MANAS Journal of Engineering
ISSN 1694-7398 | e-ISSN 1694-7398

Volume 9, Issue 2, (2021) Pages 160-168
https://doi.org/10.51354/mjen. 917837

Training of the feed-forward artificial neural networks using butterfly optimization algorithm

Büşra Irmak, Şaban Gülcü*

Necmettin Erbakan University, Faculty of Engineering, Turkey, sgulcu@erbakan.edu.tr, ORCID: 0000-0003-0972-7954, ORCID: 0000-0001-7714-8861

ABSTRACT

Artificial Neural Network (ANN) learns from inputs and outputs. The values of the weights and biases in ANN are updated according to inputs and outputs. Researchers have proposed algorithms to train Multi-Layer Perceptron (MLP). However, classical techniques often face problems in solving this optimization problem. They tend to need large amounts of computing time, large amounts of memory. More importantly, they get stuck within the local optimum and produce poor-quality solutions. To overcome these difficulties, meta-heuristic algorithms have been used to train MLP. In this article, the Butterfly Optimization Algorithm (BOA) which was designed by modeling the behaviors of butterflies was used for the first time to train the multi-layer perceptron. The developed algorithm was named BOA-MLP where the BOA algorithm optimized the values of the weights and biases in the MLP. The success of the BOA-MLP algorithm was tested on five data sets (iris, breast cancer, heart, balloon and xor) which are frequently used in the literature. In the experiments, the BOA-MLP algorithm was compared with the BAT-MLP, SMS-MLP and BP algorithms. The average and standard deviation of the mean squared error, the average classification accuracy, the sensitivity, the specificity, the precision and the F1-score were used as the performance metrics. According to the experimental results, it is seen that the BOA-MLP algorithm surpasses the BAT-MLP, SMS-MLP and BP algorithms on all data sets and shows superior success.

ARTICLE INFO

Research article

Received: 27.04.2021
Accepted: 14.06.2021

Keywords:
artificial neural networks,
butterfly optimization algorithm,
multi-layer perceptron,
optimization,
training of artificial neural networks

Corresponding author

1. Introduction

Artificial Neural Networks (ANNs) are one of the most important and biggest inventions in its field. It was developed by taking the behavior of the human brain as a role model. ANN is used in many areas such as classification, recognition, prediction and optimization. ANN developed by imitating the human brain has proven its success in the field of optimization as in many other fields.

Artificial Neural Networks include studies for computer learning. Today, digital devices and digital systems occupy an important place in our lives. Computers are used in almost every field. Computers have gained a feature that summarizes large amounts of data over time and can make comments about events using this data, while only calculating or performing data transfers in the past years. Today, computers can both make decisions about events and learn the relationship between events. It is seen that computers used today have both the ability to establish the pattern between events and to make interpretations and decisions about events. Computers have the ability to solve difficult problems that are not mathematically represented. Thanks to the superior learning ability of ANNs, they offer solutions to complex problems to solve with traditional methods. Again, thanks to the tribe of learning, it can generalize/predict situations that have not been encountered before using known examples. The most commonly used ANN method while performing these operations is the Multi-Layer Perceptron (MLP).

The ability of ANN to give correct results and to make successful classifications is provided by updating the bias and weight values in the most appropriate way. In the literature, researchers have proposed algorithms to train Multi-Layer Perceptron. However, classical techniques often face problems in solving optimization problems in the real world. They tend to need large amounts of computing time, large amounts of memory. More importantly, they get stuck within the local optimum and produce poor-quality solutions. To overcome these difficulties, meta-heuristic algorithms have been used to train ANNs [1].

The meta-heuristic algorithms designed by taking the biological movements of living things as role models, such as hunting, reproduction and feeding, aim to find the optimum result by producing solutions to problems in a search space [2]. Many challenging problems have been optimized using meta-heuristic algorithms. Meta-heuristic algorithms have been developed by taking inspiration from their unique behaviors in order to continue the vital activities of living things in nature. For this reason, these algorithms are named Natural Optimization Algorithms [3].

Many meta-heuristic algorithms have been used in the literature for training ANN. Some of these are those: Mirjalili [4] used the recently proposed Gray Wolf Optimization (GWO) algorithm for multi-layer perceptron (MKP) training. Five classifications and eight standard data sets were used to test the proposed method. It was compared with the best-known meta-heuristics algorithms. In [5], harmony search algorithms applied for supervised training of feed-forward (FF) type ANNs, which are frequently used for classification problems. Five different types of fit search algorithms were examined, paying special attention to the self-adaptive global best-fit search (SGHS) algorithm. A structure suitable for data representation of ANNs has been adapted to the SGHS algorithm. The technique has been empirically tested and validated by training ANN on the six criteria classification problem and the real-world problem. In [6], a model based on the Artificial Bee Colony (ABC) algorithm was produced to estimate the safety factors of the retaining walls. The weight and bias values of the ANN have been optimized with the ABC algorithm in order to obtain a higher level of accuracy and performance estimation in safety factors. It has been determined that the network performance is strengthened with the produced model. Tang and Fong [7] used the Dynamic Group Optimization (DGO) algorithm, which is semi-transverse and semi-evolutionary, to train a feed-forward neural network. They named the model FSADGO. In the training neural network, the DGO plays an optimization role. It optimizes the parameters and the structure of the feed-forward neural networks. It tested with two real-world problems and compared with other training algorithms. Zhang and Suganthan [8] discussed the randomized training of ANN, which attracted great attention from researchers in areas such as machine learning, statistics, and computer vision. Ojha and Abraham [9] investigated the optimization of weights of artificial neural networks, network architecture, activation nodes, learning parameters, learning environment, etc. Hacibeyoglu and Ibrahim [10] proposed a new particle swarm optimization to train multi-layered, feed-forward neural networks. It was observed that the proposed algorithm searches the solution space more than once and gives better results than particle swarm optimization by testing on ten data sets. Recently, Gülcü [11] has developed the SMS-MLP algorithm to train ANN using the States of Matter Search algorithm. He used five classification problems in the experiments. The SMS-MLP algorithm was compared with the six algorithms. The experimental results showed that the SMS-MLP algorithm is more efficient than six algorithms.

In this study, the Butterfly Optimization Algorithm is employed to train ANN for the first time. The BOA based on the swarm intelligence was developed by Arora and Singh [12]. BOA is a meta-heuristic algorithm. It is an algorithm that is made by modeling the foraging behavior of butterflies. They also use it when migrating, escaping from a predator, or laying eggs somewhere. In this study, we aimed to optimize the bias and weight values. In the experimental study part, the BOA-MLP algorithm was run on the most important five data sets in the literature and compared with three different algorithms. As a result of the comparison, the success of the BOA-MLP algorithm has been demonstrated. The main reason for the success of the BOA-MLP algorithm is that the BOA-MLP algorithm is able to escape the local optimum.

2. Material and method

2.1. Artificial Neural Network

Artificial neural networks are an information processing technique developed by modeling the operation of the nervous system of the human brain. In other words, it is the digital platform of the synaptic connections established by the neurons in the human brain. Because of the success of ANN, ANN has been used in many different areas such as system modelling [13], prediction, pattern recognition, and optimization [14, 15].

The basic building blocks of the human brain and ANN are nerve cells. The human nerve cell consists of the core, body and two extensions. The shorter of these extensions are called dendrites and dendrites are branched into thousands of branches. Its task is to get login information. The single and long extension is called the axon and its task is to carry the output information to other nerve cells [16]. Artificial nerve cell has been developed by imitating the human nerve cell. The inputs entering the nerve cell are multiplied by the respective link weights to obtain the neuron's net input and then combined with the coupling function. The result is processed by the activation function, and thus the net output of the neuron is obtained. Fig. 1 shows an artificial nerve cell.

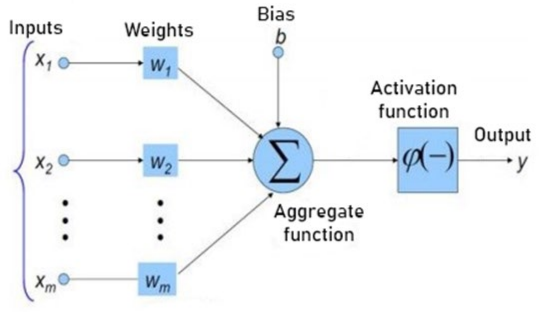

Figure 1. *Artificial Nerve Cell [17].*

ANNs are formed by the merging and grouping of artificial nerve cells. This merging consists of layers and as a result, ANN comes together from more than one interconnected layer. ANN consists of three layers as input layer, hidden layer and output layer. However, in some cases, the number of hidden layers may be more than one. In many ANN models, things go in order. That is, the hidden layer receives data from the previous input layer, processes the data and directs the results to the output layer. According to the type of ANNs, ANNS are divided into two groups as a feed-forward neural network and a feedback neural network. The feed-forward neural network is the network structure where data processing goes from input to output. A feedback neural network is the network structure where data processing can flow not only from input to output but also from output to input (backward).

The multi-layer perceptron is a feed-forward network structure with one or more hidden layers between the input layer and the output layer. In Fig. 2, the structure of the MLP with a single hidden layer is shown.

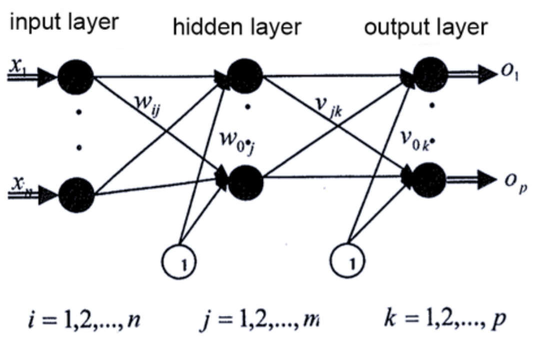

Figure 2. *MLP with a single hidden layer [18].*

An MLP consists of the following components: Artificial neural cell and layers, combine function, activation function, error function, learning algorithm.

The artificial nerve cell, consisting of inputs, concatenation and activation functions, was developed by imitation of the human nervous system. In an artificial nerve cell, input values (weights) are multiplied by node weights and sent to the merge function. The result returned from the merge function is sent to the activation function, so that the net output of the artificial nerve cell is obtained. Sigmoid, hyperbolic, ramp, identity and step functions are the most important activation functions. However, the most used one among these functions is the sigmoid function. The sigmoid function is shown in Equation (1).

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

The error function, which is one of the components of the MLP, is the objective function that finds the error in the ANN system. The most used error function is the Mean Square Error (MSE) whose formulation is given in Equation (6). Finally, the learning algorithm is the last component of the MLP. The learning algorithm mostly affects the performance of the ANN.

2.2. *Butterfly Optimization Algorithm (BOA)*

The Butterfly Optimization Algorithm (BOA) has been developed to solve challenging optimization problems. The main strategy of the BOA is to find nectar using the sense of smell. This strategy is generally as follows: Butterflies use their sense of smell to obtain the position of nectar, smell and analyze the air. The BOA algorithm takes this behavior as a role model to find the optimum value in the search space for a predefined problem. Considering scientific research, it has been observed that butterflies have a superior ability to reach the source of the scent. Butterflies are the BOA algorithm's search agents for optimization. A butterfly will produce an intensity of scent associated with its suitability.

The BOA algorithm has been developed considering the following three vital strategies:

1) All butterflies are expected to produce a scent that causes the butterflies to turn towards each other.
2) Each butterfly is expected to act randomly or move towards the superior butterfly emitting more scent.
3) The scent of a butterfly is determined or influenced by the value of its objective function.

We can explain these three stages in the BOA algorithm which is made by taking the behavior of butterflies as a role model.

1) Initialization Phase: Parameters are determined, an initial population is generated for the algorithm. When creating the starting population, the location of the butterflies is randomly assigned by calculating their odor values.
2) Iteration Stage: This stage is the part where the main processes are carried out and each butterfly tries to achieve the best result with parameters specific to the BOA algorithm. All butterflies in the search area are moved to new locations and new fitness values are calculated at each iteration.
3) Last stage: It is the part where the stopping criterion is fulfilled and the optimum result or the closest result is reached.

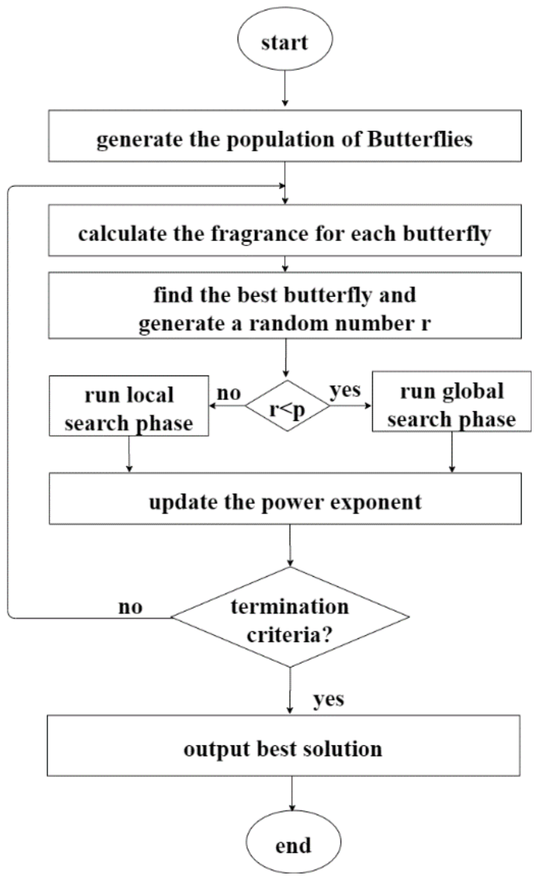
start
generate the population of Butterflies
calculate the fragrance for each butterfly
find the best butterfly and generate a random number r
run local search phase
no
r<p
yes
run global search phase
update the power exponent
no
termination criteria?
yes
output best solution
end

Figure 3. *The flowchart of the BOA algorithm.*

The BOA modality shown in Equation (2) is based on three basic concepts: sensory method (*c*), stimulus intensity (*I*) and power exponent (*a*). The sensory method refers to the raw input used by receptors to measure sensory energy form and process it in similar ways. The parameter *I*, which is the stimulus intensity, is limited to an exponential value. This is because, in previous studies by scientists, it was seen that as the stimulus gets stronger, the insects intensely go to the stimulus and this situation eventually becomes less sensitive. The parameter *a* is involved in correcting this situation. The parameter *a* is the power base depend on modality (smell in BOA). If *a* is equal to 1, it means there is no odor absorption. In other words, the rate of smell produced by a specified butterfly is perceived by other butterflies with the same capacity. This brings the butterfly closer to a single solution, usually the optimum. If *a* is equal to 0, it means that the scent emitted by the butterfly cannot be perceived by others. This allows searching locally.

$$f = c \times I^a \tag{2}$$

where *a* and *c* are random numbers in [0, 1], *f* is the perceived magnitude of the scent, and *I* is the stimulus intensity.

There are two important stages in the algorithm: local search and global search. The global search is shown in Equation (3).

$$X_i^{t+1} = X_i^t + (r^2 \times g^* - X_i^t) \times f_i \tag{3}$$

where g^* represents the best available solution among all the solutions in the current iteration. X_i^t and X_i^{t+1} are the positions of the butterfly *i* in the iterations *t* and *t+1* respectively. *r* provides a randomness that takes value in the range [0, 1] and f_i represents the perceived smell of the butterfly.

Local search is shown in Equation (4). Here, unlike the global, the X_i^t butterfly does not go towards the global best (*g*), it exhibits a random walk in the search space.

$$X_i^{t+1} = X_i^t + \left(r^2 \times X_j^t - X_k^t\right) \times f_i \tag{4}$$

where X_j^t and X_k^t are the positions of the butterflies *i* and *j* which are the randomly selected neighbors of the butterfly *i*. *r* provides a randomness that takes value in the range [0, 1] and f_i represents the perceived smell of the butterfly.

Searching for food can happen both locally and globally. Thus, a switching probability *p* is used in the BOA algorithm to switch between the global search and the local search. The key value is the probability value that decides whether the butterflies will show the best movement or a random walk. The flowchart of the BOA algorithm is presented in Fig. 3.

2.3. *Training MLP using BOA Algorithm (BOA-MLP)*

ANN learns from inputs and outputs. For this, the values of the weights and biases in ANN are updated according to inputs and outputs [11]. The ability of ANN to give correct results and to make successful classifications is provided by updating the bias and weight values in the most appropriate way. In the literature, researchers have proposed algorithms to train Multi-Layer Perceptron (MLP). However, classical techniques often face problems in solving this optimization problem. They tend to need large amounts of computing time, large amounts of memory. More importantly, they get stuck within the local optimum and produce poor-quality solutions. To overcome these difficulties, meta-heuristic algorithms have been used to train MLP [1]. In this study, the hybrid BOA-MLP algorithm is proposed to optimize the values of the weights and biases. The BOA algorithm which is a meta-heuristic algorithm was employed to train MLP for the first time. The process of the optimization of the weights and biases is shown in Fig. 4.

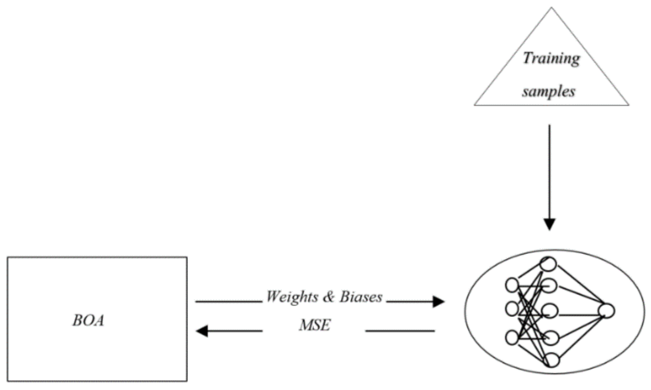

Figure 4. *The process of the optimization of the weights and biases in BOA-MLP.*

The first and most important element in training multilayer perceptron using meta-heuristic algorithms is the problem representation [19]. Meta-heuristic algorithms need a structure that shows the values of the weights and biases to solve the MLP training problem. In order to optimize the values of the weights and biases in the MLP, they must first be represented in the BOA-MLP algorithm. For this purpose, a representation vector consisting of weights and biases is used. This vector is shown in Equation (5) according to the MLP structure in Fig. 2.

$$Vector = \left(w_{i,j} \sqcup w_{j,k} \sqcup v_{0,j} \sqcup v_{0,k}\right) \tag{5}$$

where $w_{i,j}$ represents the values of weights between the input layer and the hidden layer, $w_{j,k}$ represents the values of weights between the hidden layer and the output layer. $v_{0,j}$ represents the values of biases between the input layer and hidden layer and $v_{0,k}$ represents the values of biases between the hidden layer and the output layer. The notation $\sqcup$ represents the concatenation of two sets.

At the beginning of the BOA-MLP algorithm, the population of butterflies is randomly generated. Each butterfly represents a different MLP, namely the representation vector in Equation (5). Then, the fragrance for each butterfly is calculated using the training dataset and the best butterfly in the population is found. Then the position (the values of the weights and biases in MLP) of each butterfly is updated using Equations (3) and (4). The power exponent is updated. This process continues until the termination criteria are met.

The purpose of training an MLP is to achieve the highest classification or prediction accuracy for both training and test samples. For this, Mean Square Error (MSE) is a common evaluation criterion [4]. MSE is also used as the objective function in the BOA-MLP algorithm. The formulation of MSE is given Equation (6).

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{p} e_k^{\,2} \tag{6}$$

where *N* and *p* are the number of samples in the dataset and the number of neurons in the output layer, respectively. e_k is the error of the *k*th neuron in the output layer.

3. Experimental results

To test the success of the BOA-MLP algorithm, five classification datasets with different levels of difficulty were selected: balloon, xor, iris, heart, and breast cancer. The datasets used were taken from the UCI resource pool which is used for machine learning and is available to everyone.

Table 1 shows the characteristics of the data sets. When we examine Table 1, it is seen that it is easy to solve the xor problem with 8 training and 8 test examples, 3 features and 2 classes. When we examine the balloon dataset, the difficulty is more than the difficulty of the xor dataset and less than the difficulty of the other datasets because the balloon dataset has 4 features, 20 training and 20 test samples and 2 classes. When we look at the iris dataset, it has 4 features, 150 training and 150 test samples and 3 classes. In the breast cancer data set, there are 2 classes and 9 characteristics. 599 samples are used in the training phase and 100 samples are used in the test phase. Finally, there are 22 features, 2 classes, 80 training and 187 test samples in the heart dataset.

This study does not focus on finding the optimal number of neurons in the hidden layer. In the literature, the number of neurons in the hidden layer is usually calculated by the formula $(2 \times n) + 1$ where *n* is the number of neurons in the input layer [4, 20]. Therefore in this study, the number of neurons in the hidden layer was calculated by the formula $(2 \times n) + 1$. Table 1 also shows the MLP structure and the dimension of a butterfly for each data set.

Table 1. *Properties of the classification datasets*

	Attribute number	Training	Test	Class number	MLP architecture	Vector size
Xor	3	8	8	2	3-7-1	36
Balloon	4	20	20	2	4-9-1	55
Iris	4	150	150	3	4-9-3	75
Breast Cancer	9	599	100	2	9-19-1	210
Heart	22	80	187	2	22-45-1	1081

The BOA-MLP algorithm was compared with the BAT-MLP algorithm based on the BAT algorithm [21], the SMS-MLP [11] algorithm and the Back Propagation (BP) algorithm. To fairly compare the algorithms, the experiments were run under the same situations for each algorithm. The initial values of the weights and biases were randomly selected between -10 and 10. The population size was 50 for the xor and balloon datasets, 200 for the other datasets. The maximum number of function evaluations was the same for each algorithm, 12500 for the xor and balloon datasets, 50000 for the other datasets. Each algorithm was run 30 times and the performances of the algorithms were evaluated by classification accuracy, sensitivity, specificity, precision, F1-score, the average (ave) and standard deviation (std) of the MSE results. The average (ave) and standard deviation (std) of the MSE results on training data are presented in Table 2 as ave±std. It is seen that the BOA-MLP algorithm gives better results than the other algorithms for all the datasets.

Table 3 presents the average classification accuracy on test data and average runtime of the algorithms. The BOA-MLP algorithm has better classification accuracy for all the datasets than the other algorithms. The BOA-MLP algorithm achieved 100% success on the xor and balloon datasets, 97.18% on the iris dataset, 99.37% on the breast cancer dataset and 74.46% on the heart dataset. Besides, the BOA-MLP algorithm is the slowest algorithm and the BP algorithm is the fastest algorithm. Table 4 presents the results of the sensitivity, specificity, precision and F1-score. According to Table 4, the performance of the BOA-MLP algorithm is very good for all the datasets.

Table 2. *The average and standard deviation of the MSE results on training data*

Dataset	BOA-MLP	BAT-MLP	SMS-MLP [11]	BP
Xor	**7.83E-03±9.62E-03**	1.27E-01±6.22E-02	1.33E-01±3.39E-02	1.41E-01±1.65E-01
Balloon	**2.79E-09±6.90E-09**	1.07E-02±2.83E-02	1.11E-02±1.31E-02	1.00E-01±1.51E-01
Iris	**4.74E-02±8.15E-03**	2.10E-01±1.38E-01	2.58E-01±5.12E-02	5.38E-02±1.30E-01
Breast Cancer	**1.78E-03±8.25E-05**	6.83E-03±7.92E-03	2.27E-02±4.33E-03	2.88E-02±1.21E-01
Heart	**1.15E-01±4.75E-03**	1.51E-01±3.21E-02	1.18E-01±3.13E-02	2.98E-01±1.32E-01

Table 3. *The average classification accuracy on test data and the average run time of the algorithms*

Dataset	Classification Accuracy (%)				Time (s)			
	BOA-MLP	BAT-MLP	SMS-MLP	BP	BOA-MLP	BAT-MLP	SMS-MLP	BP
Xor	**100.00**	85.00	83.75	84.17	9.2	4.7	5.5	**1.6**
Balloon	**100.00**	98.83	99.17	90.00	37.0	19.4	20.2	**1.7**
Iris	**97.18**	82.91	86.78	90.82	1408.6	775.2	881.6	**9.6**
Breast Cancer	**99.37**	96.13	85.67	93.03	5101.9	2607.3	2830.3	**29.5**
Heart	**74.46**	71.35	68.57	61.66	1479.0	937.0	892.4	**16.5**

Table 4. *The results of the sensitivity, specificity, precision and F_1-score*

Algorithm	Xor				Balloon			
	Sensitivity	Specificity	Precision	F_1-Score	Sensitivity	Specificity	Precision	F_1-Score
BOA-MLP	**100.00%**	**100.00%**	**100.00%**	**1.0000**	**100.00%**	**100.00%**	**100.00%**	**1.0000**
BAT-MLP	82.50%	87.50%	89.44%	0.8435	99.17%	98.61%	98.30%	0.9859
SMS-MLP	81.70%	86.70%	88.90%	0.8349	99.20%	99.20%	98.80%	0.9894
BP	84.17%	84.17%	85.94%	0.8221	75.00%	100.00%	86.67%	0.7838
	Iris				Breast Cancer			
	Sensitivity	Specificity	Precision	F_1-Score	Sensitivity	Specificity	Precision	F_1-Score
BOA-MLP	**97.38%**	**98.69%**	**97.42%**	**0.9738**	**99.05%**	**99.45%**	**98.02%**	**0.9850**
BAT-MLP	82.91%	91.46%	78.07%	0.7883	87.46%	98.44%	93.29%	0.8922
SMS-MLP	82.10%	91.10%	81.00%	0.8002	50.20%	95.10%	83.70%	0.5828
BP	90.82%	95.41%	88.43%	0.8899	85.87%	94.94%	81.16%	0.8255
	Heart							
	Sensitivity	Specificity	Precision	F_1-Score				
BOA-MLP	**74.71%**	71.56%	**96.79%**	**0.8422**				
BAT-MLP	71.32%	**71.78%**	96.73%	0.8180				
SMS-MLP	68.39%	70.67%	96.41%	0.7980				
BP	62.00%	57.78%	88.48%	0.6965				

Fig. 5 shows the boxplot charts of the BOA-MLP, BAT-MLP, SMS-MLP and BP algorithms on test data. It is a graphical representation of the distribution of results obtained by running the BOA-MLP, BAT-MLP, SMS-MLP and BP algorithms on the xor, balloon, iris, breast cancer and heart datasets. Fig. 6 shows the convergence graphs of the algorithms. It is seen that the BOA-MLP algorithm has a better start in many datasets and converges faster than the other algorithms. According to Fig. 5 and Fig. 6, the BOA-MLP algorithm exhibits good performance.

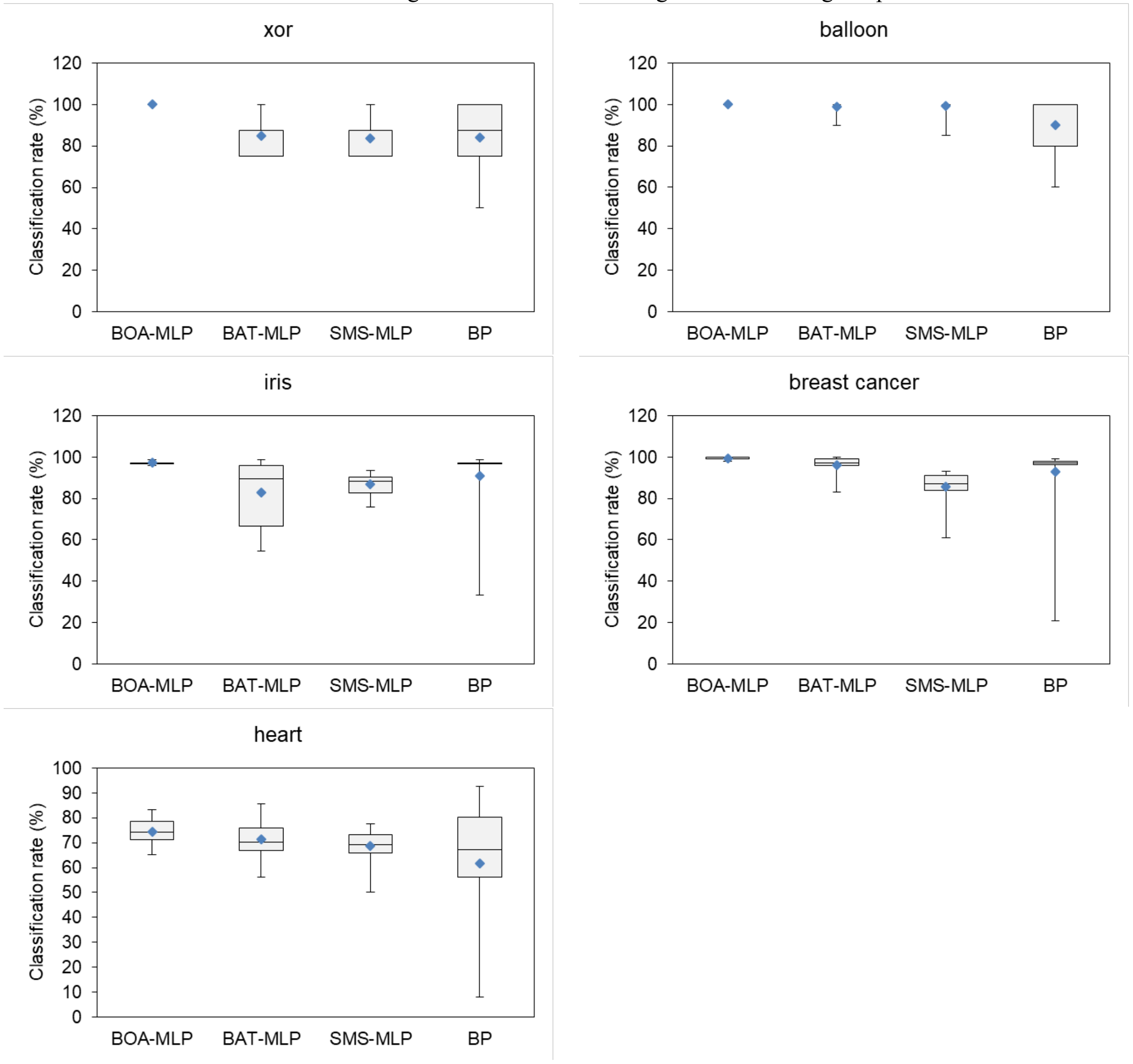

Figure 5. *The boxplot charts of the classification rate results on test data.*

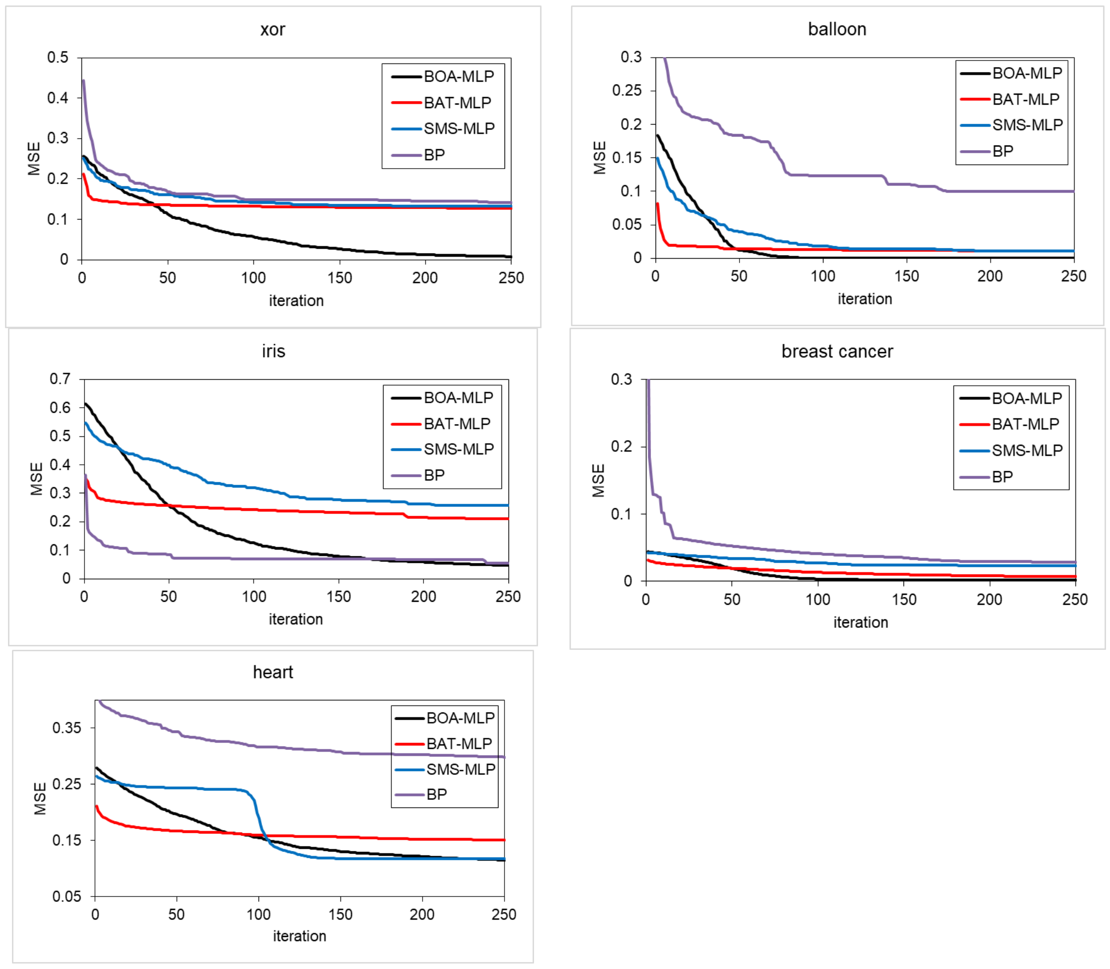

Figure 6. *The convergence graphs of the algorithms*.

4. CONCLUSION

Artificial neural networks (ANNs) are an information processing technique developed by modeling the operation of the nervous system of the human brain. In other words, it is the digital platform of the synaptic connections established by the neurons in the human brain. ANNs are formed by the merging and grouping of artificial nerve cells. This merging consists of layers and as a result, ANN comes together from more than one interconnected layer. ANN consists of three layers as input layer, hidden layer and output layer. However, in some cases, the number of hidden layers may be more than one. The multi-layer perceptron (MLP) is a feed-forward network structure with one or more hidden layers between the input layer and the output layer. MLP learns from inputs and outputs. The values of the weights and biases in MLP are updated according to inputs and outputs. Researchers have proposed algorithms to train MLP. However, classical techniques often face problems in solving this optimization problem. They tend to need large amounts of computing time, large amounts of memory. More importantly, they get stuck within the local optimum and produce poor-quality solutions. To overcome these difficulties, meta-heuristic algorithms have been used to train MLP. In this study, the Butterfly Optimization Algorithm (BOA) algorithm is used for the first time as the MLP trainer. Butterflies are the BOA algorithm's search agents for optimization. The greatest motivation in using the BOA algorithm is the superiority of its search strategy and the success of the escape ability from the local optima. The BOA algorithm is used to find the optimal values of weights and biases in MLP.

In the experiments, the proposed BOA-MLP algorithm was run on five classification data sets in the literature: iris, breast cancer, heart, xor and balloon. The datasets used were taken from the UCI resource pool used for machine learning available to everyone. The BOA-MLP algorithm was compared with the BAT-MLP algorithm, the SMS-MLP algorithm and the Back Propagation algorithm. To fairly compare the algorithms, the experiments were run under the same situations for each algorithm. Each algorithm was run 30 times and the performances of the algorithms were evaluated by classification accuracy, sensitivity, specificity, precision, F1-score, the average and standard deviation of the MSE results. The BOA-MLP algorithm has better classification accuracy for all the datasets than the other algorithms. The BOA-MLP algorithm achieved 100% success on the xor and balloon datasets, 97.18% on the iris dataset, 99.37% on the breast cancer dataset and 74.46% on the heart dataset. According to the results of the sensitivity, specificity, precision and F1-score, the performance of the BOA-MLP algorithm is very good for all the data sets. Overall, the BOA-MLP algorithm gives better results than the other algorithms for all the datasets.

References

[1]. Jaddi N.S., Abdullah S., "Optimization of neural network using kidney-inspired algorithm with control of filtration rate and chaotic map for real-world rainfall forecasting." Engineering Applications of Artificial Intelligence, 67, (2018), 246-259.

[2]. Türkoğlu B., "Artificial algae algorithm on training artificial neural networks." 2019, Selcuk University Natural Science Institute.

[3]. Haupt R.L., Ellen Haupt S., Practical genetic algorithms. 2004.

[4]. Mirjalili S., "How effective is the Grey Wolf optimizer in training multi-layer perceptrons." Applied Intelligence, 43(1), (2015), 150-161.

[5]. Kulluk S., Ozbakir L., Baykasoglu A., "Training neural networks with harmony search algorithms for classification problems." Engineering Applications of Artificial Intelligence, 25(1), (2012), 11-19.

[6]. Ghaleini E.N., et al., "A combination of artificial bee colony and neural network for approximating the safety factor of retaining walls." Engineering with Computers, 35(2), (2019), 647-658.

[7]. Tang R., Fong S., Deb Ss, Vasilakos A.V., Millham R.C., "Dynamic group optimisation algorithm for training feed-forward neural networks." Neurocomputing, 314, (2018), 1-19.

[8]. Zhang L., Suganthan P.N., "A survey of randomized algorithms for training neural networks." Information Sciences, 364, (2016), 146-155.

[9]. Ojha V.K., Abraham A., Snášel V., "Metaheuristic design of feedforward neural networks: A review of two decades of research." Engineering Applications of Artificial Intelligence, 60, (2017), 97-116.

[10]. Hacibeyoglu M., Ibrahim M.H., "A novel multimean particle swarm optimization algorithm for nonlinear continuous optimization: application to feed-forward neural network training." Scientific Programming. 2018.

[11]. Gulcu Ş., "Training of the Artificial Neural Networks using States of Matter Search Algorithm." International Journal of Intelligent Systems and Applications in Engineering, 8(3), (2020), 131-136.

[12]. Arora S.,. Singh S, "Butterfly optimization algorithm: a novel approach for global optimization." Soft Computing, 23(3), (2019), 715-734.

[13]. Tümer A., Edebali S., Gülcü Ş., "Modeling of Removal of Chromium (VI) from Aqueous Solutions Using Artificial Neural Network." Iranian Journal of Chemistry and Chemical Engineering (IJCCE), 39(1), (2020), 163-175.

[14]. Madenci E., Gülcü Ş., "Optimization of flexure stiffness of FGM beams via artificial neural networks by mixed FEM." Structural Engineering and Mechanics, 75(5), (2020), 633-642.

[15]. Karaşahin A.T., Tümer A.E., "Real time traffic signal timing approach based on artificial neural network." MANAS Journal of Engineering, 8(1), (2020), 49-54.

[16]. Pandya A., Macy (1996) "Pattern Recognition with Neural Network in C++." CRC Press, Florida.

[17]. Keskenler M.F., Keskenler E.F., "From Past to Present Artificial Neural Networks and History" Takvim-i Vekayi, 5(2), (2017), 8-18.

[18]. Hamzaçebi C., "Yapay sinir ağları: tahmin amaçlı kullanımı MATLAB ve Neurosolutions uygulamalı." 2011: Ekin Basım Yayın Dağıtım.

[19]. Belew R.K., McInerney J., Schraudolph N.N., "Evolving Networks: Using the Genetic Algorithm." 1990.

[20]. Aljarah I., Faris H., Mirjalili S., "Optimizing connection weights in neural networks using the whale optimization algorithm." Soft Computing, 22(1), (2018), 1-15.

[21]. Yang X.-S., "A new metaheuristic bat-inspired algorithm," Nature inspired cooperative strategies for optimization (NICSO 2010)., Springer, (2010), 65-74.