Evolving Systems (2022) 13:563–575
https://doi.org/10.1007/s12530-021-09401-5

ORIGINAL PAPER

Battle royale optimizer for training multi-layer perceptron

Saeid Agahian[1] · Taymaz Akan[2,3]

Received: 29 January 2021 / Accepted: 11 August 2021 / Published online: 21 August 2021

Abstract
Artificial neural network (ANN) is one of the most successful tools in machine learning. The success of ANN mostly depends on its architecture and learning procedure. Multi-layer perceptron (MLP) is a popular form of ANN. Moreover, backpropagation is a well-known gradient-based approach for training MLP. Gradient-based search approaches have a low convergence rate; therefore, they may get stuck in local minima, which may lead to performance degradation. Training the MLP is accomplished based on minimizing the total network error, which can be considered as an optimization problem. Stochastic optimization algorithms are proven to be effective when dealing with such problems. Battle royale optimization (BRO) is a recently proposed population-based metaheuristic algorithm which can be applied to single-objective optimization over continuous problem spaces. The proposed method has been compared with backpropagation (Generalized learning delta rule) and six well-known optimization algorithms on ten classification benchmark datasets. Experiments confirm that, according to error rate, accuracy, and convergence, the proposed approach yields promising results and outperforms its competitors.

Keywords Feed-forward neural network · Neural network training · Multilayer perceptron · Battle royale optimization · Metaheuristic

1 Introduction

The ultimate goal of human beings is to build machines with the ability to think, learn, and behave like humans. Therefore, understanding the structure and function of the human brain is necessary for brain-like processing (Zurada 1992; Haykin 2007). The biological nervous system was first mathematically modeled by McCulloch and Pitts in 1943 (McCulloch and Pitts 1943; Ojha et al. 2017). It is the groundwork for simulating the behavior of the neural system, which causes the emergence of Artificial Neural Networks (ANNs) and associated learning algorithms. With regard to adaptability, generalization, real-time procedure, and self-organizing capability, ANNs can be supposed as the most successful approach among other machine learning schemes in classification, clustering, pattern recognition, and regression problems (Karaboga et al. 2007; Schmidhuber 2015; Chatterjee et al. 2017; Braik et al. 2008; Linggard et al. 2012). A simple schema of McCulloch and Pitts' model is illustrated in Fig. 1.

The success of machine learning algorithms mostly depends on the learning procedure. Generally learning algorithms are divided into two main categories: supervised and unsupervised approaches. The mathematical model of neural neurons, introduced by McCulloch and Pitts, suffers from the lack of a learning strategy that tunes the neuron weights for solving a particular problem. In order to fill this gap, Hebb proposed an unsupervised rule-based learning strategy for tuning the connection weights of the network, in 1949 (Hebb 1949). Later in 1958, Rosenblatt proposed a supervised learning mechanism called perceptron with the capability of linear classification. In this mechanism, the weights of the network are updated according to the input data when errors occur. This network is the simplest case which is so-called single-layer feed-forward neural network (FFNN); however, this model is not able to solve non-linear classification problems. Consequently, developing the network and associated learning method to deal with this challenge is thus needed.

✉ Taymaz Akan
taymaz.farshi@gmail.com

1 Department of Computer Engineering, Erzurum Technical University, Erzrum 25050, Turkey

2 Faculty of Electrical Engineering and Informatics, University of Pardubice, Pardubice, Czech Republic

3 Software Engineering Department, Ayvansaray University, Istanbul 34020, Turkey

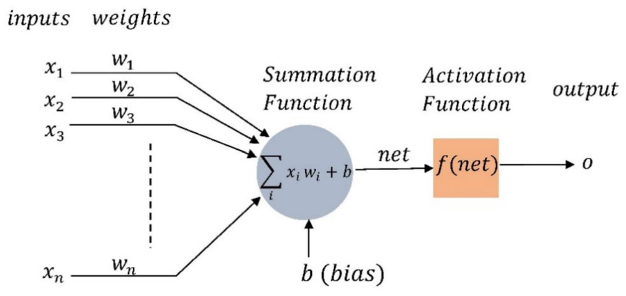


Fig. 1 McCulloch-Pitts neural model

Researchers believe that increasing the number of hidden layers overcome this problem; however, it requires a training algorithm to train the multi-layer perceptron neural network (MLP). In 1984, Werbos, as the first researcher, proposed a back-propagation (BP) algorithm for training this network (Werbos 1989). Nonetheless, this was not popular until Rumelhart (McClelland et al. 1986) brought it up again. BP tunes the weights of the network layer by layer if the output of the network is not desired. As learning algorithms directly impact the performance of neural networks, a wide variety of studies have been conducted to effectively train the procedure. Supervised training methods are divided into two main categories: gradient-based and stochastic methods. As mentioned already, BP adjusts the weights with respect to distance from the minimum point of the lost function; however, because of the low convergence rate, it may get stuck in local minima. This problem arises from the fact that movement follows the gradient descent. On the other hand, stochastic algorithms are proven to be effective in dealing with such problems.

A wide variety of stochastic optimization techniques, including heuristic and metaheuristic algorithms, have been applied to tackle this problem. Genetic Algorithm (GA) (Holland 1992; Bhattacharjee and Pant 2019), Particle Swarm Optimization (PSO) (Eberhart and Kennedy 1995; R. K. 2020), Evolutionary Strategies (ES) (Schwefel 1984), Differential Evolution (DE) (Storn and Price 1997), Ant Colony Optimization (ACO) (Dorigo and Caro 1999), Artificial Bee Colony (ABC)(Karaboga and Basturk 2007), Grey Wolf Optimization (GWO) (Mirjalili et al. 2014a), whale optimization algorithm (Mirjalili and Lewis 2016), Biogeography-Based Optimization (BBO) (Simon 2008), Firefly Optimization Algorithm (FFA) (Yang 2009), Bat Algorithm (BA) (Failed 2010), Cuckoo Search (CS) (Yang and Deb 2009), Bird Mating Optimizer (BMO) (Askarzadeh 2014), etc. have been used for this purpose. A fair balance between exploitation and exploration lets such algorithms not only find an accurate solution but also escape from poor local optima.

Montana and Davis (1989) proposed a modified version of genetic algorithm to train FFNNs, where Zhang et al. (2007) applied a hybrid PSO to tuning the weights of FFNN;. Moreover, ES was utilized for optimizing the system error of FFNN in Wienholt (1993); DE was applied to optimize the learning mechanism of classifiers (Ilonen et al. 2003; Failed 2008); and finally ACO was extended to the continuous form for training the neural networks weights (Blum and Socha 2005). In 2007, Karaboga et al. (2007) used ABC to train the neural networks; later, the authors improved their work by hybridized ABC with Levenberq-Marquardt (Ozturk and Karaboga 2011). Mirjalili et al. proposed a GWO-based algorithm for training MLPs to reduce the system errors in FFNNs (Mirjalili 2015); the authors also suggested BBO and WOA algorithms to overcome the associated problem (Mirjalili et al. 2014b; Aljarah et al. 2018). Furthermore, FFA was utilized for FFNN in order to facilitate swift learning and decrease computational complexity (Nayak et al. 2016). Additionally, BA was utilized to optimize the connection weights and structure of the FFNNs for constructing a classifier. (Jaddi et al. 2015). An alternative optimization algorithm, namely CS, which has been applied to optimize the FFNNs training for classification problems (Nawi et al. 2013). Finally, for solving the classification problems, BMO was utilized to train the FFNNs (Askarzadeh and Rezazadeh 2013).

The no-free-lunch (NFL) theorem (Wolpert and Macready 1997) emphasizes that an optimization algorithm may not outperform other algorithms in solving all kinds of optimization problems. Therefore, new approaches are required to cope with new challenges. This issue motivated us to utilize a new optimization method (BRO) that is proved to perform as good as or sometimes superior to the existing approaches for training FFNNs. In this paper, for training the feed-forward ANNs, we used a recently proposed Battle Royale Optimization (BRO) algorithm, which is proven to be successful in solving continuous optimization problems (Rahkar Farshi 2020). This paper is organized as follows: Sect. 2 describes the concepts of MLP. Section 3 provides a brief summary of BRO. BRO-based training algorithm for MLP is described in Sect. 4. Section 5 includes experimental results and discussion. Finally, Sect. 6 presents the conclusions.

2 Multi-layer perceptron neural network

MLP is a specific model of FFNN which is able to solve the non-linear classification and regression problems. Neurons are considered as the processing elements in MLP. The neurons are distributed in parallel form over each layer, where the layers are stacked on each other. Moreover, the neurons of each layer are fully connected to all other neurons in the next layer (Fig. 2). The first and last layers are respectively named input and output. The layers between the input and output, are hidden layers. The first layer feeds input variables to the network (Zurada 1992; Haykin 2007). MLP arranges neurons in one-directional mode. The neurons of the first layer receive input from the outside while other layers' neurons receive the inputs from the output of previous layers' neurons.

The connections between the layers are associated with weights to control the effect of related input on neurons. Each neuron in MLP uses summation and activation functions consecutively to produce the output, as illustrated in Fig. 1. The summation function computes the weighted sum of the inputs, see Eq. 1. Subsequently, the activation function applies a threshold on the obtained weighted sum to generate the output of the neuron.

$$net_j = \sum_{i=1}^{n} w_{ji}.x_i + b_j, \tag{1}$$

where, n indicates the total number of inputs, x_i denotes the input variable i, b_j is a bias value of jth neuron, and w_{ji} implies the weight of the connection from the ith input to jth neuron. The S-shaped sigmoid function is the most popular one among several non-linear activation functions which is broadly used in the MLP. This function is mathematically calculated per:

$$f(z) = \frac{1}{(1 + e^{-z})} \tag{2}$$

By using this activation function over the weighted sum, the output of a neuron is calculated by:

$$o_j = f\left(net_j\right) \tag{3}$$

Each classification or regression problem requires a specific network structure to be constructed. Consequently, for tuning the network connection weights, the training algorithm is triggered. Updating the weighting vectors is carried out by minimizing the total error of the network (Karaboga et al. 2007). The total error is estimated as follows:

$$E(W(t)) = \frac{1}{l} \sum_{j=1}^{l} \sum_{k=1}^{k} \left(d_k - O_k\right)^2, \tag{4}$$

where, $E(W(t))$ indicates the total error value at the tth iteration where $W(t)$ is the weights of the network connections at the tth iteration; The desired and obtained output of kth node are represented by d_k and O_k, respectively; k and l respectively indicate the number of output nodes and train instances.

Training the MLP is accomplished according to minimizing the total network error, which can be considered as an optimization problem. BP is a variation on the gradient search approach that is commonly used for training a network; however, it suffers from the scaling problem, which may cause in rapid diminishing of the of BP performance in high-dimensional problems (Montana and Davis 1989). As complex spaces are multimodal, and they consists of several local minima around the global minimum, gradient-based search approaches may get stuck in local minima; therefore, performance degradation might be observed. Metaheuristic optimization techniques would be effective in dealing with such problems. The exploitation and exploration mechanism of optimization algorithms not only prevent them from

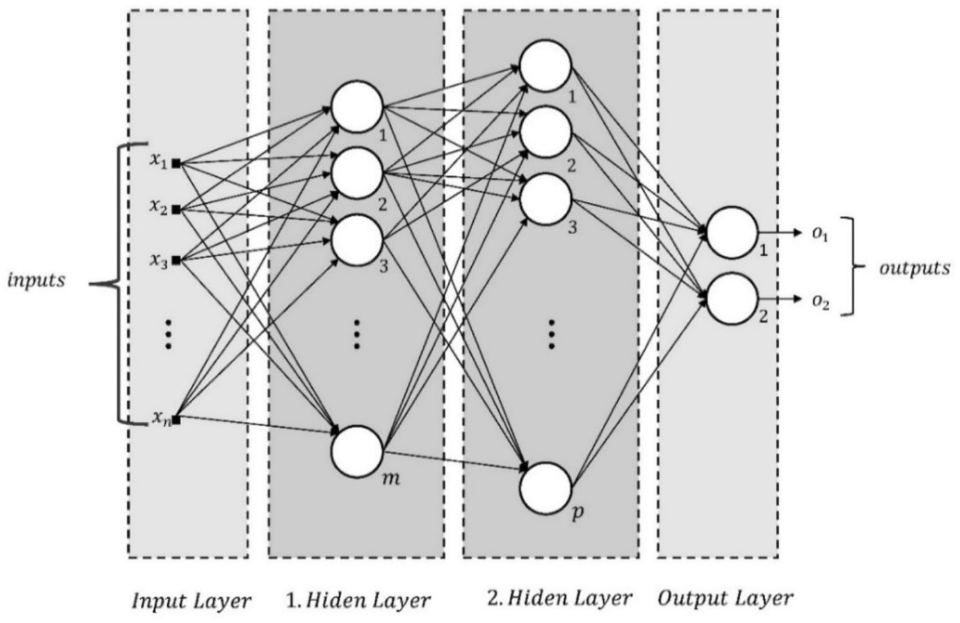


Fig. 2 Multilayer perceptron neural network

getting stuck in local optima but also provide a deep insight into the global optimum.

3 Battle royale optimizer

This section briefly overviews the BRO algorithm proposed by Rahkar-Farshi in 2020 (Rahkar Farshi 2020). The basic concept of this algorithm was inspired by the battle royale game, which is a multiplayer video game genre that blends survival. Player unknown battlegrounds (PUBG) is the well-known and most popular game among these kinds of games. Deathmatch is one of the well-played game modes of PUBG. In this mode, a player aims to kill as many other players as possible until a kill limit or a time limit is met. Generally, battles are fought on a specific battlefield map chosen by the players. Such an optimization problem space is considered as a game map in BRO. The game gets started by jumping the players from a plane onto the map. In this regard, the search agents of BRO are randomly initialized by uniform random initialization within the search space similar to many other population-based optimization algorithms. Throughout the game, a player may be killed by other rivals which causes him to be respawned at a random zoon of the battlefield. All players attempt to hurt other players by shooting. Players who take a more appropriate position, can damage their rivals. In this context, one may question who and how can shoot its opponent in the game? Actually, each search agent will be compared with the nearest agent according to the Euclidean distance. Players with a better cost function, therefore, cause damage to another one. The damage level of the player who gets damaged, will be increased by one and mathematically calculated per $x_i.damage = x_i.damage + 1$, here $x_i.damage$ is the damage level of the ith soldier among the population. Then, the player randomly move to be protected from further injury. This movement is performed by moving the player toward the best position found so far. This operation causes the exploitation which is mathematically modeled as follows:

$$x_{dam,d} = r_1 x_{dam,d} + r_2\left(x_{best,d} - x_{dam,d}\right), \tag{5}$$

where r_1, r_2 are two random variables, uniformly distributed over the interval [0,1]. However, r_1 is set to 1. $x_{dam,d}$ indicates the position of the damaged player in dimension d. Also, the level of damage for a damaged player will be reset to zero if it hurts another player. Furthermore, if a player gets damaged for a predetermined number of times (threshold value) in a row and the damaged level reaches the predefined threshold value, the player gets killed and would be respawned at a random space of the problem. In this case, the level of damage will reset to zero. Through trial and error, it is determined that the value of threshold = 3 was applicable for optimization benchmarks. This action prevents early convergence and provides good quality of exploration. Operations are mathematically calculated as:

$$x_{dam,d} = r\left(ub_d - lb_d\right) + lb_d \tag{6}$$

where lb_d and ub_d stand for the lower and upper bounds, respectively. Furthermore, in every Δ step of the iteration, the algorithm adaptively narrows the problem scale towards the best solution. The initial value for Δ is assumed as $\Delta = \frac{MaxCicle}{round(\log_{10}(MaxCicle))}$. Then, after each Δ step, Δ will be updated as $\Delta = \Delta + round(\Delta/2))$, if $i \ \geq \Delta$. This operation provides both exploration and exploitation. Consequently, the lb_d and ub_d will be updated as follows:

$$\begin{aligned} lb_d &= x_{best,d} - SD\left(\overline{x_d}\right) \\ ub_d &= x_{best,d} + SD\left(\overline{x_d}\right) \end{aligned} \tag{7}$$

where $SD\left(\overline{x_d}\right)$ indicates the standard deviation of all search agents' positions in dimension d and $x_{best,d}$ represents the position of the best-known solution. All these interactions will be repeated until the termination conditions are met.

4 BRO for training MLP

All kinds of non-linear optimization algorithms can be employed for tuning FFNN weights. BRO is a population-based approach that can be applied to single-objective optimization over continuous problem spaces. This section discusses how to apply BRO to train FFNN whit a single hidden layer. The intuition behind using a single hidden layer architecture is ease of understanding. BRO maximizes or minimizes a real objective function by systematically choosing values from a possible set for each input variable and calculating the value of the function. Integration of BRO whit MLP for generating a classifier is hinged on two important factors: encoding the search agents and constructing the objective function. In this work, each search agent (player) is encoded into a one-dimensional vector, which corresponds to network connection weights and neuron bias (see Fig. 3).

In order to evaluate the classification quality of each search agent, an objective function is required over the training set. To achieve this goal, a commonly used error function, Mean Square Error (MSE), has been used herein. The general form of MSE is mathematically calculated as follows:

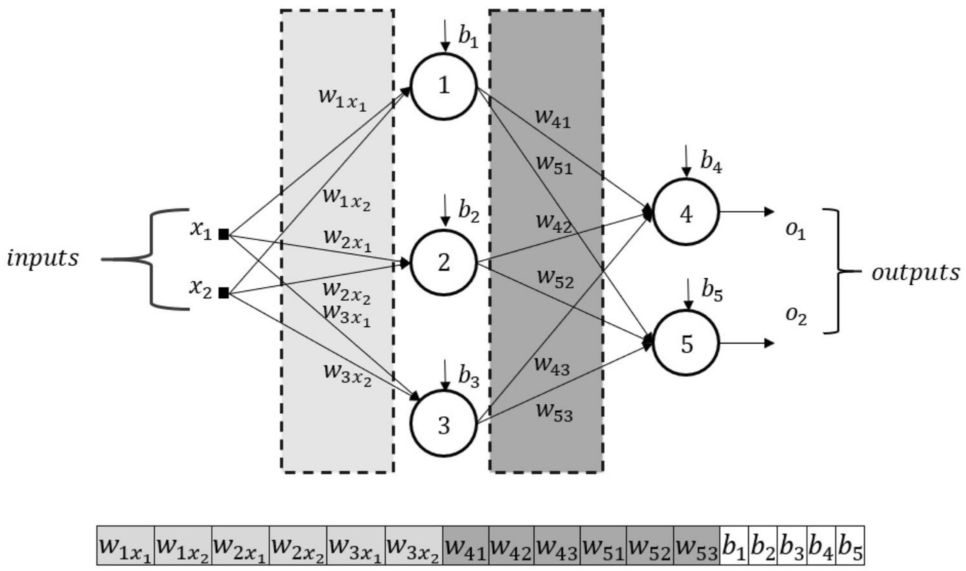


Fig. 3 Encoding MLP with a single hidden layer to a search agent vector

$$MSE = \frac{1}{U}\sum_{u=1}^{U}\left(y_u - y'_u\right)^2, \tag{8}$$

where y indicates the actual value and y' is the estimated value. U indicates the total number of instances in the training set. The number of the input variable (length of search agents' vector) is computed by Eq. (9):

$$(Ni+1)\times\left|h_i\right| + \sum_{i=2}^{Nhl}\left(\left|h_i\right|\times\left(\left|h_{i-1}\right|+1\right)\right) + \left(\left|h_{Nhl}\right|+1\right)\times No \tag{9}$$

where Ni is the number of input variables; Nhl indicates the number of hidden layers in the network; $\left|h_i\right|$ represents the number of neurons in the i^{th} hidden layer and No is the number of the output layer's neurons. For a network with one hidden layer Eq. 9 can be replaced by:

$$(Ni+1)\times\left|h_1\right| + \left(\left|h_1\right|+1\right)\times No \tag{10}$$

For instance, using Eq. (10) for the given sample in Fig. 3, the number of parameters to be tuned is calculated per $(2+1)\times 3+(3+1)\times 2 = 17$. It can be observed that we can readily estimate the vector size when $Nhl = 1$.

It should be noted that the parameter vector is commonly initialized by a uniform distribution between [− 1, 1]. However, this range can be extended to a wider range. On the other hand, in order to prevent the gradients to be vanished or exploded too quickly, the weights should be set neither much greater than 1 nor much smaller than − 1. (Ng et al. 2020).

5 Experimental results and performance evaluation

To assess the performance of the proposed approach, an extensive experimental evaluation and comparison have been carried out with backpropagation (Generalized learning delta rule) and six well-accepted population-based metaheuristic algorithms: Genetic Algorithm (GA) (Holland

Table 1 The initial values of the parameters of optimization algorithms

Algorithms	Parameter	Value
GA	Crossover probability	0.9
	Mutation probability	0.1
	Selection mechanism	Roulette wheel
	Crossover type	Whole Arithmetic Crossover
	Mutation type	Uniform mutation
PSO	Inertia weights range	[0.9, 0.6]
	acceleration coefficients	2.1 and 2.1
ABC	Limit	8
GSA	α	20
	Gravitational constant	100
	Velocity range	*rand*[0, 1]
WOA	α linearly decrease from	[2 to 0]
	r	*rand*[0, 1]
DE	Crossover probability	0.9
	Differential weight	0.5

Table 2 Characteristics of used benchmark datasets

No	Dataset	No. of Classes	No. of attributes	No. of training samples	No. of test samples
1	Iris	3	4	130	20
2	Breast Cancer	2	9	599	100
3	Heart	2	22	80	187
4	Diabetes	2	8	506	262
5	Vertebral	2	6	204	106
6	Parkinson's	2	22	128	67
7	Australian	2	14	455	235
8	tic_tac_toe	2	9	632	326
9	Ionosphere	2	33	231	120
10	Wine	3	13	117	61

Table 3 MLP architect for datasets

No	Dataset	No. of attributes	MLP structure	No. of variables
1	Iris	4	4–9-3	75
2	Cancer	9	9–19-1	210
3	Heart	22	22–45-1	1081
4	Diabetes	8	8–17-1	171
5	Vertebral	6	6–13-1	105
6	Parkinson's	22	22–45-1	1081
7	Australian	14	14–29-1	465
8	tic_tac_toe	9	9–19-1	210
9	Ionosphere	33	33–67-1	2346
10.1	Wine (one-hot)	13	13–27-3	462
10.2	Wine (int-coding)	13	13–27-1	406

1992), Particle Swarm Optimization (PSO) (Eberhart and Kennedy 1995), Artificial Bee Colony (ABC)(Karaboga and Basturk 2007), Gravitational Search Algorithm (GSA) (Rashedi et al. 2009), and Whale Optimization Algorithm (WOA) (Mirjalili and Lewis 2016) Differential Evolution (DE) (Price 1996). The control parameter of optimization algorithms may significantly affect their performance. The best control parameters for all algorithms were set based on trial and error or directly taken from Aljarah et al. (2018). These parameters are reported in Table 1. Both the number of iterations and population size for all algorithms are set to 200. Also, the method has been compared with backpropagation (Generalized learning delta rule), such that the training seek was taken as 200, η and λ were 0.1 and 1, respectively.

As the heuristic algorithms are stochastic optimization techniques, they can lead to different results. Therefore, all the empirical results are achieved by taking the average of 20 independent runs for each dataset. All algorithms were implemented in MATLAB and performed on a Core i7-7700 HQ 2.81 processor with 32 GB of RAM. In this study, ten real-world benchmark classification datasets, namely, Iris, Breast Cancer, Heart, Vertebral, Parkinson's, Australian, Tic_Tac_Toe, Ionosphere, and Wine datasets were used to validate the methods. Table 2 shows the number of classes, attributes, training samples, and test samples for each dataset. Table 3 reports the associated network architecture and the total number of the parameters for each dataset.

Additionally, in the case of scale invariance, the datasets which consist of floating point feature values, are normalized by min–max normalization method to the range [− 1, 1].

It should be noted that two different architectures of MLP classifier were used for the 10th (Wine) dataset. In the first one (10.1), one-hot encoding was used for labeling. Hence, the number of output layer's neurons equals the number of class.

Accuracy is a well-known performance criterion in order to evaluate the ability of a classifier. To assess the effect of training algorithm on MLP classifier, an accuracy performance measure was used in this paper. Accuracy is the number of correctly classified instances over the total number of instances. The numerical comparisons are conducted by taking the average of MSE and classification accuracy over 20 independent runs. Besides, in order to evaluate the stability of algorithms, standard deviation for both MSE and classification accuracy is calculated and reported in Tables 4 and 5. In these tables, "Best" indicates the best feasible solution among all runs of each algorithm.

After taking the average over the ranks of each algorithm, the highest averaged ranks are bolded in the last three rows of Table 4 and Table 5. As can be seen in Table 4, the BRO-based classifier ranked first in terms of classifier accuracy according to the mean and best values. In this table, ABC, DE, WOA, PSO, GA, BP, and GSA ranked second to eighth, respectively, according to the mean value. Also, ABC, WOA, DE, PSO, GA, GSA, and BP, correspondingly, with respect to best value. Moreover, DE performed the best in terms of the STD, whereas ABC, BRO, and BP ranked second to forth. PSO ranked fifth alongside GA. Consequently, WOA, and GSA ranked seventh to last, correspondingly. This table indicates that none of the metaheuristic algorithms could yield reasonable performance in the Heart dataset. This may arise because the number of training data is not sufficient, but too small compared to the number of test data.

Table 4 MLP classifier accuracy for each dataset

Dataset		BRO	GA	PSO	ABC	GSA	WOA	DE	BP
Iris	Mean	**0.9900**	0.79800	0.82800	0.95800	0.60200	0.80600	0.9	0.7
	Best	**1**	0.95000	0.95000	**1**	0.85000	**1**	0.95	0.75
	STD	**0.02500**	0.11409	0.08548	0.05139	0.15577	0.15700	0.0359	0.05389
Breast Cancer	Mean	0.95840	0.9836	0.9836	0.98280	0.97120	**0.98520**	0.9540	0.8535
	Best	**1**	**1**	**1**	**1**	**1**	**1**	0.96	0.97
	STD	0.01247	0.01113	0.01186	0.009797	0.01563	0.01045	**0.0060**	0.0929
Heart	Mean	0.76131	0.73190	0.7274	0.7570	0.71550	0.74242	0.75098	**0.9078**
	Best	0.80748	0.83422	0.8449	0.8021	0.87700	0.81818	0.80213	**0.9197**
	STD	0.0196	0.0413	0.0736	0.03225	0.10322	0.042551	0.0311	**0.00514**
Diabetes	Mean	**0.79221**	0.71145	0.76030	0.77664	0.69938	0.74305	0.7794	0.69160
	Best	**0.81679**	0.76717	0.79007	0.80534	0.74809	0.77862	0.7977	0.77862
	STD	0.01327	0.07046	0.03287	0.01739	0.02584	0.02107	**0.0077**	0.076602
Vertebral	Mean	0.83773	0.78603	0.78792	**0.84301**	0.73811	0.80226	0.8165	0.676415
	Best	0.86792	0.82075	0.84905	**0.87735**	0.82075	**0.87735**	0.8396	0.849056
	STD	**0.01441**	0.02491	0.04662	0.01676	0.04282	0.03932	0.0195	0.128556
Parkinson’s	Mean	**0.92000**	0.82746	0.86089	0.90268	0.81731	0.86268	0.8507	0.82089
	Best	**0.97014**	0.89552	0.92537	**0.97014**	0.86567	0.92537	0.8955	0.82089
	STD	0.02886	0.03109	0.03187	0.02243	0.01842	0.03187	0.0252	**0**
Australian	Mean	**0.88902**	0.84289	0.87455	0.88357	0.78944	0.86604	0.8857	0.78063
	Best	**0.90212**	0.87234	0.89361	0.89787	0.82978	0.89361	0.9106	0.84680
	STD	0.00893	0.01679	0.00968	**0.00841**	0.02938	0.01867	0.0141	0.05037
tic_tac_toe	Mean	0.67042	0.60552	0.58049	0.66687	0.59865	0.61631	**0.6923**	0.62883
	Best	**0.74539**	0.64723	0.64110	0.72699	0.68404	0.67484	0.7239	0.62883
	STD	0.02911	0.02750	0.05083	0.02640	0.45262	0.01951	0.0140	**0**
Ionosphere	Mean	0.94900	0.86533	0.91066	0.88700	0.77066	0.82533	**0.9629**	0.64166
	Best	0.97500	0.89166	95.0000	0.96666	0.83033	0.90833	**0.9833**	0.64166
	STD	0.01564	0.02120	0.02145	0.33857	0.27859	0.04661	0.0144	**0**
Wine (one-hot)	Mean	0.94491	0.79213	0.85901	0.92196	0.59737	0.71344	**0.99836**	0.89262
	Best	**1**	0.88524	0.96721	0.98360	0.78688	0.91803	**1**	0.95081
	STD	0.07687	0.05924	0.05910	0.03003	0.09733	0.13335	**0.0050**	0.060051
Wine (int. coding)	Mean	0.65770	0.57967	0.60655	0.62163	0.59540	0.62491	0.59590	**0.67704**
	Best	0.72131	0.63930	0.62295	0.65573	0.60655	0.70491	0.60655	**0.88524**
	STD	0.02473	0.03274	**0.00473**	0.01563	0.01028	0.02381	0.0080	0.013350
Ranks	Mean	**2.18181(1)**	5.36363(6)	4.72727(5)	3.09090(2)	7.18181(8)	4.09090(4)	3.36363(3)	5.90909(7)
	Best	**2.09090(1)**	5.27272(6)	3.81818(5)	2.81818(2)	6(7)	3.63636(3)	3.72727(4)	6.09090(8)
	STD	3.72727(3)	5.18181(5)	5.18181(5)	3.36363(2)	6.09090(8)	5.63636(7)	**2.27272(1)**	4.45454(4)

However, BP outperformed others in this dataset according to all performance criteria. As it is clear from this table BP perform best only in two cases out of eleven. In most cases, the classifier accuracy of BP is much lower than the classifier accuracy of metaheuristic-based methods. Also, in some cases, BP provides very different mean and best classifier accuracy values. For example, in Diabetes, the mean value is 0.69160, while the best value is 0.77862, and in Vertebral, although the mean value of classifier accuracy is 0.676415, the best try is 0.849056. Consequently, BP ranked worst in terms of STD due to unsatisfying results such as 0.076602 and 0.128556. All these observations prove that BP may stuck in local optima.

Apart from classifier accuracy, for each dataset, algorithms are also ranked according to the objective function. As Table 5 shows, the proposed method achieves the best

Table 5 Training results of MLP for each dataset (MSE)

Dataset		BRO	GA	PSO	ABC	GSA	WOA	DE
Iris	Mean	**0.04247**	0.31737	0.18236	0.04308	0.57529	0.22071	0.2430
	Best	**0.03188**	0.17366	0.05637	0.02900	0.41204	0.06960	0.235402
	STD	0.00797	0.06928	0.07437	0.00861	0.07892	0.12978	**0.004360**
Breast Cancer	Mean	**0.01587**	0.03370	0.03153	0.02116	0.05114	0.02425	0.0198
	Best	**0.01068**	0.02758	0.02662	0.01612	0.03873	0.01996	0.019338
	STD	0.00228	0.00346	0.00253	0.00162	0.00890	0.00266	**0.000291**
Heart	Mean	0.14622	0.18138	0.17476	0.16222	0.20443	0.16757	**0.09622**
	Best	0.13513	0.16895	0.15315	0.15214	0.18730	0.11820	**0.084854**
	STD	**0.00467**	0.00572	0.01163	0.00527	0.00820	0.02003	0.007392
Diabetes	Mean	**0.14496**	0.18623	0.21070	0.16505	0.24037	0.18633	0.1465
	Best	**0.13774**	0.16971	0.19960	0.15338	0.21083	0.15357	0.144447
	STD	0.00556	0.00793	0.00714	0.00551	0.01296	0.02124	**0.001210**
Vertebral	Mean	0.10549	0.14196	0.15236	**0.10418**	0.18080	0.13663	0.1363
	Best	**0.09001**	0.12761	0.12745	0.09476	0.13935	0.11026	0.133074
	STD	0.00719	0.00758	0.01020	0.00487	0.02029	0.01763	**0.001414**
Parkinson’s	Mean	**0.05191**	0.11360	0.09739	0.06438	0.14913	0.09934	0.0869
	Best	**0.02601**	0.07253	0.07031	0.03163	0.12362	0.07132	0.069535
	STD	0.01313	0.01431	0.01086	0.01372	0.01309	0.01722	**0.007465**
Australian	Mean	0.09383	0.11744	0.11565	0.10102	0.18149	0.11572	**0.0932**
	Best	0.08414	0.10432	0.10109	**0.07930**	0.14000	0.08361	0.089914
	STD	0.00594	0.00610	0.00507	0.00777	0.02408	0.01607	**0.001668**
tic_tac_toe	Mean	**0.20670**	0.23523	0.25197	0.21067	0.28330	0.22982	0.2089
	Best	**0.18652**	0.22008	0.23426	0.19375	0.24394	0.21557	0.207238
	STD	0.00860	0.00659	0.01525	0.00603	0.01713	0.00837	**0.000903**
Ionosphere	Mean	0.05030	0.11381	0.08387	0.10584	0.20787	0.16041	**0.0370**
	Best	0.02250	0.08276	0.03896	0.05468	0.16524	0.08968	**0.023807**
	STD	0.01494	0.01506	0.02378	0.02679	0.01924	0.03784	**0.006240**
Wine (one-hot)	Mean	0.09106	0.42134	0.30738	0.16894	0.73536	0.47471	**0.0546**
	Best	0.03911	0.29330	0.14980	0.05272	0.58589	0.24117	**0.034945**
	STD	0.06572	0.06854	0.07619	0.05091	0.08033	0.14012	**0.015217**
Wine (int. coding)	Mean	**0.31782**	0.32833	0.38554	0.32210	0.42724	0.35065	0.3622
	Best	0.28448	**0.26474**	0.37534	0.28803	0.38064	0.31382	0.360500
	STD	0.01358	0.02564	0.00500	0.01490	0.03904	0.01671	**0.001012**
Rank	Mean	**1.45454(1)**	5.18181(6)	4.72727(5)	2.63636(3)	7(7)	4.63636(4)	2.36363(2)
	Best	**1.63636(1)**	5.09090(6)	4.63636(5)	2.54545(2)	7(7)	3.90909(4)	3.27272(3)
	STD	2.90909(2)	4.36363(5)	4.18181(4)	3.18181(3)	5.90909(6)	6.18181(7)	**1.27272(1)**

results in six cases out of eleven by means of means fitness values and DE follows it with four superiority out of eleven cases. According to Mean and best values, the BRO-based MLP classifier ranks first overall, where, DE, ABC, WOA, PSO, GA, and GSA ranked second to eighth, respectively, according to the mean values. Also, ABC, DE, WOA, PSO, GA, and GSA, correspondingly, with respect to the best value. Moreover, DE performed the best in terms of the STD, whereas BRO, ABC, PS, GA, GSA and WOA ranked second to last, respectively. Furthermore, the fitness values obtained by PSO and WOA are competitive. Also, the last row proves that the DE and BRO provide more stable results

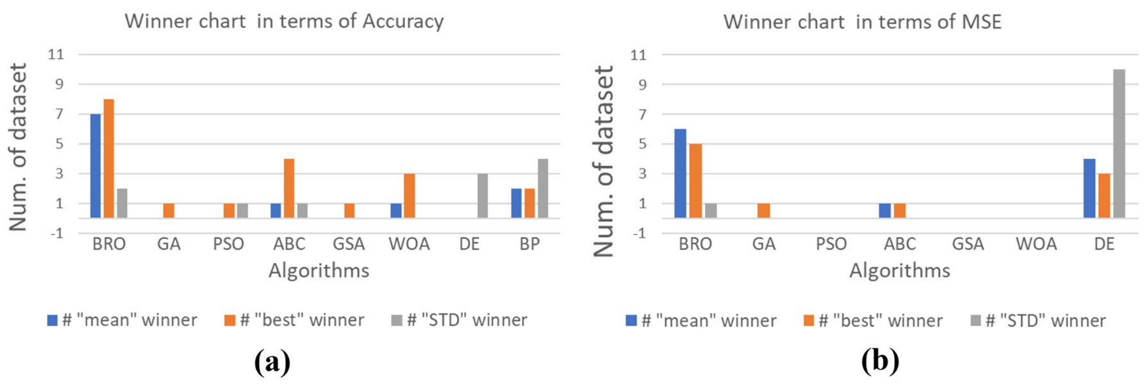


Fig. 4 Winner chart of algorithm respect to **a** accuracy and **b** MSE

than other compared algorithms. In contrast, the WOA ranks worst according to STD. Moreover, Fig. 4 illustrates how many times the algorithms have been ranked first in terms of accuracy and MSE over all data sets. As it is clear from Fig. 4, although DE offers better STD than BRO in terms of all performance metrics (accuracy and MSE), BRO outperforms DE in terms of the mean and best values. It indicates that DE may get stuck in local optima frequently.

In order to do a deep investigation, the convergence rate analysis was performed for BRO and its competitors. Figure 5 portrays convergence curves of the best fitness values among all solutions for each dataset. Almost in all cases, the convergence curve of BRO plunges rapidly, and Fig. 5 proves that the BRO and DE have a faster overall convergence rate compared to others in all datasets. On the other hand, in most cases, WOA ranked second and outperformed other compared algorithms. Also, the convergence rate of GA, PSO, and WOA are competitive., and they may have similar convergence curves. Finally, Fig. 5 also demonstrates that the GSA has the worst convergence rate, and never outperforms other algorithms.

6 Conclusion

In this paper a BRO-based MLP training algorithm has been proposed. The performance of BRO for training the MLP is compared with five well-known optimization algorithms. Also, ten most-used classification benchmark datasets are used to evaluate the performance of BRO and its competitors. The numerical comparisons were conducted by taking the average of MSE and classification accuracy over 20 independent runs. BRO-based classifier ranked first in terms of classifier accuracy and error according to the means and best values. Moreover, BRO respectively ranked third and second in STD according to accuracy and error rate. Furthermore, most cases, the convergence curve of BRO and DE plunge rapidly; they have a faster overall convergence rate compared to others in all datasets. A good balance between exploitation and exploration lets BRO not only find an accurate solution but also escape from poor local optima. Moreover, in some cases, BP provides very different mean and best classifier accuracy values due to getting stuck in local optima. Overall, experiments confirm that, according to error rate, accuracy, and convergence, the proposed approach yields promising results and outperforms its competitors Metaheuristic-based learning algorithms may perform successfully for the MLP which includes a low number of hidden layers. The number of parameters to be optimized will increase exponentially by increasing the number of hidden layers. The BRO algorithm is no exception to this handicap. In the future work, it is planned to use hybrid BP and BRO to overcome the aforementioned handicap and extend the proposed algorithm for training the convolutional neural network.

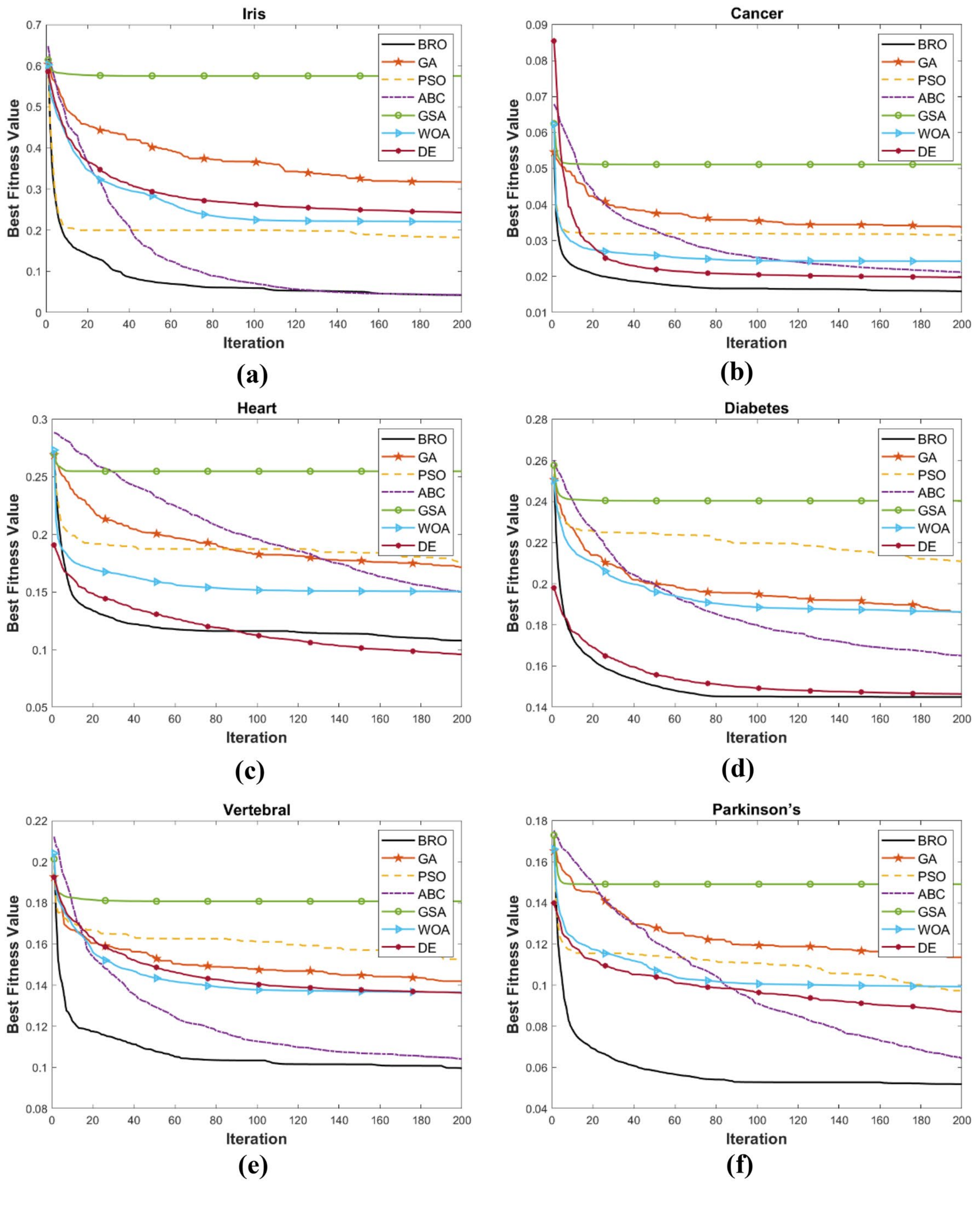
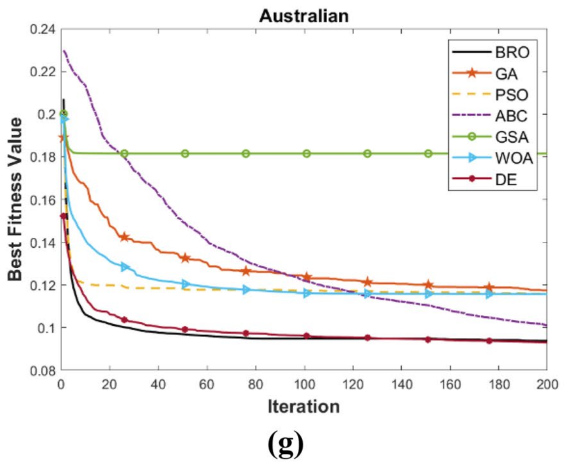
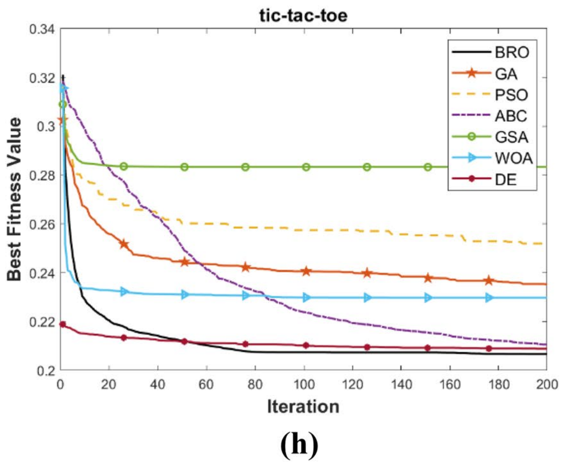
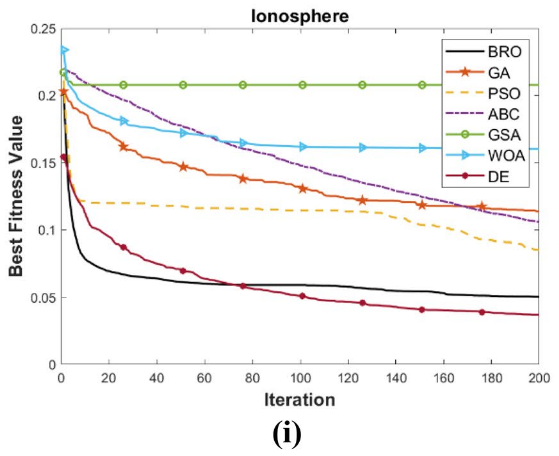
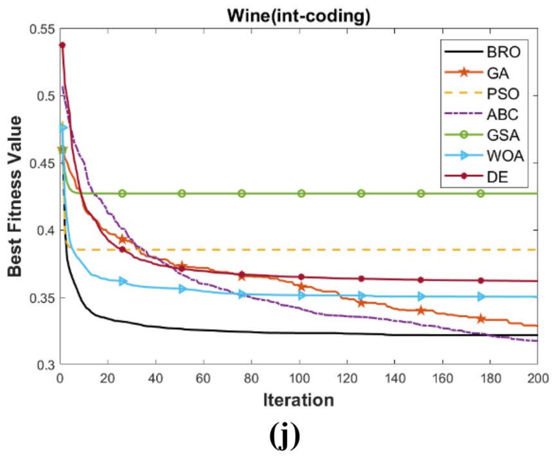


Fig. 5 MSE convergence curves









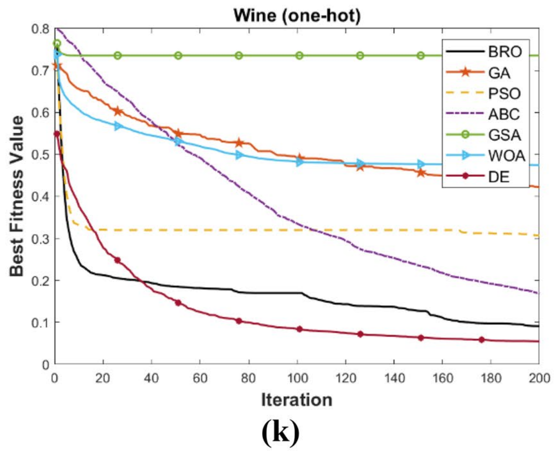


Fig. 5 (continued)

Declarations

Conflict of interest Authors declares that they have no conflict of interest.

References

Aljarah I, Faris H, Mirjalili S (2018) Optimizing connection weights in neural networks using the whale optimization algorithm. Soft Computing 22(1):1–15

Andrew NG, Katanforoosh K, Mourri YB (2020) Neural networks and deep learning. McGraw Hill, New York

Askarzadeh A (2014) Bird mating optimizer: an optimization algorithm inspired by bird mating strategies. Commun Nonlinear Sci Numer Simul 19(4):1213–1228

Askarzadeh A, Rezazadeh A (2013) Artificial neural network training using a new efficient optimization algorithm. Applied Soft Computing 13(2):1206–1213

Bhattacharjee K, Pant M (2019) Hybrid particle swarm optimization-genetic algorithm trained multi-layer perceptron for classification of human glioma from molecular brain neoplasia data. Cogn Syst Res 58:173–194

Blum C and Socha K (2005) Training feed-forward neural networks with ant colony optimization: an application to pattern classification. In Fifth International Conference on Hybrid Intelligent Systems (HIS'05), p 6.

Braik M, Sheta A, Arieqat A (2008) A comparison between GAs and PSO in training ANN to model the TE chemical process reactor. Proceedings of the AISB 2008 symposium on swarm intelligence algorithms and applications, vol 11, pp 24–30.

Chatterjee S, Sarkar S, Hore S, Dey N, Ashour AS, Balas VE (2017) Particle swarm optimization trained neural network for structural failure prediction of multistoried RC buildings. Neural Comput Appl 28(8):2005–2016

Dorigo M, Di Caro G (1999) Ant colony optimization: a new meta-heuristic. Proc Congr Evol Comput 2:1470–1477

Eberhart R and Kennedy J (1995) A new optimizer using particle swarm theory. MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, pp. 39–43: IEEE.

Haykin S (2007) Neural networks: a comprehensive foundation. Prentice-Hall, Inc., Upper Saddle River

Hebb DO (1949) The organization of behavior: a neuropsychological theory. Wiley, New York

Holland JH (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT Press, London

Ilonen J, Kamarainen J-K, Lampinen J (2003) Differential evolution training algorithm for feed-forward neural networks. Neural Process Lett 17(1):93–105

Jaddi NS, Abdullah S, Hamdan AR (2015) Optimization of neural network model using modified bat-inspired algorithm. Appl Soft Comput 37:71–86

Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Global Optim 39(3):459–471

Karaboga D, Akay B, Ozturk C (2007) Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks. Springer, Berlin, pp 318–329

Linggard R, Myers D, Nightingale C (2012) Neural networks for vision, speech and natural language. Springer, Berlin

McClelland JL, Rumelhart DE, Hinton GE (1986) The appeal of parallel distributed processing. MIT Press, Cambridge, pp 3–44

McCulloch WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys 5(4):115–133

Mirjalili S (2015) How effective is the Grey Wolf optimizer in training multi-layer perceptrons. Appl Intell 43(1):150–161

Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67

Mirjalili S, Mirjalili SM, Lewis A (2014a) Grey wolf optimizer. Adv Eng Softw 69:46–61

Mirjalili S, Mirjalili SM, Lewis A (2014) Let a biogeography-based optimizer train your multi-layer perceptron. Inform Sci 269:188–209

Montana DJ, Davis L (1989) Training feedforward neural networks using genetic algorithms. IJCAI 89:762–767

Nawi NM, Khan A, Rehman MZ (2013) A new back-propagation neural network optimized with cuckoo search algorithm. Springer, Berlin, pp 413–426

Nayak J, Naik B, Behera HS (2016) A novel nature inspired firefly algorithm with higher order neural network: performance analysis. Eng Sci Technol Int J 19(1):197–211

Ojha VK, Abraham A, Snášel V (2017) Metaheuristic design of feedforward neural networks: a review of two decades of research,". Engineering Applications of Artificial Intelligence 60:97–116

Ozturk C, Karaboga D (2011) Hybrid artificial bee colony algorithm for neural network training. IEEE Congr Evol Comput 2011:84–88

Price KV (1996) Differential evolution: a fast and simple numerical optimizer. In Proceedings of North American Fuzzy Information Processing, pp. 524–527: IEEE.

Rahkar Farshi T (2020) Battle royale optimization algorithm. Neural Comput Appl 33:1139–1157

Rashedi E, Nezamabadi-pour H, Saryazdi S (2009) GSA: A Gravitational search algorithm. Inform Sci 179(13):2232–2248

Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural Netw 61:85–117

Schwefel H-P (1984) Evolution strategies: a family of non-linear optimization techniques based on imitating some principles of organic evolution. Ann Oper Res 1(2):165–167

Simon D (2008) Biogeography-based optimization. IEEE Trans Evol Comput 12(6):702–713

Storn R, Price K (1997) Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359

Wdaa ASI and Sttar A (2008) Differential evolution for neural networks learning enhancement. Universiti Teknologi Malaysia

Werbos P (1989) Back-propagation and neurocontrol: a review and prospectus. In: IEEE Proceedings of the International Joint Conference on Neural Networks (IJCNN'89), pp. 1, I209-I216.

Wienholt W (1993) Minimizing the system error in feedforward neural networks with evolution strategy. Springer, London, pp 490–493

Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82

Yadav RK and Anubhav (2020) PSO-GA based hybrid with adam optimization for ANN training with application in medical diagnosis. Cogn Syst Res 64:191–199

Yang X-S (2009) Firefly algorithms for multimodal optimization. International symposium on stochastic algorithms. Springer, Berlin, pp 169–178

Yang X-S (2010) A new metaheuristic bat-inspired algorithm. In: González JR, Pelta DA, Cruz C, Terrazas G, Krasnogor N (eds) Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Springer, Berlin, pp 65–74

Yang X-S and Deb S (2009) Cuckoo search via Lévy flights. In 2009 World congress on nature & biologically inspired computing (NaBIC), pp 210–214

Zhang J-R, Zhang J, Lok T-M, Lyu MR (2007) A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training. Appl Math Comput 185(2):1026–1037

Zurada JM (1992) Introduction to artificial neural systems. West St. Paul, New York