Data classification using metaheuristic Cuckoo Search technique for Levenberg Marquardt back propagation (CSLM) algorithm

Nazri Mohd. Nawi, Abdullah Khan, and M. Z. Rehman

Citation: AIP Conference Proceedings **1660**, 050068 (2015); doi: 10.1063/1.4915701
View online: http://dx.doi.org/10.1063/1.4915701
View Table of Contents: http://aip.scitation.org/toc/apc/1660/1
Published by the American Institute of Physics

Data Classification Using Metaheuristic Cuckoo Search Technique For Levenberg Marquardt Back Propagation (CSLM) Algorithm

Nazri Mohd. Nawi, Abdullah Khan, M. Z. Rehman

Software and Multimedia Centre (SMC), 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.

Abstract: A nature inspired behavior metaheuristic techniques which provide derivative-free solutions to solve complex problems. One of the latest additions to the group of nature inspired optimization procedure is Cuckoo Search (CS) algorithm. Artificial Neural Network (ANN) training is an optimization task since it is desired to find optimal weight set of a neural network in training process. Traditional training algorithms have some limitation such as getting trapped in local minima and slow convergence rate. This study proposed a new technique CSLM by combining the best features of two known algorithms back-propagation (BP) and Levenberg Marquardt algorithm (LM) for improving the convergence speed of ANN training and avoiding local minima problem by training this network. Some selected benchmark classification datasets are used for simulation. The experiment result show that the proposed cuckoo search with Levenberg Marquardt algorithm has better performance than other algorithm used in this study.

Keywords: Artificial Neural Network; back propagation; local minima; Levenberg Marquardt; Cuckoo Search.
PACS: 00

INTRODUCTION

Artificial Neural Network (ANN) is a data processing model consists of a large number of particularly interconnected processing elements (neurons) functioning together in order to solving many complex real world problems. ANN has been effectively implemented in all engineering fields such as biological modeling, decision and control, health and medicine, engineering and manufacturing, marketing, ocean exploration, predicting future trends based on the huge historical data of an organization [1-8]. Because of the pleasing appearance of ANN, large numbers of applications have been proposed in recent decades. The back propagation (BP) algorithm that was introduced by Rumelhart [9] is the well-known method for training a multilayer feed-forward artificial neural networks. However, the BP algorithm based on gradient descent method suffers from two major drawbacks: low convergence rate and unpredictability. They are caused by a chance of being trapped in a local minimum and prospect of overshooting the minimum of the error surface [10-12].

In last decade, several improved learning algorithms have been proposed to overcome the defect of gradient descent based systems. These algorithms include a direct enhancement method using heuristic technique, second ordered methods. All these algorithms are derivatives of steepest gradient search, so ANN training is relatively slow. For fast and efficient training, second order learning algorithms have to be used. The most effective method is Levenberg Marquardt (LM) algorithm, which is a derived from the Newton method. LM algorithm is create as one of the most successful algorithm in increasing the convergence speed of the ANN with MLP architectures [13]. This is quite complicated algorithm since not only the gradient but also the Jacobin matrix should be calculated. The LM algorithm was developed only for layer-by-layer ANN topology, which is far from optimal[14]. LM algorithm is ranked as one of the most efficient training algorithms for small and medium sized patterns. It is a good combination of Newton's and steepest descent methods [15]. It inherits speed from Newton method but it also has the convergence capability of steepest descent method. It suits specially in training neural network in which the performance index is calculated in Mean Squared Error (MSE), but still it is not able to avoid local minimum [16-18].

In order to overcome these issues this study proposed the new method that combining Cuckoo Search (CS) and Levenberg Marquardt algorithm to train neural network for selected benchmark classification problem. The proposed method reduces the error and improved the performance by escaping from local minima. The Cuckoo Search (CS) developed in 2009 by Yang and Deb [19-20] is a new meta-heuristic algorithm replicating animal behavior. The optimal solutions obtained by the CS are far better than the finest solutions found by particle swarm optimizers and genetic algorithms [21].

The remaining of the paper is organized as follows. The next Section describes the Cuckoo Search algorithm. In the following section, the implementation of the proposed method with LMBP will be illustrates. And further

International Conference on Mathematics, Engineering and Industrial Applications 2014 (ICoMEIA 2014)
AIP Conf. Proc. 1660, 050068-1–050068-10; doi: 10.1063/1.4915701

explain the performance of the proposed method on some experimental data. The paper is concluded in final section.

THE CUCKOO SEARCH ALGORITHM

Cuckoo Search (CS) algorithm is a novel meta-heuristic technique proposed by Xin-She Yang [22]. This algorithm was stimulated by the obligate brood parasitism of some cuckoo species by laying their eggs in the nests of other host birds. The CS algorithm follows the three idealized rules:

1. Each cuckoo lays one egg at a time, and put its egg in randomly chosen nest;
2. The best nests with high quality of eggs will carry over to the next generations;
3. 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* [0, 1].

In the CS algorithm, the initial solutions have been determined by using Equation (1)

$$X_k = rand \cdot (up_i - low_i) + low_i \tag{1}$$

Where, up_i and low_i are the upper and lower bounds for each decision variable, respectively. And X_k is D dimension solution vector. The CS algorithm controls the boundary conditions in each computation steps. The general system equation of the CS algorithm is given in Equation (2);

$$X_{k(i+1)} = X_{ki} + \theta \otimes levy(\gamma) \tag{2}$$

Where, $\theta > 0$ is the step-size. The product $\otimes$ means entry-wise multiplications. The Levy flights basically provide a random walk while their random steps are drawn from a Levy distribution;

$$Levy \sim U = K^{-\gamma} \ (1 < \gamma \leq 3) \tag{3}$$

Which has an infinite variance with an infinite mean. Before starting to iterative search process, the CS algorithm detects the most successful solution as X_{best} solution. The iterative evolution phase begins with the detection step of the σ_u using Equation (4);

$$\sigma_u = \left\{ \frac{\Gamma(1+\mu).\sin(\tau.\mu/2)}{\Gamma\left[\left[\frac{(1+\mu)}{2}\right].\mu.2^{\frac{(\mu-1)}{2}}\right]} \right\}^{\frac{1}{\mu}} \tag{4}$$

In (4), the Γ denotes gamma function. The evolution phase of the X_i solution by defining the donor vector V, where $V = X_i$ After this step, the required step size value has been computed by using Equation (5);

$$Stepsize_j = 0.01 \cdot \left(\frac{U_j}{|V_j|^{\frac{1}{\mu}}}\right).(V - X_{best}) \tag{5}$$

Where $U = \sigma_u.rand[D]$ and $V = rand[D]$. The $rand[D]$ is function generated an integer within range of [1, D]. In the next step of CS algorithm the pattern V is mutated by using Equation (6);

$$V = V + Stepsize_j \cdot rand[D] \tag{6}$$

The update process of the X_{best} pattern in the CS algorithm is defined by Equation (7);

$$X_{best} \leftarrow f(X_{best}) \leq f(X_k) \tag{7}$$

The impractical patterns are manipulated by using the crossover operator given in Equation (8);

$$V = \begin{cases} X_k + rand \cdot (X_{r1} - X_{r2}) & rand_k > p\alpha \\ X_k & else \end{cases} \tag{8}$$

The algorithmic control parameters of the CS algorithm are the scale factor (μ) and mutation probability value ($p\alpha$).

THE PROPOSED CSLM ALGORITHM

In the proposed Cuckoo Search Levenberg-Marquardt (CSLM) algorithm, each best nest or solution represents a possible solution (i.e., the weight space and the corresponding biases for NN optimization in this study) to the considered problem and the size of a population represents the quality of the solution. The initialization of weights is compared with output and the best weight cycle is selected by cuckoo. The cuckoo will continue searching until the last cycle to find the best weights for the network. The solution that is neglected by the cuckoo is replaced with a new best nest. The main idea of this combined algorithm is that CS algorithm is used at the beginning stage of searching for the optimum to select the best weights. Then, the training process is continued with the LM algorithm using the best weights of CS algorithm.

The Analysis of Proposed CSLM Algorithm

The analysis of the proposed Cuckoo Search Levenberg Marquardt (CSLM) algorithm is implemented in two stages. In the first phase the Cuckoo Search algorithm via levy flight is initialized with nests as weights and integrated in to neural network optimization. And save the best nest as weight to the network. In the second phase apply the best nest as weight for the network training. In this proposed model the CS is responsible to modify the perimeters for the convergences of this algorithm. Then the overall procedure automatically adjusts its own parameter and converges with in finites period. In the proposed CSLM algorithm the CS algorithm is initiated with a set of randomly generated nests as weights for the networks. Then each randomly generated weight is passed to Levenberg Marquardt back propagation neural network for further process. The mean squared error for each weight matrix is created on illustration of all input pattern matrix through Levenberg Marquardt back propagation neural network. The set of mean square error is considered as a performances index for the proposed CSLM algorithm.

So, the weight value of a matrix is calculated as following;

$$W_c = \sum_{c=1}^{m} a.(rand - \frac{1}{2}) \tag{9}$$

$$B_c = \sum_{c=1}^{m} a.(rand - \frac{1}{2}) \tag{10}$$

Where $W_c = c^{th}$ weight value in a weight matrix. the $rand$ in the Equation (9) is the random number between[0 1]. a is any constant parameter for the proposed method it being less than one. And B_c bias value. So the list of weight matrix is as follows;

$$W^s = [{W_c}^1, {W_c}^2, {W_c}^3, \dots\dots {W_c}^{n-1}] \tag{1)}$$

Now from back propagation process sum of square error is easily intended for every weight matrix in W^s. The task of the network is to learn association between a specified set of input-output pairs, $\{(a_1, T_1), (a_2, T_2), (a_3, T_3), \dots, (a_r, T_r)\}$

So, the error can be calculated as;

$$e_r = (T_r - X_r) \tag{12}$$

The performances index for the network is calculated as;

$$V(x) = \frac{1}{2}\sum_{r=1}^{R}(T_r - X_r)^T(T_r - X_r) \tag{13}$$

$$V_F(x) = \frac{1}{2}\sum_{r=1}^{R} {e_r}^T . e_r \tag{14}$$

In the proposed method, the average sum of square as the performance index and it is calculated as following;

$$V_\mu(x) = \frac{\sum_{j=1}^{N} V_F(x)}{P_i} \tag{15}$$

$e_r = (T_r - X_r)$ is the error for the r^{th} input, $V_\mu(x)$ is the average performance, $V_F(x)$ is the performance index, and P_i is the number of cuckoo population in i^{th} iteration. The weight and bias are calculated according to the back propagation method. The sensitivity of one layer is calculated from its previous one and the calculation of the sensitivity start from the last layer of the network and move backward. To speed up convergence Levenberg Marquardt is selected a learning algorithm. The Levenberg-Marquardt (LM) algorithm is an approximation to Newton's method to get faster training speed. Then the gradient is calculated as;

$$\nabla E(t) = J^T(t)e(t) \tag{16}$$

$$\nabla^2 E(t) = J^T(t)J(t) \tag{17}$$

Where, $\nabla E(t)$ is the gradient; $\nabla^2 E(t)$ is the Hessian matrix of E (t), $e(t)$ is the error; and $J\ (t)$ is the Jacobin matrix. This is calculating in Equation (18);

$$J\ (t) = \begin{bmatrix} \frac{\partial e_1(t)}{\partial t_1} & \frac{\partial e_1(t)}{\partial t_2} & \cdots & \frac{\partial e_1(t)}{\partial t_n} \\ \frac{\partial e_2(t)}{\partial t_1} & \frac{\partial e_2(t)}{\partial t_2} & \cdots & \frac{\partial e_2(t)}{\partial t_n} \\ & \vdots & & \\ \frac{\partial e_n(t)}{\partial t_1} & \frac{\partial e_n(t)}{\partial t_2} & \cdots & \frac{\partial e_n(t)}{\partial t_n} \end{bmatrix} \tag{18}$$

For Gauss-Newton Method;

$$\nabla w = -[J^T(t)J(t)]^{-1}J(t)e(t) \tag{19}$$

For the Levenberg-Marquardt algorithm as the variation of Gauss-Newton Method;

$$w(k+1) = w(k) - [J^T(t)J(t) + \mu I]^{-1}J(t)e(t) \tag{20}$$

Where $\mu > 0$ and is a constant; I is identity matrix. So that the algorithm will approach Gauss-Newton, which should provide faster convergence. Note that when parameter λ is large, the above expression approximates gradient descent (with learning rate 1/λ) while for a small λ, the algorithm approximates the Gauss- Newton method.

The Marquardt alteration to the back propagation algorithm is presented in the following steps;

1. Present all inputs to the network and compute the corresponding network outputs and errors using Equation over all inputs. And compute sum of square of error over all input.
2. Compute the Jacobin matrix using Equation (18).
3. Solve Equation (12) to obtain Equation (19).
4. Recomputed the sum of squares of errors using Equation (15) if this new sum of squares is smaller than that computed in step 1, then reduce μ by λ, update weight using Equation(20) and go back to step 1. If the sum of squares is not reduced, then increase μ by λ and go back to step 3.
5. The algorithm is assumed to have converged when the norm of the gradient Equation (16) is less than some prearranged value, or when the sum of squares has been compact to some error goal.

At the end of each epoch the list of average sum of square error of i^{th} iteration can be calculated as;

$$SSE_i = \{V_\mu(x_1), V_\mu(x_2), V_\mu(x_3) \ldots\ldots V_\mu(x_n)\} \tag{21}$$

The Cuckoo search is replicate the minimum sum of square error and it is found when all the input is processed for each population of the cuckoo nest. So the Cuckoo search nest is calculated as:

$$x_j = Min\{V_\mu(x_1), V_\mu(x_2), V_\mu(x_3) \ldots\ldots V_\mu(x_n)\} \quad (22)$$

And the rest of the average sum of square is considered as other Cuckoo nest. A new solution ${x_i}^{t+1}$ for Cuckoo i is generated using a levy flight according to the following Equation (23);

$${x_i}^{t+1} = {x_i}^{t} + \alpha \otimes levy(\lambda) \quad (23)$$

Where $\alpha > 0$ is the step size which should be related to the scales of the problem of interests. In most cases, we can use $\alpha = 1$. The product $\otimes$ means entry-wise walk while multiplications. Levy flights essentially provide a random walk while their random steps are drawn from a Levy Distribution for large steps.

$$Levy \sim U = K^{-\gamma} \ \ (1 < \gamma \leq 3) \quad (24)$$

So, the movement of the other Cuckoo x_i toward x_j can be drawn from Equation (25)

$$V = \begin{cases} x_i + rand \cdot (x_j - x_i) \ rand_i > p\alpha \\ x_i \qquad\qquad else \end{cases} \quad (25)$$

The Cuckoo Search can move from x_i toward x_j through levy fight it can be written as;

$$\nabla V_{\mathrm{i}} = \begin{cases} x_i + \alpha \otimes levy(\lambda) \sim 0.01 \cdot \left(\frac{U_j}{|V_j|^{\frac{1}{\mu}}}\right).(V - X_{best}) \quad rand_i > p_\alpha \\ x_i \qquad\qquad else \end{cases} \quad (26)$$

Where, ∇V_{i} is a small movement of x_{i} towards x_{j}. The weight and bias for each layers then adjusted as:

$${W_c}^{n+1} = {W_c}^{n} - \nabla V_{\mathrm{i}} \quad (27)$$

$${B_c}^{n+1} = {B_c}^{n} - \nabla V_{\mathrm{i}} \quad (28)$$

RESULTS AND DISCUSSIONS

In ordered to demonstrate the presentation of the proposed CSLM algorithm in training ANN. CSLM algorithm is tested on some benchmark classification problems. The simulation experiments are performed on an AMD Athlon 1.66 GHz CPU with a 2-GB RAM. The software used for simulation process is MATLAB 2009b.

The proposed CSLM algorithm is compared with Artificial Bee Colony Levenberg-Marquardt (ABC-LM), Artificial Bee Colony Back-Propagation (ABC-BP), Artificial Bee Colony Neural Network (ABCNN) and conventional Back-Propagation Neural Network (BPNN) algorithms respectively. The performance measure for each algorithm is based on the Mean Square Error (MSE), and average 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 fixed to 5. In the network structure, the bias nodes are also used and the log-sigmoid activation function is applied. . For each problem, trial is limited to 1000 epochs. And an MSE criterion is kept to 0.0001. A total of 20 trials are run for each case. The network results are stored in the result file for each trial.

7 Bit Parity Problem

This study used 7 Bit Parity dataset. Which consist of 7 inputs to a single binary output. For 7 Bit Parity dataset the network structural design consists of 7 inputs 5 hidden neurons in the hidden layer, and 1 output in the output layer. It has forty connection weights and six biases. TABLE (1), shows the

mean square error, standard deviation and accuracy for the 7 Bit Parity test problem with five hidden neurons. From the TABLE (1), it is clear that the proposed CSLM algorithm converged to global minima with MSE of 0.014 and SD of 0.002. While the ABC-LM algorithm has an MSE of 0.083, ABCNN, BPNN and ABC-BP have the MSE of 0.217, 0.101, and 0.263. While the average accuracy rate of the proposed CSLM algorithm is 98.81%, and the other algorithms such as BPNN, ABC-LM, ABC-BP, and ABCNN have 85.12, 69.137, 82.13, and 67.86 percentile accuracies respectively. Which are still lass then the proposed CSLM algorithm. FIGURE 1 show the average accuracy, MSE, and SD of the used algorithms in this study, for the 7 Parity Bit classification problems. Which show that the proposed model CSLM model has better performance than the other algorithms, such ABC-BP, ABC-LN, ABCNN, and BPNN in term of MSE, SD, and Accuracy. Figure 1 show the MSE convergence performance of the algorithms.

TABLE (1). CPU Time, Epochs and MSE for 7 bit Parity dataset

Algorithms	ABC-BP	ABC-LM	BPNN	ABCNN	CSLM
MSE	0.217	0.083	0.263	0.101	0.014
SD	0.008	0.0124	0.014	0.0151	0.002
Accuracy (%)	82.128	69.137	85.120	67.86	98.81

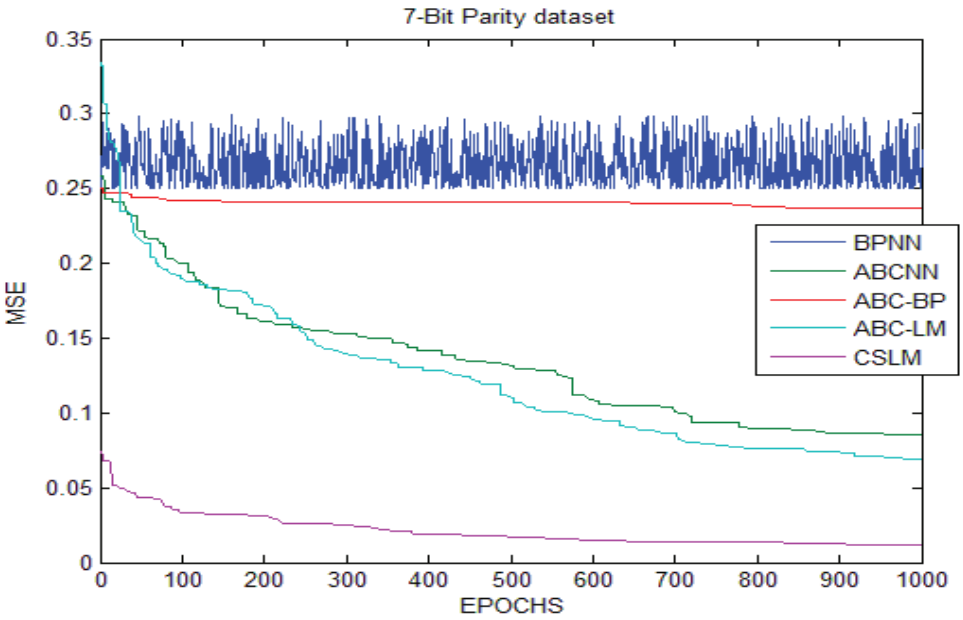

FIGURE 1. MSE convergence performance on the selected algorithms for IRIS dataset

Iris Classification Problem

The Iris classification Problem was created to display the values of different analysis. This is maybe the best famous database to be found in the pattern recognition literature. There were 150 instances, 4 inputs, and 3 outputs in this dataset. The classification of Iris dataset involving the data of petal width, petal length, sepal length, and sepal width into three classes of species, which consist of Iris Santos, Iris Vesicular, and Iris Virginia. The selected network configuration for Iris classification dataset is 4-5-3. Which consist of 4 inputs nodes, 5 hidden nodes and 3 outputs nodes. 75 instances are used for training dataset and the rest as for testing dataset. TABLE (2), described the MSE, SD, and accuracy of the proposed CSLM algorithm compare with simple BPNN and with the variants of ABC algorithm. TABLE (2), show the simulation result, from the result it's clear that the proposed CSLM model has better performance in case of MSE, SD, and accuracy.

FIGURE 2 show the MSE convergence performance of the used algorithms in this study.

TABLE (2). CPU Time, Epochs and MSE for IRIS dataset

Algorithms	ABC-BP	ABC-LM	BPNN	ABCNN	CSLM
MSE	0.155	0.058	0.312	0.048	0.001
SD	0.023	0.005	0.022	0.004	0.0004
Accuracy (%)	86.88	79.559	87.192	80.23	99.66

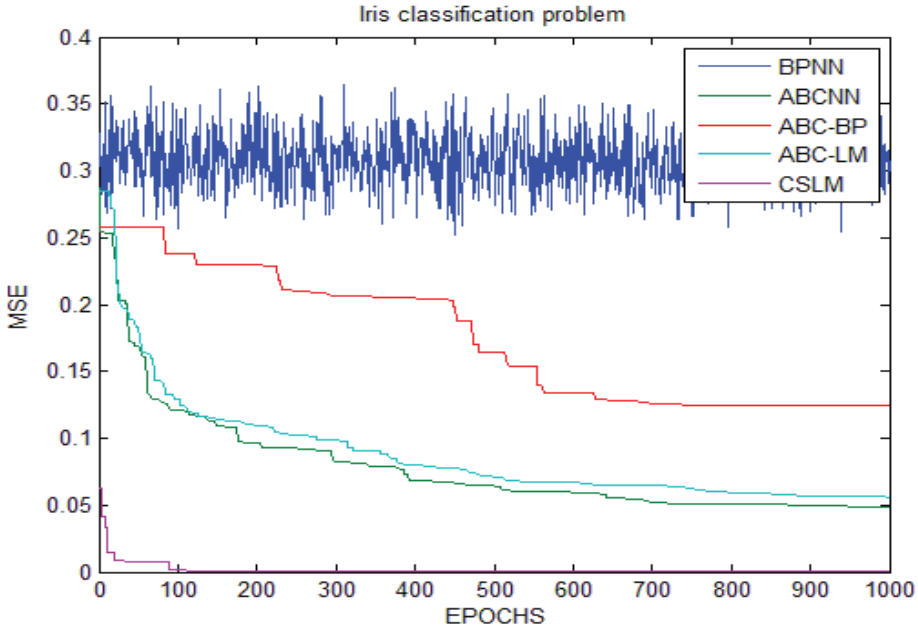

FIGURE 2. MSE convergence performance on the selected algorithms for IRIS dataset

Wisconsin Breast Cancer Classification Problem

Wisconsin Breast Cancer dataset is taken from University California Irvine Machine Learning Repository (UCIMLR) database. This problem tried to diagnosis of Wisconsin breast cancer by trying to classify a tumor as either benign or malignant based from continues clinical variable. This dataset consist of 9 inputs and 2 outputs with 699 instances. The input attribute are, for instance, the clump thickness, the uniformity of cell size, the uniformity of cell shape, the amount of marginal adhesion, the single epithelial cell size, frequency of bare nuclei, bland chromatin, normal nucleoli, and mitoses. The selected network architecture is used for the breast cancer classification problem is consists of 9 inputs nodes, 5 hidden nodes and 2 output nodes.

TABLE (3). CPU Time, Epochs and MSE for Breast Cancer Classification dataset

Algorithms	ABC-BP	ABC-LM	BPNN	ABCNN	CSLM
MSE	0.184	0.0139	0.271	0.014	0.001
SD	0.05	0.0011	0.017	0.0002	0.005
Accuracy (%)	92.021	93.831	90.7138	88.968	99.80

TABLE (3) describes in the detail simulation result of the comparing algorithm for Breast Cancer problem in terms of MSE, SD, and accuracy. From the table 3, it's easily understand that the proposed CSLM algorithm achieve better result than the ABC-BP, ABC-LM, ABCNN, and BPNN in term of MSE, SD, and accuracy. The proposed CSLM model has MSE of 0.001; SD is 0.005 with 99.80 % of accuracy. While the other algorithms such as BP, ABC-BP, ABC-LM, and ABCNN, have MSE 0.271, 0.014, 0.0139, 0.184, and SD of 0.05, 0.0011, 0.017, 0.0002, which shows less performances than the proposed methods in term of MSE, SD, and accuracy. FIGURE 3 show the MSE comprising performance of the used algorithms.

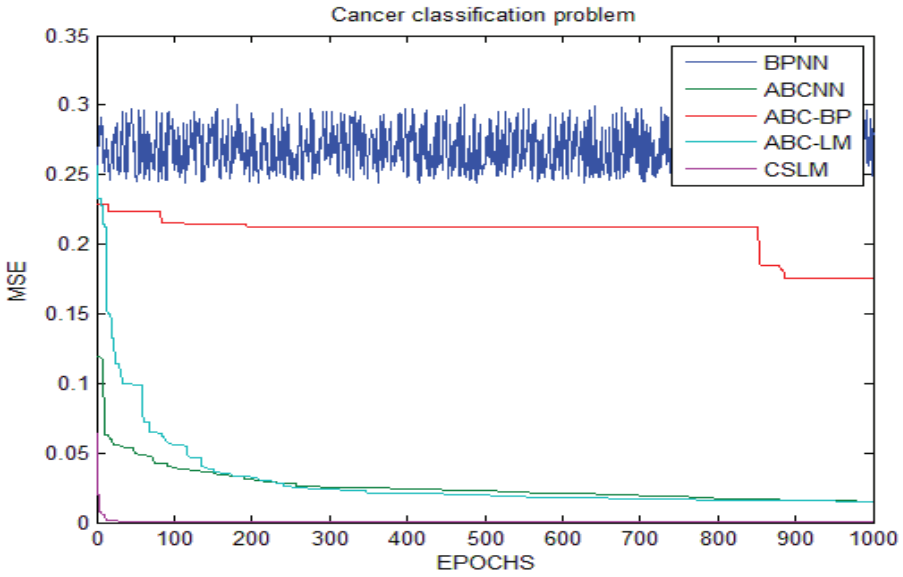

FIGURE 3. MSE convergence performance on the selected algorithms for breast cancer dataset

Diabetes Classification Problem

The Diabetes dataset is taken from UCI Machine learning Repository, created based on Pima India diabetes. This dataset consist of 768 examples, 8 inputs and 2 outputs; consist of all the information of the chemical change in a female body whose disparity can cause diabetes. The feed forward network topology for this network is set to 8-5-2. TABLE (4) shows the MSE, SD, and accuracy for the proposed CSLM, conventional BP, ABC-BP, ABC-LM and ABCNN algorithms. The table 4 clearly illustrate that the proposed CSLM model perform better than the other methods such BPNN, ABC-BP, and ABC-LM on the Diabetes classification problem in term of MSE and accuracy. The TABLE (4) given below describes that the proposed CSLM algorithm achieve 0.015 MSE and SD, of 0.003 with 98.72 % of accuracy. Which is far better than other algorithms in case of MSE, SD, and accuracy. While the BPNN, ABC-BP, ABC-LM, ABCNN, algorithms have MSE of 0.201, 0.141, 0.269, 0.131, and SD of 0.002, 0.0334, 0.026, 0.021. While the FIGURE 4 give the comparison performances of the algorithms for the Diabetes classification problem.

TABLE (4). CPU Time, Epochs and MSE for Diabetes Benchmark Classification Problem

Algorithms	ABC-BP	ABC-LM	BPNN	ABCNN	CSLM
MSE	0.201	0.141	0.269	0.131	0.015
SD	0.002	0.0334	0.026	0.021	0.003
Accuracy (%)	91.468	65.098	84.94	68.090	98.72

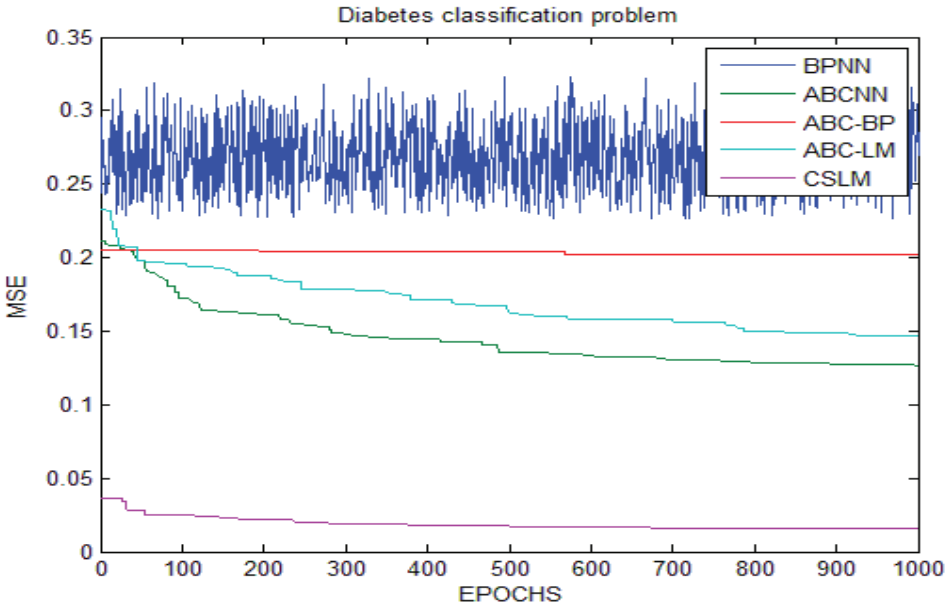

FIGURE 4. MSE convergence performance on the selected algorithms for diabetes dataset

CONCLUSION

Derivative free metaheuristic techniques based on nature inspired behavior provide solutions to complex problems. One of the most recent additions to the collection of nature inspired optimization procedure is Cuckoo Search (CS) algorithm. CS is used to find optimal weight set for neural network in training process. Traditional training algorithms have some limitation such as getting trapped in local minima and slow convergence rate. This study proposed a new technique CSLM by combining the best features of two known algorithms back-propagation (BP) and Levenberg Marquardt algorithm (LM) for improving the convergence speed of ANN training and avoiding local minima problem by training this network. Some selected benchmark classification datasets are used for simulation. The experiment result show that the proposed cuckoo search with Levenberg Marquardt algorithm has better performance than other algorithm used in this study.

ACKNOWLEDGMENTS

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. M. Z. Rehman, N. M. Nawi, A. Khan, "A New Bat Based Back-Propagation (BAT-BP) Algorithm," *ICSS 2013 Proceedings*, Bangi, Malaysia, 2013, pp. 395-404.
2. N. M. Nawi, A. Khan, M. Z. Rehman, "A New Back-propagation Neural Network optimized with Cuckoo Search Algorithm," *ICCSA-2013*, Vietnam, 2013, pp. 413-426.
3. N. M. Nawi, A.K., M. Z. Rehman, A New Cuckoo Search based Levenberg-Marquardt (CSLM) Algorithm, *ICCSA-2013*, Vietnam, 2013, pp. 438-451.

4. N. M. Nawi, A. Khan, M. Z. Rehman, "A New Levenberg-Marquardt based Back-propagation Algorithm trained with Cuckoo Search," *ICEEI-2013*, Malaysia, 2013, pp. 18-24.
5. M. Z. Rehman, N. M. Nawi, *International Journal on New Computer Architectures and Their Applications (IJNCAA)*, **1**(4), 838-847 (2011).
6. B. Coppin, "Artificial Intelligence Illuminated," *Jones and Bartlet illuminated Series*, USA, 291-324, (2004).
7. V. M. Krasnopolsky, & Chevallier, *Neural Networks*, **16** (3), 335-348 (2003).
8. I. A. Basheer, & Hajmeer, *Journal of Microbiological Methods*, **43** (1), 03-31 (2000).
9. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Nature*, **323**(9), 533-535 (1986).
10. N. M. Nawi, R.S. Ransing, N. AbdulHamid, *Journal of Science and Technology*, **2**(2) (2011).
11. W. A. M. Ahmed, E. Saad, and E. Aziz, "Modified back propagation algorithm for learning artificial neural networks," *National Radio Science Conference (NRSC 2001)*, 2001, pp. 345-352.
12. W. Jin *et al*., "The improvements of BP neural network learning algorithm," *WCCC-ICSP*, 2000, pp. 1647-1649.
13. M.T. Hagan, and M.B. Menhaj, *IEEE Transactions on Neural Networks*, **5**(6), 989-993 (1994).
14. B. Wilamowski, N. Cotton, and J. Hewlett, "Neural network trainer with second order learning algorithms," *International Conference on Intelligent Engineering Systems (INES 2007)*, 2007, pp. 127-132.
15. M. I. Lourakis, "A brief description of the Levenberg-Marquardt algorithm implemented by levmar," *Institute of Computer Science, Foundation for Research and Technology - Hellas (FORTH)*, Greece (2005).
16. Q. Xue, *et al.*, "Improved LMBP algorithm in the analysis and application of simulation data," *2010 International Conference on Computer Application and System Modeling (ICCASM)*, 2010, pp. 545-547.
17. J. Yan, *et al.,* "Levenberg-Marquardt algorithm applied to forecast the ice conditions in Ningmeng Reach of the Yellow River," *IEEE Fifth International Conference on Natural Computation (ICNC'09)* (2009).
18. C. Ozturk, and D. Karaboga, "Hybrid artificial bee colony algorithm for neural network training," *IEEE Congress on Evolutionary Computation (CEC)*, 2011, pp. 84-88.
19. X. S. Yang and S. Deb, "Cuckoo Search via Levy flights," *World Congress on Nature & Biologically Inspired Computing*, 2009, pp. 210-214.
20. X. S. Yang, and S. Deb, *Int. J. of Mathematical Modelling and Numerical Optimisation*, **1** (4), 330-343 (2010).
21. X. S. Yang, and S. Deb, "Cuckoo search via Lévy flights," *Proceeings of World Congress on Nature & Biologically Inspired Computing*, 2009, pp. 210-214.
22. X. S. Yang, "Nature-inspired metaheuristic algorithms," *Luniver Press* (2010).