Hindawi Publishing Corporation
Mathematical Problems in Engineering
Article ID 868375

Research Article

Weight Optimization in Recurrent Neural Networks with Hybrid Metaheuristic Cuckoo Search Techniques for Data Classification

Nazri Mohd Nawi,[1] Abdullah Khan,[1] M. Z. Rehman,[1] Haruna Chiroma,[2] and Tutut Herawan[3]

[1]*Software and Multimedia Centre, Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, Parit Raja, 86400 Batu Pahat, Johor, Malaysia*
[2]*Department of Artificial Intelligence, University of Malaya, 50603 Lembah Pantai, Kuala Lumpur, Malaysia*
[3]*Department of Information Systems, University of Malaya, 50603 Lembah Pantai, Kuala Lumpur, Malaysia*

Correspondence should be addressed to Tutut Herawan; tutut@um.edu.my

Received 16 September 2014; Accepted 15 November 2014

Academic Editor: Yudong Zhang

Recurrent neural network (RNN) has been widely used as a tool in the data classification. This network can be educated with gradient descent back propagation. However, traditional training algorithms have some drawbacks such as slow speed of convergence being not definite to find the global minimum of the error function since gradient descent may get stuck in local minima. As a solution, nature inspired metaheuristic algorithms provide derivative-free solution to optimize complex problems. This paper proposes a new metaheuristic search algorithm called Cuckoo Search (CS) based on Cuckoo bird's behavior to train Elman recurrent network (ERN) and back propagation Elman recurrent network (BPERN) in achieving fast convergence rate and to avoid local minima problem. The proposed CSERN and CSBPERN algorithms are compared with artificial bee colony using BP algorithm and other hybrid variants algorithms. Specifically, some selected benchmark classification problems are used. The simulation results show that the computational efficiency of ERN and BPERN training process is highly enhanced when coupled with the proposed hybrid method.

1. Introduction

Artificial neural network (ANN) is a well-known procedure that has the ability to classify nonlinear problem and is experimental in nature [1]. However it can give almost accurate solution for clearly or inaccurately formulated problems and for phenomena that are only understood during experiment. An alternate neural network approach is to use recurrent neural networks (RNN), which have inside feedback loops within network allowing them to store previous memory to train past history [2–5]. The RNN model has been implemented in various applications, such as forecasting of financial data [6], electric power demand [7], tracking water quality and minimizing the additives needed for filtering water [7], and data classification [8]. In order to understand the advantage of the dynamical processing of recurrent neural networks, researchers have developed an amount of schemes by which gradient methods and, in particular, back propagation learning can be extended to recurrent neural networks [8].

Werbos [9] introduced the back propagation through time approach approximating the time evolution of a recurrent neural network as a sequence of static networks using gradient methods. The simple recurrent networks were first trained by Elman with the standard back propagation (BP) learning algorithm, in which errors are calculated and weights are updated at each time step. The BP is not as effective as the back propagation through time (BPTT) learning algorithm, in which error signal is propagated back through time [10]. However, certain properties of the RNN make many of the algorithms less capable, and it often takes huge amount of time to train a network of even a moderate size. In addition, the complex error surface of the RNN network makes many training algorithms more flat to being trapped in local minima [5]. The back propagation (BP) algorithm is the well-known method for training network. Otherwise, the BP algorithm suffers from two main drawbacks, that is, low convergence rate and instability. They are caused by a possibility of being trapped in a local minimum and view of overshooting the minimum of the error surface [11–17].

To overcome the weaknesses of the above algorithms, there have been many researches on dynamic system modeling with recurrent neural network. This ability of dynamic modeling system formulate a kind of neural network that is more superior to the conventional feed forward neural networks because the system outputs are function of both the current inputs as well as their inner states [18, 19]. Ahmad et al. in [20] investigated a new method using fully connected recurrent neural network (FCRNN) and back propagation through time (BPTT) algorithm to observe the difference of Arabic alphabetic like "*alif*" to "*ya*." The algorithm is also used to improve the people's knowledge and understanding of Arabic words using the proposed technique. In 2010, Xiao, Venayagamoorthy, and Corzine trained recurrent neural network integrated with particle swarm optimization (PSO) and BP algorithm (PSOBP) to provide the optimal weights to avoid local minima problem and also to identify the frequency dependent impedance of power electronic system such as rectifiers, inverter, and AC-DC conversion [5]. The experimental results described that the proposed method successfully identified the impedance characteristic of the three-phase inverter system, which not only can systematically help avoiding the training process being trapped in local minima, but also has better performance as compared to both sample BP and PSO algorithms. Similarly, Zhang and Wu [21] used adaptive chaotic particle swarm optimization (ACPSO) algorithm for classification of crops from synthetic aperture radar (SAR) images. During simulations, ACPSO was found to be superior to the back propagation (BP), adaptive BP (ABP), momentum back propagation (MBP), particle swarm optimization (PSO), and resilient back propagation (RPROP) methods [21]. Aziz et al. [22] carried out a study on the performance of particle swarm optimization algorithm with training Elman RNN to discover the classification accuracy and convergence rate compared with Elman recurrent network with BP algorithm. Based on the simulated result it is illustrated that the proposed Elman recurrent network particle swarm optimization (ERNPSO) algorithm is better than the back propagation Elman recurrent network (BPERN) in terms of classification accuracy. However in terms of convergence time the BPERN is much better than the proposed ERNPSO algorithm.

Cheng and Shen [23] proposed an improved Elman RNN to calculate radio propagation loss, with three-dimensional parabola equation method in order to decrease calculation time and to improve approximation performance of the network. Based on results the proposed improved Elman networks show an efficient and feasible performance to predict propagation loss compared with the simple Elman RNN. However, the Elman RNN loses necessary significant data to train the network for predicating propagation. Wang et al. [24] used the Elman RNN to compute the total nitrogen (TN), total phosphorus (TP), and dissolved oxygen (DO) at three different sites of Taihu during the period of water diversion. The conceptual form of the Elman RNN for different parameters was used by means of the principle component analysis (PCA) and validated on water quality diversion dataset. The values of TS, TP, and DO calculated by the model were intimately related to their respective values. The simulated result shows that the PCA can efficiently accelerate the input parameters for the Elman RNN and can precisely compute and forecast the water quality parameter during the period of water diversion, but not free of local minim problem.

In [25], the proposed LM algorithm based on Elman and Jordan recurrent neural network has been used to forecast annual peak load of Java, Madura, and Bali interconnection for 2009–2011. The study is carried out to check the performance of the proposed LM based recurrent network with respect to their forecasting accuracy over the given period. From the simulation results, it is clear that the proposed LM based recurrent neural network has better performance than the LM based feed forward neural network. After reviewing the above algorithms, it is found that the traditional ANN training has a drawback, that is, slow speed of convergence, which is not definite to find the global minimum of the error function since gradient descent may get stuck in local minima.

To overcome the weaknesses and to improve the convergence rate, this work proposes new hybrid metaheuristic search algorithms that make use of the Cuckoo Search via Levy flight (CS) by Yang and Deb [26] to train Elman recurrent network (ERN) and back propagation Elman recurrent network (BPERN). The main goals of this study are to improve convergence to global minima, decrease the error, and accelerate the learning process using a hybridization method. The proposed algorithms called CSERN and CSBPERN imitate animal behaviour and are valuable for global optimization [27, 28]. The performance of the proposed CSERN and CSBPERN is verified on selected benchmark classification problems and compared with artificial bee colony using BPNN algorithm and other similar hybrid variants.

The remainder of the paper is organized as follows. Section 2 describes the proposed method. Result and discussion are explained in Section 3. Finally, the paper is concluded in Section 4.

2. Proposed Algorithms

In this section, we describe our proposed Cuckoo Search (CS) to train Elman recurrent network (ERN) and back propagation Elman recurrent network (BPERN).

2.1. CSERN Algorithm. In the proposed CSERN algorithm, each best nest represents a possible solution, that is, the weight space and the corresponding biases for ERN optimization. The weight optimization problem and the size of a population represent the quality of the solution. In the first epoch, the weights and biases are initialized with CS and then those weights are passed on to the ERN. The weights in ERN are calculated. In the next cycle, CS will update the weights with the best possible solution and CS will continue searching the best weights until either the last cycle/epoch of the network is reached or the MSE is achieved. The CS is a population based optimization algorithm; it starts with a random initial population. In the proposed CSERN algorithm, the weight space and the corresponding biases for ERN optimization are calculated by the weight matrices given in (1) and (2) as follows:

$$W_n = U_n = \sum_{n=1}^{N} a \cdot \left(\text{rand} - \frac{1}{2}\right), \tag{1}$$

$$B_n = \sum_{n=1}^{N} a \cdot \left(\text{rand} - \frac{1}{2}\right), \tag{2}$$

where $W_n = N$th weight value in a weight matrix. The rand in (1) is the random number in the range $[0, 1]$, where a is any constant parameter for the proposed method and it is less than 1, and B_n is a bias value. Hence, the list of weights matrix is given as follows:

$$W^c = \left[W_n^{\ 1}, W_n^{\ 2}, W_n^{\ 3}, \ldots, W_n^{\ N-1}\right]. \tag{3}$$

Now from neural network process sum of square errors is easily planned for every weight matrix in W^c. For the ERN structure three layers' network with one input layer, one hidden or "state" layer, and one "output" layer are used. Each layer will have its own index variable, that is, k for output nodes, j and l for hidden nodes, and i for input nodes. In a feed forward network, the input vector x is propagated through a weight layer and

$$\text{net}_j(t) = \sum_{i}^{n} x_i(t)\, w_{n(ji)} + B_{n(j)}, \tag{4}$$

where n is the number of inputs and $B_{n(j)}$ is a bias.

In a simple recurrent network, the input vector is similarly propagated through a weight layer but also combined with the previous state activation through an additional recurrent weight layer, U, and

$$\begin{gathered} y_j(t) = f\left(\text{net}_j(t)\right), \\ \text{net}_j(t) = \sum_{i}^{n} x_i(t)\, W_{n(ji)} + \sum_{l}^{m} y_l(t-1)\, U_{n(jl)} + B_{n(j)}, \\ y_j(t) = f\left(\text{net}_j(t)\right), \end{gathered} \tag{5}$$

where m is the number of "state" nodes. The output of the network is in both cases determined by the state and a set of output weights W and

$$\begin{gathered} \text{net}_k(t) = \sum_{j}^{M} y_j(t)\, W_{n(k\ j)} + B_{n(k)}, \\ Y_k(t) = g\left(\text{net}_k(t)\right), \end{gathered} \tag{6}$$

where g is an output function. Hence, the error can be calculated as follows:

$$E = \left(T_k - Y_k\right). \tag{7}$$

The performances index for the network is given as follows:

$$\begin{gathered} V(x) = \frac{1}{2}\sum_{k=1}^{K} \left(T_k - X_k\right)^T \left(T_k - Y_k\right), \\ V_F(x) = \frac{1}{2}\sum_{k=1}^{K} E^T \cdot E. \end{gathered} \tag{8}$$

In the proposed method the average sum of squares is the performance index and it is calculated as follows:

$$V_\mu(x) = \frac{\sum_{j=1}^{N} V_F(x)}{P_i}, \tag{9}$$

where y_r is the output of the network when the kth input net$_i$ is presented. The equation $E = (T_k - Y_k)$ is the error for the output layer, $V_\mu(x)$ is the average performance, $V_F(x)$ is the performance index, and P_i is the number of Cuckoo populations in ith iteration. At the end of each epoch the list of average sums of square errors of ith iteration SSE can be calculated as follows:

$$\text{SSE}_i = \left\{V_\mu^{\ 1}(x), V_\mu^{\ 2}(x), V_\mu^{\ 3}(x), \ldots, V_\mu^{\ n}(x)\right\}. \tag{10}$$

The Cuckoo Search is replicating the minimum sum of square error (MSE). The MSE is found when all the inputs are processed for each population of the Cuckoo nest. Hence, the Cuckoo Search nest x_j is calculated as follows:

$$x_j = \text{Min}\left\{V_\mu^{\ 1}(x), V_\mu^{\ 2}(x), V_\mu^{\ 3}(x), \ldots, V_\mu^{\ n}(x)\right\}. \tag{11}$$

The rest of the average sum of squares is considered as other Cuckoo nests. A new solution x_i^{t+1} for Cuckoo i is generated using a Levy flight according to the following equation:

$$x_i^{t+1} = x_i^{\ t} + \alpha \oplus \text{levy}(\lambda). \tag{12}$$

Hence, the movement of the other Cuckoo x_i toward x_j can be drawn from (13) as follows:

$$X = \begin{cases} x_i + \text{rand} \cdot \left(x_j - x_i\right) & \text{rand}_i > p_\alpha \\ x_i & \text{else.} \end{cases} \tag{13}$$

The Cuckoo Search can move from x_i toward x_j through Levy flight; it can be written as

$$\nabla X_i = \begin{cases} x_i \oplus \alpha\text{levy}(\lambda) \sim 0.01 \cdot \left(\dfrac{U_j}{\left|V_j\right|^{1/\mu}}\right) \cdot \left(X - X_{\text{best}}\right) & \text{rand}_i > p_\alpha \\ x_i & \text{else,} \end{cases} \tag{14}$$

where ∇X_i is a small movement of x_i toward x_j. The weights and bias for each layer are then adjusted as follows:

$$W_n^{s+1} = U_n^{s+1} = W_n^{s} - \nabla X_i,$$
$$B_n^{s+1} = B_n^{s} - \nabla X_i. \quad (15)$$

The pseudocode for CSERN algorithm is given in Pseudocode 1.

2.2. CSBPERN Algorithm. In the proposed CSBPERN algorithm, each best nest represents a possible solution, that is, the weight space and the corresponding biases for BPERN optimization. The weight optimization problem and the size of the solution represent the quality of the solution. In the first epoch, the best weights and biases are initialized with CS and then those weights are passed on to the BPERN. The weights in BPERN are calculated. In the next cycle CS will update the weights with the best possible solution, and CS will continue searching the best weights until either the last cycle/epoch of the network is reached or the MSE is achieved.

The CS is a population based optimization algorithm, and like other metaheuristic algorithms it starts with a random initial population. In the proposed CSBPERN algorithm, each best nest represents a possible solution, that is, the weight space and the corresponding biases for BPERN optimization. The weight optimization problem and the size of a nest represent the quality of the solution. In the first epoch, the best weights and biases are initialized with CS and then those weights are passed on to the BPERN. The weights in BPERN are calculated. In the next cycle CS will update the weights with the best possible solution and CS will continue searching the best weights until either the last cycle/epoch of the network is reached or the MSE is achieved.

In CSBPERN, the weight value of a matrix is calculated with (1) and (2) as given in Section 2.1. Also, the weight matrix is updated with (3). Now from neural network process sum of square errors (SSE) is easily planned for every weight matrix in W^c. For the BPERN structure three layers' network with one input layer, one hidden or "state" layer, and one "output" layer are used. In CSBPERN network, the input vector x is propagated through a weight layer W using (4). In a simple recurrent network, the input vector is not only similarly propagated through a weight layer, but also combined with the previous state activation through an additional recurrentweight layer U, as given in (5). The output of the network in both cases is determined by the state and a set of output weights W, as given in (6).

According to gradient descent, each weight change in the network should be proportional to the negative gradient of the cost with respect to the specific weights as given in

$$\delta_{pk} = -\eta \frac{\delta E}{\delta y_{pk}}. \quad (16)$$

Thus, the error for output nodes is calculated as follows:

$$\delta_{pk} = \left(T_{pk} - Y_{pk}\right) Y_{pk} \left(1 - Y_{pk}\right), \quad (17)$$

and for the hidden nodes the error is given as follows:

$$\delta_{pj} = \sum_{k}^{m} \delta_{pk} W_{n(kj)} f'\left(Y_{pj}\right). \quad (18)$$

Thus the weights and bias are simply changed for the output layer as

$$\nabla W^{n+1}{}_{(kj)} = \eta \sum_{p}^{n} \delta_{pk} y_{pj},$$
$$\nabla B^{n+1}{}_{(kj)} = \eta \sum_{p}^{n} \delta_{pk} y_{pj}, \quad (19)$$

and for the input layer the weight change is given as

$$\nabla W^{n+1}{}_{(ji)} = \eta \sum_{p}^{n} \delta_{pj} x_{pi},$$
$$\nabla B^{n+1}{}_{(ji)} = \eta \sum_{p}^{n} \delta_{pj} x_{pi}. \quad (20)$$

Adding a time subscript, the recurrent weights can be modified according to (21) as follows:

$$\nabla U^{n+1}{}_{(jh)} = \nabla U^{n}{}_{(jh)} + \eta \sum_{p}^{n} \delta_{pj}(t)\, y_{ph}(t-1). \quad (21)$$

The network error is calculated for CSBPERN using (7) from Section 2.1. The performance indices for the network are measured with (8) and (9). At the end of each epoch the list of average sums of square errors of ith iteration SSE can be calculated with (10). The Cuckoo Search is imitating the minimum SSE, which is found when all the inputs are processed for each population of the Cuckoo nest. Hence, the Cuckoo Search nest x_j is calculated using (11). A new solution x_i^{t+1} for Cuckoo i is generated using a Levy flight according to (12). The movement of the other Cuckoo x_i toward x_j is controlled through (13). The Cuckoo Search can move from x_i toward x_j through Levy flight as written in (14). The weights and bias for each layer are then adjusted with (15).

The pseudocode for CSBPERN algorithm is given in Pseudocode 2.

3. Result and Discussion

3.1. Datasets. This study focuses on two criteria for the performances analysis: (a) to get less mean square error (MSE) and (b) to achieve high average classification accuracy on testing data from the benchmark problem. The benchmark datasets were used to validate the accuracy of the proposed algorithms taken from UCI Machine Learning Repository. For the experimentation purpose, the data has to be arranged into training and testing datasets; the algorithms are trained on training set, and their performance accuracy is calculated on the corresponding test set. The workstation used for carrying out the experimentation comes equipped with

```
(1) Initializes CS population size dimension and ERN structure
(2) Load the training data
(3) While MSE < stopping criteria
(4) Pass the Cuckoo nests as weights to network
(5) Feed forward network runs using the weights initialized with CS
(6) Calculate the error using (7)
(7) Minimize the error by adjusting network parameter using CS
(8) Generate Cuckoo egg ($x_j$) by taking Levy flight from random nest
                              $x_i = x_j$
(9) Abandon a fraction $p_\alpha \in [0, 1]$ of the worst nest. Build new nest at new location via Levy flight to replace the old one
(10) Evaluate the fitness of the nest, Chose a random nest $i$
        If
        (a) $X_j > X_i$   Then
        (b) $x_i \leftarrow x_j$
        (c) $X_i \leftarrow X_j$
        End if
(11) CS keeps on calculating the best possible weight at each epoch until the network is converged.
End While
```

Pseudocode 1: Pseudocode of CSERN algorithm.

```
(1) Initializes CS population size dimension and BPERN structure
(2) Load the training data
(3) While MSE < stopping criteria
(4) Pass the Cuckoo nests as weights to network
(5) Feed forward network runs using the weights initialized with CS
(6) The sensitivity of one layer is calculated from its previous one and the calculation of the sensitivity start from the last
    layer of the network and move backward using (17) and (18).
(7) Update weights and bias using (19) to (20)
(8) Calculate the error using (7)
(9) Minimize the error by adjusting network parameter using CS.
(10) Generate Cuckoo egg ($x_j$) by taking Levy flight from random nest.
                              $x_i = x_j$
(11) Abandon a fraction $p_\alpha \in [0, 1]$ of the worst nest. Build new nest at new location via Levy flight to replace the old one.
(12) Evaluate the fitness of the nest, Chose a random nest $i$
    If
    (a) $X_j > X_i$   Then
    (b) $x_i \leftarrow x_j$
    (c) $X_i \leftarrow X_j$
    End if
(13) CS keeps on calculating the best possible weight at each epoch until the network is converged.
End While
```

Pseudocode 2: Pseudocode of CSBPRNN algorithm.

2 GHz processor, 2-GB of RAM, while the operating system used is Microsoft XP (Service Pack 3). Matlab version R2010a software was used to carry out simulation of the proposed algorithms. For performing simulation, seven classification problems, that is, Thyroid Disease [29], Breast Cancer [30], IRIS [31], Glass [32], Australia Credit Card Approval [33], Pima Indian Diabetes [34], and 7-Bit Parity [35, 36] datasets, are selected. The following algorithms are analyzed and simulated on these problems:

(a) Conventional back propagation neural network (BPNN) algorithm.

(b) Artificial bee colony back propagation (ABC-BP) algorithm.

(c) Artificial bee colony neural network (ABCNN) algorithm.

(d) Artificial bee colony Levenberg Marquardt (ABC-LM) algorithm.

(e) Cuckoo Search recurrent Elman network (CSERN) algorithm.

(f) Cuckoo Search back propagation Elman recurrent network (CSBPERN) algorithm.

To compare the performance of proposed algorithms such as CSERN and CSBPERN with conventional BPNN, ABC-BP, and ABC-LM, the network parameters such as number of hidden layers, node in the hidden layer, the value for the weight initialization, and value of learning rate are used similarly. Three layers' NN is used for training and testing of the model. For all problems the NN structure has single hidden layer consisting of five nodes while the input and output layers nodes vary according to the data given. From the input layer to hidden layer and from hidden to output layer log-sigmoid activation function is used as the transform function.

Although the simple Elman neural network (SENN) used the pure line as the activation function for the output layer, learning rate of 0.4 is selected for the entire test. All algorithms were tested using the initial weights and biases are randomly initialized in range [0, 1]; for each problem, one trial is limited to 1000 epochs. A total of 20 trials are run for each dataset to validate these algorithms. For each trial the network results are stored in the result file. Mean square error (MSE), standard deviation of error mean square (SD), the number of epochs, and the average accuracy are recorded in separate file for each trial for selected classification problem.

Table 1: Summary of algorithms performance for Wisconsin Breast Cancer Classification Problem.

Algorithms	Accuracy	MSE	SD
ABC-BP	92.02	0.184	0.459
ABC-LM	93.83	0.0139	0.0010
ABCNN	88.96	0.014	0.0002
BPNN	90.71	0.271	0.017
CSERN	**99.95**	**3.23E − 05**	**2.9E − 05**
CSBPERN	**97.37**	**0.00072**	**0.0004**

Table 2: Summary of algorithms performance for Iris Classification Problem.

Algorithms	Accuracy	MSE	SD
ABC-BP	86.87	0.155	0.022
ABC-LM	79.55	0.058	0.0057
ABCNN	80.23	0.048	0.004
BPNN	87.19	0.311	0.022
CSERN	**99.97**	**5.1E − 06**	**3.2E − 06**
CSBPERN	**87.31**	**0.022**	**0.006**

3.2. Wisconsin Breast Cancer Classification Problem. The Breast Cancer dataset was created by William H. Wolberg. This dataset deals with the breast tumor tissue samples collected from different patients. The cancer analysis are performed to classify the tumor as benign or malignant. This dataset consists of 9 inputs and 2 outputs with 699 instances. The input attributes are, for instance, the clump thickness, the uniformity of cell size, the uniformity of cell shape, the amount of marginal adhesion, the single epithelial cell size, frequency of bare nuclei, bland chromatin, normal nucleoli, and mitoses. The selected network architecture used for the Breast Cancer Classification Problem consists of 9 input nodes, 5 hidden nodes, and 2 output nodes.

Table 1 illustrates that the proposed CSERN and CSBPERN algorithms show better performance than BPNN, ABC-BP, and ABC-LM algorithms. The proposed algorithms achieve small MSE ($3.23E-05$, 0.00072) and SD ($2.9E-05$, 0.0004) with 99.95 and 97.37 percent accuracy, respectively. Meanwhile, the other algorithms such as BPNN, ABC-BP, and ABC-LM fall behind the proposed algorithms with large MSE (0.271, 0.014, 0.184, and 0.013) and SD (0.017, 0.0002, 0.459, and 0.001) and lower accuracy. Similarly, Figure 1 shows the performances of MSE convergence for the used algorithms. From the simulation results, it can be easier to understand that the proposed algorithms show better performance than the BPNN, ABC-BP, and ABC-LM algorithms in terms of MSE, SD, and accuracy.

3.3. IRIS Classification Problem. The Iris flower multivariate dataset was introduced by Fisher to demonstrate the discriminant analysis in pattern recognition and machine learning to find a linear feature sets that either merge or separates two or more classes in the classification process. This is maybe the best famous database to be found in the pattern recognition literature. There were 150 instances, 4 inputs, and 3 outputs in this dataset. The classification of Iris dataset involves the data of petal width, petal length, sepal length, and sepal width into three classes of species, which consist of *Iris setosa*, *Iris versicolor*, and *Iris virginica*. The selected network structure for Iris classification dataset is 4-5-3, which consists of 4 input nodes, 5 hidden nodes, and 3 output nodes. In total 75 instances are used for training dataset and the rest for testing dataset.

Table 2 shows the comparison between performances of the proposed CSERN and CSBPERN algorithms with the BPNN, ABCNN, ABC-BP, and ABC-LM algorithms in terms of MSE, SD, and accuracy. From Table 2 it is clear that the proposed algorithms have better performances by achieving less MSE and SD and higher accuracy than that of the BPNN, ABCNN, ABC-BP, and ABC-LM algorithms. Figure 2 illustrates the MSE convergences performances of the algorithms. From Figure 2, it is clear that the proposed algorithms show higher performances than the other algorithms in terms of MSE, SD, and accuracy.

3.4. Thyroid Classification Problem. This dataset is taken for UCI Learning Repository, created based on the "Thyroid Disease" problem. This dataset consists of 21 inputs, 3 outputs, and 7200 patterns. Each case contains 21 attributes, which can be allocated to any of the three classes, which were hyper-, hypo-, and normal function of thyroid gland, based on the patient query data and patient examination data. The selected network architecture for Thyroid classification dataset is 21-5-3, which consists of 21 input nodes, 5 hidden nodes, and 3 output nodes.

Table 3 summarizes the comparison of performance of the all algorithms in terms of MSE, SD, and accuracy. From the table, it is easy to understand that the proposed CSERN and CSBPERN algorithms have small MSE and SD and

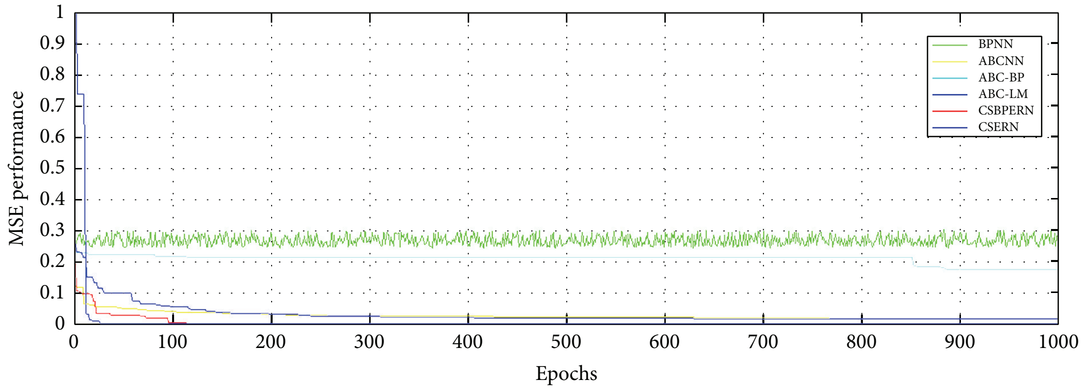

Figure 1: MSE via epochs convergence for Wisconsin Breast Cancer Classification Problem.

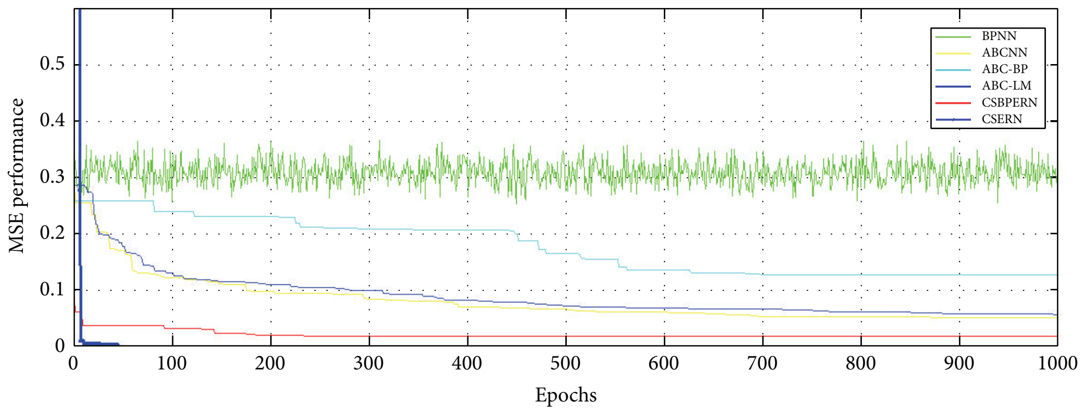

Figure 2: MSE via epochs convergence on Iris Benchmark Classification Problem.

Table 3: Summary of algorithms performance for Thyroid Classification Problem.

Algorithms	Accuracy	MSE	SD
ABC-BP	93.28	0.046	0.0006
ABC-LM	91.66	0.0409	0.0007
ABCNN	88.18	0.050	0.00064
BPNN	85.88	0.311	0.033
CSERN	**99.97**	**6.6E − 06**	**2.45E − 06**
CSBPERN	**95.008**	**0.0042**	**0.0086**

high accuracy, while the BPNN, BACNN, ABC-BP, and ABC-LM still have large MSE and SD with low accuracy. Figure 3 also shows the MSE convergence performances of the proposed and compared algorithms. From the simulation results, it is realized that the proposed algorithms have better performances in terms of MSE, SD, and accuracy than that of the other compared algorithms.

3.5. Diabetes Classification Problem. This dataset consists of 768 examples, 8 inputs, and 2 outputs and consists of all the information of the chemical change in a female body whose disparity can cause diabetes. The feed forward network topology for this network is set to 8-5-2. The target error for the Diabetes Classification Problem is set to 0.00001 and the maximum number of epochs is 1000. It is evident from Table 4 that the proposed CSERN and CSBPERN algorithms show better performance than the BPNN, ABCNN, ABC-BP, and ABC-LM algorithms in terms of MSE, SD, and accuracy. From Table 4, it is clear that the proposed algorithms have MSE of $1.7E - 05$, 0.039, and SD of $2.05E - 05$, 0.003, and achieved 99.96, 89.53 percent of accuracy. Meanwhile, the other algorithms such as BPNN, ABCNN, ABC-BP, and ABC-LM have MSE of 0.26, 0.131, 0.2, and 0.14, SD of 0.026, 0.021, 0.002, and 0.033, and accuracy of 86.96, 68.09, 88.16, and 56.09 percent, which is quite lower than the proposed algorithms. Figure 4 describes the MSE convergence performance of the used algorithms for Diabetes Classification Problem.

3.6. Glass Classification Problem. The Glass dataset is used for separating glass splinters in criminal investigation into six classes taken from UCI Repository or Machine Learning database which consists of float processed or non-float processed building windows, vehicle windows, containers, tableware, or head lamp. This dataset is made of 9 inputs and 6 outputs which consist of 214 examples. The selected feed forward network architecture for this network is set to 9-5-6.

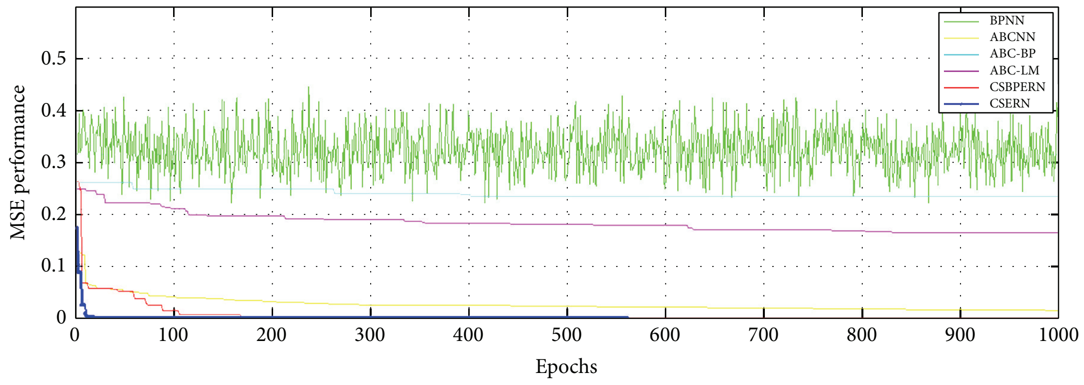

Figure 3: MSE via epochs convergence for Thyroid Benchmark Classification Problem.

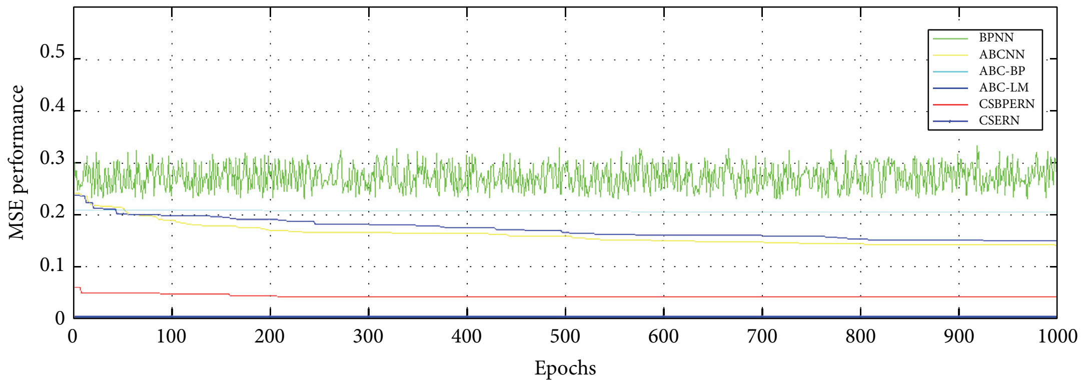

Figure 4: MSE via epochs convergence for Diabetes Classification Problem.

Table 4: Summary of algorithms performance for Diabetes Classification Problem.

Algorithms	Accuracy	MSE	SD
ABC-BP	88.16	0.20	0.0023
ABC-LM	65.09	0.14	0.033
ABCNN	68.09	0.131	0.021
BPNN	86.96	0.26	0.026
CSERN	**99.96**	**1.79E – 05**	**2.05E – 05**
CSBPERN	**89.53**	**0.039**	**0.003**

Table 5: Summary of algorithms performance for Glass Classification Problem.

Algorithms	Accuracy	MSE	SD
ABC-BP	94.09	0.025	0.009
ABC-LM	93.96	0.005	0.002
ABCNN	91.93	$1.8E - 03$	0.003
BPNN	94.04	0.36	0.048
CSERN	**99.96**	**2.2E – 05**	**2.5E – 05**
CSBPERN	**97.81**	**0.0005**	**0.0002**

Table 5 summarises the comparison performances of the algorithms. From the table it is clear to understand that the proposed algorithms outperform the other algorithms. The proposed CSERN and CSBPERN algorithms achieve small MSE of $2.20E - 05$, 0.0005, SD of $2.50E - 05$, 0.0002, and high accuracy of 99.96 and 97.81 percent. Meanwhile, the BPNN, ABCNN, ABC-BP, and ABC-LM algorithms have large MSE of 0.36, $1.80E - 03$, 0.025, and 0.005, SD of 0.048, 0.003, 0.002, and 0.009, and accuracy of 94.04, 91.93, 94.09, and 93.96 percent, which is quite lower than the proposed algorithms. Figure 5 shows the convergence performance of the algorithms for MSE via epochs. From the overall results, it is clear that the proposed algorithms have better performances than the other compared algorithms in case of MSE, SD, and accuracy.

3.7. Australian Credit Card Approval Classification Problem. This dataset is taken from UCI Machine Learning Repository, which contains all the details on the subject of card and application. The Australian Credit Card dataset consists of 690 instances, 51 inputs, and 2 outputs. Each example in this dataset represented a real detail about credit card application, whether the bank or similar institute generated the credit card or not. All attributes names and value have been changed to meaningless symbols to defend the privacy of the data. The selected architecture of NN is 51-5-2.

Table 6 gives the detailed result of the proposed algorithms with the compared algorithms which shows that

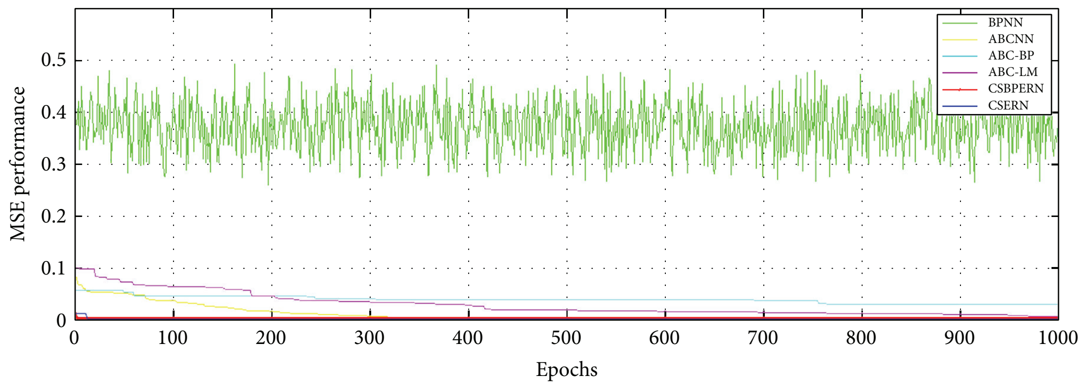

FIGURE 5: MSE via epochs convergence for Glass Benchmark Classification Problem.

TABLE 6: Summary of algorithms performance for Australian Credit Card Approval Classification Problem.

Algorithms	Accuracy	MSE	SD
ABC-BP	89.99	0.17	0.04
ABC-LM	77.78	0.055	0.005
ABCNN	76.79	0.13	0.012
BPNN	88.89	0.271	0.015
CSERN	**99.92**	**2.15E − 05**	**2.5E − 05**
CSBPERN	**85.75**	**0.021**	**0.0091**

TABLE 7: Summary of algorithms performance for 7-Bit Parity Classification Problem.

Algorithms	Accuracy	MSE	SD
ABC-BP	82.12	0.12	0.008
ABC-LM	69.13	0.08	0.012
ABCNN	67.85	0.10	0.015
BPNN	85.12	0.26	0.014
CSERN	**99.98**	**2.31E − 06**	**2.6E − 06**
CSBPERN	**89.28**	**0.052**	**0.005**

the proposed CSERN and CSBPERN algorithms achieve high accuracy of 99.92, 85.75, with MSE of $2.15E - 05$, 0.021, and SD of $2.5E - 05$, 0.0091. Meanwhile, the other algorithms, that is, BPNN, ABCNN, ABC-BP, and ABC-LM, have accuracy of 88.89, 76.79, 77.78, and 89.99, SD of 0.015, 0.012, 0.005, and 0.04, and MSE of 0.271, 0.13, 0.055, and 0.17, which is quite larger than the proposed algorithms. Similarly, Figure 6 shows the MSE convergence performances of the algorithms for the Australian Credit Card Approval Classification Problem. From these figures it can be easy to understand that the proposed algorithms have better result than that of the other compared algorithms.

3.8. Seven-Bit Parity Classification Problem. The parity problem is one of the most popular initial testing tasks and is a very demanding classification problem for neural network. In parity problem if given input vectors contain an odd number of one, the corresponding target value is 1; otherwise the target value is 0. The N-bit parity training set consists of $2N$ training pairs, with each training pair comprising an N-length input vector and a single binary target value. The $2N$ input vector represents all possible combinations of the N binary numbers. The selected architecture of NN is 7-5-1. Table 7 gives the detailed summary of the algorithms in terms of MSE, SD, and accuracy. From the table, it is clear that the proposed CSERN and CSBPERN algorithms have better performance than BPNN, ABCNN, ABC-BP, and ABC-LM algorithms in terms of MSE, SD, and accuracy. The proposed algorithms have MSE of $2.3E - 06$, 0.052, and SD of $2.6E - 06$, 0.005, and achieve 99.98 and 89.28 percent of accuracy. Meanwhile, the other BPNN, ABCNN, ABC-BP, and ABC-LM algorithms converge with MSE of 0.26, 0.10, 0.12, and 0.08, SD of 0.014, 0.015, 0.008, and 0.012, and 85.12, 67.85, 82.12, and 69.13 percent of accuracy, which is quite lower than that of the proposed algorithms. Finally, Figure 7 represents the MSE convergence performance of the algorithms for the 7-Bit Parity Classification Problem.

4. Conclusion

This paper has studied the data classification problem using the dynamic behavior of RNN trained by nature inspired metaheuristic Cuckoo Search algorithm which provides derivative-free solution to optimize complex problems. This paper has also proposed a new metaheuristic Cuckoo Search based on ERN and BPERN algorithms in order to achieve fast convergence rate and to avoid local minima problem in conventional RNN. The proposed algorithms called CSERN and CSBPERN are unlike the existing algorithms; CSERN and CSBPERN imitate animal behaviour and are valuable for global convergence. The convergence behaviour and performance of the proposed CSERN and CSBPERN are simulated on some selected benchmark classification problems. Specifically, 7-Bit Parity and some selected UCI benchmark classification datasets are used for training and testing the network. The performances of the proposed models are compared with artificial bee colony using BPNN algorithm and other hybrid variants. The simulation results show that the proposed CSERN and BPERN algorithms are far better than the baseline algorithms in terms of convergence rate.

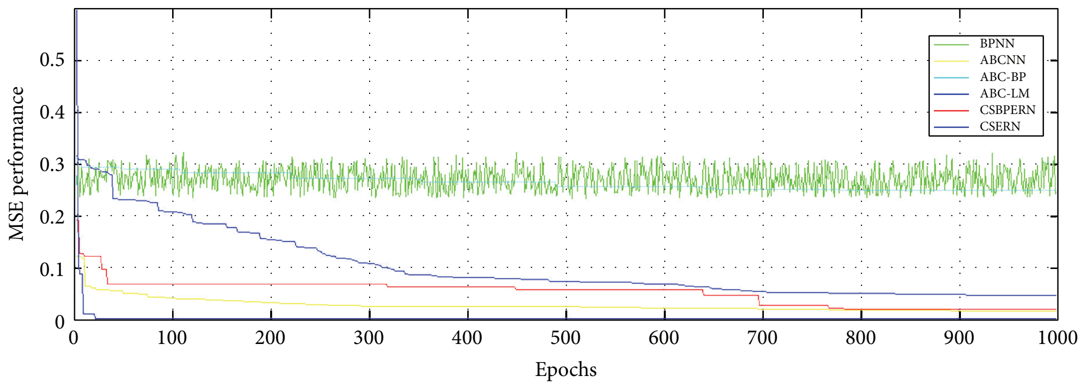

FIGURE 6: MSE via epochs convergence for Credit Card Benchmark Classification Problem.

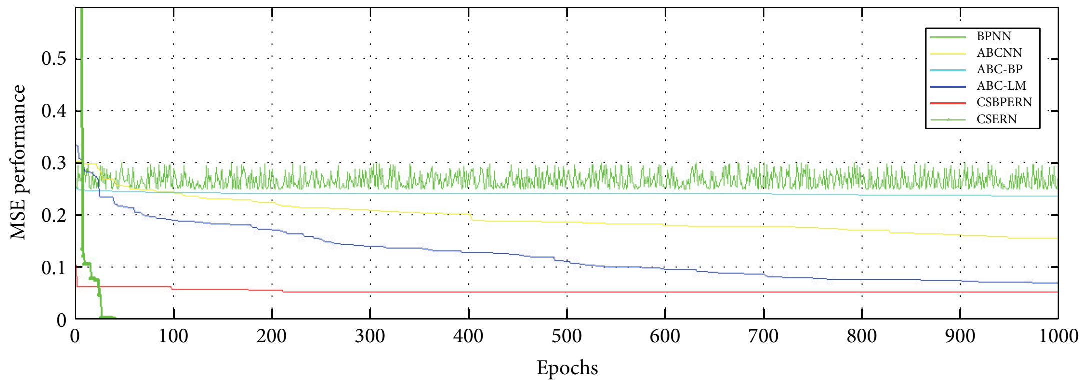

FIGURE 7: MSE via epochs convergence for 7-Bit Parity Classification Problem.

Furthermore, CSERN and BPERN achieved higher accuracy and less MSE on all the designated datasets.

Summary of Acronyms, Mathematical Symbols, and Their Meanings Used

ABP:	Adaptive back propagation
ANN:	Artificial neural network
ACPSO:	Adaptive chaotic particle swarm optimization
BP:	Back propagation
BPERN:	Back propagation Elman recurrent network
BPTT:	Back propagation through time
CS:	Cuckoo Search
CSERN:	Cuckoo Search Elman recurrent network
CSBPERN:	Cuckoo Search back propagation Elman recurrent network
DO:	Dissolved oxygen
ERN:	Elman recurrent network
ERNPSO:	Elman recurrent network particle swarm optimization
FCRNN:	Fully connected recurrent neural network
MBP:	Momentum back propagation
PCA:	Principle component analysis
PSO:	Particle swarm optimization
PSO-BP:	Particle swarm optimization back propagation
RPROP:	Resilient back propagation
RNN:	Recurrent neural network
SAR:	Synthetic aperture radar
SSE:	Sum of square errors
TN:	Total nitrogen
TP:	Total phosphorus
W_n:	Weight value at each layer in the feed forward network
U_n:	Weight value at each addition layer in the recurrent feedback
B_n:	Bias values for the network
W^c:	Total weight matrix for the network
net_j:	Output function for the hidden layer
net_k:	Output function for the output layer
f:	Net activation function for the hidden layer
g:	Net input activation function for the output layer
T_k:	Actual output
Y_k:	Predicted output
$V_\mu(x)$:	Average performance
$V_F(x)$:	Performance index
rand:	Random function for generating random variables
E:	Error at the output layer

x_i:	Cuckoo nest at position i
x_j:	Cuckoo nest at position j
x_i^{t+1}:	New solution for Cuckoo
$\oplus$:	Stepwise multiplication
∇X_i:	Small movement of Cuckoo x_i towards x_j
δ_{pk}:	Error for output nodes
δ_{pk}:	Error for hidden nodes
∇W:	Change in weights for the layers
∇B:	Change in bias weights for the layers
∇U:	Change in weights for the recurrent layer.

Conflict of Interests

There is no conflict of interests reported regarding the publication of this paper.

Authors' Contribution

All authors equally contributed to this paper.

Acknowledgments

This work is supported by Fundamental Research Grant Scheme (FRGS) Vote no. 1236 from MoHE Malaysia and Program Rakan Penyelidikan University of Malaya (PRPUM) Grant Vote no. CG063-2013.

References

[1] Y. Zhang and L. Wu, "Stock market prediction of S&P 500 via combination of improved BCO approach and BP neural network," *Expert Systems with Applications*, vol. 36, no. 5, pp. 8849–8854, 2009.

[2] L. Fauseeti, *Fundamental of Neural Network Architecture, Algorithm and Application*, Prentice Hall, Englewood Cliffs, NJ, USA, 1994.

[3] S. Haykin, *Neural Network: A Comprehensive Foundation*, Macmillan, New York, NY, USA, 1994.

[4] M. H. Hassoun, *Foundamental of Artificial Neural Network*, Massachusetts Institute of Technology Press, London, UK, 1995.

[5] P. Xiao, G. K. Venayagamoorthy, and K. A. Corzine, "Combined training of recurrent neural networks with particle swarm optimization and backpropagation algorithms for impedance identification," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '07)*, pp. 9–15, April 2007.

[6] C. L. Giles, S. Lawrence, and A. C. Tsoi, "Rule inference for financial prediction using recurrent neural networks," in *Proceedings of the IEEE/IAFE Conference on Computational Intelligence for Financial Engineering*, pp. 253–259, IEEE, March 1997.

[7] S. Li, D. C. Wunsch, E. O'Hair, and M. G. Giesselmann, "Wind turbine power estimation by neural networks with Kalman filter training on a SIMD parallel machine," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '99)*, pp. 3430–3434, Washington, DC , USA, July 1999.

[8] N. M. Nawi, A. Khan, and M. Z. Rehman, "CSBPRNN: a new hybridization technique using cuckoo search to train back propagation recurrent neural network," in *Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013)*, vol. 285 of *Lecture Notes in Electrical Engineering*, pp. 111–118, 2014.

[9] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[10] P. Coulibaly, F. Anctil, and J. Rousselle, "Real-time short-term natural water inflows forecasting using recurrent neural networks," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '99)*, vol. 6, pp. 3802–3805, IEEE Press, July 1999.

[11] N. M. Nawi, R. S. Ransing, and N. A. Hamid, "BPGD-AG: a new improvement of back-propagation neural network learning algorithms with adaptive gain," *Journal of Science and Technology*, vol. 2, pp. 83–102, 2011.

[12] A. Khan, N. M. Nawi, and M. Z. Rehman, "A new back-propagation neural network optimized with cuckoo search algorithm," in *Computational Science and Its Applications—ICCSA 2013: Proceedings of the 13th International Conference, Ho Chi Minh City, Vietnam, June 24-27, 2013, Part I*, vol. 7971 of *Lecture Notes in Computer Science*, pp. 413–426, Springer, Berlin, Germany, 2013.

[13] A. Khan, N. M. Nawi, and M. Z. Rehman, "A new cuckoo search based Levenberg-Marquardt (CSLM) algorithm," in *Computational Science and Its Applications—ICCSA 2013: Proceedings of the 13th International Conference, Ho Chi Minh City, Vietnam, June 24-27, 2013, Part I*, vol. 7971 of *Lecture Notes in Computer Science*, pp. 438–451, Springer, Berlin, Germany, 2013.

[14] N. M. Nawi, A. Khan, and M. Z. Rehman, "A new Levenberg Marquardt based back propagation algorithm trained with cuckoo search," *Procedia Technology*, vol. 11, pp. 18–23, 2013.

[15] N. M. Nawi, M. Z. Rehman, and A. Khan, "A new bat based back-propagation (BAT-BP) algorithm," in *Advances in Systems Science: Proceedings of the International Conference on Systems Science 2013 (ICSS 2013)*, vol. 240 of *Advances in Intelligent Systems and Computing*, pp. 395–404, Springer, 2014.

[16] N. M. Nawi, R. Ghazali, and M. N. M. Salleh, "The development of improved back-propagation neural networks algorithm for predicting patients with heart disease," in *Information Computing and Applications: Proceedings of the 1st International Conference, ICICA 2010, Tangshan, China, October 15–18, 2010*, vol. 6377 of *Lecture Notes in Computer Science*, pp. 317–324, Springer, Berlin, Germany, 2010.

[17] Y. Zhang, S. Wang, G. Ji, and P. Phillips, "Fruit classification using computer vision and feedforward neural network," *Journal of Food Engineering*, vol. 143, pp. 167–177, 2014.

[18] T. G. Barbounis, J. B. Theocharis, M. C. Alexiadis, and P. S. Dokopoulos, "Long-term wind speed and power forecasting using local recurrent neural network models," *IEEE Transactions on Energy Conversion*, vol. 21, no. 1, pp. 273–284, 2006.

[19] E. D. Übeyli, "Recurrent neural networks employing Lyapunov exponents for analysis of doppler ultrasound signals," *Expert Systems with Applications*, vol. 34, no. 4, pp. 2538–2544, 2008.

[20] A. M. Ahmad, S. Ismail, and D. F. Samaon, "Recurrent neural network with backpropagation through time for speech recognition," in *Proceedings of the IEEE International Symposium on Communications and Information Technologies: Smart Info-Media Systems (ISCIT '04)*, pp. 98–102, October 2004.

[21] Y. Zhang and L. Wu, "Crop classification by forward neural network with adaptive chaotic particle swarm optimization," *Sensors*, vol. 11, no. 5, pp. 4721–4743, 2011.

[22] M. F. A. Aziz, H. N. A. Hamed, and S. M. H. Shamsuddin, "Augmentation of Elman Recurrent Network learning with

particle swarm optimization," in *Proceedings of the 2nd Asia International Conference on Modelling & Simulation (AMS '08)*, pp. 625–630, May 2008.

[23] F. Cheng and H. Shen, "An improved recurrent neural network for radio propagation loss prediction," in *Proceedings of the International Conference on Intelligent Computation Technology and Automation (ICICTA '10)*, pp. 579–582, May 2010.

[24] H. Wang, Y. Gao, Z. Xu, and W. Xu, "An recurrent neural network application to forecasting the quality of water diversion in the water source of Lake Taihu," in *Proceedings of the International Conference on Remote Sensing, Environment and Transportation Engineering (RSETE '11)*, pp. 984–988, June 2011.

[25] Y. Tanoto, W. Ongsakul, and C. O. P. Marpaung, "Levenberg-Marquardt recurrent networks for long-term electricity peak load forecasting," *Telkomnika (Telecommunication, Computing, Electronics and Control)*, vol. 9, pp. 257–266, 2013.

[26] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proceedings of the World Congress on Nature and Biologically Inspired Computing (NABIC '09)*, pp. 210–214, Coimbatore, India, December 2009.

[27] X.-S. Yang and S. Deb, "Engineering optimisation by cuckoo search," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 1, no. 4, pp. 330–343, 2010.

[28] M. Tuba, M. Subotic, and N. Stanarevic, "Modified cuckoo search algorithm for unconstrained optimization problems," in *Proceedings of the 5th European Computing Conference (ECC '11)*, pp. 263–268, April 2011.

[29] J. R. Quinlan, P. J. Compton, K. A. Horn, and L. Lazarus, "Inductive knowledge acquisition: a case study," in *Proceedings of the 2nd Australian Conference on Applications of Expert Systems*, pp. 137–156, 1986.

[30] W. H. Wolberg and O. L. Mangasarian, "Multisurface method of pattern separation for medical diagnosis applied to breast cytology," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 87, no. 23, pp. 9193–9196, 1990.

[31] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[32] I. W. Evett and E. J. Spiehler, "Rule induction in forensic science," in *Knowledge Based Systems in Government*, pp. 107–118, 1987.

[33] J. R. Quinlan, "Simplifying decision trees," *International Journal of Man-Machine Studies*, vol. 27, no. 3, pp. 221–234, 1987.

[34] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Networks*, vol. 1, no. 4, pp. 295–307, 1988.

[35] 7-Bit Parity Training data, http://homepages.cae.wisc.edu/~ece539/data/parity7r.

[36] 7-Bit Parit Testing data, http://homepages.cae.wisc.edu/~ece539/data/parity7t.