Countering the problem of oscillations in Bat-BP gradient trajectory by using momentum

Nazri Mohd. Nawi[1], M. Z. Rehman[1], Abdullah Khan[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, Batu Pahat, Johor Darul Takzim, Malaysia.

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

Abstract. Metaheuristic techniques have been recently used to counter the problems like slow convergence to global minima and network stagnancy in back-propagation neural network (BPNN) algorithm. Previously, a meta-heuristic search algorithm called Bat was proposed to train BPNN to achieve fast convergence in the neural network. Although, Bat-BP algorithm achieved fast convergence but it had a problem of oscillations in the gradient path, which can lead to sub-optimal solutions. In-order to remove oscillations in the BAT-BP algorithm, this paper proposed the addition of momentum coefficient to the weights update in the Bat-BP algorithm. The performance of the modified Bat-BP algorithm is compared with simple Bat-BP algorithm on XOR and OR datasets. The simulation results show that the convergence rate to global minimum in modified Bat-BP is highly enhanced and the oscillations are greatly reduced in the gradient path.

Keywords: metaheuristics, slow convergence, global minima, bat algorithm, back-propagation neural network algorithm, Bat-BP algorithm, momentum.

1 Introduction

Back-propagation Neural Network (BPNN) is a very old optimization technique applied on the Artificial Neural Networks (ANN) to speed up the network convergence to global minima during training process [1-3]. BPNN follows the basic principles of ANN which mimics the learning ability of a human brain. Similar to ANN architecture, BPNN consists of an input layer, one or more hidden layers and an output layer of neurons. In BPNN, every node in a layer is connected to every other node in the adjacent layer. Unlike normal ANN architecture, BPNN learns by calculating the errors of the output layer to find the errors in the hidden layers [4]. This qualitative ability makes it highly suitable to be applied on problems in which no relationship is found between the output and the inputs. Due to its high rate of plasticity and learning capabilities, it has been successfully implemented in wide range of applications [5].

T. Herawan et al. (eds.), *Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013)*, Lecture Notes in Electrical Engineering 285, DOI: 10.1007/978-981-4585-18-7_12,

Despite providing successful solutions BPNN has some limitations. Since, it uses gradient descent learning which requires careful selection of parameters such as network topology, initial weights and biases, learning rate, activation function, and value for the gain in the activation function. An improper use of these parameters can lead to slow network convergence or even network stagnancy [5-6]. Previous researchers have suggested some modifications to improve the training time of the network. Some of the variations suggested are the use of learning rate and momentum to stop network stagnancy and to speed-up the network convergence to global minima. These two parameters are frequently used in the control of weight adjustments along the steepest descent and for controlling oscillations [7].

Besides setting network parameters in BPNN, evolutionary computation is also used to train the weights to avoid local minima. To overcome the weaknesses of gradient-based techniques, many new algorithms have been proposed recently. These algorithms include global search techniques such as hybrid PSO-BP [8], artificial bee colony back-propagation (ABC-BP) algorithm [9-10], evolutionary artificial neural networks algorithm (EA) [11], genetic algorithms (GA) [12] and Bat based back-propagation (Bat-BP) algorithm [13] etc. Unlike other algorithms, Bat-BP algorithm [13] shows high accuracy and avoids local minima. But, still some oscillations are detected in the gradient trajectory. In-order to avoid extreme changes in the gradient due to local anomalies [4-6], momentum coefficient is introduced in Bat-BP algorithm [13].

In this paper, Bat-BP with momentum is compared with simple Bat-BP algorithm and validated on XOR and OR datasets. We find that by using an appropriate metaheuristic technique such as BAT with BPNN enhanced with momentum coefficient can answer many limitations of gradient descent efficiently. The next two sections provide a brief discussion on BAT algorithm, followed by the proposed BAT-BP with momentum algorithm, simulation results, and conclusions.

2 The Bat Algorithm

Bat is a metaheuristic optimization algorithm developed by Xin-She Yang in 2010 [14]. Bat algorithm is based on the echolocation behavior of microbats with varying pulse rates of emission and loudness. Yang [14] has idealized the following rules to model Bat algorithm;

1) All bats use echolocation to sense distance, and they also “know” the difference between food/prey and back-ground barriers in some magical way.
2) A bat fly randomly with velocity (v_i) at position (x_i) with a fixed frequency ($f_{\min}$), varying wavelength λ and loudness A_0 to search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission $r \in [0,1]$, depending on the proximity of their target.
3) Although the loudness can vary in many ways, Yang [14] assume that the loudness varies from a large (positive) $A0$ to a minimum constant value $Amin$.

Firstly, the initial position xi, velocity vi and frequency fi are initialized for each bat bi. For each time step t, the movement of the virtual bats is given by updating their velocity and position using Equations 2 and 3, as follows:

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

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

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

Where β denotes a randomly generated number within the interval [0,1]. Recall that x_i^t denotes the value of decision variable j for bat i at time step t. The result of fi in Equation 1 is used to control the pace and range of the movement of the bats. The variable x^* represents the current global best location (solution) which is located after comparing all the solutions among all the n bats. In order to improve the variability of the possible solutions, Yang [12] has employed random walks. Primarily, one solution is selected among the current best solutions for local search and then the random walk is applied in order to generate a new solution for each bat;

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

Where, A^t stands for the average loudness of all the bats at time t, and $\epsilon \in [-1,1]$ is a random number. For each iteration of the algorithm, the loudness A_i and the emission pulse rate r_i are updated, as follows:

$$A_i^{t+1} = \propto A_i^t \tag{5}$$

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

Where α and γ are constants. At the first step of the algorithm, the emission rate, r_i^0 and the loudness, A_i^0 are often randomly chosen. Generally, $A_i^0 \epsilon [1,2]$ and $r_i^0 \epsilon [0,1]$[12].

3 The Proposed BAT-BP Algorithm

BAT is a population based optimization algorithm, and like other meta-heuristic algorithms, it starts with a random initial population. In Bat algorithm, each virtual bat flies randomly with a velocity v_i at some position x_i, with a varying frequency f_i and loudness A_i, as explained in the Section 2. As, it searches and finds its prey, it changes frequency, loudness and pulse emission rate r_i. Search is intensified by a local random walk. Selection of the best continues until stopping criterion are met. To control the dynamic behavior of a swarm of bats, Bat algorithm uses a frequency-tuning technique and the searching and usage is controlled by changing the algorithm-dependent parameters [14].

In the proposed BAT-BP algorithm [13], each position 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 position of a food source represent the quality of the solution. In the first epoch, the best weights and biases are initialized with BAT and then those weights are passed on to the BPNN where momentum coefficient, α is appended. The weights in BPNN are calculated and compared in the reverse cycle. In the next cycle BAT will again update the weights with the best possible solution and BAT 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 Bat-BP algorithm is shown in the Figure 1:

Step 1: *BAT initializes and passes the best weights to BPNN*
Step 2: *Load the training data*
Step 3: While *MSE < Stopping Criteria*
Step 4: *Initialize all BAT Population*
Step 5: *Bat Population finds the best weight in Equation 4 and pass it on to the network, the weights,* w_{ij} $with\ momentum, \alpha$ *and biases,* b_i *in BPNN are then adjusted using the following formulae;*

$$w_{ij}(k+1) = (w_{ij}k + \mu\partial_j y_i)\alpha_{ij}$$
$$b_i(k+1) = b_i k + \mu\partial_j$$

Step 6: *Feed forward neural network runs using the weights initialized with BAT*
Step 7: *Calculate the backward error*
Step 8: *Bat keeps on calculating the best possible weight at each epoch until the Network is converged.*
End While

Fig. 1. Pseudo code of the proposed Bat-BP algorithm

4 Results and Discussions

The simulations are carried-out on workstation equipped with a 2.33GHz Core-i5 processor, 4-GB of RAM, Microsoft Windows 7 and MATLAB 2010. Three datasets such as 2-bit XOR, 3-Bit XOR and 4-bit OR were used. The performance of the Simple BAT BP [13] algorithm is analyzed and compared with the modified BAT-BP algorithm. Three layer back-propagation neural networks is used for testing of the models, the hidden layer is kept fixed to 10-nodes while output and input layers nodes vary according to the datasets 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. Momentum coefficient of 0.03 is found to be optimal for the weight updating in modified Bat-BP algorithm. A total of 20 trials, each trial consisting of 1000 epochs are run for each dataset. CPU time, average accuracy, and Mean Square Error (MSE) are recorded for each independent trials on XOR and OR datasets and stored in a separate file.

4.1 2-Bit XOR Dataset

The first test problem is the 2 bit XOR Boolean function consisting of two binary inputs and a single binary output. In simulations, we used 2-10-1 network architecture for two bit XOR. For the simple Bat-BP and modified Bat-BP, Table 1, shows the CPU time, number of epochs and the MSE for the 2 bit XOR test problem. Figure 2 shows the 'MSE performance vs. Epochs' of simple BAT-BP and modified Bat-BP for the 2-10-1 network architecture.

Table 1. CPU Time, Epochs and MSE for 2-bit XOR dataset with **2-10-1** ANN Architecture

Algorithms	**Simple BAT-BP**	**Modified Bat-BP**
CPUTIME	**2.39**	**0.44**
EPOCHS	**23.25**	**7.9**
MSE	**0**	**0**
Accuracy (%)	**100**	**100**

The modified Bat-BP algorithm avoids the local minima and converges on the provided network architecture successfully within 100 epochs as seen in the Table 1. The average CPU time is also reduced to a mere 0.44 from 2.39 CPU cycles in modified Bat-BP in Table 1. In Figure 2, the modified BAT-BP algorithm can be seen to converge within 5 epochs and shows a smooth gradient while simple BAT-BP converges within 10 epochs and shows a lot of oscillations in the trajectory path.

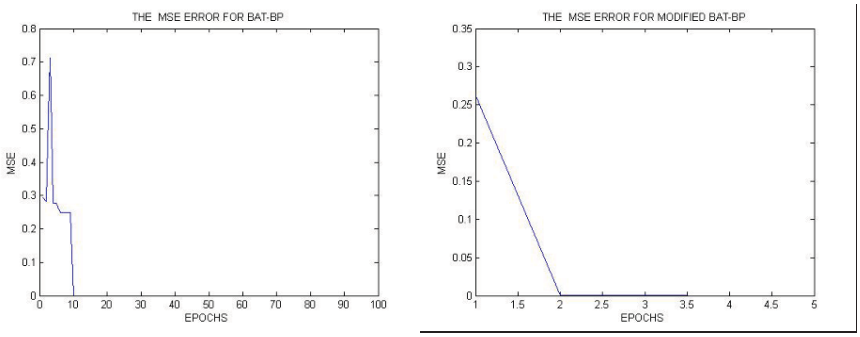

Fig. 2. (From Left to Right) Simple Bat-BP and modified Bat-BP convergence performance on 2-bit XOR with 2-10-1 ANN Architecture

4.2 3-Bit XOR Dataset

In the second phase, we used 3 bit XOR dataset consisting of three inputs and a single binary output. For the three bit input we apply 3-10-1, network architecture. The parameter range is same as used for two bit XOR problem, for the 3-10-1 the network it has forty connection weights and eleven biases. For the simple Bat-BP, and modified

Bat-BP, Table 2 shows the CPU time, number of epochs and the MSE for the 2 bit XOR test problem.

Table 2. CPU Time, Epochs and MSE for 3-bit XOR dataset with **2-10-1** ANN Architecture

Algorithms	Simple BAT-BP	Modified Bat-BP
CPUTIME	**4.05**	1.68
EPOCHS	**23**	25
MSE	**0.0625**	0.05
Accuracy (%)	**93.69**	94.99

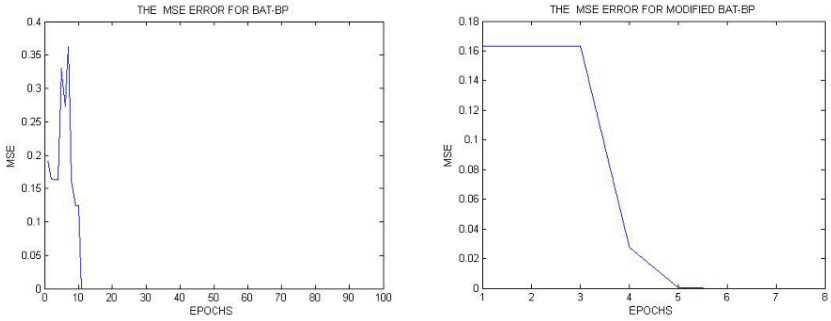

Fig. 3. (From Left to Right) Simple Bat-BP and modified Bat-BP convergence performance on 3-bit XOR with 2-10-1 ANN Architecture

In Table 2, modified BAT-BP algorithm can be seen converging with superior 1.68 CPU cycles. While 1 percent improvement in average accuracy is recorded but no improvement in MSE is detected in modified Bat-BP, as seen in the Table 2. In Figure 3, we can see the simulation results 'MSE vs. Epochs' convergence performance for 3-bit XOR dataset on simple and modified Bat-BP algorithms. Oscillations in modified Bat-BP can be seen controlled in the Figure 3.

4.3 4-Bit OR Dataset

The third dataset is based on the logical operator OR which indicates whether either operand is true. If one of the operand has a nonzero value, the result has the value 1. Otherwise, the result has the value 0. The network architecture used here is 4-10-1 in which the network has fifty connection weights and eleven biases. Table 3, illustrates the CPU time, epochs, and MSE performance of the simple Bat-BP, and modified Bat-BP, algorithms respectively. Figure 4, shows the 'MSE performance vs. Epochs' for the 4-10-1 network architecture of the proposed Bat-BP algorithm. In Figure 4, we can see that modified Bat-BP is converging within 9 epochs which is much better than the 22 epochs offered by simple Bat-BP. Also, it can be noted from the Table 3 that Bat-

BP comes with less CPU overheads and converges within 0.48 CPU cycles. Accuracy and the MSE was same for both algorithms, as shown in the Table 3.

Table 3. CPU Time, Epochs and MSE for 4-bit OR dataset with **2-10-1** ANN Architecture

Algorithms	BAT-BP	Modified Bat-BP
CPUTIME	**2.88**	0.48
EPOCHS	**46.8**	10.25
MSE	**0**	0
Accuracy (%)	**100**	100

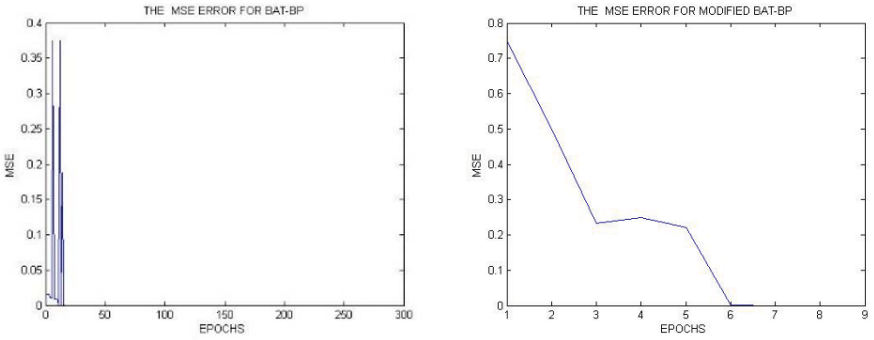

Fig. 4. (From Left to Right) Simple Bat-BP and modified Bat-BP convergence performance on 4-bit OR with 2-10-1 ANN Architecture

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.

5 Conclusions

BPNN algorithm is one of the most widely used procedure to train Artificial Neural Networks (ANN). But 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. Previously, a meta-heuristic search algorithm, called Bat was proposed to train BPNN to achieve fast convergence rate in the neural network. Although, Bat-BP [13] algorithm achieved fast convergence but it had a problem of oscillations in the gradient path, which can lead to

sub-optimal solutions. In-order to remove oscillations in the BAT-BP algorithm [13] this paper, proposed the addition of momentum coefficient to the weights update in the Bat-BP algorithm. The performance of the modified Bat-BP is compared with the simple Bat-BP [13] algorithm by means of simulations on 2-bit, 3-bit XOR and 4-bit OR datasets. The simulation results show that the modified Bat-BP algorithm converges to the global minimum successfully showing a 0 MSE, less CPU cycles and 100 percent accuracy with almost no oscillations in the gradient descent path.

References

1. Deng, W. J., Chen, W. C., and Pei, W.: Back-propagation neural network based importance-performance analysis for determining critical service attributes, J. Expert Systems with Applications, vol. 34 (2). 2008
2. Kosko, B.: Neural Network and Fuzzy Systems, 1st Edition, Prentice Hall, India. (1992)
3. Rumelhart, D. E., Hinton, G. E., and Williams, R. J.: Learning Internal Representations by error Propagation, J. Parallel Distributed Processing: Explorations in the Microstructure of Cognition. (1986)
4. Rehman, M.Z., Nawi, N. M., and Ghazali, R.: Studying the effect of adaptive momentum in improving the accuracy of gradient descent back propagation algorithm on classification problems, J. International Journal of Modern Physics (IJMPCS), vol.1 (1). (2012)
5. Nawi, N. M., Ransing, M. R., and Ransing, R. S.: An improved Conjugate Gradient based learning algorithm for back propagation neural networks, J. Computational Intelligence, vol. 4. (2007)
6. Nawi, N. M., Rehman, M. Z., and Ghazali, M. I.: Noise-Induced Hearing Loss Prediction in Malaysian Industrial Workers using Gradient Descent with Adaptive Momentum Algorithm, J.International Review on Computers and Software (IRECOS), vol. 6 (5). (2011)
7. Lee, K., Booth, D., and Alam, P. A.: Comparison of Supervised and Unsupervised Neural Networks in Predicting Bankruptcy of Korean Firms, J. Expert Systems with Applications, vol. 29. (2005)
8. Mendes, R., Cortez, P., Rocha, M., and Neves, J.: Particle swarm for feed forward neural network training. In: Proceedings of the International Joint Conference on Neural Networks, vol. 2, pp. 1895--1899. (2002)
9. 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)
10. Karaboga, D., Akay, B., and Ozturk, C.: Artificial Bee Colony (ABC) Optimization Algorithm for Training Feed-Forward Neural Networks, In: 4th International Conference on Modeling Decisions for Artificial Intelligence (MDAI 2007), Kitakyushu, Japan, August 16-18. (2007)
11. Yao, X.: Evolutionary artificial neural networks, J. International Journal of Neural Systems, vol. 4(3), pp. 203--222. (1993)
12. Montana, D. J., & Davis, L.: Training feedforward neural networks using genetic algorithms, In: Proceedings of the eleventh international joint conference on artificial Intelligence, vol. 1, pp. 762--767. (1989)
13. 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)
14. Yang, X. S.: A new metaheuristic bat-inspired algorithm, In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), pp. 65--74. (2010)