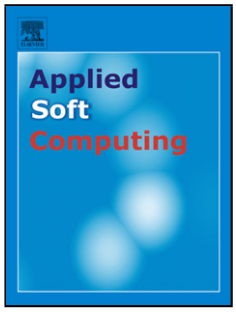


Title: Optimization of Neural Network Model using Modified Bat-inspired Algorithm

Author: Najmeh Sadat Jaddi Salwani Abdullah Abdul Razak Hamdan

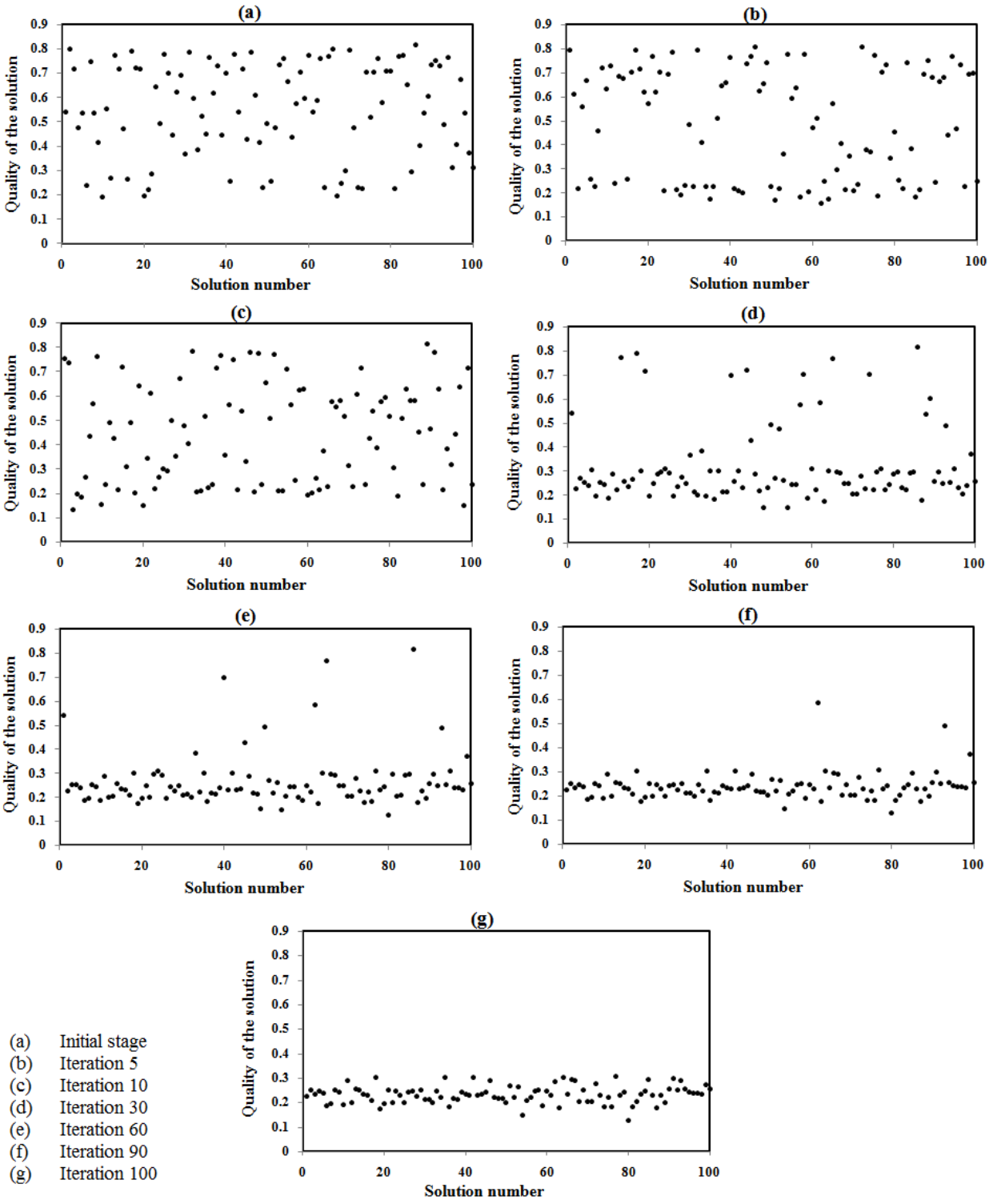
PII: S1568-4946(15)00495-0
DOI: http://dx.doi.org/doi:10.1016/j.asoc.2015.08.002
Reference: ASOC 3121

To appear in: *Applied Soft Computing*

Received date: 14-3-2014
Revised date: 19-4-2015
Accepted date: 1-8-2015

Please cite this article as: N.S. Jaddi, S. Abdullah, Optimization of Neural Network Model using Modified Bat-inspired Algorithm, *Applied Soft Computing Journal* (2015), http://dx.doi.org/10.1016/j.asoc.2015.08.002

- A modified bat algorithm with a new solution representation for both optimizing the weights and structure of ANNs is proposed.
- To improve the exploration and exploitation capability of bat algorithm some modifications based on chaotic map on bat algorithm is studied.
- The Taguchi method is used to tune the parameters of the algorithm.
- Six classification and two time series benchmark datasets are used to test the performance of the proposed approach in terms of classification and prediction accuracy.
- Statistical tests are applied to compare the performance of the methods.
- Finally, our best method is applied to a real-world problem, namely to predict the future values of rainfall data in Selangor at Malaysia.



(a) Initial stage
(b) Iteration 5
(c) Iteration 10
(d) Iteration 30
(e) Iteration 60
(f) Iteration 90
(g) Iteration 100

Optimization of Neural Network Model using Modified Bat-inspired Algorithm

Najmeh Sadat Jaddi, Salwani Abdullah, Abdul Razak Hamdan

Data Mining and Optimization Research Group (DMO),
Centre for Artificial Intelligence Technology, Faculty of Information Science and Technology
Universiti Kebangsaan Malaysia (UKM), 43600 Bangi, Selangor, Malaysia
najmehjaddi@gmail.com, salwani@ukm.edu.my, arh@ukm.edu.my

Abstract: The success of an artificial neural network (ANN) strongly depends on the variety of the connection weights and the network structure. Among many methods used in the literature to accurately select the network weights or structure in isolate; a few researchers have attempted to select both the weights and structure of ANN automatically by using metaheuristic algorithms. This paper proposes modified bat algorithm with a new solution representation for both optimizing the weights and structure of ANNs. The algorithm, which is based on the echolocation behaviour of bats, combines the advantages of population-based and local search algorithms. In this work, ability of the basic bat algorithm and some modified versions which are based on the consideration of the personal best solution in the velocity adjustment, the mean of personal best and global best solutions through velocity adjustment and the employment of three chaotic maps are investigated. These modifications are aimed to improve the exploration and exploitation capability of bat algorithm. Different versions of the proposed bat algorithm are incorporated to handle the selection of the structure as well as weights and biases of the ANN during the training process. We then use the Taguchi method to tune the parameters of the algorithm that demonstrates the best ability compared to the other versions. Six classification and two time series benchmark datasets are used to test the performance of the proposed approach in terms of classification and prediction accuracy. Statistical tests demonstrate that the proposed method generates some of the best results in comparison with the latest methods in the literature. Finally, our best method is applied to a real-world problem, namely to predict the future values of rainfall data and the results show satisfactory of the method.

Keywords: Bat-inspired algorithm; Artificial neural network; Chaotic map; Time series prediction; Classification; Real-world rainfall data

1. Introduction

An artificial neural network (ANN) is an imitation of a biological natural neural network. Each ANN is structured into several interconnections of simple processing units which are called nodes or neurons. An ANN can have one or more layers of nodes between the input layer and output layer. Each of these interlayers is called a hidden layer, and they can be completely or partially connected. Each connection between two nodes has a specified weight. An ANN can be taught by training it using on hand cases with inputs and expected outputs. Due to the learning ability and nonlinearity method of ANNs [1], they have been used extensively in many applications of data mining such as classification, associate rule mining and clustering [2, 3]. ANN is regarded as one of computational intelligence techniques and an effective machine learning tool capable of data mining applications.

Although many researches have been performed to generate an accurate ANN model in the literature, the automatic creation of a near-optimal structure for an ANN for a given task is still considered to have potential. Several researchers have used evolutionary algorithms in the design of ANNs. For example, Palmes, Hayasaka, and Usui [4] applied a mutation-based genetic algorithm for neural network training, while Gepperth and Roth [5] proposed an evolutionary multiobjective procedure to optimize the feedforward neural network. Hung and Du [6] applied particle swarm optimization (PSO) to optimize an ANN. Hervás-Martínez, Martínez-Estudillo, and Carbonero-Ruz [7] designed the structure and weights of an ANN using an evolutionary algorithm and a backward stepwise procedure. Chi-Keong, Eu-Jin, and Kay Chen [8] proposed a hybrid multiobjective evolutionary approach in their design for an ANN. a Taguchi-based parameter designing of genetic algorithm for ANN training was proposed in [9]. However, most of these strategies need the chromosome length to be predefined. Since this user-defined length is problem-dependent, it can affect the efficiency of the method. Besides this issue, the input variables are fixed (no feature selection), but the selection of features is important especially for the classification problem [10-13]. Even selecting all features as the input does not guarantee that the best accuracy in classification will be achieved [14].

The use of merging and growing algorithms in the design of ANNs was proposed by Islam, Amin, Ahmmed, and Murase [15] and later on in another approach by Islam, Sattar, Amin, Yao, and Murase [16]. Kaylani, Georgiopoulos, Mollaghasemi, and Anagnostopoulos [17] employed a prune operator for a genetic algorithm to design the ARTMAP architecture. Later on, they extended their study for multiobjective approach [18]. Carry and Morgan [19] used a modified and pruned neural network for seasonal data. Mantzaris, Anastassopoulos, and Adamopoulos [20] applied a pruned probabilistic neural network using a genetic algorithm to minimize the structure of an ANN. The important problem with merging and growing algorithms is that when to add or delete the nodes need to be carefully predefined.

In an attempt to address designing ANN, a family of multi-layer self-organizing neural networks has been studied in recent years [21-24]. Their method starts with one hidden layer and then the next hidden layers are added until the stopping criteria are met. Masutti and de Castro [25] used a combination of self-organizing networks and an artificial immune system to minimize the neurons in an ANN. However, the main disadvantage of the self-organizing method is that by growing the ANN during the process, the time will also incrementally grow so the stopping criteria need to be carefully predefined.

Other metaheuristic algorithms have been applied for ANN model optimization in recent years. Ludermir, Yamazaki and Zanchettin [26] used the advantages of combined simulated annealing, tabu search and backpropagation to optimize an ANN model during the training process. Yu, Wang, and Xi [27] evaluated an ANN using modified PSO and discrete PSO (DPSO). A PSO-based single multiplicative neuron model for time series prediction was proposed by Zhao and Yang [28]. In 2011, a hybrid approach have been proposed to optimize the operation of the ANN in terms of weights and architecture [29].

In the literature a few number of methods [1, 26, 29] have been applied for optimizing both weights and structure of neural network simultaneously. It is the significance of the research in this area to find a superior solution for the neural network model.

In our method presented in this paper, both the weights and the structure of ANN are varied during optimization procedure. The variation of weights and structure of ANN is provided using a solution representation proposed in this paper. With aid of this solution representation, the troubles with user-defined length and feature selection in evolutionary algorithms, time growing with self-organizing neural networks and predefined period for adding and deleting process in merging and growing algorithms are overcome. In order to have more accurate model compared to proposed methods in the literature, optimization technique in our method is provided derived from bat algorithm. The bat algorithm as recent and powerful algorithm has been proposed based on the echolocation behaviour of bats by Yang [30]. The advantage of bat algorithm is for gaining the combination of population based algorithm and local search. This combination initially provides the algorithm capability of global diverse exploration and local intensive exploitation which is the key point in metaheuristic algorithms. This interesting metaheuristic optimization technique behaves as a group of bats looking for prey using their ability of echolocation [31-33]. This new metaheuristic algorithm very quickly revealed its superior ability in many areas of optimization. A work comparing the bat algorithm and many other algorithms was recently published [34]. Also, a solution for a scheduling problem using a bat algorithm was presented by Musikapun and Pongcharoen [35], while a fuzzy bat clustering method was proposed by Khan, Nikov, and Sahai [36], and many other applications of the bat algorithm have been put forward [37-40].

In this paper, the basic bat algorithm and several modified versions which are toward enhancing exploration and exploitation capability of bat algorithm are evaluated and tested using six classification and two time series prediction problems. Among these, the ability of three versions using the chaotic map approach to generate a chaotic sequence instead of a random sequence is investigated. Chaos is a random-like procedure set up in nonlinear and dynamic systems. Most of the metaheuristic algorithms in the literature use uniform probability to generate random numbers. When a random number is needed by the algorithm, it can be produced by iterating one action of the selected chaotic map that is started from a random initial value in the first step. In recent years, new approaches using the chaotic map have been presented [41-43]. In some works, chaotic sequences have been assumed instead of random sequences and to some extent good results have been achieved in many applications [44-46]. Following this same line of study in the literature, we combine three versions of chaotic maps with our method to improve the quality of the convergence in bat algorithm. The Taguchi method is then applied to the best method to adjust the parameters of the algorithm. Then statistical tests are undertaken to determine the efficacy of our proposed algorithm.

The rest of this paper is organized as follows. Section 2 provides a brief description of classification and time series prediction problems. Section 3 explains the proposed method in detail. Section 4 reports the experimental results and discusses the outcomes of the statistical tests, and finally, Section 5presents the conclusion of this work.

2. Problem description

In this work we solve two different data mining tasks: classification and time series prediction. Many algorithms have been widely applied to solve these kinds of problems which are significant in the field of data mining.

2.1 Classification problem

The task of assigning an object to a proper group (based on a number of attributes describing that object) is defined as a classification problem. The classification process is as follows:

- Given a set of records called a training set, each record contains a set of attributes; one of the attributes is selected as the class attribute;
- A proper model for the class attribute as a function of other attributes is sought;
- Earlier unseen records are allocated to a class as accurately as possible and a testing set is employed to determine the accuracy of the model.

Generally, the given dataset is divided into training and testing sets, where the training set is employed to build the model and the testing set is applied to validate the model.

2.2 Time series prediction

Time series prediction is the use of a specified model to forecast future values based on earlier values. The time series prediction procedure is as follows:

- Given a set of points in the past as a training set, each point is considered as input for the model;
- A proper model for the future values as a function of past values is sought;
- For every point in the past, the model is trained using past data as the inputs and what follows is the desired output.

The training is referred to when building the model and the model is then tested using test data.

3. Methodology

Several ANNs have been proposed in the literature. In this work, the optimization of the weights and structure of ANN is considered by applying a bat-inspired algorithm. Each solution in the population of the bat algorithm

contains both a structure and a weights solution. In this method, various weights, inputs, number of hidden layers and number of nodes in each hidden layer are measured based on a specified fitness function that is dependent on the structure of the ANN and the weights. Diversification of each solution in this method is achieved by adjusting the frequency, velocity and position of the bat population in the bat algorithm. The probability of applying the diversity function of the bat algorithm on the structure of the ANN and the weights and biases (W&B) are equal. The details of the method are given in the following subsections.

3.1 Bat algorithm

As mentioned above, the bat algorithm was first proposed by Yang [30] and is based on the echolocation activity of bats. Bats release a very loud sound pulse and pay attention to the echo that returns from objects. Bats fly randomly using frequency, velocity and position to search for prey. In the bat algorithm, the frequency, velocity and position of each bat in the population is updated for further movements. The algorithm is formulated to imitate the ability of bats to find their prey. The bat algorithm follows many simplifications and idealization rules of bat behaviour that were considered and proposed by Yang [30].

The bat algorithm has the advantage of combining a population-based algorithm with local search. This algorithm involves a sequence of iterations, where a collection of solutions changes through random modification of the signal bandwidth which is increased using harmonics. The pulse rate and loudness is updated only if the new solution is accepted. The frequency, velocity and position of the solutions are calculated based on following formulas:

$$f_i = f_{min} + (f_{max} - f_{min})\,\beta \tag{1}$$

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x_{gbest}^t) f_i \tag{2}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{3}$$

where the value of β is a random number within the range of [0,1], f_i is the frequency of the i^{th} bat that controls the range and speed of movement of the bats, v_i and x_i denote the velocity and position of i^{th} bat, respectively, and x_{gbest}^t stands for the current global best position at time step t.

In order to enhance the diversity of the possible solutions a local search approach is applied to those solutions that meet a certain condition in the bat algorithm. If the solution meets the condition, then random walk (Eq. 4) is employed to generate a new solution:

$$x_{new} = x_{old} + \epsilon A^t \tag{4}$$

in which $\epsilon \in [-1,1]$ is a random number that efforts to the power and direction of the random walk and A^t denotes the average loudness of all bats so far.

The loudness A_i and the pulse rate r_i have to be updated in each iteration. The loudness typically decreases when a bat find its prey while the pulse rate increases. The loudness A_i and pulse rate r_i are updated as follows:

$$A_i^{t+1} = \alpha A_i^t \quad (5)$$

$$r_i^{t+1} = r_i^0[1 - exp(-\gamma t)] \quad (6)$$

in which α and γ are constant values and both are equal to 0.9 as in [30]. The loudness and pulse rate are updated only if the new solution is accepted. Fig. 1 shows the flowchart of the bat algorithm.

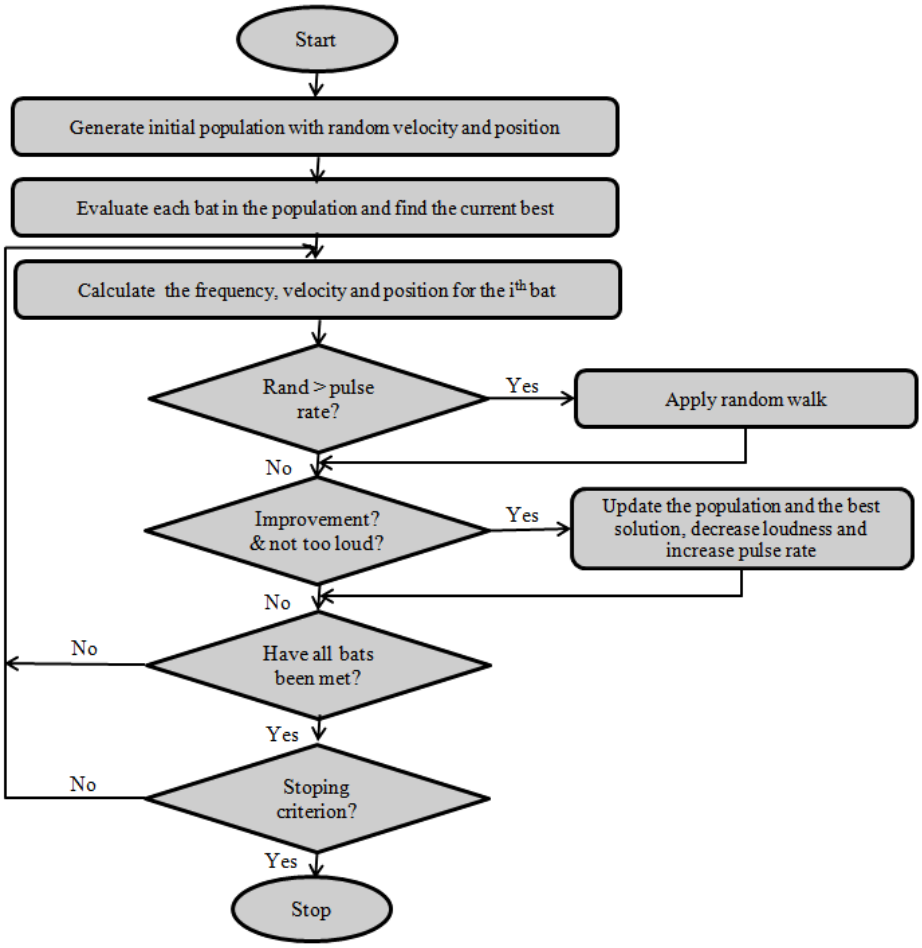


Fig. 1. Flowchart of bat algorithm

3.2 Modifications of bat algorithm for a dynamic neural network

In this paper, several versions of bat algorithm are employed in order to optimize the ANN model. The bat algorithm and its modifications are:

A. Basic bat for dynamic neural network(*BatDNN*)
B. Modified bat for dynamic neural network(*MBatDNN*)
C. Mean bat for dynamic neural network (*MeanBatDNN*)
D. Piecewise map for bat dynamic neural network (*PiecewiseBatDNN*)
E. Logistic map for bat dynamic neural network (*LogisticBatDNN*)
F. Sinusoidal map for bat dynamic neural network (*SinBatDNN*)

A. *BatDNN*: The standard version of the bat algorithm (described in Fig. 1) is applied for optimizing the ANN model. The solutions in the population consist of two parts. The first part encodes the structure of the ANN and the second part determines the W&B in the ANN structure. For the initial population, the structure of the solutions is randomly assigned, then the number of W&B is calculated to fit each structure and at the end the value of the W&B are randomly generated. In the *do while* loop of the bat algorithm there is a probability that the adjustment of frequency, velocity and position will be applied to the structure solution or W&B solution. This probability is the same for both cases.

As is evident from Fig.2 and relevant formulas, in this bat algorithm the movement of each bat in the search space is towards continuous valued places. In our study, when there is a chance for the structure solution to be adjusted by the new frequency, velocity and position, the search space is formed as a binary pattern [47]. In order to represent the bat's position in the binary vector we use a sigmoid function:

$$f(x_i^t) = \frac{1}{1+e^{-x_i^t}} \tag{7}$$

Eq. (7) is applied to the output of Eq. (3) when the adjustment procedure of the bat algorithm is applied to the structure of the ANN. If the output of $f(x_i^t)$ is greater than a certain value within the range of (0, 1) then the value of x_i^t is set to one, otherwise this value is adjusted to 0. Therefore with the aid of this sigmoid function we provide only binary values for the structure of each bat candidate that is selected for the bat algorithm adjustment. Then the W&B solution is organized based on the new structure. If the adjustment procedure of the bat algorithm is affected by the W&B solution then the structure solution will remain the same as the previous structure. After generating a new solution, a random walk procedure is applied to the solutions under certain conditions. Then the solution is accepted if the required conditions are met. The loudness and the pulse rate are updated when the solution is accepted. This process is continued until the stopping criterion is met. The pseudo code of the proposed method is shown in Fig. 2.

Set population size, popSize← 100

Set number of iterations, numOfIte← 100

Set minimum frequency, f_{min}← 0

Set maximum frequency, f_{max}← 2

Set loudness, A_i←0.25

Set pulse rate, r_i← 0.5

Set probability of applying operators on structure or W&B, prob← 0.5

Set the initial population

Set the best fit, $f(x_{best})$

Set iteration ←0

do while(*iteration < numOfIte*)

for *all bats in the population*

if (*rand < prob*)

Generate new solution by adjusting frequency (f_i), velocity (v_i), and position (x_i) on structure solution using binary bat, Eqs. (*1*),(*2*),(*3*),(*7*)

Add or delete the random cells in W&B based on new structure

else

Generate new solution by adjusting frequency (f_i), velocity (v_i), and position (x_i) on W&B solution with the same structure, Eqs. (*1*),(*2*),(*3*)

endif

if($\beta > r_i$) *then generate a local solution around the solution, Eq.* (*4*)

endif

Evaluate the solution

if ($\beta < A_i$ & ($f(x_i) < f(x_{best})$) *then accept the new solution and increase r_i and decrease A_i,Eqs.* (*5*) *and* (*6*)

Update the x_{best} and population

endif

endfor

Increase the iteration

endwhile

Return the x_{best}

Fig. 2. Pseudo code of proposed bat algorithm for ANN model optimization

B. *MBatDNN*: The basic bat algorithm updates the velocity and position of each bat in a similar way to the procedure in standard PSO [48] while f_i controls the movement of bats. In some respects, this bat algorithm is an integration of PSO and local search which uses control of loudness and pulse rate. The adjustment procedure in the basic bat algorithm is similar to PSO but the idea of adjusting velocity based on personal best (*pbest*) and global best (*gbest*) in PSO motivated us to continue our study to enhance the quality of the bat algorithm by improving the bat movement by incorporating the superiority of *pbest*. In the standard bat algorithm the bats are moved toward the

gbest which leads the algorithm to the exploration, while in the modified version, the effect of *pbest* is considered to improve the exploitation of the algorithm. In this case, we altered Eq. (3) into Eq. (8):

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x_{gbest}^t)f_i + (x_i^{t-1} - x_{pbest}^t)f_i \quad (8)$$

In this new version of the bat algorithm each bat keeps the history of its path in the search space which is linked with the best solution reached so far. This value is called *pbest*, while *gbest* is the best value gained so far by any bat among all the bats in the population.

These two different kinds of neighbourhoods help the algorithm to achieve better convergence. In the *gbest* group, all the bats are neighbours of all the other bats; therefore, the *gbest* position is used in the public expression of the velocity update equation. It is assumed that *gbest* groups converge speedily, because all the bats are involved at the same time in the best fraction of the search space. However, it is possible that the global optimum is not close to the best bat; thus, it might be impossible for the group to discover other areas and this is where we know as local optima. In the *pbest* group a specific number of bats can affect the velocity of a given bat. This means that the group will converge more slowly but they will be able to find the global optimum with superior probability. Both *gbest* and *pbest* can be considered social neighbourhoods because the relations among the bats do not depend on their positions. The pseudo code of this modified bat algorithm is the same as in Fig. 2 with a small modification, that is, it uses Eq. (8) instead of Eq. (3) when it updates the velocity of the solution.

C. *MeanBatDNN*: In this part of study we used a linear combination of *pbest* and *gbest* in the velocity update equation[49]. In this case, the current position of each bat is compared with the linear combination of the *pbest* and *gbest* positions instead of comparing *gbest* and *pbest*. We propose a new velocity update equation as follows:

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - (x_{gbest}^t + x_{pbest}^t)/2)\, f_i + (x_i^{t-1} - (x_{pbest}^t - x_{gbest}^t)/2)\, f_i \quad (9)$$

In this experiment we used Eq. (9) in place of Eq. (2) to observe the effect of this modification. A comparison of the movements of a bat in the basic bat algorithm and in the two modified versions of the bat algorithm discussed thus far, namely MBatDNN and MeanBatDNN, is illustrated in Fig. 3. As can be seen from the figure, we attempt to direct the bat's movement toward the *gbest* and *pbest* position by manipulating the velocity equation for each move. This may improve the convergence ability of the bat algorithm.

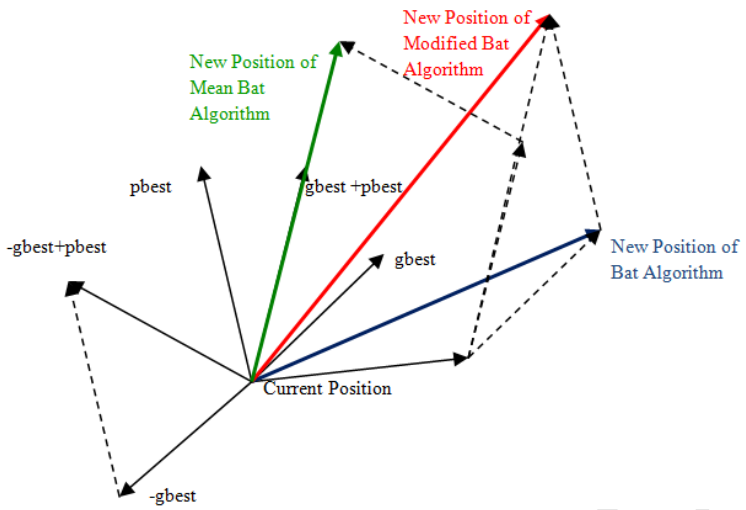


Fig. 3. Comparison of bat movement in the basic bat algorithm, MBatDNN and MeanBatDNN

D. *PiecewiseBatDNN*: The concept of the piecewise linear chaotic map has been the subject of growing interest in chaos research due to the simplicity of its implementation. The simplest piecewise linear chaotic map is described as follows:

$$y_{k+1} = \begin{cases} \dfrac{y_k}{p} & y \in (0, p) \\ \dfrac{1-y_k}{1-p} & y \in [p,1) \end{cases} \tag{10}$$

where $y \in (0,1)$ is a variable. Let k = 1, 2, 3,... then y_k and y_{k+1} are k^{th} and $(k+1)^{th}$ values of y, respectively. P is a control parameter. Variable y performs chaotically between (0,1) when parameter p is in use in the range of (0,0.5) $\cup$ (0.5,1).

E. *LogisticBatDNN*: The logistic map is a form of polynomial mapping. Mathematically, the logistic map is written as:

$$y_{k+1} = r y_k (1 - y_n) \tag{11}$$

In the case of r = 4 the values finally leave the interval (0,1). Since we need a random number between [0,1] in the velocity equation, r=4 is used in this experiment.

F. *SinBatDNN*: The properties of the sinusoidal map are similar to those of the logistic map. The simplest form of sinusoidal map in Eq. (12) generates the chaotic sequence in interval [0, 1].

$$x_{k+1} = Sin(\pi x_k) \tag{12}$$

3.3 Solution representation

In this work, two one-dimensional vectors are considered for solution representation. One vector represents the structure solution which contains binary values 0 and 1. The second vector holds the W&B of the neural network. The first solution has three parts. Two parts are for the number of hidden layers and number of nodes in each hidden layer. These occupy three cells each. The binary values of the number of hidden layers and number of nodes are kept in these three cells. For the time series problem the input values and the number of inputs are set as in other work [21-23]. We have done this in order to make our work comparable with others. For classification, because the values of the inputs and number of inputs are important [11], the feature selection part is added to the structure solution representation. The length of this part is equal to the number of features in the dataset. If the i^{th} value in this part is equal to 1, this means that the i^{th} feature in the full feature set is contained in the subset of selected features. If the cell shows a 0 value, this means the subset of selected features does not hold this feature. The second vector holds the W&B in which the number of W&B is calculated based on the structure solution. The W&B vector represents the real values of the weights and biases. Figure 4 illustrates an example of solution representation along with the related network architecture scheme.

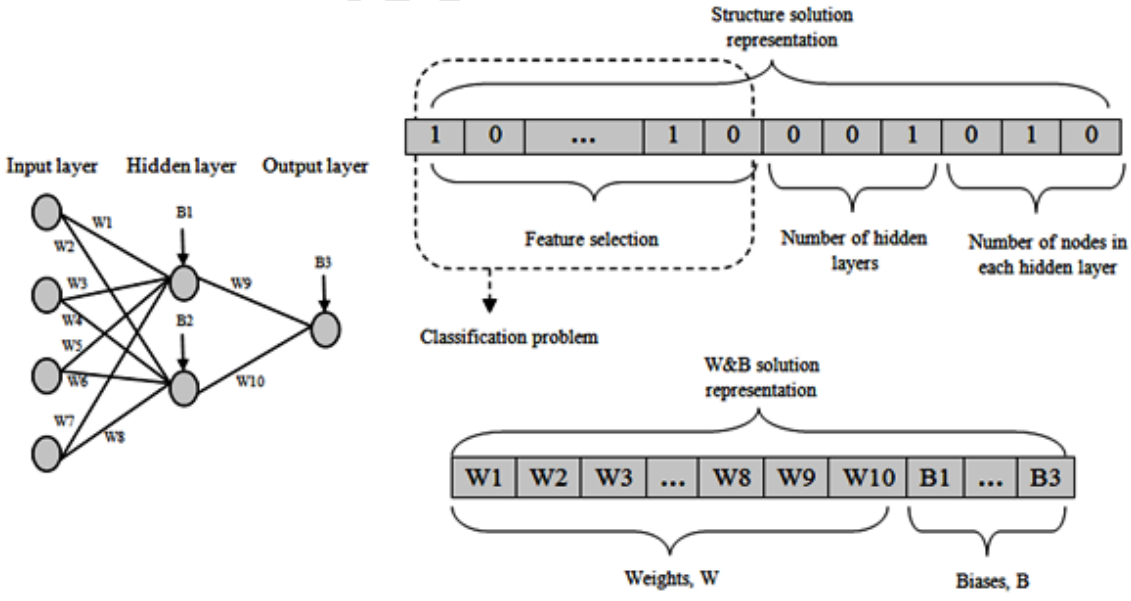


Fig. 4. Example of solution representation

3.4 Cost function

To evaluate the quality of the solution in consecutive iterations we need to use an effective cost function in order to select the solution that optimizes the objective function.

For classification problems, the classification error, which represents the percentage of misclassified training examples, is considered as:

$$E(p_t) = \frac{100}{n_t} \sum_{x \in n_t} \varepsilon(x) \tag{13}$$

Where n_t is the number of examples and $\varepsilon(x)$ is the number misclassified instances. The second part of the cost function evaluates the number of connections (weights) in the ANN. The percentage of the connections occupied by the network is specified by:

$$p_c = \frac{100}{n_c} \sum_{i=1}^{n_c} w_i \tag{14}$$

Where n_c is the maximum number of connections that can be used by the network and w_i is the number of weights or connections. Therefore for the classification problem the cost function $f(s)$ of the solution s is calculated by the average of the classification error and the percentage of the number of connections in the network, as shown in Eq. (15).

$$f(s) = \frac{1}{2}(E(p_t) + p_c) \tag{15}$$

For prediction problems, the fitness function $f(s)$ of solution s is set by the average of the mean squared error (MSE) or root mean squared error (RMSE) and the percentage of the number of connections. This is represented by the same equation (15), where $E(p_t)$ is considered to be MSE or RMSE.

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{16}$$

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2} \tag{17}$$

In these equations N is the number of instances and y_i and $\hat{y}_i$ are the actual value and predicted value, respectively.

4. Experimental results

4.1 Benchmark classification and time series prediction problems

In this section, we show the performance of the proposed methods using six classification and two time series prediction problems. The classification problems are Iris, Diabetes diagnoses, Thyroid dysfunction, Breast cancer, Credit card, and Glass identification. The time series prediction problems are Mackey-Glass and Gas Furnace. The classification problems are taken from the UCI machine learning repository [50]. The Gas Furnace dataset is a multivariate dataset from http://datasets.connectmv.com/datasets/, while Mackey-Glass is a univariate dataset which was produced from Eq. (18), where t_d =17:

$$\frac{dx(t)}{dt} = -bx(t) + \frac{ax(t-t_d)}{1+x^{10}(t-t_d)} \tag{18}$$

A summary of the description of the datasets is presented in Table 1 and the initial parameters (Table 2) were selected based on other experiments in the literature [30].

Table 1. Characteristics of datasets

Dataset	Examples	Features	Classes
Iris	150	4	3
Diabetes	768	8	2
Thyroid	7,200	21	3
Cancer	699	10	2
Card	690	15	2
Glass	214	10	6
Mackey-Glass	1,000	1	0
Gas Furnace	296	2	0

Table 2. Configuration of parameters

Parameter	Definition	Value
popSize	Size of population	100
numOfIte	Maximum number of iterations	100
f_{min}	Minimum frequency	0
f_{max}	Maximum frequency	2
A_i	Loudness of emission	0.25
r_i	Pulse rate	0.5
prob	Probability of bat adjustments on structure or W&B solutions	0.5

For the Mackey-Glass dataset, we considered the output of $x(t+6)$ with the input variables of $x(t)$, $x(t-6)$, $x(t-12)$ and $x(t-18)$.For the Gas Furnace dataset the input variables were $u(t-3)$, $u(t-2)$, $u(t-1)$, $y(t-3)$, $y(t-2)$, $y(t-1)$ and the output variable was $y(t)$, as used in earlier reported works (Oh, Pedrycz, & Park, 2003; Park, Park, Kim, & Oh, 2004; Oh &

Pedrycz, 2005; Oh, Pedrycz, & Roh, 2009). We programmed the proposed algorithm using Java. We used 30 two-fold iterations [29]to estimate the performance of the model. The data were randomly divided into two parts for each run. One half was employed for the training set and the other half was used as the testing set to test the final model. The patterns in the datasets were normalized into the range of [-1, 1] using the min-max normalization method. The activation function chosen for this experiment was the hyperbolic tangent because it has higher performance compared to others [51]. We compare the results in the following two subsections. The first evaluates the proposed algorithms in comparison with each other and the second compares the best proposed method with the methods in the literature.

4.1.1Results of comparison of proposed methods

The performance of the six versions of the algorithm were evaluated and compared with each other based on two criteria:

- The percentage of the error (classification error, MSE / RMSE)
- The percentage of the number of connections used by the model.

A summary of the results obtained by six versions of the algorithm is shown in Table 3. Note that in this work the best method found from first three versions (*MeanBatDNN*) has been used for the chaotic maps application. Another notation is that in this table the training and testing errors for the case of classification datasets (first six datasets) means classification error for training and testing while for Mackey-Glass dataset is RMSE and for Gas Furnace dataset is MSE. From Table 3 it can be seen that the LogisticBatDNN has slightly superior performance compared to the other proposed methods. An example of the performance of the LogisticBatDNN method in Mackey-Glass dataset is provided in Fig. 5, which shows the scatter plots and optimization progress for the Mackey-Glass dataset. The scatter plots show the correlation between the actual data and the predicted value. In addition, the simultaneous minimization progress of the RMSE and the number of connections in the ANN are illustrated in this figure. To prove the above finding we performed the average ranking test to find the first ranked algorithm. These results are calculated using the RANK function in Microsoft Excel and the average of the ranks are shown in Table 4.

Table 3. Results of proposed versions of bat algorithm

Dataset	Criteria	BatDNN	MBatDNN	MeanBatDNN	MeanBatDNN		
					PiecewiseBatDNN	LogisticBatDNN	SinBatDNN
Iris	Training error %	2.5546	2.5466	2.5133	3.2488	2.5008	2.5775
	Std. Dev.	0.0114	0.0109	0.0111	0.0117	0.0113	0.0108
	Testing error%	3.6133	3.7733	3.8235	4.9666	2.4213	3.9187
	Std. Dev.	0.0013	0.0060	0.0117	1.7607	0.0088	0.0096
	Connection%	5.4812	4.5288	4.5491	4.6403	4.4680	4.4984
Diabetes	Training error %	21.824	21.718	21.730	27.592	21.648	22.008
	Std. Dev.	0.0061	0.0105	0.0105	0.0440	0.0104	0.0144
	Testing error%	22.391	22.472	22.427	28.238	21.824	22.412
	Std. Dev.	0.0097	0.0308	0.0309	0.0493	0.0061	0.0310
	Connection%	5.3642	5.1548	5.0819	4.7442	5.0455	4.9817
Thyroid	Training error %	7.2792	7.2465	7.2545	7.4407	7.0911	7.1585
	Std. Dev.	0.0041	0.0041	0.0041	0.0030	0.0049	0.0052
	Testing error%	7.2318	7.4796	7.7050	7.5963	7.2545	7.7657
	Std. Dev.	0.0077	0.0168	0.0213	0.0091	0.0041	0.0186
	Connection%	11.531	7.8020	7.4473	6.7541	6.0029	6.9219
Cancer	Training error %	3.1499	3.0880	3.0018	3.1614	2.9336	2.9693
	Std. Dev.	0.0042	0.0058	0.0639	0.0050	0.0058	0.0055
	Testing error%	4.2378	3.8844	3.8491	3.7268	3.1614	3.8574
	Std. Dev.	0.0155	0.0144	0.0135	0.0114	0.0050	0.0134
	Connection%	7.5087	7.4649	7.2105	8.7631	7.0000	6.9736
Card	Training error %	13.700	13.227	13.140	13.536	12.872	13.439
	Std. Dev.	0.0124	0.0081	0.0084	0.0106	0.0063	0.0095
	Testing error%	14.586	14.561	14.557	14.708	14.471	14.572
	Std. Dev.	0.0049	0.0044	0.0044	0.0051	0.0059	0.0040
	Connection%	7.0197	6.7159	6.7816	7.1264	6.0098	6.6502
Glass	Training error %	40.965	40.529	39.906	40.124	38.748	41.900
	Std. Dev.	0.0328	0.0345	0.0303	0.0352	0.0464	0.0366
	Testing error%	44.633	43.484	43.380	43.305	41.429	44.726
	Std. Dev.	0.0227	0.0166	0.0166	0.0141	0.0288	0.0195
	Connection%	7.0618	7.4366	6.6970	8.2776	7.3657	7.2069
Mackey-Glass	Training error %	0.0007	0.0672	0.0675	0.0213	0.0020	0.0211
	Std. Dev.	0.0001	0.0001	0.0001	2.6E-05	7.8E-06	2.0E-05
	Testing error%	0.0005	0.0525	0.0525	1.8E-02	0.0029	0.0180
	Std. Dev.	8.6E-05	0.0001	0.0001	3.6E-05	1.0E-05	3.9E-05
	Connection%	4.8021	4.3594	3.2930	3.5412	3.2930	3.3333
Gas Furnace	Training error %	0.3335	0.3329	0.3321	0.3333	0.3289	0.3333
	Std. Dev.	8.0E-05	0.0001	0.0002	7.5E-05	0.0003	7.5E-05
	Testing error%	0.4861	0.4510	0.4512	0.4842	0.4476	0.4842
	Std. Dev.	0.0014	0.0013	0.0013	0.0011	0.0014	0.0011
	Connection%	4.9355	5.0140	4.9859	4.8739	4.6610	4.8739

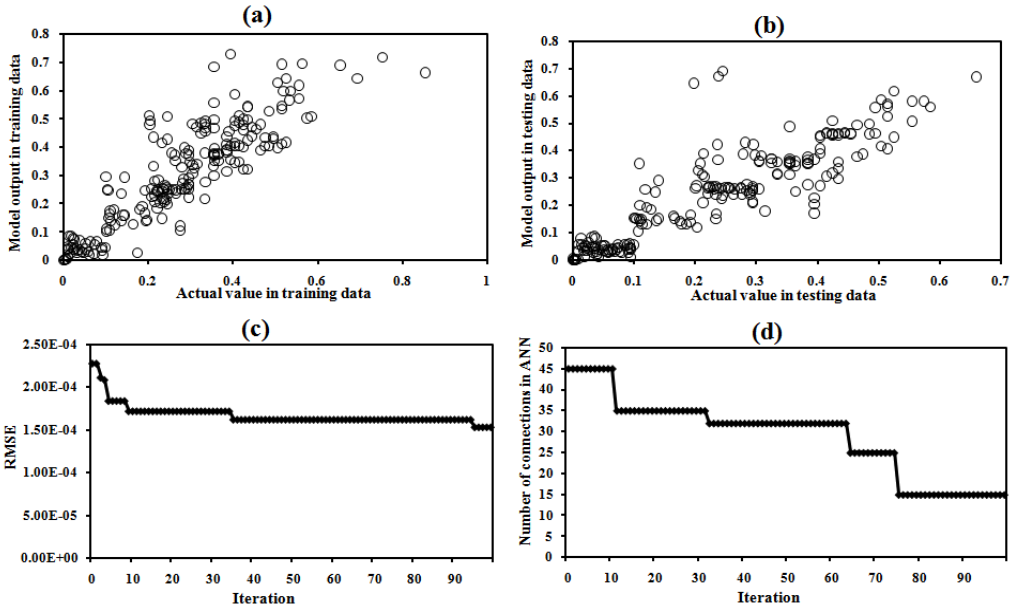


Fig. 5. Relation between actual and model output value in: (a) the training data and (b) the testing data, along with optimization progress in: (c) RMSE and (d) the number of connections of ANN.

Table 4. Average of the ranks for proposed algorithms

Training error		Testing error		Number of connections	
Algorithm	Rank	Algorithm	Rank	Algorithm	Rank
LogisticBatDNN	1.25	LogisticBatDNN	1.125	LogisticBatDNN	1.875
MeanBatDNN	2.75	MeanBatDNN	3.5	SinBatDNN	2.375
MBatDNN	3.625	MBatDNN	3.75	MeanBatDNN	3.25
PiecewiseBatDNN	3.75	PiecewiseBatDNN	4.125	PiecewiseBatDNN	4.125
SinBatDNN	4.25	BatDNN	4.125	MBatDNN	4.375
BatDNN	5.375	SinBatDNN	4.375	BatDNN	4.625

The results show that the LogisticBatDNN is ranked first in all cases for training error, testing error and number of connections. The box plot graphs in Fig. 6 and Fig.7 graphically depict the distribution of the results for all datasets from running the program 30 times. These graphs show the maximum and minimum observed values. The boxes represent the centre 50 percent of the data. The upper boundaries of the boxes place the 75th percentile of the data while the lower boundaries point to the 25th percentile. There is a line in the boxes that specifies the median of the data. This is an assessment of central tendency or in other words, the location of the centre of the data. From the graphs, it can be seen that LogisticBatDNN shows superior performance in most of the cases. This superiority of LogisticBatDNN is due to the incorporation of the advantage of the logistic map which is sensitive to the effects of a small change from the initial state.

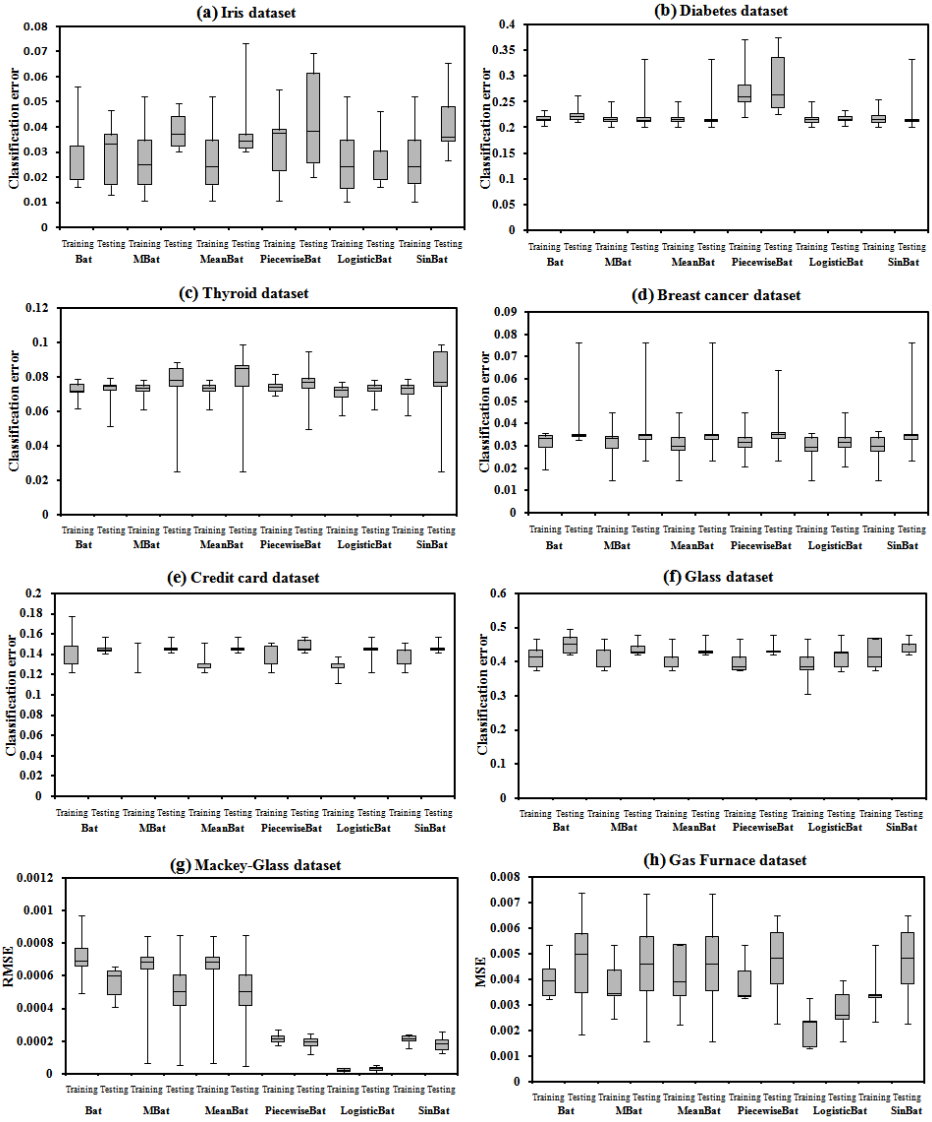


Fig. 6. Box plots of training and testing errors for all datasets: (a) Iris dataset, (b) Diabetes dataset, (c) Thyroid dataset, (d) Breast cancer dataset, (e) Credit card dataset, (f) Glass dataset, (g) Mackey- Glass dataset, and (h) Gas Furnace dataset.

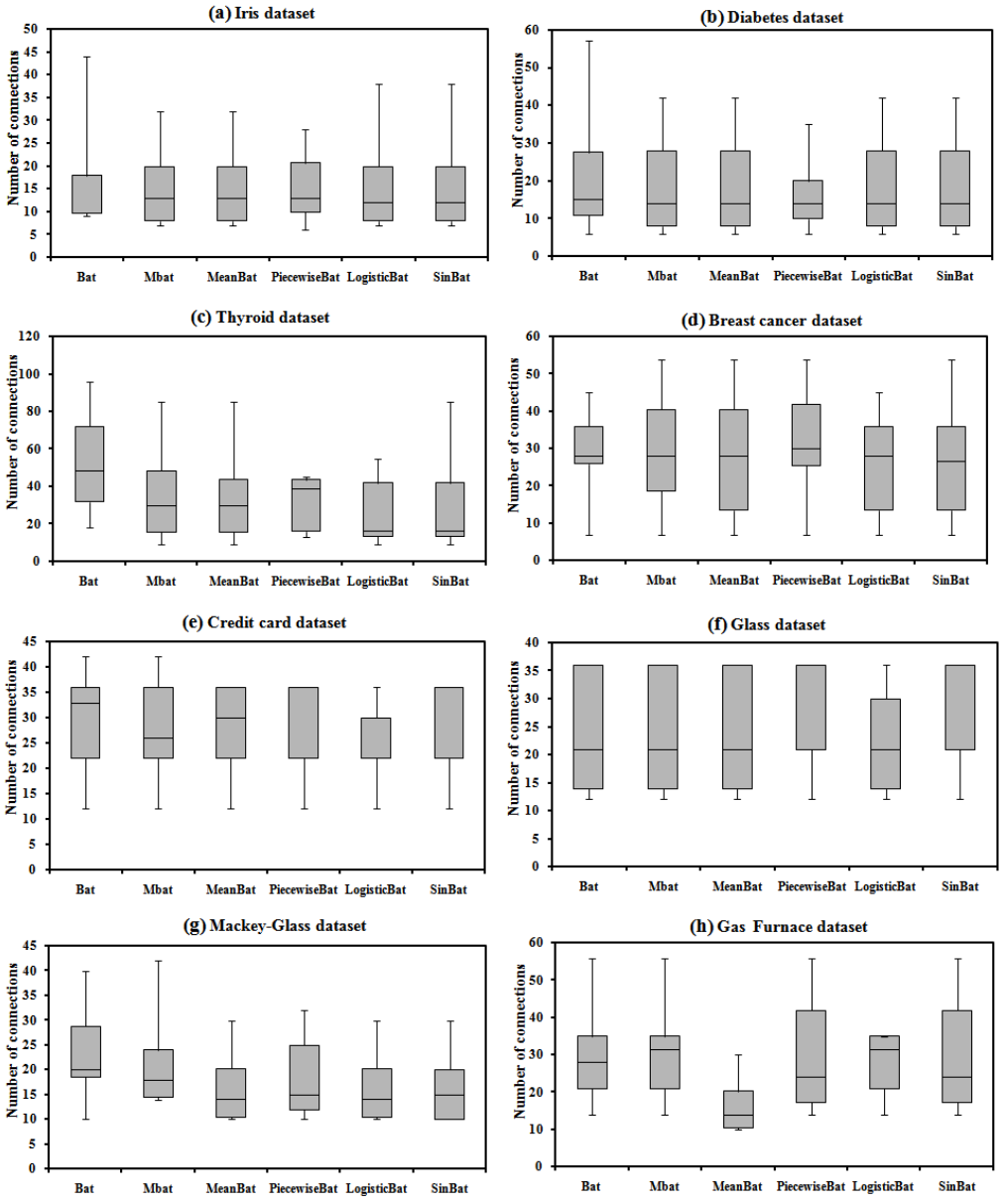


Fig. 7. Box plots of number of connections for all datasets: (a) Iris dataset, (b) Diabetes dataset, (c) Thyroid dataset, (d) Breast cancer dataset, (e) Credit card dataset, (f) Glass dataset, (g) Mackey- Glass dataset, and (h) Gas Furnace dataset.

To assess whether LogisticBatDNN is statistically different from the other proposed algorithms, next we calculated the *p*-values for all datasets, where the critical value α is equal to 0.05. This assessment was performed for testing error and number of connections. The *p*-values computed for LogisticBatDNNas compared with other algorithms are shown in Table 5. The values below the critical level (highlighted in bold) prove the ability of LogisticBatDNN. In the case of testing error, LogisticBatDNN showed better ability in a minimum of two and a

maximum of six out of the eight datasets. In the case of the number of connections, LogisticBatDNN outperforms in at least one and in a maximum in three datasets out of eight. Therefore we can conclude that LogisticBatDNN is able to train the ANN with more accuracy using less complex network model because it is able to obtain the best results in both of the examined problems, namely classification and time series prediction.

Table 5. *p*-value results for pairwise comparison of LogisticBatDNN versus other proposed algorithms

Dataset	BatDNN		MBatDNN		MeanBatDNN		PiecewiseBatDNN		SinBatDNN	
	Testing error	Number of connections	Testing error	Number of connections	Testing error	Number of connections	Testing error	Number of connections	Testing error	Number of connections
Iris	**1.5E-07**	0.0588	**2.2E-07**	0.2862	**3.3E-05**	0.2510	**1.3E-05**	0.3672	**2.3E-06**	0.1627
Diabetes	**0.0027**	0.3269	0.1289	0.0803	0.1472	0.0803	0.1289	0.0803	0.1533	0.1627
Thyroid	0.4442	**0.0001**	0.2638	0.0602	0.1465	0.1465	**0.0307**	0.4973	0.0842	**0.0249**
Cancer	**0.0002**	0.1361	**0.0073**	0.0792	**0.0082**	0.0846	**0.0236**	**0.0368**	**0.0071**	0.1627
Card	0.0931	**0.0265**	0.1209	0.080	0.1306	**0.0173**	**0.0278**	**0.0028**	0.1078	**0.0118**
Glass	**1.8E-05**	0.2723	**7.7E-05**	0.4469	**0.0001**	0.1391	**0.0001**	**0.0067**	**1.3E-06**	0.1321
Mackey-Glass	**2.5E-25**	**0.0006**	**6.5E-17**	**2.04E-11**	**7.0E-17**	0.1627	**2.4E-19**	0.2377	**6.1E-19**	0.4480
Gas Furnace	0.1787	0.2812	0.1627	0.0238	0.1627	**0.0238**	0.1394	0.3144	0.1394	0.3144

In order to further improve our method, we ran the LogisticBatDNN using the parameters selected by the Taguchi method. The Taguchi method permits the analysis of many diverse parameters using an orthogonal array without high amount of testing. This enables the recognition of the key parameters that have the most effect on the performance value so that further testing of these parameters can be carried out and, conversely, the parameters that have little effect can be disregarded.

Table 6 shows the parameters found by the Taguchi method using Minitab software. The parameters that are different from the parameters chosen without Taguchi method are shown in bold. From Table 7 it can be observed that there is a significant difference between LogisticBatDNN with the Taguchi method (T-LogisticBatDNN). The results below the critical value ($\alpha = 0.05$) are highlighted in bold. This shows that the results of the T-LogisticBatDNN, whose parameters were tuned using the Taguchi method, has better ability compared to LogisticBatDNN whose parameters were selected, based on other approaches in the literature.

Table 6. Configuration of parameters with Taguchi method

Parameter	Definition	Value with Taguchi method
popSize	Size of population	100
numOfIte	Maximum number of iterations	100
f_{min}	Minimum frequency	0.25
f_{max}	Maximum frequency	2
A_i	Loudness of emission	0. 5
r_i	Pulse rate	0.75
prob	Probability of bat adjustments on structure or W&B solutions	0.5

Table 7. *p*-value results comparison of T-LogisticBatDNN versus LogisticBatDNN

Dataset	LogisticBatDNN	
	Testing error	Number of connections
Iris	**0.00548**	0.11175
Diabetes	**0.00996**	0.21415
Thyroid	**0.00174**	**0.04950**
Cancer	**0.00147**	0.08504
Card	**0.00042**	**0.04644**
Glass	**0.00071**	**0.04316**
Mackey-Glass	0.08103	0.08374
Gas Furnace	**0.00064**	**0.00582**

As it is mentioned before, the aim of modifications on the bat algorithm is to achieve higher quality of convergence by improving the trade-off of exploration and exploitation. Fig. 8 is given in order to perceive how this modifications assist the bat algorithm to balance the exploration and exploitation. This figure shows the status of the initial and improved solutions in the population in different iteration number.

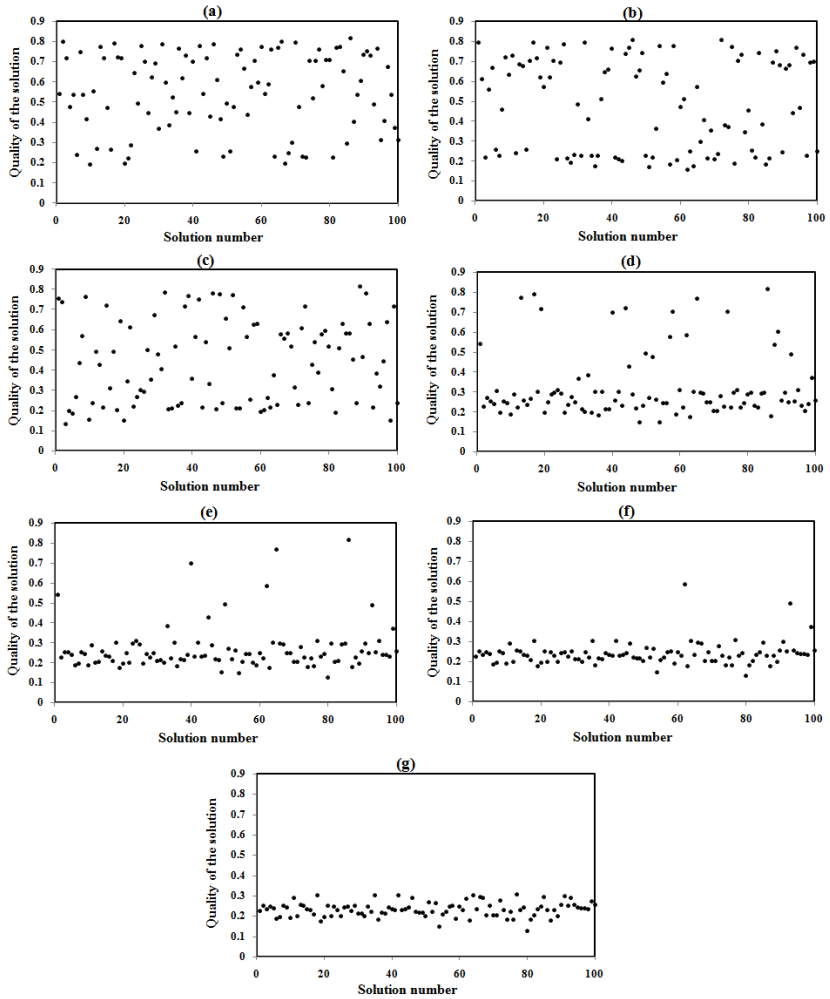


Fig. 8. Convergence action with exploration and exploitation illustration in the population: (a) initial stage, (b) iteration 5, (c) iteration 10, (d) iteration 30, (e) iteration 60, (f) iteration 90, (g) iteration 100.

4.1.2 Results of comparison of the best proposed method with other methods in the literature

In order to improve our evaluation we also compared our best method (T-LogisticBatDNN) with other approaches in the literature. Since there are plenty of results available in the literature, especially in the case of classification, for the purposes of comparison we chose the latest and the most similar approaches to our method [29] which has optimized both weights and structure of ANN. Table 8 shows the details of this comparison in the case of testing error and number of connections for classification problems. Table 9 shows a comparison with other approaches [23, 52] for time series prediction problems. The best results are shown in bold.

Table 8. Comparison of T-LogisticBatDNN and other approaches in the literature for classification problem

Dataset	Criteria	T-LogisticBatDNN	SA	TS	GA	TSa	GaTSa+BP
Iris	Testing error%	**1.8313**	12.649	12.478	2.5641	4.6154	5.2564
	Connection%	**4.0729**	26.072	25.937	22.979	24.260	31.853
Diabetes	Testing error%	**20.139**	27.156	27.404	25.994	25.876	27.061
	Connection%	**5.2732**	30.383	30.816	18.703	25.506	9.0975
Thyroid	Testing error%	**6.7545**	7.3813	7.3406	7.2850	7.3322	7.1509
	Connection%	**5.8059**	34.887	35.891	14.390	29.811	12.640
Cancer	Testing error%	**2.8947**	7.1729	7.2779	7.4220	6.2846	15.242
	Connection%	**6.6578**	35.727	33.454	57.606	34.363	27.337
Card	Testing error%	**13.738**	23.469	18.042	31.724	21.269	15.242
	Connection%	**5.6321**	41.745	44.512	55.122	43.410	38.441
Glass	Testing error%	**38.429**	58.381	56.412	58.031	57.777	55.142
	Connection%	**6.7578**	31.802	32.963	55.651	31.037	31.173

Table 9. Comparison of T-LogisticBatDNN with other approaches in the literature for time series prediction problem

Dataset	Criteria	T-LogisticBatDNN	gHFSNN (triangular)	gHFSPNN (Gaussian)	gFSPNNT*(triangular)	gFSPNNT*(Gaussian)	gFSPNNT(triangular)	gFSPNNT(Gaussian)	gFPNN(Triangular)	gFPNN(Gaussian)
Mackey-Glass	Training error%	**0.0018**	0.0163	0.0152	0.0448	0.0229	0.1240	0.0453	-	-
	Testing error%	**0.0027**	0.0315	0.0262	0.0441	0.0289	0.1180	0.0441	-	-
Gas Furnace	Training error%	0.3020	0.8200	**0.1330**	-	-	1.1000	1.0000	1.5000	1.6000
	Testing error%	**0.3300**	11.520	10.250	-	-	11.200	10.300	10.300	10.000

Note: The percentage of error has been calculated for the results in the literature.

From Table 8 it can be seen that the results of T-LogisticBatDNN outperformed the other approaches in all cases. It is clear that there is a large difference in the number of connections in T-LogisticBatDNN compared with other approaches. We believe that this difference is due to the optimization strategy that attempts to minimize the structure of the ANN model as well as weights and biases. In the case of time series prediction, our approach was

better than the other approaches apart from the training error for the Gas Furnace dataset. Although the training error in the Gas Furnace dataset is inferior compared to others, our proposed model showed great ability in testing data (Table 9).

In order further improve the comprehensiveness of our assessment, we also performed the Friedman test and Nemenyi test to find out whether there are significant differences between the performance of our proposed method and other approaches for classification error, number of connections for the classification problem and prediction error for the time series prediction problem.

4.1.3 Friedman and post-hoc tests for classification error

The value computed by the Friedman test was18.57143, which is greater than 10.57 (critical value) for the testing error of classification problems. Therefore we rejected the null hypothesis. This evaluation showed that there is a significant difference between the performances of the algorithms in the case of classification error.

We also carried out a Nemenyi test as a post-hoc test to find the group of algorithms that are differ from the others. After calculating the standard error (SE) and its posterior computing the minimum significant difference (MSD) we needed to see where any differences in means went beyond the MSD. The MSD in our case was equal to 7.53944 and the result highlighted in bold in Table 10 shows that T-LogisticBatDNN had statistically significant results in two cases.

Table 10. Nemenyi test for classification error

		T-LogisticBatDNN	SA	TS	GA	TSa	GaTSa+BP
	Mean	13.96468	22.70167	21.49282	22.1704	20.52602	19.50763
T-LogisticBatDNN	13.96468	-	**8.7369**	7.5281	**8.2057**	6.5613	5.5429
SA	22.70167	-	-	1.2088	0.5312	2.1756	3.1940
TS	21.49282	-	-	-	0.6775	0.9668	1.9851
GA	22.1704	-	-	-	-	1.6443	2.6627
TSa	20.52602	-	-	-	-	-	1.0183
GaTSa+BP	19.50763	-	-	-	-	-	-

4.1.4 Friedman and post-hoc tests for number of connections of classification problem

We repeated the same process reported in the previous subsection for the number of connections achieved for the classification problem. The Friedman test result showed a value equal to 17.42857, which is greater than the critical level (10.57). Therefore the null hypothesis was rejected for the number of connections as well .Hence we can conclude that there is a significant difference between the performances of the compared algorithms. The Nemenyi

test result for MSD was equal to 7.53944. In Table 11, the bolded cases illustrate the higher score compared to other algorithms. The T-LogisticBatDNN algorithm achieved the highest difference in all the cases.

Table 11. Nemenyi test for number of connections in classification problem

Algorithm		T-LogisticBatDNN	SA	TS	GA	TSa	GaTSa+BP
	Mean	5.700014	33.43645	33.92932	32.40878	31.39817	25.09068
T-LogisticBatDNN	5.700014	-	**27.736**	**28.229**	**26.708**	**25.698**	**19.390**
SA	33.43645	-	-	0.4928	1.0276	2.0382	**8.3457**
TS	33.92932	-	-	-	1.5205	2.5311	**8.8386**
GA	32.40878	-	-	-	-	1.0106	7.3181
TSa	31.39817	-	-	-	-	-	6.3074
GaTSa+BP	25.09068	-	-	-	-	-	-

4.1.5 Friedman and post-hoc tests for prediction error of time series prediction

In the case of time series prediction, due to the lack of availability of results in the literature for the number of connections, we performed the Friedman test on prediction error (test data) only. The Friedman test result was equal to 12.8. This value is greater than the critical level (7.6) so we rejected the null hypothesis and continued our study by performing the Nemenyi post-hoc test. From Table 12 it can be seen that the T-LogisticBatDNN algorithm performed better than the others, where the MSD is equal to 4.315611.

Table 12. Nemenyi test of prediction error for time series prediction

Algorithm		T-LogisticBatDNN	gHFSPNN (triangular)	gHFSPNN (Gaussian)	gFSPNN T(triangular)	gFSPNN T(Gaussian)
	Mean	0.16638	5.77577	5.138145	5.659	5.17205
T-LogisticBatDNN	0.16638	-	**5.6093**	**4.9717**	**5.4926**	**5.0056**
gHFSPNN(triangular)	5.77577	-	-	0.6376	0.1167	0.6037
gHFSPNN(Gaussian)	5.138145	-	-	-	0.5208	0.0339
gFSPNN T(triangular)	5.659	-	-	-	-	0.4869
gFSPNN T(Gaussian)	5.17205	-	-	-	-	-

4.2 Time series prediction for real-world data

In the last part of our assessment, the best method (T-LogisticBatDNN) was applied to real-world rainfall data. Multi-step ahead prediction is the task of predicting a sequence of values in a time series. Rainfall time series data was gathered from the Institute of Climate Change, Universiti Kebangsaan Malaysia (UKM), Malaysia. Station1 is a daily rainfall data record from years 1975 to 2003 which was reported from 5R-Mardi Serdang-Sel station. Station2

is also a daily rainfall record, in this case from years 1989 to 2003, which was recorded from 8R-Ampang Semenyih-Sel station.

Table 13. Results of T-LogisticBatDNN for real-world rainfall data

Dataset	Criteria	T-LogisticBatDNN
Station1	Training error %	0.28494
	Std. Dev.	0.09832
	Testing error%	0.36604
	Std. Dev.	0.02747
	Connection%	41.2348
Station2	Training error %	0.33593
	Std. Dev.	0.04812
	Testing error%	0.28925
	Std. Dev.	0.01392
	Connection%	36.9378

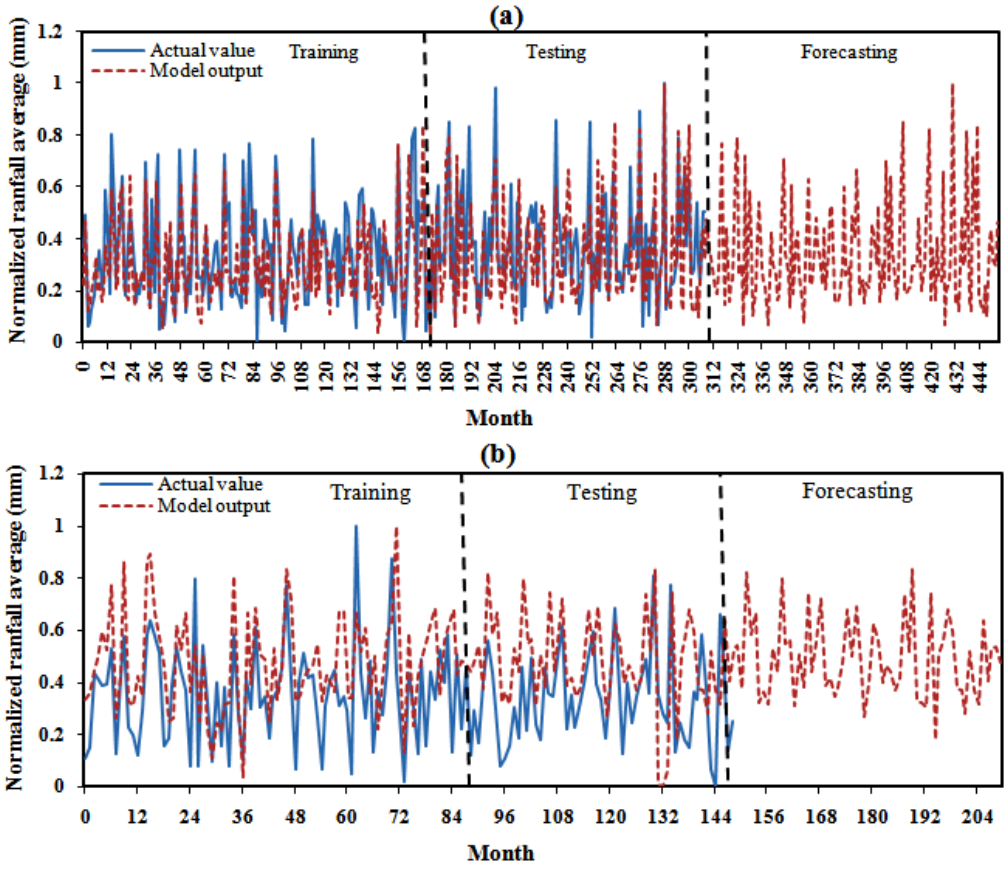


Fig. 9.T-LogisticBatDNN for real data forecasting: (a) station1 and (b) station2.

Monthly averages of the daily data were obtained. The first 24 corresponding averages were employed as the input for the ANN and the rest was divided into two parts; one part was used as a training set and the second as a testing set. The data were normalized into the range of [-1, 1]. The results for rainfall data using T-LogisticBatDNN

are shown in Table13. The details of these rainfall forecasts are given in Fig. 9. The model was used to predict three years of future rainfall changes based on previously observed values. This three years prediction as a sequence of values was performed to investigate the performance of the method for multistep-ahead prediction of real-world data. To achieve this goal we built a model by optimizing the fitness of the model for the training data with aid of T-LogisticBatDNN. Then the best model thus found was used to evaluate the predicted data using the test data. The forecasting part consists of the values predicted by the model for future rainfall changes. These three steps of the forecasting procedure are shown in Fig. 9. Both the actual value and the model output were normalized into the interval [0, 1] in this figure. It is clear that, although the rainfall changes have an irregular pattern, the performance of the model shows that the proposed method has good ability to predict real-world data.

5. Conclusion

This paper investigated the ability of the bat algorithm to undertake simultaneous optimization of the structure and weights of an ANN in order to allow the creation of a more accurate and less complex neural network model. To achieve this major aim, the basic bat algorithm was applied to optimize the proposed method. Then two modifications (MBatDNN and MeanBatDNN) of the bat algorithm were proposed in order to improve the exploration and exploitation strategy of the algorithm in the population. For enhancement of the balancing exploration and exploitation, three types of chaotic maps were then included in the more accurate version (MeanBatDNN) in order to generate a sequence map instead of a random sequence. The efficacy of these three chaotic map versions were compared with each other where the algorithms were applied to classification and time series prediction problems. With the aid of statistical tests the best version among the six proposed bat algorithms was identified, namely LogisticBatDNN. Then the Taguchi method was used to tune the parameters of LogisticBatDNN and the result showed superior performance. This subsequent version, namely T-LogisticBatDNN, was compared statistically with other methods in the literature. Based on our extensive evaluations it was concluded that the T-LogisticBatDNN algorithm has the ability to outperform two of the compared algorithms (SA and GA) in the case of classification error. It also showed better performance for the number of connections of the ANN which has higher achievement for all cases. Moreover, this algorithm was able to obtain more accurate model for time series prediction. Finally, the proposed method was applied to a real-world time series prediction problem (rainfall data) and it was found that the proposed method is able to predict the future instances for rainfall data accurately. These promising results motivate us to find ways to extend our method as future work. This extension could be in the form of the incorporation of other chaotic maps in the method. Another research direction is that the method could be used in solving other real-world problems such as regression and pattern recognition.

Acknowledgements

This work was supported by the Ministry of Education, Malaysia (ERGS/1/2013/ICT07/UKM/02/5) and the Universiti Kebangsaan Malaysia (DIP-2012-15).

References

[1] S.-H. Yang, Y.-P. Chen, An evolutionary constructive and pruning algorithm for artificial neural networks and its prediction applications, Neurocomputing, 86 (2012) 140-149.

[2] E. Cantu-Paz, C. Kamath, An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 35 (2005) 915-927.

[3] S. Singh, P. Bhambri, J. Gill, Time Series based Temperature Prediction using Back Propagation with Genetic Algorithm Technique, International Journal of Computer Science Issues, 8 (2011) 28-32.

[4] P.P. Palmes, T. Hayasaka, S. Usui, Mutation-based genetic neural network, IEEE Transactions on Neural Networks, 16 (2005) 587-600.

[5] A. Gepperth, S. Roth, Applications of multi-objective structure optimization, Neurocomputing, 69 (2006) 701-713.

[6] D.S. Huang, J.X. Du, A Constructive Hybrid Structure Optimization Methodology for Radial Basis Probabilistic Neural Networks, IEEE Transactions on Neural Networks, 19 (2008) 2099-2115.

[7] C. Hervás-Martínez, F.J. Martínez-Estudillo, M. Carbonero-Ruz, Multilogistic regression by means of evolutionary product-unit neural networks, Neural Networks, 21 (2008) 951-961.

[8] G. Chi-Keong, T. Eu-Jin, T. Kay Chen, Hybrid Multiobjective Evolutionary Design for Artificial Neural Networks, IEEE Transactions on Neural Networks, 19 (2008) 1531-1548.

[9] N.S. Jaddi, S. Abdullah, A.R. Hamdan, Taguchi-Based Parameter Designing of Genetic Algorithm for Artificial Neural Network Training, in: Informatics and Creative Multimedia (ICICM), 2013 International Conference on, IEEE, 2013, pp. 278-281.

[10] S. Abdullah, N.S. Jaddi, Great deluge algorithm for rough set attribute reduction, in: Database Theory and Application, Bio-Science and Bio-Technology, Springer, 2010, pp. 189-197.

[11] N.S. Jaddi, S. Abdullah, Nonlinear Great Deluge Algorithm for Rough Set Attribute Reduction, Journal of Information Science and Engineering 29 (2013) 49-62.

[12] N.S. Jaddi, S. Abdullah, Hybrid of genetic algorithm and great deluge for rough set attribute reduction, Turkish Journal of Electerical Engineering and Computer Sciences 21 (2013) 1737-1750.

[13] N.S. Jaddi, S. Abdullah, An Interactive Rough Set Attribute Reduction Using Great Deluge Algorithm, in: Advances in Visual Informatics, Springer, 2013, pp. 285-299.

[14] N.S. Jaddi, S. Abdullah, A.R. Hamdan, Multi-population cooperative bat algorithm-based optimization of artificial neural network model, Information Sciences, 294 (2015) 628-644.

[15] M.M. Islam, M.F. Amin, S. Ahmmed, K. Murase, An adaptive merging and growing algorithm for designing artificial neural networks, in: IEEE International Joint Conference on Neural Networks, 2008, pp. 2003-2008.

[16] M.M. Islam, M.A. Sattar, M.F. Amin, X. Yao, K. Murase, A new adaptive merging and growing algorithm for designing artificial neural networks, Trans. Sys. Man Cyber. Part B, 39 (2009) 705-722.

[17] A. Kaylani, M. Georgiopoulos, M. Mollaghasemi, G.C. Anagnostopoulos, AG-ART: An adaptive approach to evolving ART architectures, Neurocomputing, 72 (2009) 2079-2092.

[18] A. Kaylani, M. Georgiopoulos, M. Mollaghasemi, G.C. Anagnostopoulos, C. Sentelle, Z. Mingyu, An Adaptive Multiobjective Approach to Evolving ART Architectures, IEEE Transactions on Neural Networks, 21 (2010) 529-550.

[19] B. Curry, P.H. Morgan, Seasonality and neural networks: a new approach, Int. J. Metaheuristics, 1 (2010) 181-197.

[20] D. Mantzaris, G. Anastassopoulos, A. Adamopoulos, Genetic algorithm pruning of probabilistic neural networks in medical disease estimation, Neural Networks, 24 (2011) 831-835.

[21] S.-K. Oh, W. Pedrycz, B.-J. Park, Polynomial neural networks architecture: analysis and design, Computers & Electrical Engineering, 29 (2003) 703-725.

[22] S.-K. Oh, W. Pedrycz, A new approach to self-organizing fuzzy polynomial neural networks guided by genetic optimization, Physics Letters A, 345 (2005) 88-100.

[23] S.-K. Oh, W. Pedrycz, S.-B. Roh, Hybrid fuzzy set-based polynomial neural networks and their development with the aid of genetic optimization and information granulation, Applied Soft Computing, 9 (2009) 1068-1089.

[24] H.-S. Park, B.-J. Park, H.-K. Kim, a.S.-K. Oh, Self-Organizing Polynomial Neural Networks Based on Genetically Optimized Multi-Layer Perceptron Architecture, International Journal of Control, Automation, and Systems, 2 (2004) 423-434.

[25] T.A.S. Masutti, L.N. de Castro, Neuro-immune approach to solve routing problems, Neurocomputing, 72 (2009) 2189-2197.
[26] T.B. Ludermir, A. Yamazaki, C. Zanchettin, An Optimization Methodology for Neural Network Weights and Architectures, IEEE Transactions on Neural Networks, 17 (2006) 1452-1459.
[27] J. Yu, S. Wang, L. Xi, Evolving artificial neural networks using an improved PSO and DPSO, Neurocomputing, 71 (2008) 1054-1060.
[28] L. Zhao, Y. Yang, PSO-based single multiplicative neuron model for time series prediction, Expert Systems with Applications, 36 (2009) 2805-2812.
[29] C. Zanchettin, T.B. Ludermir, L.M. Almeida, Hybrid Training Method for MLP: Optimization of Architecture and Training, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 41 (2011) 1097-1109.
[30] X.-S. Yang A New Metaheuristic Bat-Inspired Algorithm, in: Nature Inspired Cooperative Strategies for Optimization (NISCO 2010), Springer 2010, pp. 65-74.
[31] P.W. Tsai, J.S. Pan, B.Y. Liao, M.J. Tsai, V. Istanda, Bat Algorithm Inspired Algorithm for Solving Numerical Optimization Problems, Applied Mechanics and Materials, 148 - 149 (2011) 134-137.
[32] X.-S. Yang Bat algorithm for multi-objective optimisation, Int. J. Bio-Inspired Comput., 3 (2011) 267-274.
[33] T.C. Bora, L.S. Coelho, L. Lebensztajn, Bat-Inspired Optimization Approach for the Brushless DC Wheel Motor Problem, Magnetics, IEEE Transactions on, 48 (2012) 947-950.
[34] K. Khan, A. Sahai, A Comparison of BA, GA, PSO, BP and LM for Training Feed forward Neural Networks in e-Learning Context, International Journal of Intelligent Systems and Applications(IJISA), 4 (2012) 23-29.
[35] P. Musikapun, P. Pongcharoen, Solving Multi-Stage Multi-Machine Multi-Product Scheduling Problem Using Bat Algorithm, in: 2nd International Conference on Management and Artificial Intelligence IACSIT Press, Singapore 2012.
[36] K. Khan, A. Nikov, A. Sahai, A Fuzzy Bat Clustering Method for Ergonomic Screening of Office Workplaces, in: Third International Conference on Software, Services and Semantic Technologies S3T 2011 Advances in Intelligent and Soft Computing, Springer-Verlag Berlin Heidelberg, 2011, pp. 59-66.
[37] J.W. Zhang, G.G. Wang, Image Matching Using a Bat Algorithm with Mutation, Applied Mechanics and Materials, 203 (2012) 88-93.
[38] A. Natarajan, S. Subramanian, K. Premalatha, A comparative study of cuckoo search and bat algorithm for Bloom filter optimisation in spam filtering, Int. J. Bio-Inspired Comput., 4 (2012) 89-99.
[39] G. Komarasamy, A. Wahi, An Optimized K-Means Clustering Technique using Bat Algorithm, European Journal of Scientific Research 84 (2012) 263-273.
[40] S. Mishra, K. Shaw, D. Mishra, A New Meta-heuristic Bat Inspired Classification Approach for Microarray Data, Procedia Technology, 4 (2012) 802-806.
[41] S. Talatahari, B. Farahmand Azar, R. Sheikholeslami, A.H. Gandomi, Imperialist competitive algorithm combined with chaos for global optimization, Communications in Nonlinear Science and Numerical Simulation, 17 (2012) 1312-1319.
[42] K. Chandramouli, E. Izquierdo, Image Classification using Chaotic Particle Swarm Optimization, in: Image Processing, 2006 IEEE International Conference on, 2006, pp. 3001-3004.
[43] R. Charrier, C. Bourjot, F. Charpillet, A deterministic metaheuristic approach using "logistic ants" for combinatorial optimization, in: Proceedings of the 7th international conference on Swarm intelligence, Springer-Verlag, Brussels, Belgium, 2010, pp. 344-351.
[44] H. Min, F. Jianchao, W. Jun, A Dynamic Feedforward Neural Network Based on Gaussian Particle Swarm Optimization and its Application for Predictive Control, IEEE Transactions on Neural Networks 22 (2011) 1457-1468.
[45] E. Araujo, L.d.S. Coelho, Particle swarm approaches using Lozi map chaotic sequences to fuzzy modelling of an experimental thermal-vacuum system, Applied Soft Computing, 8 (2008) 1354-1364.
[46] Q.-K. Pan, L. Wang, L. Gao, A chaotic harmony search algorithm for the flow shop scheduling problem with limited buffers, Appl. Soft Comput., 11 (2011) 5270-5280.
[47] R.Y.M. Nakamura, L.A.M. Pereira, K.A. Costa, D. Rodrigues, J.P. Papa, X.S. Yang, BBA: A Binary Bat Algorithm for Feature Selection, in: 2012 25th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), 2012, pp. 291-297.
[48] J. Kennedy, R. Eberhart, Particle swarm optimization, in: IEEE International Conference on Neural Networks 1995, pp. 1942-1948
[49] K. Deep, J.C. Bansal, Mean particle swarm optimisation for function optimisation, Int. J. Comput. Intell. Stud., 1 (2009) 72-92.

[50] C.L. Blake, C.J. Merz, UCI Repository of Machine Learning Databases in, University of California at Irvine, 1998.
[51] B. Karlik, A.V. Olgac, Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks International Journal of Artificial Intelligence and Expert Systems, 1 (2010) 111-122.
[52] S.-K. Oh, S.-B. Roh, W. Pedrycz, A new approach to genetically optimized hybrid fuzzy set-based polynomial neural networks with FSPNs and PNs, in: Proceedings of the Second international conference on Modeling Decisions for Artificial Intelligence, Springer-Verlag, Tsukuba, Japan, 2005, pp. 328-337.