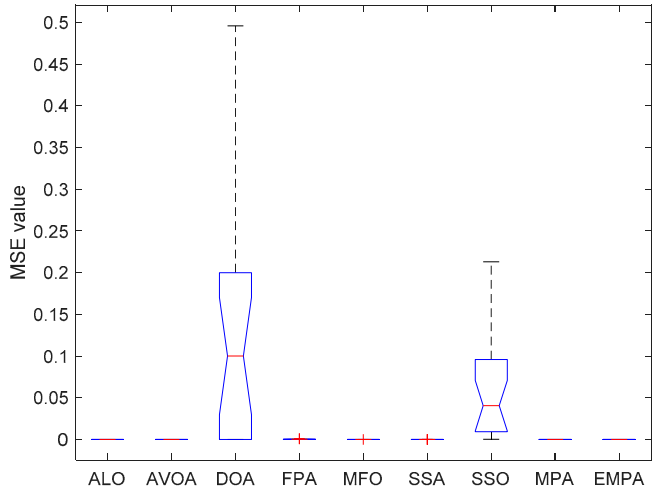

Figure 29. The ANOVA test of Balloon.

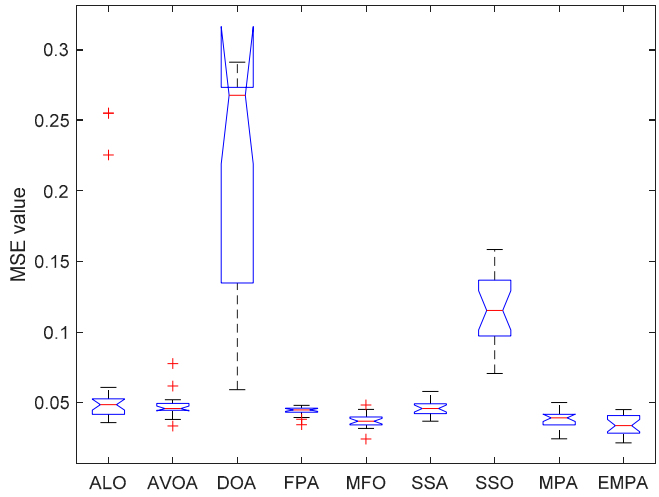

Figure 30. The ANOVA test of Cancer.

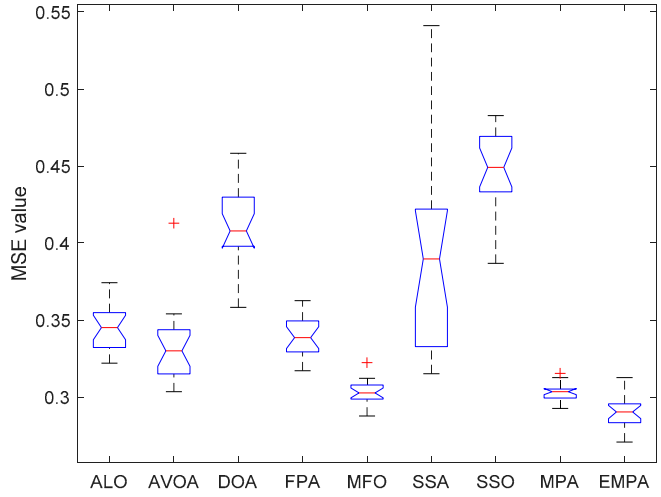

Figure 31. The ANOVA test of Diabetes.

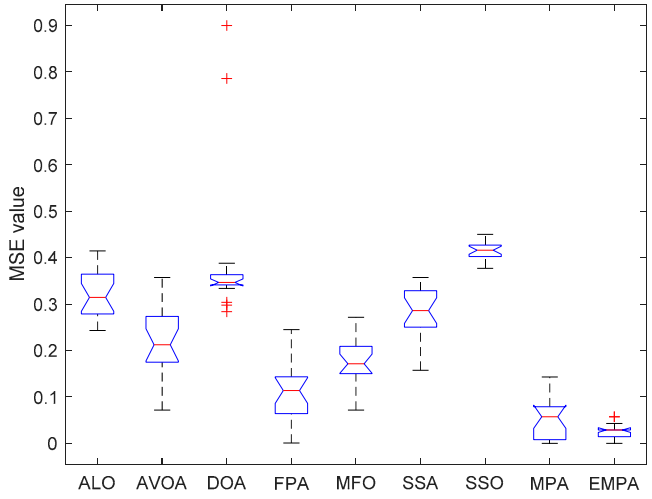

Figure 32. The ANOVA test of Gene.

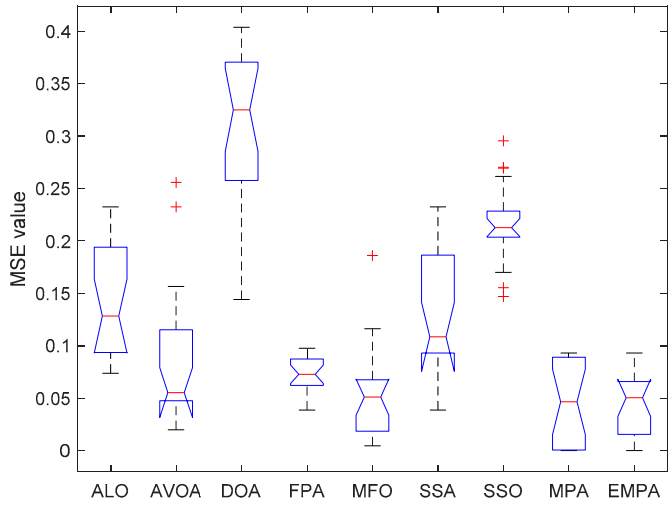

Figure 33. The ANOVA test of Parkinson.

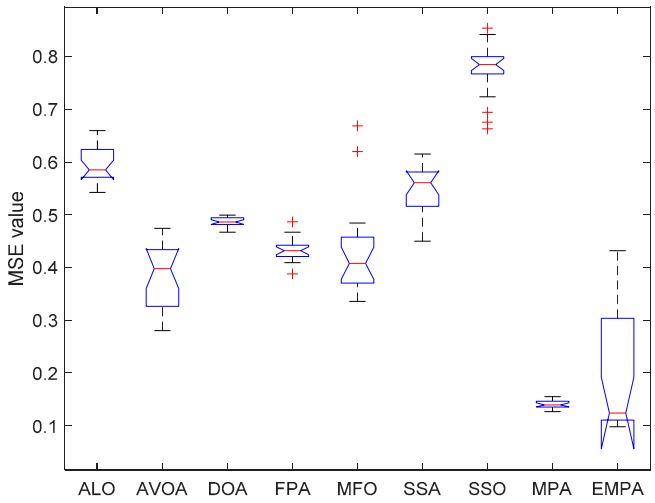

Figure 34. The ANOVA test of Splice.

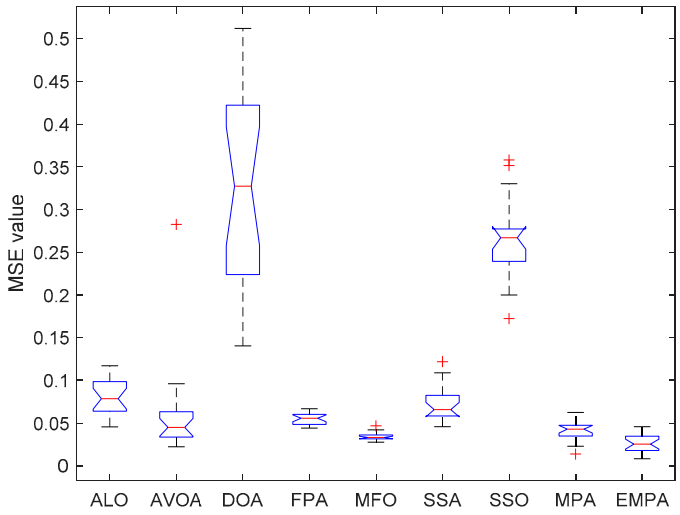

Figure 35. The ANOVA test of WDBC.

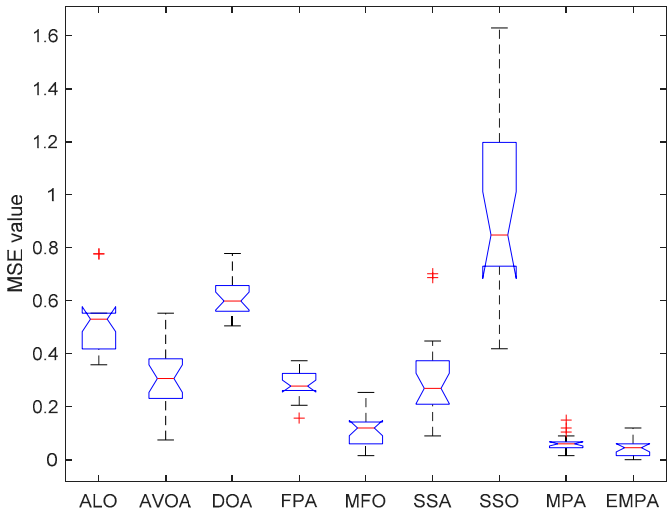

Figure 36. The ANOVA test of Zoo.

Statistically, the EMPA is based on the marine predators foraging strategy to imitate Lévy flight, Brownian motion, and the optimal encounter rate policy to arrive at the overall best solution. The EMPA was employed to resolve FNNs for the following reasons. First, the EMPA has the properties of straightforward algorithm architecture, excellent control parameters, great traversal efficiency, strong stability, and easy implementation. Second, the EMPA utilizes the Lévy flight, Brownian motion, and the optimal encounter rate policy to determine the best solution. The Lévy flight can increase the population diversity, expand the search space, enhance the exploitation ability, and improve the calculation accuracy. The Brownian motion and optimal encounter rate policy can filter out the best solution, avoid search stagnation, enhance the exploration ability, and accelerate the convergence speed. Third, the ranking-based mutation operator was introduced into the MPA. The EMPA not only balances exploration and exploitation to avoid falling into the local optimum and premature convergence, but it also utilizes a unique search mechanism to renew the position and identify the best solution. To summarize, the EMPA has a quicker convergence speed, greater calculation accuracy, higher classification rate, and strong stability and robustness. The EMPA has a strong overall optimization ability to train FNNs.

7. Conclusions and Future Research

In this paper, an enhanced MPA based on the ranking-based mutation operator was presented to train FNNs, and the objective was not only to determine the best combination

of connection weight and deviation value, but also to acquire the global best solution according to the given input value. The ranking-based mutation operator not only enhanced the selection probability to filter out the optimal search agent, but also mitigated search stagnation to accelerate convergence speed. The EMPA utilized the distinctive mechanisms of Lévy flight, Brownian motion, the optimal encounter rate policy, and the ranking-based mutation operator to attain the minimum classification, prediction and approximation errors. The EMPA had strong robustness, parallelism, and scalability to determine the best value. Compared with the other algorithms, the EMPA had excellent reliability and superiority to train FNNs. The experimental results demonstrate that the convergence speed, calculation accuracy and classification rate of the EMPA were superior to those of the other algorithms. Furthermore, the EMPA had strong practicability and feasibility for training FNNs.

In future research, we will utilize the DL methods, ML methods, and CNN. We will modify the activation function, such as RELU and sRELU. We will employ the random forest, XGBBOST, KNN, and FNN with other optimization algorithms. The EMPA will be utilized to resolve complex optimization problems, such as intelligent vehicle path planning, intelligent-temperature-controlled self-adjusting electric fans, and sensor information fusion.

Author Contributions: Conceptualization, J.Z. and Y.X.; methodology, J.Z. and Y.X..; software, J.Z. and Y.X.; validation, J.Z. and Y.X.; formal analysis, J.Z.; investigation, Y.X.; resources, J.Z. and Y.X.; data curation, J.Z. and Y.X.; writing—original draft preparation, J.Z.; writing—review and editing, J.Z. and Y.X.; visualization, J.Z. and Y.X.; supervision, Y.X.; project administration, J.Z. and Y.X.; funding acquisition, J.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Start-up Fee for Scientific Research of High-level Talents in 2022 under Grant No. 00701092336.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All data used to support the findings of this study are included within the article.

Acknowledgments: This research was funded by the Start-up Fee for Scientific Research of High-level Talents in 2022 under Grant No. 00701092336. The authors would like to thank everyone involved for their contribution to this article. They also would like to thank the editor and anonymous reviewers for the helpful comments and suggestions.

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

Abbreviations

The following abbreviations are used in this manuscript:

FNNs	Feedforward Neural Networks
MPA	Marine Predators Algorithm
EMPA	Enhanced Marine Predators Algorithm
UCI	University of California Irvine
ALO	Ant Lion Optimization
AVOA	African Vultures Optimization Algorithm
DOA	Dingo Optimization Algorithm
FPA	Flower Pollination Algorithm
MFO	Moth Flame Optimization
SSA	Salp Swarm Algorithm
SSO	Sperm Swarm Optimization
MLP	Multilayer Perception
MSE	Mean Squared Error
N/A	Not Applicable

References

1. Üstün, O.; Bekiroğlu, E.; Önder, M. Design of highly effective multilayer feedforward neural network by using genetic algorithm. *Expert Syst.* **2020**, *37*, e12532. [CrossRef]
2. Yin, Y.; Tu, Q.; Chen, X. Enhanced Salp Swarm Algorithm based on random walk and its application to training feedforward neural networks. *Soft Comput.* **2020**, *24*, 14791–14807. [CrossRef]
3. Troumbis, I.A.; Tsekouras, G.E.; Tsimikas, J.; Kalloniatis, C.; Haralambopoulos, D. A Chebyshev polynomial feedforward neural network trained by differential evolution and its application in environmental case studies. *Environ. Model. Softw.* **2020**, *126*, 104663. [CrossRef]
4. Al-Majidi, S.D.; Abbod, M.F.; Al-Raweshidy, H.S. A particle swarm optimisation-trained feedforward neural network for predicting the maximum power point of a photovoltaic array. *Eng. Appl. Artif. Intell.* **2020**, *92*, 103688. [CrossRef]
5. Truong, T.T.; Dinh-Cong, D.; Lee, J.; Nguyen-Thoi, T. An effective deep feedforward neural networks (DFNN) method for damage identification of truss structures using noisy incomplete modal data. *J. Build. Eng.* **2020**, *30*, 101244. [CrossRef]
6. Mirjalili, S. The ant lion optimizer. *Adv. Eng. Softw.* **2015**, *83*, 80–98. [CrossRef]
7. Abdollahzadeh, B.; Gharehchopogh, F.S.; Mirjalili, S. African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems. *Comput. Ind. Eng.* **2021**, *158*, 107408. [CrossRef]
8. Bairwa, A.K.; Joshi, S.; Singh, D. Dingo optimizer: A nature-inspired metaheuristic approach for engineering problems. *Math. Probl. Eng.* **2021**, *2021*, 2571863. [CrossRef]
9. Yang, X.-S.; Karamanoglu, M.; He, X. Flower pollination algorithm: A novel approach for multiobjective optimization. *Eng. Optim.* **2014**, *46*, 1222–1237. [CrossRef]
10. Mirjalili, S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowl. Based Syst.* **2015**, *89*, 228–249. [CrossRef]
11. Mirjalili, S.; Gandomi, A.H.; Mirjalili, S.Z.; Saremi, S.; Faris, H.; Mirjalili, S.M. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw.* **2017**, *114*, 163–191. [CrossRef]
12. Eslami, M.; Babaei, B.; Shareef, H.; Khajehzadeh, M.; Arandian, B. Optimum design of damping controllers using modified Sperm swarm optimization. *IEEE Access* **2021**, *9*, 145592–145604. [CrossRef]
13. Zhang, S.; Xie, L. Grafting constructive algorithm in feedforward neural network learning. *Appl. Intell.* **2022**, 1–18. [CrossRef]
14. Fan, Y.; Yang, W. A Backpropagation Learning Algorithm with Graph Regularization for Feedforward Neural Networks. *Inf. Sci.* **2022**, *607*, 263–277. [CrossRef]
15. Qu, Z.; Liu, X.; Sun, L. Learnable antinoise-receiver algorithm based on a quantum feedforward neural network in optical quantum communication. *Phys. Rev. A* **2022**, *105*, 052427. [CrossRef]
16. Admon, M.R.; Senu, N.; Ahmadian, A.; Majid, Z.A.; Salahshour, S. A new efficient algorithm based on feedforward neural network for solving differential equations of fractional order. *Commun. Nonlinear Sci. Numer. Simul.* **2022**, *177*, 106968. [CrossRef]
17. Guo, W.; Qiu, H.; Liu, Z.; Zhu, J.; Wang, Q. An integrated model based on feedforward neural network and Taylor expansion for indicator correlation elimination. *Intell. Data Anal.* **2022**, *26*, 751–783. [CrossRef]
18. Zhang, G. Research on safety simulation model and algorithm of dynamic system based on artificial neural network. *Soft Comput.* **2022**, *26*, 7377–7386. [CrossRef]
19. Venkatachalapathy, P.; Mallikarjunaiah, S.M. A feedforward neural network framework for approximating the solutions to nonlinear ordinary differential equations. *Neural Comput. Appl.* **2022**, *35*, 1661–1673. [CrossRef]
20. Liao, G.; Zhang, L. Solving flows of dynamical systems by deep neural networks and a novel deep learning algorithm. *Math. Comput. Simul.* **2022**, *202*, 331–342. [CrossRef]
21. Shao, R.; Zhang, G.; Gong, X. Generalized robust training scheme using genetic algorithm for optical neural networks with imprecise components. *Photon. Res.* **2022**, *10*, 1868–1876. [CrossRef]
22. Wu, C.; Wang, C.; Kim, J.-W. Welding sequence optimization to reduce welding distortion based on coupled artificial neural network and swarm intelligence algorithm. *Eng. Appl. Artif. Intell.* **2022**, *114*, 105142. [CrossRef]
23. Raziani, S.; Ahmadian, S.; Jalali, S.M.J.; Chalechale, A. An Efficient Hybrid Model Based on Modified Whale Optimization Algorithm and Multilayer Perceptron Neural Network for Medical Classification Problems. *J. Bionic Eng.* **2022**, *19*, 1504–1521. [CrossRef]
24. Dong, Z.; Huang, H. A training algorithm with selectable search direction for complex-valued feedforward neural networks. *Neural Netw.* **2021**, *137*, 75–84. [CrossRef]
25. Fontes, C.H.; Embiruçu, M. An approach combining a new weight initialization method and constructive algorithm to configure a single Feedforward Neural Network for multi-class classification. *Eng. Appl. Artif. Intell.* **2021**, *106*, 104495. [CrossRef]
26. Zheng, M.; Luo, J.; Dang, Z. Feedforward neural network based time-varying state-transition-matrix of Tschauner-Hempel equations. *Adv. Space Res.* **2022**, *69*, 1000–1011. [CrossRef]
27. Yılmaz, O.; Bas, E.; Egrioglu, E. The training of Pi-Sigma artificial neural networks with differential evolution algorithm for forecasting. *Comput. Econ.* **2022**, *59*, 1699–1711. [CrossRef]
28. Luo, Q.; Li, J.; Zhou, Y.; Liao, L. Using spotted hyena optimizer for training feedforward neural networks. *Cogn. Syst. Res.* **2021**, *65*, 1–16. [CrossRef]
29. Askari, Q.; Younas, I. Political optimizer based feedforward neural network for classification and function approximation. *Neural Process. Lett.* **2021**, *53*, 429–458. [CrossRef]

30. Duman, S.; Dalcalı, A.; Özbay, H. Manta ray foraging optimization algorithm–based feedforward neural network for electric energy consumption forecasting. *Int. Trans. Electr. Energy Syst.* **2021**, *31*, e12999. [CrossRef]
31. Pan, X.-M.; Song, B.-Y.; Wu, D.; Wei, G.; Sheng, X.-Q. On Phase Information for Deep Neural Networks to Solve Full-Wave Nonlinear Inverse Scattering Problems. *IEEE Antennas Wirel. Propag. Lett.* **2021**, *20*, 1903–1907. [CrossRef]
32. Wu, Q.; Chen, Z.; Chen, D.; Li, S. Beetle antennae search strategy for neural network model optimization with application to glomerular filtration rate estimation. *Neural Process. Lett.* **2021**, *53*, 1501–1522. [CrossRef]
33. Mahmoud, M.A.B.; Guo, P.; Fathy, A.; Li, K. SRCNN-PIL: Side Road Convolution Neural Network Based on Pseudoinverse Learning Algorithm. *Neural Process. Lett.* **2021**, *53*, 4225–4237. [CrossRef]
34. Jamshidi, M.B.; Daneshfar, F. A Hybrid Echo State Network for Hypercomplex Pattern Recognition, Classification, and Big Data Analysis. In Proceedings of the 2022 12th International Conference on Computer and Knowledge Engineering (ICCKE), Mashhad, Iran, 17–18 November 2022; IEEE: Piscataway, NJ, USA; pp. 7–12.
35. Khalaj, O.; Jamshidi, M.B.; Saebnoori, E.; Masek, B.; Stadler, C.; Svoboda, J. Hybrid machine learning techniques and computational mechanics: Estimating the dynamic behavior of oxide precipitation hardened steel. *IEEE Access* **2021**, *9*, 156930–156946. [CrossRef]
36. Daneshfar, F.; Jamshidi, M.B. An Octonion-Based Nonlinear Echo State Network for Speech Emotion Recognition in Metaverse. *SSRN Electron. J.* **2022**, 4242011.
37. Elminaam, D.S.A.; Nabil, A.; Ibraheem, S.A.; Houssein, E.H. An efficient marine predators algorithm for feature selection. *IEEE Access* **2021**, *9*, 60136–60153. [CrossRef]
38. Zhang, J.; Li, X.; Tian, J.; Jiang, Y.; Luo, H.; Yin, S. A variational local weighted deep sub-domain adaptation network for remaining useful life prediction facing cross-domain condition. *Reliab. Eng. Syst. Saf.* **2023**, *231*, 108986. [CrossRef]
39. Zhang, J.; Zhang, K.; An, Y.; Luo, H.; Yin, S. An Integrated Multitasking Intelligent Bearing Fault Diagnosis Scheme Based on Representation Learning Under Imbalanced Sample Condition. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, 1–12. [CrossRef]
40. Zhang, J.; Li, X.; Tian, J.; Luo, H.; Yin, S. An integrated multi-head dual sparse self-attention network for remaining useful life prediction. *Reliab. Eng. Syst. Saf.* **2023**, *233*, 109096. [CrossRef]
41. Zhang, J.; Tian, J.; Li, M.; Leon, J.I.; Franquelo, L.G.; Luo, H.; Yin, S. A parallel hybrid neural network with integration of spatial and temporal features for remaining useful life prediction in prognostics. *IEEE Trans. Instrum. Meas.* **2022**, *72*, 1–12. [CrossRef]
42. Faramarzi, A.; Heidarinejad, M.; Mirjalili, S.; Gandomi, A.H. Marine Predators Algorithm: A nature-inspired metaheuristic. *Expert Syst. Appl.* **2020**, *152*, 113377. [CrossRef]
43. Yan, Z.; Zhang, J.; Zeng, J.; Tang, J. Nature-inspired approach: An enhanced whale optimization algorithm for global optimization. *Math. Comput. Simul.* **2021**, *185*, 17–46. [CrossRef]
44. Bridge, P.D.; Sawilowsky, S.S. Increasing physicians' awareness of the impact of statistics on research outcomes: Comparative power of the t-test and Wilcoxon rank-sum test in small samples applied research. *J. Clin. Epidemiol.* **1999**, *52*, 229–235. [CrossRef] [PubMed]