International Journal of Mathematics and
Computer Science, **18**(2023), no. 3, 415–427

Artificial neural network optimization using differential evolution algorithm with adaptive weight bound adjustment for pattern classification

Saithip Limtrakul, Jeerayut Wetweerapong

Department of Mathematics
Faculty of Science
Khon Kaen University
Khon Kaen, Thailand

email: saithiplim@kkumail.com, wjeera@kku.ac.th

(Received January 3, 2023, Accepted February 8, 2023,
Published March 31, 2023)

Abstract

Artificial Neural Network (ANN) is a popular machine learning technique applied to various advanced applications of Artificial Intelligence (AI). In this work, a Differential Evolution Algorithm with adaptive Weight bound adjustment (DEAW) is used to optimize the neural networks for solving classification problems. The DEAW algorithm initials the weights in a small range of bounds and gradually expands them in the mutation step when needed. The experiments are performed on five synthesis scatter-points datasets and four real-world UCI datasets to compare with other well-known evolutionary algorithms: harmony search, ant colony optimization, and self-adaptive differential evolution. The results show that the DEAW method is comparable to the other algorithms and performs better in several cases.

Keywords and phrases: Neural network, Differential Evolution, Training Neural Network, Classification.
AMS (MOS) Subject Classifications: 68T05, 68T07, 68T20.
ISSN 1814-0432, 2023, http://ijmcs.future-in-tech.net

1 Introduction

Classification methods are frequently used to support decision-making tasks. Many problems need to predefine or classify the objects based on their related attributes to provide a tentative decision; for example, medical diagnosis, speech recognition, quality control of products, customer service.

Artificial neural networks are essential intelligent tools for various classification tasks. An ANN has a concept of a learning procedure that imitates the behavior of the nervous system. Similar to the synaptic transferring the signal via the dendrite and axon, the ANN transforms forward input data to output data using weights and the transfer function. The optimal weights minimize the error between the actual and target outputs at the sample input data. Because the approach is data-driven, it thus extracts the underlying model without any knowledge of data properties and the prior model of the data. The obtained model can identify a functional relationship between the group members and their attributes and provide the basis for the classification rules. Moreover, the nonlinear model obtained from an ANN is flexible in modeling real-world problems with complex relations.

The training algorithm applied to solve the objective function or optimize the weights strongly affects the performance of an ANN. The traditional methods for network learning, the same as training, such as the Backpropagation method [1], rely on the gradient-based algorithm. The algorithm has a powerful local search ability but possibly gets stuck into local minima and has a slow convergence speed [2]-[6].

Many researchers proposed global search techniques to overcome the limitation of the gradient-based algorithm. In particular, evolutionary algorithms are increasingly and rapidly developed to evolve the learning process of neural networks [7]-[14]. Kulluk et al. [15] studied five different variances of the harmony search algorithm. The technique was empirically tested and verified on six benchmark classification problems and one real-world problem. In 2015, Chen et al. [16] presented a hybrid algorithm based on the Particle Swarm Optimization (PSO) and Cuckoo Search (CS) algorithm as a training method for feed-forward neural networks. They investigated the performance of the proposed algorithm by applying it to two benchmark problems: function approximation and classification problem. The results show that the hybrid algorithms outperform both PSO and CS. Mavrovouniotis and Yang [17] applied an Ant Colony Optimization (ACO) algorithm to train a feed-forward neural network compared to the ACO hybridized with gradient descent training method. The algorithms were evaluated on

several benchmark classification problems. The results show that the ACO hybridized with BP has a good performance.

Differential Evolution (DE) [18] is an attractive global optimization method compared with other evolutionary algorithms because of ease of implementation, convergence speed, accuracy, robustness, and a few control parameters. Also, it is possible to adapt some of the parameters during the process execution. Bhatia and Vishwakarma [19] proposed a Self-adaptive Differential Evolution (SDE) to optimize the weights and adjust the mutation factor and the crossover rate. The SDE algorithm enhanced the training performance compared to DE and GA algorithms on four benchmark datasets considering convergence rate and mean-squared error.

Since DE is a population-based algorithm, the value of each component depends on the range of the search space and significantly affects the efficiency of ANN performance. The search space size can be defined as a range of weight bound. In our proposed method, the automatic adjusting of the bound in the mutation process is allowed.

The rest of this paper is organized into the following sections. In Section 2, we present the methodology of feed-forward neural network and an implementation of DE with adaptive weight bound adjustment as a training method for ANN. Then the experimental results and discussion are presented in Section 3. Finally, we conclude our paper in section 4.

2 Methodology

2.1 Feed-forward Neural Network

The Multi-Layer Perceptron (MLP) is a general structure of our study. MLP is a feed-forward neural network consisting of multiple layers: an input layer, one or more hidden layers, and an output layer. Each layer has nodes where each node is fully weight-interconnected to all nodes in the subsequent layer (see Figure 1). An MLP transforms the inputs into the outputs through the non-linear function, called the transfer function or activation function, expressed by:

$$x_{hout} = f_1(\sum(x_{input} * w_{i,h})) \tag{2.1}$$

$$x_{output} = f_2(\sum(x_{hout} * w_{h,o})) \tag{2.2}$$

where f_1, f_2 are the activation functions of the h hidden neurons and the o output neuron, respectively. x_{hout} and x_{output} are the output of each hidden

and output node. $w_{i,h}$. $w_{h,o}$ are connected weights between the input-hidden layer and hidden-output layer. In general, the activation function applied to the hidden node is the sigmoid function and, for the output, usually be a linear function.

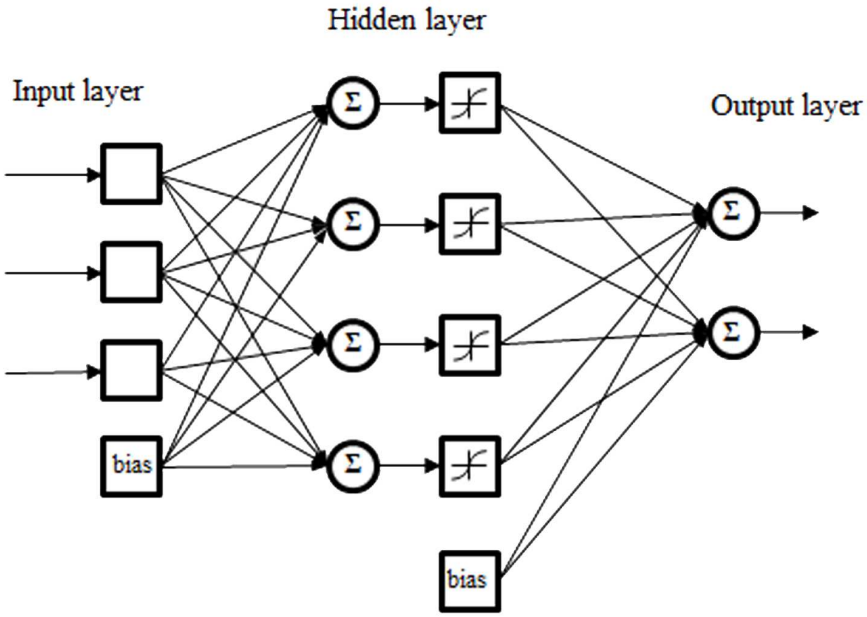


Figure 1: Feed-forward neural network model

Since the ANN learns the mathematical model by training the network, the training rule is essential for a learning algorithm. The updating rules are used to determine how connection weights are changed. The objective function defined by the mean of the distance error between the target value and the output from the network needs to be minimized to find the optimal weights. Suppose the weights vector is $W = [w_0, w_1, ., w_n]$, 1 to n are indices of connection weights, and 0 is an index for the bias node. The optimal solutions can be obtained by minimizing the following objective function:

$$E_W = \frac{1}{2}\sum_{d \in D}(t_d - o_d)^2 \tag{2.3}$$

where D is a set of training data and d is the index of each training data.

t_d and o_d are target and output values, respectively. The output o_d can be calculated by

$$o_d = f_2(\sum(f_1(\sum(x_{input} * W_{i,h})) * W_{h,o})) \quad (2.4)$$

$$x_{output} = f_1(\sum(x_{input} * W_{i,h})) \quad (2.5)$$

2.2 Training Neural Network by using Differential Evolution

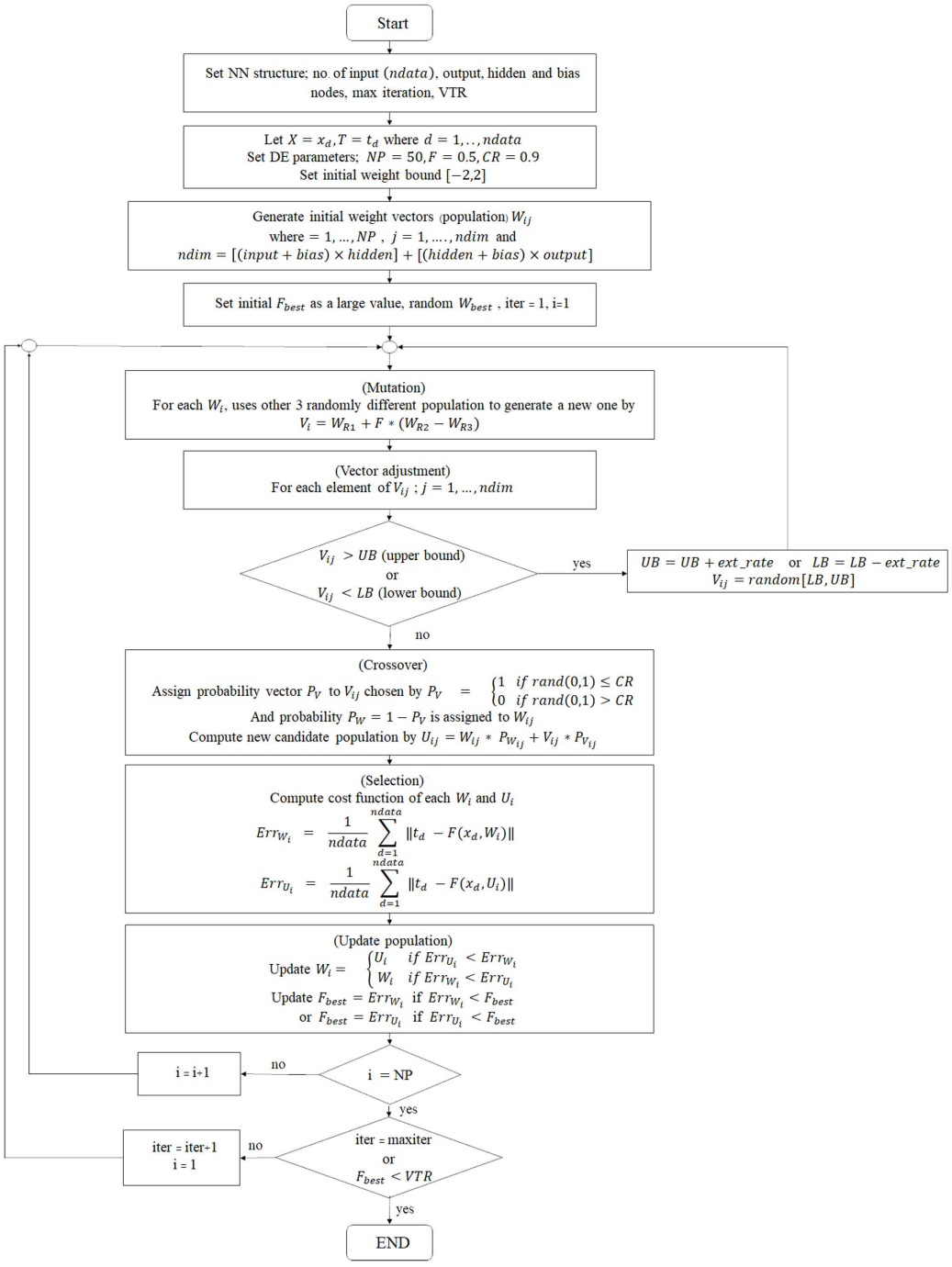
The Differential Evolution (DE) algorithm is an efficient population-based method developed by Storn and Price [18] for solving continuous optimization problems. The method consists of three operations; i.e., mutation, crossover, and selection for generating new candidate solutions. The scaling factor F multiplies the difference between the randomized two population vectors to add to the third one and obtain the mutant vector. The Crossover Rate CR in the range $[0, 1]$ is used to combine the mutant vector with the target vector to create a trial vector. In selection, the trial candidate vector replaces the target vector if it gives a superior solution; otherwise, the original candidate remains unchanged. Many variant DE methods are different in the step of mutation and crossover [20]-[21]. In this work, we apply the basic mutation scheme called DE/rand/1/bin.

Applying DE to train the network, the vectors of connecting weights in ANN are used as the individual population vector of the DE algorithm; i.e., a set of population vectors $W = W_i$, where $i = 1, ..., NP$, and NP is the number of populations. Let $X = x_d$ and $T = t_d$ where $d = 1, , ndata$ be the sets of training and target data vectors, respectively. $F(\cdot)$ is the composite of activation functions in the network and $F(x_d, W_i)$ is an output corresponding to x_d and W_i. In each iteration, the current weight vectors of ANN are obtained from the DE approach by reducing the error between the target T and the output obtained from $F(X, W_i)$ in the selection process.

The DEAW algorithm applies an adaptive process to the mutation step for automatic bound adjustment. The amount of extending rate depends on the current iteration. This strategy increases the rapid exploration ability of the algorithm at the beginning stage and decreases it at the later stage. The equation for adjustment is expressed as:

$$ext\_rate = \frac{1}{iter} \quad (2.6)$$

where iter is the number of current iterations. Figure 3 shows the algorithm of our enhanced DE with adaptive weights bound for training ANN.


Start
Set NN structure; no. of input (ndata), output, hidden and bias nodes, max iteration, VTR
Let $X = x_d, T = t_d$ where $d = 1, .., ndata$
Set DE parameters; $NP = 50, F = 0.5, CR = 0.9$
Set initial weight bound $[-2,2]$
Generate initial weight vectors (population) W_{ij} where $= 1, ..., NP$, $j = 1, ..., ndim$ and $ndim = [(input + bias) \times hidden] + [(hidden + bias) \times output]$
Set initial F_{best} as a large value, random W_{best} , iter = 1, i=1
(Mutation)
For each W_i, uses other 3 randomly different population to generate a new one by $V_i = W_{R1} + F * (W_{R2} - W_{R3})$
(Vector adjustment)
For each element of V_{ij} ; $j = 1, ..., ndim$
$V_{ij} > UB$ (upper bound) or $V_{ij} < LB$ (lower bound)
yes
$UB = UB + ext\_rate$ or $LB = LB - ext\_rate$
$V_{ij} = random[LB, UB]$
no
(Crossover)
Assign probability vector P_V to V_{ij} chosen by $P_V = \begin{cases} 1 & if\ rand(0,1) \le CR \\ 0 & if\ rand(0,1) > CR \end{cases}$
And probability $P_W = 1 - P_V$ is assigned to W_{ij}
Compute new candidate population by $U_{ij} = W_{ij} * P_{W_{ij}} + V_{ij} * P_{V_{ij}}$
(Selection)
Compute cost function of each W_i and U_i
$Err_{W_i} = \frac{1}{ndata} \sum_{d=1}^{ndata} \|t_d - F(x_d, W_i)\|$
$Err_{U_i} = \frac{1}{ndata} \sum_{d=1}^{ndata} \|t_d - F(x_d, U_i)\|$
(Update population)
Update $W_i = \begin{cases} U_i & if\ Err_{U_i} < Err_{W_i} \\ W_i & if\ Err_{W_i} < Err_{U_i} \end{cases}$
Update $F_{best} = Err_{W_i}$ if $Err_{W_i} < F_{best}$
or $F_{best} = Err_{U_i}$ if $Err_{U_i} < F_{best}$
i = NP
no
i = i+1
yes
iter = maxiter or $F_{best} < VTR$
no
iter = iter+1
i = 1
yes
END

3 Experimental Result and Discussion

3.1 Experimental Setup

In this work, we applied the DE as a learning algorithm to feed-forward neural networks with a hidden layer. The number of nodes in the input layer depends on each problem with one bias. In the hidden layer, we considered various appropriated numbers of nodes with one bias obtained by preliminary experiments. We used two different activation functions. The first one is Sigmoid which is assigned to each hidden node. Secondly, the linear function is applied to the output layer. Each experimental study is tested for 30 runs. The configuration of DE is simple: F=0.5,CR=0.9. The population size is 50 for all experiments.

3.2 The 2D scatter points classification problems

To demonstrate the efficiency of the DEAW algorithm, 5 labeled scatter points data are used for training and classification. The data, Crescent-Fullmoon, Cluster-in-cluster, and Half-kernel are linearly divided into 2 classes, Corners and Outliers are separated into 4 classes. Each data set is randomly generated via generating function, 200 data for Crescent-Fullmoon, Half-kernel, and Corners, 228 data for Cluster-in-cluster, and 240 data for Outliers. In order to avoid bias separation, we used 5-fold cross-validation for evaluating the results. The structure of ANN and settings are described in Table 1.

Table 1: The settings for 2D classification problems.

Data	Input	Class	Hidden nodes	Training	Testing	no. runs
Cresent-Fullmoon	2	2	3	160	40	30
Cluster-in-cluster	2	2	3	180	48	30
Half-kernel	2	2	3	160	40	30
Corner	2	4	3	160	40	30
Outlier	2	4	3	192	48	30

By using the initial weight bound [-2,2] with activation scale 1, we obtain 100% accuracy of classification by this combination for all datasets. The classification results are illustrated in Fig. 3. All training and testing outputs are perfectly matched to their targets.

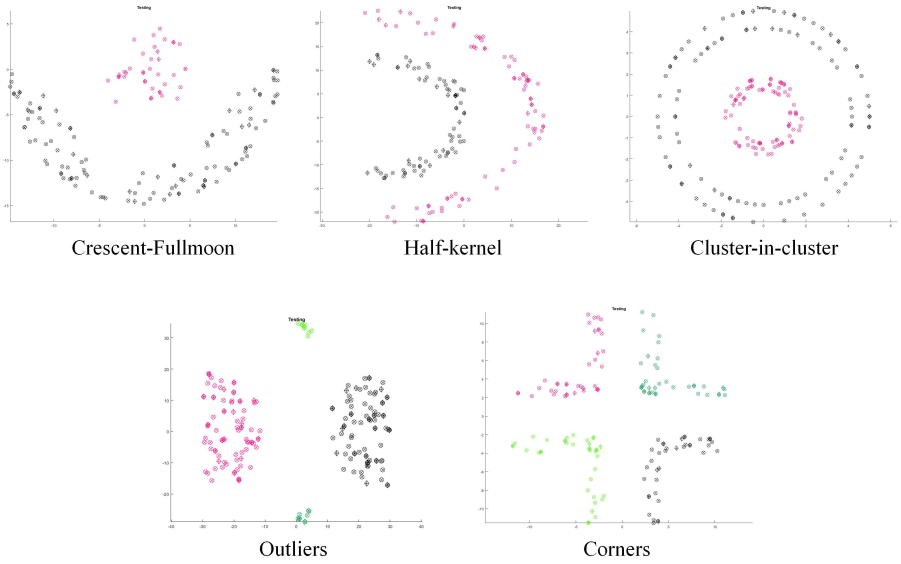


Figure 3: Results of scatter point classification by using our DEAW algorithm. Training and Testing results are shown in the same images. Cross ($\times$) and Plus ($+$) represent Training and Testing target, respectively, while Circle ($\circ$) and Diamond ($\diamond$) represent the output. Each class is colored by different colors.

3.3 High dimensional classification problems

The performances of the proposed method are also tested by using four benchmarks datasets obtained from the UCI Machine Learning Repository [15, 19, 17]. The main characteristics of the data sets show in the Table. 2. Each data is described as follows.

Iris dataset - This dataset contains three classes referring to 3 types of iris plants. Each type consists of 50 samples with four inputs. The remarkable characteristic of this dataset is that one of the classes can separate from the other two groups, while the latter overlap.

Glass identification dataset The dataset describes the types of glass to 6 classes; float processed building windows, non-float processed building windows, vehicle windows, container, tableware, and headlamps, by using nine input characteristics.

Wine dataset This dataset introduces the classes of results of chemical

analysis of wines grown in the same region in Italy but derived from three different cultivars. 178 instances are classified into three classes using 13 input characteristics.

Cancer dataset The dataset of diagnostics of breast cancer was obtained from Wisconsin Hospitals, Madison. The data consists of 699 instances, each of which contains 9 inputs and 2 outputs.

Table 2: Main characteristics and partitioning of the benchmark datasets

Dataset	Attributes Char.	Attributes	Classes	Instances	Training	Testing	Training/ Testing ratio
Iris	Real	4	3	150	120	30	4:1
Glass	Real	9	6	214	193	21	9:1
Wine	Integer, Real	13	3	178	144	34	4:1
Cancer	Integer, Real	9	2	699	560	139	4:1

The accuracy performance metric is calculated by:

$$Acc = 100\frac{\text{no. of correctly classified objects}}{\text{no. of objects in the datasets}}.$$

Since the previous works [15, 19, 17] have individual settings, we made 3 different experimental configurations for comparison. Tables 3, 4, 5 show the results compared to [15, 19, 17], respectively. Table 3 shows the result compared with the harmony search algorithm [15]. Similar results are obtained from testing the Iris dataset, while the DEAW method with a lower number of hidden nodes provides more accurate testing results for the Glass dataset. Next, Table 4 shows the comparison result between DEAW and the Self-adaptive DE method (SDE) [19]. Although the SDE method presented only maximum training accuracy, the result shows that our proposed method gives slightly better accuracy for both Iris and Wine datasets. Finally, Table 5 presents the results of the ACO method [17] and our method on the Cancer dataset. DEAW method shows lower accuracy for training while expressing better results for testing. The results imply that the ACO method may overfit the dataset in the training process while the DEAW algorithm can overcome the problem.

Table 3: The performance comparison between HS [15] and DEAW

Data	.	Arch	Training	Testing	CV	Bound	Min train	Min test	Max train	Max test
Iris	HS	4-3-3	135	15	9:1	[-1,1]	**97.04**	**93.33**	98.52	**100**
	DEAW	4-3-3	120	30	4:1	ext	96.67	90.00	**100**	**100**
Glass	HS	9-8-6	192	22	9:1	[-1,1]	63.73	45.45	70.98	80.95
	DEAW	9-3-6	193	21	9:1	ext	**66.94**	**95.08**	**71.81**	**96.48**

Table 4: The performance comparison between SDE[19] and DEAW

Data	.	Arch	Training	Testing	CV	Bound	Min train	Min test	Max train	Max test
Iris	SDE	4-3-3	135	15	9:1	[-1,1]	n/a	n/a	99.96	n/a
	DEAW	4-3-3	120	30	4:1	ext	**96.67**	**90.00**	**100**	**100**
Wine	SDE	13-10-2	142	36	4:1	[-1,1]	n/a	n/a	99.96	n/a
	DEAW	13-3-2	144	34	4:1	ext	**100**	**95.59**	**100**	**97.35**
		13-10-2	144	34	4:1	ext	**100**	**96.18**	**100**	**97.94**

Table 5: The performance comparison between ACO [17] and DEAW

Data	.	Arch	Training	Testing	CV	Bound	Min train	Min test	Max train	Max test
Cancer	ACO	9-6-2	525	174	3:1	[-1,1]	n/a	n/a	**97.54**	96.34
	DEAW	9-3-2	560	139	4:1	ext	**86.98**	**91.94**	91.94	**98.13**

4 Conclusion

In this study, an enhanced differential evolution algorithm with adaptive weight bound adjustment (DEAW) was used to train feed-forward neural networks for classification problems. The algorithms are evaluated on five synthesis scatter-point datasets and four real-world datasets. The results show that the DEAW algorithm with simple configuration can efficiently train the ANN. The accuracy rate of the proposed algorithm was comparable with other evolutionary methods and gave better results in several cases.

Acknowledgment. This work was partially supported by a research grant from the Development and Promotion of Science and Technology Talents Project and the Department of Mathematics, Faculty of Science, Khon Kaen University, Fiscal Year 2022.

References

[1] David E. Rumelhart, Geoffrey E. Hinton, Ronald J. Williams, Learning representations by back-propagating errors. Nature, **323**, (1986), 533–536.

[2] V.K. Dhar, A.K. Tickoo, R. Koul, B.P. Dubey, Comparative performance of some popular artificial neural network algorithms on benchmark and function approximation problems, Pramana, **74,** no. 2, (2010), 307–324.

[3] Lianhui Chen, A global optimization algorithm for neural network training, Proceedings of the 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan), Nagoya, Japan, (1993), 443–446.

[4] Lutz Prechelt, A quantitative study of experimental evaluations of neural network learning algorithms: Current research practice. Neural Networks, **9,** Issue 3, (1996), 457–462.

[5] Alberto Prieto, Beatriz Prieto, Eva Martinez Ortigosa, Eduardo Ros, Francisco Pelayo, Julio Ortega, Ignacio Rojas, Neural networks: An overview of early research, current frameworks and new challenges. Neurocomputing, **214,** (2016), 242-268.

[6] Sibo Yang, T.O. Ting, K.L. Man, Sheng-Uei Guan, Investigation of Neural Networks for Function Approximation, Procedia Computer Science, **17,** (2013), 586–594.

[7] Gouqiang Peter Zhang, Neural Networks for Classification: A survey, IEEE Transaction on System, Man and CyberneticsPart C: Application and Review, **30,** no. 4, (2000), 451–461.

[8] Tatt Hee Oong, Nor Ashidi Mat Isa, Adaptive evolutionary artificial neural networks for pattern classification, IEEE Transaction on Neural Network, **22,** no. 11, (2011), 1823–1836.

[9] Yongquan Zhou, Yanbiao Niu, Qifang Luo, Ming Jiang, Teaching learning-based whale optimization algorithm for multi-layer perception neural network training, Math. Biosci. Engg., **17,** no. 5, (2020), 5987–6025.

[10] Fei Han, Jian-Sheng Zhu, Improved Particle Swarm Optimization Combined with Back propagation for Feed forward Neural Networks, Int. J. Intell. Syst., **28,** no. 3, (2013), 271–288.

[11] Sudarshan Nandy, Partha Pratim Sarkar, Achintya Das, Training a Feed-Forward Neural Network with Artificial Bee Colony based Back propagation Method, International Journal of Computer Science and Information Technology, **4,** no. 4, (2012), 33–46.

[12] Najmeh Sadat Jaddi, Salwani Abdullah, Abdul Razak Hamdan, Optimization of neural network model using modified bat-inspired algorithm, Applied Soft Computing, **37,** (2015), 71–86

[13] Fei Han, Jing Jiang, Qing-Hua Ling, Ben-Yue Su, A survey on metaheuristic optimization for random single-hidden layer feed forward neural network, Neurocomputing, **335,** (2019), 261–273.

[14] Ashraf Mohamed Hemeida, Somaia Awad Hassan, Al-Attar Ali Mohamed, Salem Alkhalaf, Mountasser Mohamed Mahmoud, Tomonobu Senjyu, Ayman Bahaa El-Din, Nature-inspired algorithms for feed-forward neural network classifiers: A survey of one decade of research, Ain Shams Engineering Journal, **11,** Issue 3, (2020), 659–675.

[15] Sinem Kulluk, Lale Ozbakir, Adil Baykasoglu, Training neural networks with harmony search algorithms for classification problems, Engineering Applications of Artificial Intelligence, **25,** Issue 1, (2012), 11–19.

[16] Jeng-Fung Chen, Quang Hung Do, and Ho-Nien Hsieh, Training Artificial Neural Networks by a Hybrid PSO-CS Algorithm, Algorithms, **8,** no. 2, (2015), 292–308.

[17] Michalis Mavrovouniotis, Shengxiang Yang, Training neural networks with ant colony optimization algorithms for pattern classification, Soft Computing, **19**, (2015), 1511–1522.

[18] Rainer Storn, Kenneth Price, Differential Evolution A Simple and Efficient Heuristic for global Optimization over Continuous Spaces, Journal of Global Optimization, **11,** (1997), 341–359.

[19] Shivani Bhatia, Virendra P. Vishwakarma, Feed-Forward Neural Network Optimization using Self-adaptive Differential Evolution for Pattern Classification, 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology, Bangalore, India, (2016), 184–188.

[20] Swagatam Das, Ponnuthurai Nagaratnam Suganthan, Differential Evolution: A Survey of the State-of-the-Art, IEEE Transactions on Evolutionary Computation, **15,** (2011), 4–31.

[21] Swagatam Das, Sankha Subhra Mullick, P.N. Suganthan, Recent advances in differential evolutionAn updated survey, Swarm and Evolutionary Computation, **27,** (2016), 1–30.