CSBPRNN: A New Hybridization Technique Using Cuckoo Search to Train Back Propagation Recurrent Neural Network

Nazri Mohd. Nawi[1], Abdullah khan[1], M. Z. Rehman[1]

[1]Software and Multimedia Centre, Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia (UTHM).
P.O. Box 101, 86400 Parit Raja, BatuPahat, Johor Darul Takzim, Malaysia.

nazri@uthm.edu.my, hi100010@siswa.uthm.edu.my, zrehman862060@gmail.com

Abstract. Nature inspired meta-heuristic algorithms provide derivative-free solution to optimize complex problems. Cuckoo Search (CS) algorithm is one of the most modern addition to the group of nature inspired optimization meta-heuristics. The Simple Recurrent Networks (SRN) were initially trained by Elman with the standard back propagation (SBP) learning algorithm which is less capable and often takes enormous amount of time to train a network of even a moderate size. And the complex error surface of the SBP makes many training algorithms are prone to being trapped in local minima. This paper proposed a new meta-heuristic based Cuckoo Search Back Propagation Recurrent Neural Network (CSBPRNN) algorithm. The CSBPRNN is based on Cuckoo Search to train BPRNN in order to achieve fast convergence rate and to avoid local minima problem. The performance of the proposed CSBPRNN 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, cuckoo search, local minima, and artificial bee colony algorithm, meta-heuristic optimization, recurrent neural network.

1 Introduction

Recurrent neural networks have been an important focus of research and development since1990's [1].There has been large number of research on dynamic system modelling with recurrent neural networks (RNN) [2-4]. They are calculated to learn sequential or time-varying patterns. All recurrent neural networks have feedback (closed loop) connections and are categorized as; BAM, Hopfield, Boltzmann machine, and recurrent back propagation nets [5]. RNN techniques have been applied to a wide variety of problems. Such as forecasting of financial data [6], electric power demand [7] and track water quality and minimize the additives needed for filtering water [8]. The main step of the RNN-based impedance extraction method is to train the neural network in such a way that it learns the dynamic performance of the test system. There are a number of training algorithms available for neural networks [9-12]. The Simple Recurrent Networks were initially trained by Elman with the standard back propagation (BP) learning algorithm, in which errors are computed 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 [13]. However, the BP algorithm that was introduced by Rumelhart [14] suffers from two major drawbacks: low convergence rate and instability. They are caused by a possibility of being trapped in a local minimum and prospect of overshooting the minimum of the error surface [15-18]. Also, RNN generate complex error surfaces with multiple local minima, the BP fall into local minima in place of a global minimum [15, 19-21]. Many methods have been proposed to speed up the back propagation based training algorithms. A recent development in evolutionary computation technique has enabled the application of various population based search algorithm in the training of neural network. Many researchers have focused on using genetic algorithms to change the weight parameters of neural networks [12, 15]. As a relatively new stochastic algorithm the particle swarm optimization (PSO) method has gained more and more attention. A preliminary study of PSO trained feed forward networks was presented in [11]. Test results based on the training of some simple problems showed that the performance of PSO is not much better than other methods. However, the authors argued that PSO is still promising in cases where a high number of local minima are known to exist [6].

In-order to improve convergence rate in gradient descent, this papers proposed a new meta-heuristic algorithm called Cuckoo search (CS) [22] integrated with BPRNN algorithm. The proposed Cuckoo Search Back-propagation Recurrent Neural Network (CSBPRNN) convergence behavior and performance will be analyzed on XOR and OR datasets. The results are compared with artificial bee colony using BPNN algorithm, and similar hybrid variants. The main goal is to decrease the computational cost and to accelerate the learning process using a hybridization method.

The remaining paper is organized as follows: Section II gives literature review of learning algorithm. Section III described the proposed CSBPRNN. Result and discussion are explained in Section IV. Finally, the paper is concluded in the Section V.

2 Cuckoo Search (CS) Algorithm

Cuckoo Search (CS) algorithm is a novel meta-heuristic technique proposed by Xin-She Yang [22-28]. 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. 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 fractionpa $\in [0,1]$ of the n nests that are replaced by new nests (with new random solutions).

2.1 Back Propagation Recurrent Neural Network

A Simple Recurrent Network (SRN) in its simplest form is a three layer feed forward (FF) network, but in SRN the hidden layer is self-recurrent which have short-term memory [1]. It is a special case of a general Recurrent Neural Network (RNN) and trained with full back propagation through time (BPTT) and its variants [29]. In this papers we used three layers network with one input layer, one hidden or 'state' layer, and one 'output' layer. Each layer will have its own index variable: $k$for output nodes, $j$and l for hidden, and i for input nodes. In a feed forward network, the input vector,x is propagated through a weight layer, V;

$$net_j(t) = \sum_i^n x_i(t)v_{ji} + b_j \tag{1}$$

Where, n the *number of inputs*, $b_j$is a bias, and $f$is an output function. In a SRN 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;

$$net_j(t) = \sum_i^n x_i(t)v_{ji} + \sum_l^m y_l(t-1)u_{jl} + b_j \tag{2}$$

$$y_j(t) = f(net_j(t)) \tag{3}$$

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;

$$net_k(t) = \sum_j^m y_j(t)w_{kj} + b_k \tag{4}$$

$$y_k(t) = g(net_k(t)) \tag{5}$$

Where, g is an output function. Finally the output is read off fromy_k, and the error Eagainst a target vector T is computed:

$$E = \frac{1}{2}\sum_p^n \sum_k^m (T_{pk} - y_{pk})^2 \tag{6}$$

Where, T is the desired output, n is the total number of available training samples and m is the total number of output nodes. And p, adds to the cost, overall output units k.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 weight.

$$\delta_{pk} = -\eta \frac{\delta E}{\delta y_{pk}} \tag{7}$$

Thus, the error for output nodes is;

$$\delta_{pk} = -\frac{\delta E}{\delta y_{pk}}\frac{\delta y_{pk}}{\delta net_{pk}} = \left(T_{pk} - y_{pk}\right)g'\left(y_{pk}\right) \tag{8}$$

$$\delta_{pk} = \left(T_{pk} - y_{pk}\right)y_{pk}\left(1 - y_{pk}\right) \tag{9}$$

And for the hidden nodes;

$$\delta_{pj} = -\left(\sum_k^m \frac{\delta E}{\delta y_{pk}}\frac{\delta y_{pk}}{\delta net_{pk}}\frac{\delta net_{pk}}{y_{pj}}\right)\frac{\delta y_{pj}}{\delta net_{pj}} \tag{10}$$

$$\delta_{pj} = \sum_k^m \delta_{pk} w_{kj} f'\left(y_{pj}\right) \tag{11}$$

Thus, the output weights are calculated as;

$$\Delta w_{kj} = \eta \sum_p^n \delta_{pk} y_{pj} \tag{12}$$

And for the input weights;

$$\Delta v_{ji} = \eta \sum_p^n \delta_{pj} x_{pi} \tag{13}$$

Adding a time subscript, the recurrent weights can be modified according to Equation 12;

$$\Delta u_{jh} = \eta \sum_p^n \delta_{pj}(t) y_{ph}(t-1) \tag{14}$$

3 The Proposed CSBPRNN Algorithm

```
Step 1:CS and BPRNN are initialized
Step 2: Load the training data
Step 3: Initialize all cuckoo nests
Step 4: Pass the cuckoo nests as weights to network
Step 5: While MSE<STOPPING CRITERIA
Step 6: Feed forward network runs using the weights initialized with CS
Step 7: Calculate the error
Step8: CS keeps on calculating the best possible weight at each epoch until the network is converged.
  End While
```

In the proposed CSBPRNN algorithm, each best nest represents a possible solution (i.e., the weight space and the corresponding biases for BPRNN optimization in this paper). 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 BPRNN. The weights in BPRNN are calculated. 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 CSBPRNN algorithm is:

4 Results and Discussion

The simulation experiments are performed on an AMD E-450, 1.66 GHz CPU with 2-GB RAM. The software used for simulation process is MATLAB 2009b. For performing simulations the following algorithms are trained on the 2-bit XOR and 4-bit OR datasets:

a. Back-Propagation Neural Network (BPNN) [14]
b. Artificial Bee Colony Back-Propagation (ABCBP) [30],
c. Artificial Bee Colony Levenberg Marquardt (ABCLM), [31]
d. Artificial Bee Colony Neural Network (ABCNN),[32] and
e. Proposed Cuckoo Search Back propagation Recurrent Neural Network (CSBPRNN).

Three layer neural networks is used for testing of the models, the hidden layer is kept fixed to 5-nodes while output and input layers nodes vary according to the data set given. Log-sigmoid activation function is used as the transfer function from input layer to hidden layer and from hidden layer to the output layer. And for CSBPRNN there is a feedback connection from hidden to input layers. For each problem, trial is limited to 1000 epochs. A total of 20 trials are run for each case. The network results are stored in the result file for each trial. Mean, standard deviation (SD) and accuracy are recorded for each independent trial on 2-bit XOR, and 4-bit OR dataset.

4.1 The 2-Bit XOR Dataset

The first test problem is the 2-bit XOR Boolean function of two binary inputs and a single binary output. In the simulations, we used 2-5-1, CSBPRNN network for two bit XOR dataset. Table 1, shows the CPU time, number of epochs, MSE and the Standard Deviation (SD) for the 2 bit XOR tested on CSBPRNN, ABCLM, ABCBP, ABCNN, and BPNN algorithms. Table 1 shows the MSE performance comparison of CSBPRNN, ABCBP, ABCLM, ABCNN, and BPNN for the 2-bit XOR. Table. 1 shows that the proposed CSBPRNN algorithm successfully avoided the local minima and converge the network within 327 epochs. From this, we can easily see that the proposed CSBPRNN can converge successfully for almost every kind of network structure. Also CSBPRNN has less MSE with high accuracy when compared with other algorithms.

Table 1. CPU Time, Epochs, MSE Accuracy, and Standard deviation for 2-5-1 ANN Structure

Algorithm	BPNN	ABCBP	ABCNN	ABCLM	CSBPRNN
CPUTIME	42.64	172.33	121.74	123.95	54.65
EPOCH	1000	1000	1000	1000	327
MSE	0.22066	2.39E-04	0.12500	0.1250	0
SD	0.01058	6.70E-05	1.05E-06	1.51E-06	0
Accuracy (%)	54.6137	96.4723	74.1759	71.6904	100

4.2 The 4-Bit OR dataset

The second dataset selected for training the proposed CSBPRNN is 4-bit OR dataset. In 4-bit OR, if the number of inputs all is 0, the output is 0, otherwise the output is 1. Again for the four bit input we applied 4-5-1, recurrent neural network structure. Table 2 illustrates the CPU time, epochs, and MSE performance accuracy and SD of the proposed CSBPRNN, ABCBP, ABCLM, ABCNN and BPNN algorithms respectively. From Table 2, we can see that the proposed CSBPRNN algorithm outperforms other algorithms in-terms of CPU time, epochs, MSE, accuracy, and SD. For the 4-5-1 network structure CSBPRNN again has 0 MSE, 100 percent accuracy and 0 SD of the MSE which is better than the BPNN, ABCNN, ABCBP, ABCLM algorithms.

Table 2. CPU Time, Epochs, MSE Accuracy, and Standard deviation for 4-5-1 ANN Structure

Algorithm	**BPNN**	**ABCBP**	**ABCNN**	**ABCLM**	**CSBPRNN**
CPUTIME	63.28	162.49	116.31	118.73	10.122
EPOCH	1000	1000	1000	1000	63
MSE	0.0528	1.9E-10	1.8E-10	1.8E-10	0
SD	0.008	1.5E-10	2.3E-11	2.1E-11	0
Accuracy (%)	89.83	99.97	99.99	99.99	100

5 Conclusion

Nature inspired meta-heuristic algorithms provide derivative-free solution to optimize complex problems. A new meta-heuristic search algorithm, called cuckoo search (CS) is used to train BPRNN to achieve faster convergence rate, minimize the training error, and to increase the convergence accuracy. The proposed CSBPRNN algorithm is used to train network on the 2-bit XOR, and 4-Bit OR benchmark dataset. The results show that the proposed CSBPRNN is simple and generic for optimization problems and has better convergence rate, Standard Deviation (SD), and high accuracy than the ABCLM, ABCBP, ABCNN, and BPNN algorithms. In future, the proposed algorithm CSBPRNN will be trained and tested on different benchmarks data classification tasks.

Acknowledgements

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. Elman, J. L.: Finding structure in time, J. Cognitive Science, Vol.14 (2), pp. 179--211, (1990)

2. Barbounis, T.G., Theocharis, J.B., et al., Long-term wind speed and power forecasting using local recurrent neural network models, J. IEEE Transactions on Energy Conversion, vol. 21,(1,) pp. 273--284, (2006)
3. Goedtel, A., Dasilva, I.N., and Amaral Semi, P.J., Recurrent Neural Network for Induction Motor Speed Estimation in Industry Application, In: IEEE MELECON, pp. 1134--1137, August (2006)
4. Xiao, P., Venayagamoorthy, G. K., and Corzine, K. A.: Combined Training of Recurrent Neural Networks with Particle Swarm Optimization and Back propagation Algorithms for Impedance Identification, In: Proceedings of the IEEE Swarm Intelligence Symposium (2007)
5. Hecht-Nielsen, R.: Neurocomputing, Addison-Wesley, Reading, PA, (1990)
6. Giles, C. L., Lawrence, S., and Tsoi, A. C.: Rule inference for financial prediction using recurrent neural networks, In: IEEE Conference on Computational Intelligence for Financial Engineering, IEEE Press. (1997)
7. Li, S., Wunsch II, D. C., O'Hair, E., and Giesselmann, M. G.: Wind turbine power estimation by neural networks with Kalman filter training on a SIMD parallel machine, In: International Joint Conference on Neural Networks, (1999)
8. Coulibay, P., Anctil, F., and Rousselle, J.: Real-time short-term water inflows forecasting using recurrent neural networks, In: International Joint Conference on Neural Networks, (1999)
9. Gudise, V.G. and Venayagamoorthy, G.K.: Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks, In: IEEE Swarm Intelligence Symposium, pp. 110--117. April (2003)
10. Janson, D.J. and Frenzel, J.F.: Training product unit neural networks with genetic algorithms, J. IEEE Intelligent Systems and Their Applications, vol. 8 (5), pp. 26--33, Oct. (1993)
11. Salerno, J., Using the particle swarm optimization technique to train a recurrent neural model, In: Ninth IEEE International Conference on Tools with Artificial Intelligence, pp. 45--49, Nov. (1997)
12. Werbos, P.: Back propagation through time: what it does and how to do it, In: Proceedings of the IEEE, vol. 78 (10), pp. 1550--1560. (1990)
13. Temurtas, F., Yumusak, N., Gunturkun, R., Temurtas, H., and Cerezci, O.: Elman's Recurrent Neural Networks Using Resilient Back Propagation for Harmonic Detection, In: 8th Pacific Rim International Conference on Artificial Intelligent Proceedings, vol. 3157, pp. 422--428. (2004)
14. Rumelhart D.E. Hinton G.E., and Williams R.J.: Learning Representations by Back– Propagating Errors Nature, vol. 323, pp. 533-536. (1986)
15. Nawi, N. M., Rehman, M. Z., and Khan, A.: A New Bat-Based Back-propagation (BAT-BP) Algorithm, In: ICSS 2013, Wrocław, Poland, September 10-12. (2013)
16. Nawi, N. M., Khan, A., and Rehman, M. Z.: A New Back-propagation Neural Network optimized with Cuckoo Search Algorithm, In: ICCSA-2013, pp. 413--426. (2013)
17. Nawi, N. M., Khan, A., and Rehman, M. Z.: A New Cuckoo Search based Levenberg-Marquardt (CSLM) Algorithm, In: ICCSA-2013, pp. 438--451. (2013)
18. Nawi, N. M., Khan, A., and Rehman, M. Z.: A New Levenberg-Marquardt based Back-propagation Algorithm trained with Cuckoo Search, In: ICEEI-2013, pp.18--24. (2013)
19. Lahmiri, S.: A comparative study of backpropagation algorithms in financial prediction. J. International Journal of Computer Science, Engineering and Applications (IJCSEA), vol.1 (4). (2011)

20. Nawi, N. M., Ransing, R. S., Salleh, M.N.M., Ghazali,R., and Hamid, N.A, An improved back propagation neural network algorithm on classification problems, J. Communications in Computer and Information Science vol. 118, pp. 177--188. (2010)
21. Nawi, N.M., Ghazali, R., Salleh, M.N.M. The development of improved back-propagation neural networks algorithm for predicting patients with heart disease, J. LNCS, vol. 6377 (4), pp. 317--324. (2010)
22. Yang. XS., and Deb. S.: Cuckoo search via Lévy flights, In: World Congress on Nature & Biologically Inspired Computing, India, pp. 210--214. (2009)
23. Yang, X. S., and Deb, S.: Engineering Optimisation by Cuckoo Search, Int. J. of Mathematical Modelling and Numerical Optimisation, vol. 1 (4), pp. 330-- 343. (2010)
24. Tuba, M., Subotic, M., Stanarevic, N.: Modified cuckoo search algorithm for unconstrained optimization problems, In: European Computing Conference, pp. 263--268. (2011)
25. Tuba, M., Subotic, M., Stanarevic, N.: Performance of a Modified Cuckoo Search Algorithm for Unconstrained Optimization Problems, J. Faculty of Computer Science, vol. 11 (2), pp. 62--74. (2012)
26. Chaowanawate, K., and 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)
27. Pavlyukevich, I.: Levy flights, non-local search and simulated annealing, J. Journal of Computational Physics, vol. 226 (2), pp. 1830--1844. (2007)
28. Walton, S., Hassan, O., Morgan, K., and Brown, M.: Modified cuckoo search: A new gradient free optimisation algorithm. J. Chaos, Solitons& Fractals, vol. 44 (9), pp. 710--718. (2011)
29. Williams, R. J. and Peng, J.: An efficient gradient-based algorithm for on-line training ofrecurrent network trajectories, J. Neural Computation, vol.2, pp. 490--501. (1990)
30. Nandy, S., Sarkar, P. P., and Das, A.: Training a Feed-forward Neural Network with Artificial Bee Colony Based Backpropagation Method, J. International Journal of Computer Science & Information Technology (IJCSIT), vol. 4 (4), pp. 33--46. (2012)
31. Ozturk, C., and Karaboga, D.: Hybrid Artificial Bee Colony algorithm for neural network training, In: IEEE Congress of Evolutionary Computation (CEC), pp. 84--88. (2011)
32. Karaboga, D., and Ozturk, C..: Neural networks training by artificial bee colony algorithm on pattern classification, J. Neural Network World, vol. 19, no. 10, pp. 279--292, (2009).