A New Back-Propagation Neural Network Optimized with Cuckoo Search Algorithm

Nazri Mohd. Nawi, Abdullah Khan, and Mohammad Zubair Rehman

Faculty of Computer Science and Information Technology,
Universiti Tun Hussein Onn Malaysia (UTHM)
P.O. Box 101, 86400 Parit Raja, Batu Pahat, Johor Darul Takzim, Malaysia
nazri@uthm.edu.my, hi100010@siswa.uthm.edu.my,
zrehman862060@gmail.com

Abstract. Back-propagation Neural Network (BPNN) algorithm is one of the most widely used and a popular technique to optimize the feed forward neural network training. Traditional BP algorithm has some drawbacks, such as getting stuck easily in local minima and slow speed of convergence. Nature inspired meta-heuristic algorithms provide derivative-free solution to optimize complex problems. This paper proposed a new meta-heuristic search algorithm, called cuckoo search (CS), based on cuckoo bird's behavior to train BP in achieving fast convergence rate and to avoid local minima problem. The performance of the proposed Cuckoo Search Back-Propagation (CSBP) is compared with artificial bee colony using BP algorithm, and other hybrid variants. Specifically OR and XOR datasets are used. The simulation results show that the computational efficiency of BP training process is highly enhanced when coupled with the proposed hybrid method.

Keywords: Back propagation neural network, cuckoo search algorithm, local minima, and artificial bee colony algorithm.

1 Introduction

Artificial Neural Networks (ANN) provide main features, such as: flexibility, competence, and capability to simplify and solve problems in pattern classification, function approximation, pattern matching and associative memories [1-3]. Among many different ANN models, the multilayer feed forward neural networks (MLFF) have been mainly used due to their well-known universal approximation capabilities [4]. The success of ANN mostly depends on their design, the training algorithm used, and the choice of structures used in training. ANN are being applied for different optimization and mathematical problems such as classification, object and image recognition, Signal processing, seismic events prediction, temperature and weather forecasting, bankruptcy, tsunami intensity, earthquake, and sea level etc. [5-9]. ANN has the aptitude for random nonlinear function approximation and information processing which other methods does not have [10]. Different techniques are used in the past for optimal network performance for training ANNs such as back propagation neural network

B. Murgante et al. (Eds.): ICCSA 2013, Part I, LNCS 7971, pp. 413–426, 2013.

(BPNN) algorithm [11]. However, the BPNN algorithm suffers from two major drawbacks: low convergence rate and instability. The drawbacks are caused by a risk of being trapped in a local minimum [12], [13] and possibility of overshooting the minimum of the error surface [14]. Over the last years, many numerical optimization techniques have been employed to improve the efficiency of the back propagation algorithm including the conjugate gradient descent [15] [16]. However, one limitation of this procedure, which is a gradient-descent technique, is that it requires a differentiable neuron transfer function. Also, as neural networks generate complex error surfaces with multiple local minima, the BPNN fall into local minima instead of a global minimum [17], [18]. Evolutionary computation is often used to train the weights and parameters of neural networks. In recent years, many improved learning algorithms have been proposed to overcome the weakness of gradient-based techniques. These algorithms include global search technique such as hybrid PSO-BP [19], artificial bee colony algorithm [20-22], evolutionary algorithms (EA) [23], particle swarm optimization (PSO) [24], differential evolution (DE) [25], ant colony, back propagation algorithm [26], and genetic algorithms (GA) [27].

In order to overcome the weaknesses of the conventional BP, this paper proposed a new meta-heuristic search algorithm, called cuckoo search back propagation (CSBP). Cuckoo search (CS) is developed by Yang and Deb [28] which imitates animal behavior and is useful for global optimization [29], [30], [31]. The CS algorithm has been applied independently to solve several engineering design optimization problems, such as the design of springs and welded beam structures [32], and forecasting [33]. For these problems, Yang and Deb showed that the optimal solutions obtained by CS are far better than the best solutions obtained by an efficient particle swarm optimizer or genetic algorithms. In particular, CS can be modified to provide a relatively high convergence rate to the true global minimum [34].

In this paper, the convergence behavior and performance of the proposed Cuckoo Search Back-propagation (CSBP) on XOR and OR datasets is analyzed. The results are compared with artificial bee colony using BPNN algorithm, and similar hybrid variants. The main goals are to decrease the computational cost and to accelerate the learning process using a hybridization method.

The remaining paper is organized as follows: Section 2 gives literature review on BPNN. Section 3, explains Cuckoo Search via levy flight. In section 4, the proposed CSBP algorithm, and simulation results are discussed in Section 5. Finally, the paper is concluded in the Section 6.

2 Back-Propagation Neural Network (BPNN)

The Back-Propagation Neural Network (BPNN) is one of the most novel supervised learning ANN algorithm proposed by Rumelhart, Hinton and Williams in 1986 [35]. BPNN learns by calculating the errors of the output layer to find the errors in the hidden layers. Due to this ability of back-propagating, it is highly suitable for problems in which no relationship is found between the output and inputs. The gradient descent method is utilized to calculate the weights and adjustments are made to the network to

minimize the output error. The BPNN algorithm has become the standard algorithm used for training multilayer perceptron. It is a generalized least mean squared (LMS) algorithm that minimizes a criterion equals to the sum of the squares of the errors between the actual and the desired outputs. This principle is;

$$E_p = \sum_{i=1}^{j} (e_i)^2 \tag{1}$$

Where the nonlinear error signal is

$$e_i = d_i - y_i \tag{2}$$

d_i and y_i are respectively, the desired and the current outputs for the i^{th} unit. P denotes in (1) the p^{th} pattern; j is the number of the output units. The gradient descent method is given by;

$$w_{ki} = -\mu \frac{\partial E_p}{\partial w_{ki}} \tag{3}$$

Where w_{ki} is the weight of the i^{th} unit in the $(n\text{-}1)^{th}$ layer to the k^{th} unit in the nth layer. The BP calculates errors in the output layer ∂_l, and the hidden layer, ∂_j are using the formulas in Equation (4) and Equation (5) respectively [36] [37]:

$$\partial_l = \mu(d_i - y_i) f'(y_i) \tag{4}$$

$$\partial_j = \mu \sum_i \partial_l w_{lj} f'(y_i) \tag{5}$$

Here d_i is the desired output of the ith output neuron, y_i is the actual output in the output layer, y_i is the actual output value in the hidden layer, and k is the adjustable variable in the activation function. The back propagation error is used to update the weights and biases in both the output and hidden layers. The weights, w_{ij} and biases, b_i, are then adjusted using the following formulae;

$$w_{ij}(k+1) = w_{ij}(k) + \mu \partial_j y_i \tag{6}$$

$$w_{lj}(k+1) = w_{lj}(k) + \mu \partial_j x_l \tag{7}$$

$$b_i(k+1) = b_i(k) + \mu \partial_j \tag{8}$$

Here k is the number of the epoch and μ is the learning rate.

3 Cuckoo Search Viva Levy Flight

Levy Flights have been used in many search algorithms [38]. In Cuckoo Search algorithm levy flight is an important component for local and global searching [34]. In sub sections, levy flight and then cuckoo search algorithm based on levy flights is explained.

a. Levy Flight

In nature, the flight movement of many animals and insects is recognized as a random manner. The foraging path of an animal commonly has the next move based on the current state and the variation probability to the next state. Which way it chooses be determined by indirectly on a probability that can be modeled mathematically. Levy Flights is a random walk that is characterized by a series of straight jumps chosen from a heavy-tailed probability density function [34]. In statistical term, it is a stochastic algorithm for global optimization that finds a global minimum [38]. A levy flight process step length can be calculated by using Mantegna algorithm using equation (9).

$$S = \frac{u}{|v|^{\frac{1}{\beta}}} \tag{9}$$

Note that u and v are drawn from normal distribution with respect to these two random variables;

$$u \sim N(0.\sigma_u{}^2), \qquad v \sim N(0.\sigma_v{}^2) \tag{10}$$

The symbol ~ in (10) means the random variable obeys the distribution on right hand side; that is, samples should draw from the distribution. The $\sigma_u{}^2$ and $\sigma_v{}^2$ present in equation (10) are the variance of distributions which come from equation (11);

$$\sigma_u = \left\{\frac{\Gamma(1+\beta)\sin(\tau\beta/2)}{\Gamma[(1+\beta)/2]\beta 2^{(\beta-1)^2}}\right\}^{\frac{1}{\beta}}, \qquad \sigma_v = 1 \tag{11}$$

b. Cuckoo Search (CS) Algorithm

Cuckoo Search (CS) algorithm is a novel meta-heuristic technique proposed by Xin-Shen Yang [28]. This algorithm was stimulated by the obligate brood parasitism of some cuckoo species by laying their eggs in the nests of other host birds. Some host nest can keep direct difference. If an egg is discovered by the host bird as not its own, it will either throw the unknown egg away or simply abandon its nest and build a new nest elsewhere. Some other species have evolved in such a way that female parasitic cuckoos are often very specialized in the mimic in color and pattern of the eggs of a few chosen host species. This reduces the probability of their eggs being abandoned and thus increases their population. The CS algorithm follows three idealized rules:

a. Each cuckoo lays one egg at a time, and put its egg in randomly chosen nest;
b. The best nests with high quality of eggs will carry over to the next generations;
c. The number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability $pa \in [0, 1]$.

In this case, the host bird can either throw the egg away or abandon the nest, and build a completely new nest. The rule-c defined above can be approximated by the fraction pa $\in [0, 1]$ of the n nests that are replaced by new nests (with new random solutions). For a maximization problem, the quality or fitness of a solution can simply be proportional to the value of the objective function. In this algorithm, each egg in a nest represents a solution, and a cuckoo egg represents a new solution, the aim is to use a new and a potentially better solution (cuckoo) to replace a not so good solution in the nests. Based on these three rules, the basic steps of the Cuckoo Search (CS) can be summarized as the following pseudo code:

When generating new solutions x^{t+1} for a cuckoo i, a Levy flight is performed

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

Where $\alpha > 0$ is the step size, which should be related to the scales of the problem of interest. The product $\oplus$ means entry wise multiplications. The random walk via Levy flight is more efficient in exploring the search space as its step length is much longer in the long run. The Levy flight essentially provides a random walk while the random step length is drawn from a Levy distribution as shown in the Equation 12:

$$\text{Lavy} \sim u = t^{-\lambda}, 1 < \lambda \leq 3 \tag{13}$$

This has an infinite variance with an infinite mean. Here the steps essentially construct a random walk process with a power-law step-length distribution and a heavy tail. Some of the new solutions should be generated by Levy walk around the best solution obtained so far, this will speed up the local search. However, a substantial fraction of the new solutions should be generated by far field randomization whose locations should be far enough from the current best solution. This will make sure the system will not be trapped in local minima.

The pseudo code for the CS algorithm is given below:

Step 1: Generate initial population of N host nest i= 1... N
Step 2: while (fmin<MaxGeneration) or (stop criterion)
Do
Step 3: Get a cuckoo randomly by Levy flights and evaluate its fitness F_i
Step 4: Choose randomly a nest j among N.
Step 5: if $F_i > F_j$Then
Step 6: Replace j by the new solution.
End if
Step 7: A fraction pa of worse nest are abandoned and new ones are built.
Step 8: Keep the best solutions (or nest with quality solutions).
Step 9: Rank the solutions and find the current best.
End while

4 The Proposed CSBP Algorithm

The CS is a population based optimization algorithm, and like other meta-heuristic algorithms, it starts with a random initial population, The CS algorithm basically works in three steps: selection of the best source by keeping the best nests or solutions, replacement of host eggs with respect to the quality of the new solutions or cuckoo eggs produced based randomization via Levy flights globally (exploration) and discovering of some cuckoo eggs by the host birds and replacing according to the quality of the local random walks (exploitation) [29]. Each cycle of the search consists of several steps initialization of the best nest or solution, the number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability $pa\ \varepsilon\ [0, 1]$.

In the proposed CSBP algorithm, each best nest represents a possible solution (i.e., the weight space and the corresponding biases for BPNN optimization in this paper). The weight optimization problem and the size of population represent the quality of the solution. In the first epoch, the best weights and biases are initialized with CS and then these weights are passed to the BPNN. The weights in BPNN are calculated and compared with best solution in the backward direction. In the next cycle CS will updated the weights with the best possible solution and CS will continue searching the best weights until the last cycle/ epoch of the network is reached or either the MSE is achieved.

The pseudo code of the proposed CSBP algorithm is:

Step 1: CS is initializes and passes the best weights to BPNN
Step 2: Load the training data
Step 3**: While** MSE<stopping criteria
Step 4: Initialize all cuckoo nests
Step 5: Pass the cuckoo nests as weights to network
Step 6: Feed forward neural network runs using the weights initialized with CS
Step 7: Calculate the error backward
Step8: CS keeps on calculating the best possible weight at each epoch until the network is converged.
End While

5 Experiments and Results

To test the performance of the proposed CSBP, Boolean datasets of 4-bit OR, 2-bit and 3-bit XOR were used. The simulation experiments were performed on a 1.66 GHz AMD Processor with 2GB of RAM. For performing simulations Matlab 2009b software was used. The proposed (CSBP) algorithm is compared with artificial bee colony Levenberg Marquardt algorithm (ABC-LM), artificial bee colony back propagtion (ABC-BP) algorithm and standard BPNN based on mean squarer error, epochs and CPU time. The three layer feed forward neural network are used for each problem.

I.e. input layer, one hidden layer, and output layers. The number of hidden nodes is design of five and ten neurons. In the network structure the biases nodes are also used and the log sigmoid activation function is placed as the activation function for the hidden and output layers nodes. For each problem, trial is limited to 1000 epochs. And minimum error is keeping 0. A total of 20 trials are run for each case. The network results are stored in the result file for each trial.

The first test problem is the Exclusive OR (XOR) Boolean function of two binary input to a single binary output as (0 0; 0 1; 1 0 ; 1 1) -to -(0; 1; 1 ;0). From the Table 1, we can see that the proposed CSBP method performs well on 2-bit XOR dataset. The CSBP converges to global minima in 21.23 second of CPU time with an average accuracy of 100% and achieves a MSE of 0. While other algorithms fall behind in-terms of MSE, CPU time and accuracy. The Figure 1 shows that the CSBP performs well and converges to global minima within 134 epochs for the 2-5-1 network structure.

Table 1. CPU time, Epochs and MSE error for 2- bit XOR dataset with 2-5-1 ANN architecture

Algorithm	ABC-BP	ABC-LM	BPNN	CSBP
CPU TIME	172.3388	123.9488	42.64347	21.23
EPOCHS	1000	1000	1000	149
MSE	2.39E-04	0.125	0.220664	0
Accuracy (%)	96.47231	71.69041	54.6137	100

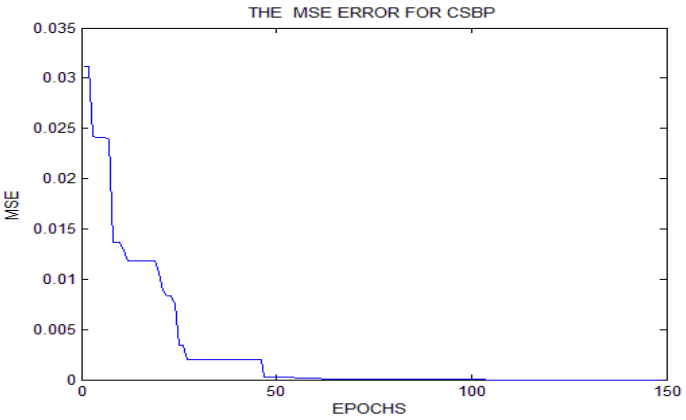

Fig. 1. The CSBP convergence performance on 2 bit XOR with 2-5-1 ANN architecture

Table 2 give an idea about the CPU time, number of epochs and the mean square error for the 2 bit XOR data sets with ten hidden neurons. From the table, we can identify that the proposed CSBP method has better result than the ABC-BP, ABC-LM and BPNN algorithm. The CSBP achieves a MSE of 0 in 100 epochs and in 28.9 second of CPU time. Meanwhile, the other algorithms have short performed with large MSE's and CPU times. The Figure 2 shows the convergence performance of CSBP algorithm for the 2-10-1 network architecture.

Table 2. CPU time, Epochs and MSE error for 2- bit XOR dataset with 2-10-1 ANN architecture

Algorithm	ABC-BP	ABC-LM	BPNN	CSBP
CPU TIME	197.34	138.96	77.63	46.99
EPOCHS	1000	1000	1000	203
MSE	8.39E-04	0.12578	0.120664	0
Accuracy (%)	96.8	71.876	54.6137	100

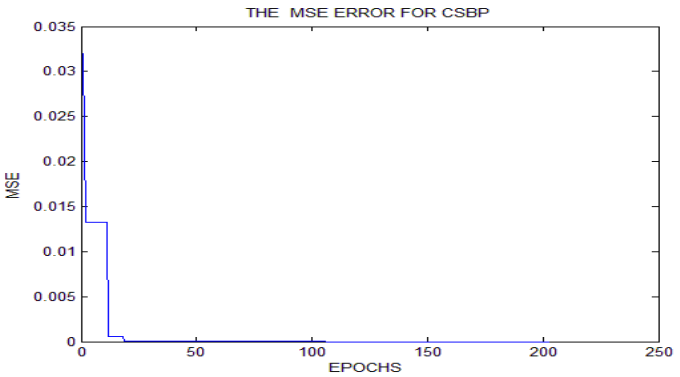

Fig. 2. The CSBP convergence performance on 2 bit XOR with 2-10-1 ANN architecture

In the second case we used 3- bit Exclusive OR test problem of three input to a single binary output as (1 1 1; 1 1 0 ; 1 0 1; 1 0 0; 0 1 1; 0 1 0 ; 0 0 1; 0 0 0) –to-(1; 0; 0; 1; 0; 1; 1; 0). In the simulation, we selected a 3- bit exclusive OR data-set for 3-5-1 architecture. In three bit XOR input if the number of binary inputs is odd, the output is 1, otherwise the output is 0. Tables 3 describe the CPU time, number of epochs and the MSE of the XOR dataset with five hidden neurons. We can see

in the Table 3 that the proposed algorithm has an MSE of 5.4E-04 within 938 epochs and in 149.53 of CPU time. In 3-bit XOR dataset has better performances in MSE and accuracy. But it has take long CPU time for the processing then the BPNN, and ABC

Table 3. CPU time, Epochs and MSE error for 3- bit XOR dataset with 3-5-1 ANN architecture

Algorithm	ABC-BP	ABC-LM	BPNN	CSBP
CPU TIME	172.3388	123.7925	50.03059	149.53
EPOCHS	1000	1000	1000	938
MSE	0.08	0.01063	0.25077	5.4E-04
Accuracy (%)	86.47231	78.83231	47.626	98.7351

Table 4 provides an idea about the CPU time, number of epochs and the mean square error for the 3 bit XOR test problem with ten hidden neurons. The proposed CSBP method with 3-10-1 network structure also outperforms the other algorithm in term of CPU time and MSE errors. The CSBP gets an MSE of 7.4E-04 in 161.53 CPU time. While for the same network structure the ABC-BP achieves a less MSE of 0.0845. The conventional back propagation has 0.256 MSE achieved in 75.13748 CPU cycles.

Table 4. CPU time, Epochs and MSE error for 3- bit XOR dataset with 3-10-1 ANN architecture

Algorithm	ABC-BP	ABC-LM	BPNN	CSBP
CPU TIME	185.3388	153.7925	75.13748	161.53
EPOCHS	1000	1000	1000	1000
MSE	0.0845	0.01054	0.256	7.4E-04
Accuracy (%)	86.8971	79.83231	49.626	98.88

In the third case we used four bit OR test problem of four input to a single binary output as (1 1 1 1;1 1 1 0;1 1 0 1;1 1 0 0; 1 0 1 1;1 0 0 1; 1 0 0 0;0 1 1 1;0 1 1 0; 0 1 0 1; 0 1 0 0; 0 0 1 1 ; 0 0 1 0; 0 0 0 1;1 0 1 0; 0 0 0 0) –to –(1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 0). In four bit OR dataset, if all inputs are 0, the output is 0, otherwise the output will be 1. For the 4-5-1 network architecture, it has twenty five connection weights and six biases. Tables 5, confirms the CPU time, number of epochs, the mean square error, and accuracy for the 4 bit OR test problem with five hidden

neurons. The proposed CSBP converged to a MSE of 0 within 51 epochs. While the ABC-LM algorithm has an MSE of 1.82E-10 and the ABC-BP has the MSE of 1.91E-05.

Table 5. CPU time, Epochs and MSE error for 4- bit OR dataset with 4-5-1 ANN architecture

Algorithm	ABC-BP	ABC-LM	BPNN	CSBP
CPU TIME	162.4945	118.7274	63.28089	8.486525
EPOCHS	1000	1000	1000	51
MSE	1.91E-10	1.82E-10	0.052778	0
Accuracy (%)	99.97	99.99572	89.83499	100

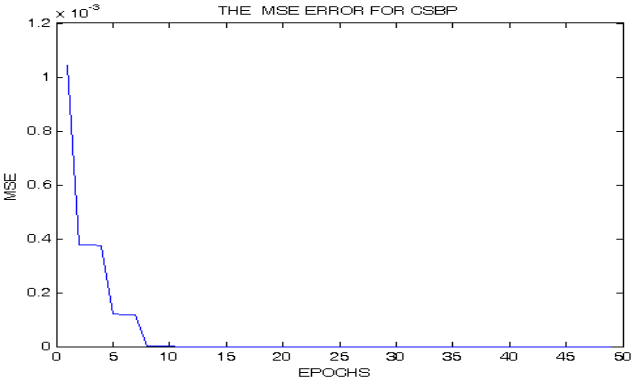

Fig. 3. The CSBP convergence performance on 4 bit OR with 4-5-1 ANN architecture

Table 6 illustrates the CPU time, number of epochs and the MSE for the 4 bit OR test problem with ten hidden neurons. In four bit OR datasets the proposed algorithm has outperformed other algorithms with 99 percent accuracy. Figure 4 shows the convergence performance of the proposed CSBP for the 4-10-1 network architecture.

Table 6. CPU time, Epochs and MSE error for 4- bit OR dataset with 4-10-1 ANN architecture

Algorithm	ABC-BP	ABC-LM	BPNN	CSBP
CPU TIME	180.4945	129.7274	67.28089	8.678
EPOCHS	1000	1000	1000	36

Table 6. *(Continued)*

MSE	1.67E-10	1.76E-10	0.05346	0
Accuracy(%)	99.47	99.78	89.83499	100

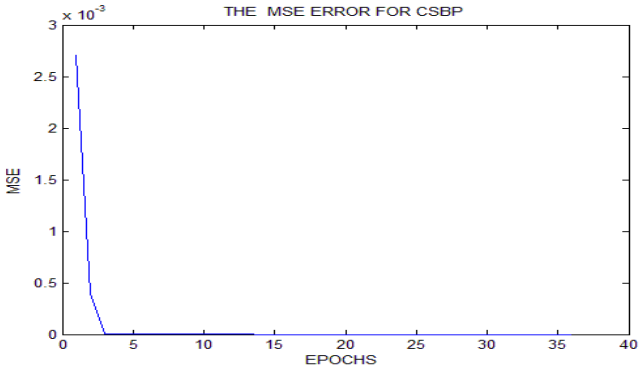

Fig. 4. The CSBP convergence performance on 4 bit OR with 4-10-1 ANN architecture

6 Conclusion

BPNN algorithm is one of the most widely used and a popular procedure to optimize the feed forward neural network training. Conventional BPNN algorithm has some drawbacks, such as getting stuck in local minima and slow speed of convergence. Nature inspired meta-heuristic algorithms provide derivative-free solution to optimize complex problems. A new meta-heuristic search algorithm, called cuckoo search (CS) is proposed to train BPNN to achieve fast convergence rate and to minimize the training error. The performance of the proposed CSBP algorithm is compared with the ABC-LM, ABC-BP and BPNN algorithms by means of simulation on three datasets such as 2-bit, 3-bit XOR and 4-bit OR. The simulation results show that the proposed CSBP is far better than the previous methods in terms of simplicity, convergence rate and accuracy. In future the proposed model will be used on the benchmarks classification datasets collected from UCI machine learning repository.

Acknowledgment. The Authors would like to thank Office of Research, Innovation, Commercialization and Consultancy Office (ORICC), Universiti Tun Hussein Onn Malaysia (UTHM) and Ministry of Higher Education (MOHE) Malaysia for financially supporting this Research under Fundamental Research Grant Scheme (FRGS) vote no. 1236.

References

1. Dayhoff, J.E.: Neural-Network Architectures: An Introduction, 1st edn. Van Nostrand Reinhold Publishers, New York (1990)
2. Rehman, M.Z., Nazri, M.N.: Studying the Effect of adaptive momentum in improving the accuracy of gradient descent back propagation algorithm on classification problems. International Journal of Modern Physics (IJMPCS) 9(1), 432–439 (2012)
3. Ozturk, C., Karaboga, D.: Hybrid Artificial Bee Colony algorithm for neural network training. In: IEEE Congress of Evolutionary Computation (CEC), pp. 84–88 (2011)
4. Haykin, S.: Neural Networks, A Comprehensive Foundation. Prentice Hall, New Jersey (1999)
5. Du, K.L.: Clustering: A neural network approach. Neural Networks 23(1), 89–107 (2010)
6. Guojin, C., Miaofen, Z., et al.: Application of Neural Networks in Image Definition Recognition, Signal Processing and Communications. In: ICSPC, pp. 1207–1210 (2007)
7. Romano, M., Liong, S., et al.: Artificial neural network for tsunami forecasting. Asian Earth Sciences 36, 29–37 (2009)
8. Hayati, M., Mohebi, Z.: Application of Artificial Neural Networks for Temperature forecasting. World Academy of Science, Engineering and Technology 28(2), 275–279 (2007)
9. Perez, M.: Artificial neural networks and bankruptcy forecasting: a state of the art. Neural Computing & Application 15, 154–163 (2006)
10. Contreras, J., Rosario, E., Nogales, F.J., Conejos, A.J.: ARIMA Models to Predict Next-Day Electricity Prices. IEEE Transactions on Power Systems 18(3), 1014–1020 (2003)
11. Leung, C., Member, C.: A Hybrid Global Learning Algorithm Based on Global Search and Least Squares Techniques for back propagation neural network Networks. In: International Conference on Neural Networks, pp. 1890–1895 (1994)
12. Ahmed, W.A.M., Saad, E.S.M., Aziz, E.S.A.: Modified Back Propagation Algorithm for Learning Artificial Neural Networks. In: Eighteenth National Radio Science Conference (NRSC), pp. 345–352 (2001)
13. Leigh, W., Hightower, R., Modena, N.: Forecasting the New York Stock exchange composite index with past price and invest rate on condition of volume spike. Expert System with Applications 28(1), 1–8 (2005)
14. Wen, J., Zhao, J.L., Luo, S.W., Han, Z.: The Improvements of BP Neural Network Learning Algorithm. In: 5th Int. Conf. on Signal Processing WCCC-ICSP, pp. 1647–1649 (2000)
15. Lahmiri, S.: Wavelet transform, neural networks and the prediction of s & p price index: a comparativepaper of back propagation numerical algorithms. Business Intelligence Journal 5(2), 235–244 (2012)
16. Nawi, N.M., Ransing, R.S., Salleh, M.N.M., Ghazali, R., Hamid, N.A.: An improved back propagation neural network algorithm on classification problems. In: Zhang, Y., Cuzzocrea, A., Ma, J., Chung, K.-I., Arslan, T., Song, X. (eds.) DTA and BSBT 2010. CCIS, vol. 118, pp. 177–188. Springer, Heidelberg (2010)
17. Gupta, J.N.D., Sexton, R.S.: Comparing backpropagation with a genetic algorithm for neural network training. The International Journal of Management Science 27, 679–684 (1999)
18. Nawi, N.M., Ghazali, R., Salleh, M.N.M.: The development of improved back-propagation neural networks algorithm for predicting patients with heart disease. In: Zhu, R., Zhang, Y., Liu, B., Liu, C. (eds.) ICICA 2010. LNCS, vol. 6377, pp. 317–324. Springer, Heidelberg (2010)

19. Zhang, J., Lok, T., Lyu, M.: A hybrid particle swarm optimization back propagation algorithm for feed forward neural network training. Applied Mathematics and Computation 185, 1026–1037 (2007)
20. Shah, H., Ghazali, R., Nawi, N.M., Deris, M.M.: Global hybrid ant bee colony algorithm for training artificial neural networks. In: Murgante, B., Gervasi, O., Misra, S., Nedjah, N., Rocha, A.M.A.C., Taniar, D., Apduhan, B.O. (eds.) ICCSA 2012, Part I. LNCS, vol. 7333, pp. 87–100. Springer, Heidelberg (2012)
21. Shah, H., Ghazali, R., Nawi, N.M.: Hybrid ant bee colony algorithm for volcano temperature prediction. In: Chowdhry, B.S., Shaikh, F.K., Hussain, D.M.A., Uqaili, M.A. (eds.) IMTIC 2012. CCIS, vol. 281, pp. 453–465. Springer, Heidelberg (2012)
22. Karaboga, D., Akay, B., Ozturk, C.: Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks. In: Torra, V., Narukawa, Y., Yoshida, Y. (eds.) MDAI 2007. LNCS (LNAI), vol. 4617, pp. 318–329. Springer, Heidelberg (2007)
23. Yao, X.: Evolutionary artificial neural networks. International Journal of Neural Systems 4(3), 203–222 (1993)
24. Mendes, R., Cortez, P., Rocha, M., Neves, J.: Particle swarm for feedforward neural network training. In: Proceedings of the International Joint Conference on Neural Networks, vol. 2, pp. 1895–1899 (2002)
25. Ilonen, J., Kamarainen, J.I., Lampinen, J.: Differential Evolution Training Algorithm for Feed-Forward Neural Networks. Neural Processing Letters 17(1), 93–105 (2003)
26. Liu, Y.-P., Wu, M.-G., Qian, J.-X.: Evolving Neural Networks Using the Hybrid of Ant Colony Optimization and BP Algorithms. In: Wang, J., Yi, Z., Żurada, J.M., Lu, B.-L., Yin, H. (eds.) ISNN 2006. LNCS, vol. 3971, pp. 714–722. Springer, Heidelberg (2006)
27. Khan, A.U., Bandopadhyaya, T.K., Sharma, S.: Comparisons of Stock Rates Prediction Accuracy using Different Technical Indicators with Backpropagation Neural Network and Genetic Algorithm Based Backpropagation Neural Network. In: Proceedings of the First International Conference on Emerging Trends in Engineering and Technology. IEEE Computer Society, Nagpur (2008)
28. Yang, X.S., Deb, S.: Cuckoo search via Lévy flights. In: Proceedings of World Congress on Nature & Biologically Inspired Computing, India, pp. 210–214 (2009)
29. Yang, X.S., Deb, S.: Engineering Optimisation by Cuckoo Search. International Journal of Mathematical Modelling and Numerical Optimisation 1(4), 330–343 (2010)
30. Tuba, M., Subotic, M., Stanarevic, N.: Modified cuckoo search algorithm for unconstrained optimization problems. In: Proceedings of the European Computing Conference (ECC 2011), Paris, France, pp. 263–268 (2011)
31. Tuba, M., Subotic, M., Stanarevic, N.: Performance of a Modified Cuckoo Search Algorithm for Unconstrained Optimization Problems. WSEAS Transactions on Systems 11(2), 62–74 (2012)
32. Yang, X.S., Deb, S.: Engineering optimisation by cuckoo search. International Journal of Mathematical Modelling and Numerical Optimisation 1(4), 330–343 (2010)
33. Chaowanawate, K., Heednacram, A.: Implementation of Cuckoo Search in RBF Neural Network for Flood Forecasting. In: Fourth International Conference on Computational Intelligence, Communication Systems and Networks, pp. 22–26 (2012)
34. Walton, S., Hassan, O., Morgan, K., Brown, M.: Modified cuckoo search: A new gradient free optimisation algorithm. Chaos, Solitons & Fractals 44(9), 710–718 (2011)
35. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning Internal Representations by error Propagation. In: Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1 (1986)

36. Cichocki, A., Unbehauen, R.: Neural Network for Optimization and Signal Processing. Wiley, Chichester (1993)
37. Lippman, R.P.: An introduction to computing with neural networks. IEEE ASSP. Mag. 4(2) (April 1987)
38. Pavlyukevich, I.: Levy flights, non-local search and simulated annealing. Journal of Computational Physics 226(2), 1830–1844 (2007)