

Optimizing Weights in Elman Recurrent Neural Networks with Wolf Search Algorithm

Nazri Mohd. Nawi^(✉), M.Z. Rehman, Norhamreeza Abdul Hamid,
Abdullah Khan, Rashid Naseem, and Jamal Uddin

Faculty of Computer Science and Information Technology,
Soft Computing and Data Mining Centre (SMC),
Universiti Tun Hussein Onn Malaysia (UTHM),
P.O. Box 101 Parit Raja, 86400 Batu Pahat, Johor, Malaysia
nazri@uthm.edu.my, zrehman862060@gmail.com,
norhamreeza@gmail.com, abdullahdirvi@gmail.com,
rnsqau@gmail.com, jamalmaths2014@gmail.com

Abstract. This paper presents a Metahybrid algorithm that consists of the dual combination of Wolf Search (WS) and Elman Recurrent Neural Network (ERNN). ERNN is one of the most efficient feed forward neural network learning algorithm. Since ERNN uses gradient descent technique during the training process; therefore, it is not devoid of local minima and slow convergence problem. This paper used a new metaheuristic search algorithm, called wolf search (WS) based on wolf's predatory behavior to train the weights in ERNN to achieve faster convergence and to avoid the local minima. The performance of the proposed Metahybrid Wolf Search Elman Recurrent Neural Network (WRNN) is compared with Bat with back propagation (Bat-BP) algorithm and other hybrid variants on benchmark classification datasets. The simulation results show that the proposed Metahybrid WRNN algorithm has better performance in terms of CPU time, accuracy and MSE than the other algorithms.

Keywords: Elman recurrent network · Wolf search algorithm · Local minima · Metaheuristic optimization · Nature inspired algorithms

1 Introduction

Artificial Neural Networks (ANN) consists of interconnected nodes that simulates that calculation process inside human brain [1–3]. This interconnected calculation process makes ANN flexible enough to perform large complex problems such as pattern recognition, engineering, biological modelling, health & medicine, decision control, manufacturing & management, marketing, and sea & space modelling etc. [4–10].

Usually ANN structures are fully connected feed-forward networks that starts from one layer through the connected layer and finally generates results on the output layer [4, 11]. This ability of learning makes ANN quite useful but an addition of short-term memory makes them more powerful and dynamic. Recurrent Neural Networks (RNN) provides short-term memory or states and show dynamic temporal behavior [11].

RNN have the ability to store earlier patterns thus paving way for the modelling of active systems [12, 13].

Until today, full and partial RNN have been extensively used to conduct associative memories, spatio-temporal pattern classification, optimization and prediction [14–19]. When the ease of use is considered, partial recurrent neural network (RNN) especially Elman recurrent neural network (ERNN) memorize previous states, executes, and learn very efficiently. ERNN performs efficiently on normal networks but when network size becomes larger, the calculation becomes more complex thus making ERNN more prone to converging to local minima. It also consumes more memory during experimentation due to extra connection weights [11].

Therefore, to overcome the problem of inefficient weights that cause the ERNN to converge to local minima due to flat surfaces and to avoid CPU overheads. This paper proposed a hybrid Wolf Search Elman Recurrent Neural Network (WRNN). The proposed WRNN algorithm is tested on classification datasets and compared with Artificial Bee Colony Back Propagation (ABCBP) [20], Artificial Bee Colony-Levenberg-Marquardt algorithm (ABC-LM) [21], and Bat with Back-Propagation [22] algorithms. The selected performance metrics are mean squared error (MSE), CPU time, accuracy, and convergence rate.

The remaining paper is organized as follows: Sect. 2 explains the Wolf Search Recurrent Neural Network's (WRNN) implementation. Meanwhile, the Sect. 3 discuss the results and finally, the paper is concluded in the Sect. 4.

2 Wolf Search (WS) Algorithm

Wolf Search (WS) algorithm is a metaheuristic algorithm inspired by the silent predatory behavior of the wolves and the intelligent avoidance of the enemy. Proposed by Rui Tang [23], WS follows three idealized rules for finding the best solution or prey;

- (a) Each wolf has a fixed visual area with a radius. In 2D, the coverage would simply be the area of a circle by the radius v . In hyper-plane, where multiple attributes dominate, the distance would be estimated by Minkowski distance.
- (b) The result or the fitness of the objective function represents the quality of the wolf's current position. The wolf always tries to move to better terrain but rather than choose the best terrain it opts to move to better terrain that already houses a companion. If the new position is better, the incentive is stronger provided that it is already inhabited by a companion wolf.
- (c) At some point, it is possible that the wolf will sense an enemy. The wolf will then escape to a random position far from the threat and beyond its visual range.

2.1 The Proposed WRNN Algorithm

In the proposed Wolf Search Elman Recurrent Neural Network (WRNN) algorithm, each best position in WS algorithm represents a possible solution (i.e., the initial weight space and the corresponding biases for Elman Recurrent Neural Network (ERNN)).

The weight optimization problem and the size of the population represents the quality of the solution. In the first epoch, the best weights and biases are initialized with WS and then those weights are passed on to the ERNN. The weights in the ERNN are calculated. In the next cycle, WS updates the weights with the best possible solution and WS will continue searching the best weights until the last cycle/epoch of the network is reached or either the MSE is achieved. Figure 1 shows the proposed flowchart for the WRNN algorithm.

The WS is a population based optimization algorithm. It starts with a random initial population. In the proposed WRNN algorithm, the weight value of a matrix is calculated as following;

$$W_n = U_n = \sum_{n=1}^N a.(rand - \frac{1}{2}) \quad (1)$$

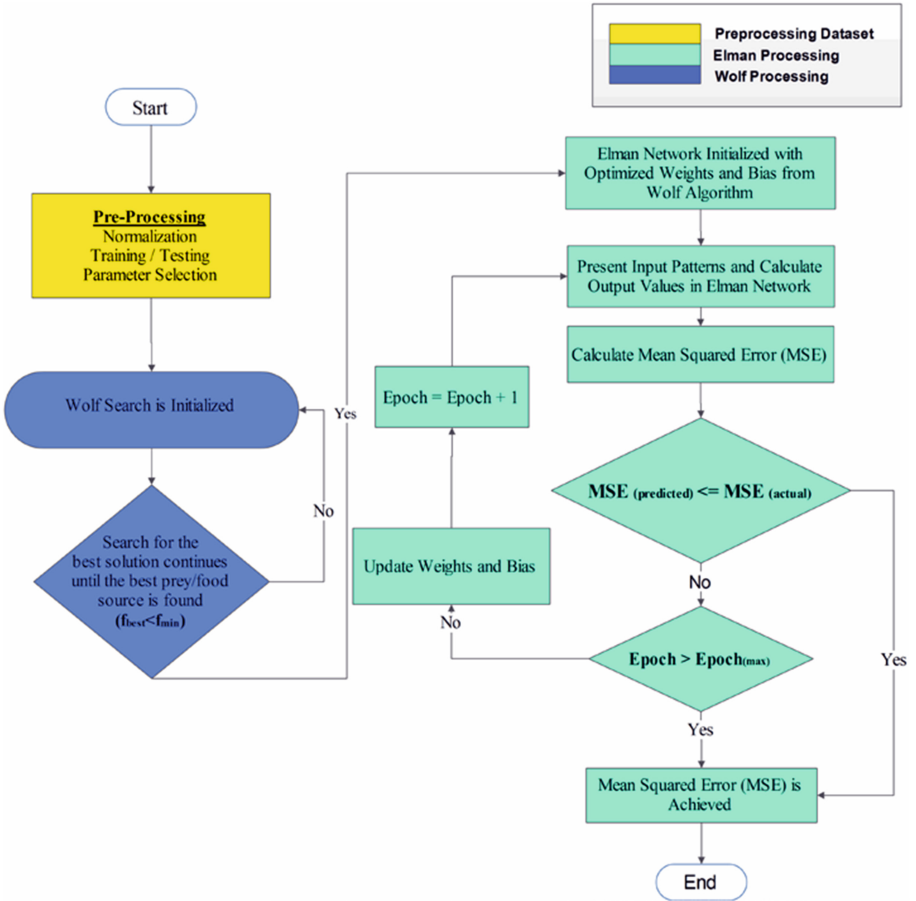


Fig. 1. The flowchart of the proposed WRNN algorithm

$$B_n = \sum_{n=1}^N a \cdot (rand - \frac{1}{2}) \quad (2)$$

Where, $W_n = N^{th}$ is the weight value in a weight matrix. The *rand* in the Eq. (1) is the random number between $[0, 1]$, a is any constant parameter for the proposed method it being less than one, and B_n is bias value. So the list of weight matrix is as follows;

$$W^c = [W_n^1, W_n^2, W_n^3, \dots, W_n^{N-1},] \quad (3)$$

Now from neural network process, MSE is easily calculated from every weight matrix, in W^c . For the ERN structure three layered network consisting of one input layer, one hidden or 'state' layer, and one 'output' layer is used. Each layer 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.

$$net_j(t) = \sum_i^n x_i(t)w_{n(ji)} + B_{n(j)} \quad (4)$$

Where n the number of inputs is, $B_{n(j)}$ is a bias. In a simple recurrent network (RN), 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 ;

$$y_j(t) = f(net_j(t)) \quad (5)$$

$$net_j(t-1) = \sum_i^n x_i(t)W_{n(ji)} + \sum_l^m y_l(t-1)U_{n(jl)} + B_{n(j)} \quad (6)$$

$$y_j(t-1) = f(net_j(t-1)) \quad (7)$$

Where, m is the number of 'state' nodes. The output of the network in both cases is determined by the state and a set of output weights, W ;

$$net_k(t) = \sum_j^M y_j(t-1)w_{n(kj)} + B_{n(k)} \quad (8)$$

$$Y_k(t) = g(net_k(t)) \quad (9)$$

Where, g is an output function. So, the error can be calculated as;

$$E = (T_k - Y_k) \quad (10)$$

The performances index for the network is calculated as;

$$V_F(x) = \frac{1}{2} \sum_{k=1}^K E^T \cdot E \quad (11)$$

In the proposed method, the MSE is considered as the performance index and calculated as;

$$V_{\mu}(x) = \frac{\sum_{j=1}^N V_F(x)}{P_i} \quad (12)$$

Where, y_r is the output of the network when the k^{th} input net_i is presented. And $E = (T_k - Y_k)$ is the error for the k^{th} input, $V_{\mu}(x)$ is the average performance, $V_F(x)$ is the performance index, and P_i is the number of Wolf population in i^{th} iteration. At the end of each epoch, the list of average sum of Mean Square Error of i^{th} iteration MSE can be calculated as;

Begin

Step 1: Initialize Wolf population size, dimensions and ERNN structure

Step 2: Load the training and testing data

Step 3: **While** MSE < stopping criteria

Step 4: Pass the wolf preys as weights to network

Step 5: Feed forward network runs using the weights initialized with Wolf Search

Step 6: Calculate the error using the Equation (10)

Step 7: Minimize the error by adjusting network parameter using Wolf Search.

Step 8: Generate Wolf prey (x_j) by using Brownian motion from a certain position.

Step 9: If enemy detected, abandon the current position and jump to a safe distance.

Step 10: Evaluate the fitness of the prey, Chose a random wolf i

If

a. $X_j > X_i$ Then

b. $x_i \leftarrow x_j$

c. $X_i \leftarrow X_j$

End if

Step 11: Wolf keeps on calculating the best possible weight at each epoch until the network is converged.

End While

Step 12: Post Process Results and Visualization.

End

Fig. 2. The pseudo code of the proposed WRNN algorithm

$$MSE_i = \{V_\mu^1(x), V_\mu^2(x), V_\mu^3(x) \dots V_\mu^n(x)\} \quad (13)$$

The Wolf search duplicates the MSE and is found when all the inputs are processed for each Wolf in the population. The Wolf search position x_j is calculated as;

$$x_j = \text{Min}\{V_\mu^1(x), V_\mu^2(x), V_\mu^3(x) \dots V_\mu^n(x)\} \quad (14)$$

And the rest of the Average MSE is considered as other wolves. A new solution x_i^{t+1} for Wolf i is generated using the following Equation;

$$x_i^{t+1} = x_{min}^t + \text{rand} * (x_{max}^t - x_{min}^t) \quad (15)$$

So, the movement of the other Wolf x_i^{t+1} toward x_j can be drawn from Eq. (16);

$$X_i = \begin{cases} x_i^{t+1} + \text{rand} \cdot \text{stepb} * (x_j - x_i^{t+1}) \\ x_i^{t+1} \text{ else} \end{cases} \quad (16)$$

Where, ∇V_i is a small movement of x_i towards x_j . The weights and biases for each layer is then adjusted as;

$$W_n^{s+1} = U_n^{s+1} = W_n^s - X_i \quad (17)$$

$$B_n^{s+1} = B_n^s - X_i \quad (18)$$

The pseudo code for the WRNN is given in the Fig. 2:

3 Results and Discussion

Basically, the main focus of this paper is to compare different algorithms based on mean squared error (MSE), accuracy during network convergence and the CPU time. Before discussing the simulation results, there are certain things that needs to be explained such as tools and technologies, network topologies, testing methodology and the classification problems used for the entire experimentation. The discussion is as follows:

3.1 Preliminary Study

In order to demonstrate the performance of the proposed BARNN algorithm for training RNN. The proposed algorithms are tested on breast cancer, thyroid, Iris and Australian credit card datasets taken from University of California, Irvine Machine Learning Repository (UCIMLR). The simulation experiments are carried out on Intel Core i5 Processor with 8 GB main memory. Simulation software is MATLAB 2012 running on Windows 7 home edition. The performance measure for each algorithm is based on the Mean Squared Error (MSE), standard deviation and accuracy. The three

layers feed forward neural network architecture (i.e. input layer, one hidden layer, and output layers.) is used for each problem. The number of hidden nodes is kept fixed to 5 node. In the network structure, the bias nodes and the log-sigmoid activation function are used. For each problem, trial is limited to 1000 epochs. A total of 20 trials are run for each dataset with target error set to 0.0001. The network results are stored in the result file for each trial.

3.2 Classification Datasets

The first datasets selected is breast cancer that consists of a total 699 instances [24]. The selected network architecture used for this dataset has 9 inputs nodes, 5 hidden nodes and 2 output nodes. Table 1, shows the simulation results of all the algorithms used in this study. From the Table 1, it can be easily seen that the proposed WRNN algorithm achieves better convergence than the ABCBP, ABC-LM, and Bat-BP in terms of MSE, CPU time, epochs and accuracy. The proposed WRNN converged to global minima with an MSE of 0.0001, and 99.99% accuracy.

The second dataset for classification is Thyroid which consists of a total of 7200 instances [25]. The selected network architecture for Thyroid has 21 input nodes, 5 hidden nodes and 3 output nodes. The simulation result of Thyroid classification problem are shown in the Table 1. From Table 1, it can be seen that the proposed

Table 1. Summary of algorithms performance on classification problems

	Algorithms	Epochs	CPU Time	Accuracy	MSE
Breast Cancer	ABC-BP	1000	1482.9	92.02	0.184
	ABC-LM	1000	1880.64	93.83	0.0139
	BAT-BP	1000	345.42	97.81	0.0219
	WRNN	37	79	99.99	0.0001
Thyroid	ABC-BP	1000	1747.23	93.28	0.046
	ABC-LM	1000	1382.91	91.66	0.0409
	BAT-BP	1000	4610.79	99.35	0.01
	WRNN	114	1563	99.99	5.33E-05
IRIS	ABC-BP	1000	156.43	86.88	0.155
	ABC-LM	1000	171.52	79.56	0.058
	BAT-BP	1000	475.38	98.06	0.021
	WRNN	105	161	99.99	6.62E-05
Australian	ABC-BP	1000	6894	89.99	0.173
Credit Card	ABC-LM	1000	4213	77.78	0.05
Approval	BAT-BP	967.03	892.31	99.67	0.0033
	WRNN	44	177	99.99	4.90E-05
Note: The highlighted area represents the proposed WRNN algorithm					

WRNN algorithm outperforms the other comparison algorithms in terms of epochs, CPU time, MSE, and accuracy. The proposed algorithm converged to the global minima within a mere 114 epochs with an MSE of $5.33\text{E-}05$.

Collected by Fisher, IRIS classification datasets is one of the best pattern recognition dataset in use since 1936 [26]. It consists of 150 samples collected from three Iris species. The selected network architecture for Iris has 4 inputs, 5 hidden, and 3 outputs nodes. Table 1 shows the MSE, SD, Epochs, CPU time and accuracy of the proposed WRNN algorithm when compared with other hybrid variants. From the Table 1, it is clear that the proposed WRNN algorithm has better performance in terms of MSE, CPU time, and accuracy. WRNN converged with an MSE of $6.62\text{E-}05$, an accuracy of 99.99%, and within 105 epochs when compared with the other algorithms.

The last dataset is Australian Credit Card Approval which consists of 690 instances [27]. The network architecture selected has 51 inputs, 5 hidden nodes, and 2 outputs. This dataset contains the information of the bank clients. All the attributes are changed to hide the client's data. Table 1 shows the proposed WRNN algorithm's performance comparison algorithms with other algorithms. From Table 1, it can be clearly realized that WRNN achieved higher accuracy of 99.99% with an MSE of $4.90\text{E-}05$ within 44 epochs. Whereas, the other algorithms fell behind the proposed WRNN algorithm in terms of MSE, accuracy and epochs.

4 Conclusion

In this paper, Nature Inspired Metaheuristic Wolf search algorithm is used to train the weights in Elman Recurrent Neural Network (ERNN). The proposed Metahybrid Wolf Search with Elman Recurrent Neural Network (WRNN) algorithm has been able to overcome the inherent slow convergence problem of the ERNN. The performance of the proposed WRNN is compared with Bat with back propagation (Bat-BP) and other hybrid variant algorithms on benchmark classification datasets taken from UCIMLR. The simulation results show that the proposed WRNN algorithm has better convergence performance with high accuracy, small MSE and less CPU time than other algorithms.

Acknowledgments. The Authors would like to thank Office of Research, Innovation, Commercialization and Consultancy (ORICC), Universiti Tun Hussein Onn Malaysia (UTHM) and Ministry of Education (MOE) Malaysia for financially supporting this Research under Fundamental Research Grant Scheme (FRGS) vote no. 1236. This research is also supported by Gates IT Solution Sdn. Bhd under its publication scheme.

References

1. Nawi, N.M., Khan, A., Rehman, M.Z.: A new optimized Cuckoo Search Recurrent Neural Network (CSRNN) algorithm. In: The 8th International Conference on Robotic, Vision, Signal Processing and Power Applications, pp. 335–341. Springer, Singapore (2013)

2. Radhika, Y., Shashi, M.: Atmospheric temperature prediction using support vector machines. *Int. J. Comput. Theory Eng.* **1**, 55–58 (2009)
3. Akcayol, M.A., Cinar, C.: Artificial neural network based modeling of heated catalytic converter performance. *Appl. Therm. Eng.* **25**, 2341–2350 (2005)
4. Rehman, M.Z., Nawi, N.M.: Improving the accuracy of Gradient Descent Back Propagation algorithm (GDAM) on classification problems. *Int. J. New Comput. Archit. Appl.* **1**, 861–870 (2011)
5. Kosko, B.: *Neural Network and Fuzzy System*. Prentice Hall, Upper Saddle River (1994)
6. Krasnopolsky, V.M., Chevallier, F.: Some neural network applications in environmental sciences. part II: advancing computational efficiency of environmental numerical models. *Neural Netw.* **16**, 335–348 (2003)
7. Coppin, B.: *Artificial Intelligence Illuminated*. Jones and Bartlett Publishers Inc., Sudbury (2004)
8. Basheer, I.A., Hajmeer, M.: Artificial neural networks: fundamentals, computing, design, and application. *J. Microbiol. Methods* **43**, 3–31 (2000)
9. He, Z., Wu, M., Gong, B.: Neural network and its application on machinery fault diagnosis. In: *IEEE International Conference on Systems Engineering*. pp. 576–579 (1992)
10. Li, B., Chow, M.Y., Tipsuwan, Y., Hung, J.C.: Neural-network-based motor rolling bearing fault diagnosis. *IEEE Trans. Ind. Electron.* **47**, 1060–1069 (2000)
11. Nawi, N.M., Khan, A., Rehman, M.Z.: CSBPRNN: a new hybridization technique using cuckoo search to train back propagation recurrent neural network. In: Herawan, T., Deris, M. M., Abawajy, J. (eds.) *Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013)*. LNEE, vol. 285, pp. 111–118. Springer, Singapore (2014). doi:[10.1007/978-981-4585-18-7_13](https://doi.org/10.1007/978-981-4585-18-7_13)
12. Zhang, J., Lok, T., Lyu, M.R.: A hybrid particle swarm optimization-back-propagation algorithm for feedforward neural network training. *Appl. Math. Comput.* **185**, 1026–1037 (2007)
13. Ab Aziz, M.F., Hj Shamsuddin, S.M., Alwee, R.: Enhancement of particle swarm optimization in elman recurrent network with bounded Vmax function. In: *Proceedings 2009 3rd Asia International Conference on Modelling and Simulation*, AMS 2009, pp. 125–130 (2009)
14. Sutskever, I., Hinton, G., Taylor, G.: The Recurrent temporal restricted Boltzmann machine. *Neural Inf. Process. Syst.* **21**, 1601–1608 (2008)
15. Gupta, L., McAvoy, M., Phegley, J.: Classification of temporal sequences via prediction using the simple recurrent neural network. *Pattern Recognit.* **33**, 1759–1770 (2000)
16. Saad, E.W., Prokhorov II, D., Donald, C.W.: Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks. *IEEE Trans. Neural Netw.* **9**, 1456–1470 (1998)
17. Guo, L., Rivero, D., Pazos, A.: Epileptic seizure detection using multiwavelet transform based approximate entropy and artificial neural networks. *J. Neurosci. Methods* **193**, 156–163 (2010)
18. Güler, N.F., Übeyli, E.D., Güler, I.: Recurrent neural networks employing Lyapunov exponents for EEG signals classification. *Expert Syst. Appl.* **29**, 506–514 (2005)
19. Übeyli, E.D.: Recurrent neural networks employing Lyapunov exponents for analysis of doppler ultrasound signals. *Expert Syst. Appl.* **34**, 2538–2544 (2008)
20. 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. Springer, Heidelberg (2007). doi:[10.1007/978-3-540-73729-2](https://doi.org/10.1007/978-3-540-73729-2)

21. Karaboga, D., Akay, B.: A comparative study of Artificial Bee Colony algorithm. *Appl. Math. Comput.* **214**, 108–132 (2009)
22. Nawi, N.M., Rehman, M.Z., Khan, A.: A new Bat Based Back-Propagation (BAT-BP) algorithm. In: Swiątek, J., Grzech, A., Swiątek, P., Tomczak, J.M. (eds.) *Advances in Systems Science*, pp. 395–404. Springer, Cham (2014)
23. Tang, R., Fong, S., Yang, X.-S., Deb, S.: Wolf search algorithm with ephemeral memory. In: *Seventh International Conference on Digital Information Management (ICDIM 2012)*, pp. 165–172 (2012)
24. Wolberg, W.H., Mangasarian, O.L.: Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proc. Natl. Acad. Sci. U.S.A.* **87**, 9193–9196 (1990)
25. Quinlan, J.R.: Induction of decision trees. *Mach. Learn.* **1**, 81–106 (1986)
26. Fisher, R.: The use of multiple measurements in taxonomic problems. *Ann. Eugen.* **7**, 179–188 (1936)
27. Quinlan, J.R.: Simplifying decision trees. *Int. J. Man-Mach. Stud. Spec. Issue: Knowl. Acquisition Knowl.-Based Syst. Part 5* **27**(3), 221–234 (1987)

Recent Advances on Soft Computing and Data Mining
The Second International Conference on Soft
Computing and Data Mining (SCDM-2016), Bandung,
Indonesia, August 18-20, 2016 Proceedings
Herawan, T.; Ghazali, R.; Nawi, N.M.; Deris, M.M. (Eds.)
2017, XXI, 649 p. 215 illus., 117 illus. in color., Softcover
ISBN: 978-3-319-51279-2