International Journal of Electrical and Computer Engineering (IJECE)
Vol. 10, No. 4, August 2020, pp. 3725~3733
ISSN: 2088-8708, DOI: 10.11591/ijece.v10i4.pp3725-3733

A hybrid constructive algorithm incorporating teaching-learning based optimization for neural network training

Mahdieh Khorashadizade[1], Morteza Jouyban[2], Mohammadreza Asghari Oskoei[3]
[1]Department of Computer Science, Sistan and Baluchestan University, Iran
[2,3]Department of Computer Science, Allameh Tabataba'i University, Iran

Article Info

Article history:

Received May 29, 2019
Revised Jan 16, 2020
Accepted Feb 1, 2020

Keywords:

Algorithm incorporating
Teaching-learning
Optimization
Neural network training

ABSTRACT

In neural networks, simultaneous determination of the optimum structure and weights is a challenge. This paper proposes a combination of teaching-learning based optimization (TLBO) algorithm and a constructive algorithm (CA) to cope with the challenge. In literature, TLBO is used to choose proper weights, while CA is adopted to construct different structures in order to select the proper one. In this study, the basic TLBO algorithm along with an improved version of this algorithm for network weights selection are utilized. Meanwhile, as a constructive algorithm, a novel modification to multiple operations, using statistical tests (MOST), is applied and tested to choose the proper structure. The proposed combinatorial algorithms are applied to ten classification problems and two-time-series prediction problems, as the benchmark. The results are evaluated based on training and testing error, network complexity and mean-square error. The experimental results illustrate that the proposed hybrid method of the modified MOST constructive algorithm and the improved TLBO (MCO-ITLBO) algorithm outperform the others; moreover, they have been proven by Wilcoxon statistical tests as well. The proposed method demonstrates less average error with less complexity in the network structure.

Corresponding Author:

Mahdieh Khorashadizade,
Department of Computer Science,
Sistan and Baluchestan University,
Zahedan, Iran.
Email: khorashadizade.m@gmail.com

1. INTRODUCTION

Artificial neural networks (ANN), having a strong similitude to biological networks, have the ability to learn noise data as well as the ability to classify and recognize different types of input patterns. These take place only if the neural network is well trained without an over-fitting or under-fitting model. The most well-known training algorithm is the back propagation [1], but it has numerous drawbacks such as trapping in local minima [2]. Hence, researchers have decided to utilize evolutionary algorithms instead. In addition to the training and determination of optimal weights, another critical issue is the design of an appropriate ANN architecture. Many studies have been conducted for architecture as well as weight optimization. For instance, in applying a novel method based on Gaussian-PSO and fuzzy reasoning [3] ANN weight and structure optimization is presented. In literature, there are other methods to optimize ANN architecture, namely constructive algorithms and pruning algorithms. Constructive algorithms have many advantages over pruning algorithms, such as easy initiation, less complexity of the final solution, and lighter load of computation. Furthermore, CA's are able to freeze the existing weights in the neural network if they are useful in output; as a result, resulting in the reduction of the required time and memory. In pruning algorithms, several

problem-dependent parameters are to be properly identified in order to obtain an acceptable network with a satisfactory performance. This makes it difficult to be used in real-world applications [4].

This paper portrays a combination of random search procedures and systematic methods, proposing hybridizing improved teaching-learning algorithms with constructive algorithms for the purpose of ANN design. The hybrid is advantageous, for teaching-learning algorithm is a parameter-independent optimization algorithm that balance between exploration and exploitation. Meanwhile, constructive algorithms are adopted to select an appropriate ANN architecture. Since using constructive algorithms is cost-effective in terms of the training-time and complexity of ANN, it hinders the production of networks with an inefficient very large architecture. This paper, with the aim of simultaneously optimizing the ANN weights and architecture, combines training and constructive algorithms applied to ten classification problems and two-time series prediction problems, as the benchmark. After evaluating the performance of proposed hybrid algorithms and comparing their results, it was found that the proposed method outperformed other algorithms. The proposed combination method proves to have a lower mean error in most cases. The rest of this study is organized as follows: Section 2 provides a brief description of the algorithms that we provided. Then, in Section 3, a hybrid proposed method to ANN optimization is presented. In Section 4, the experimental results of the application of the proposed approaches to the ANN problems are reported, and finally, the conclusion is drawn in the last section.

2. ALGORITHM DESCRIPTION

2.1. Improved teaching-learning based optimization (ITLBO)

Although TLBO provides high-quality solutions in the least amount of time and has a great stability in convergence [5], in the learner phase of this algorithm, learners randomly choose another learner from the population. This difficulty leads to a lack of balance between the two concepts of diversity and convergence. ITLBO with an improvement into basic TLBO overcomes this difficulty. In this algorithm, the teacher phase is the same as the teacher phase in the basic TLBO algorithm and the learner phase is expressed as follows. The ITLBO has been developed to improve the weaknesses of TLBO algorithm; for example, in TLBO random choices due to low local search capability, but in ITLBO with addition concept of neighborhood we trying to reduce random choices and utilize of neighborhood abilities. This issue increases local search and global search capability. The main sections of ITLBO are as follows:

2.1.1. ITLBO learner phase

In this phase, each learner is encoded with an integer and placed in a rectangular array. learners may learn from their neighbors or from the best individual in whole class. This process is based on local search ability; furthermore, balance between global search and local search ability is applied. In local search, each learner updates his position with Pc probability by the best learner in his neighborhood (or $X_{i,teacher}$) and also global best learner that in population.

$$X_{i,new} = X_{i,old} + r_2.\left(X_{i,teacher} - X_{i,old}\right) + r_3.\left(X_{teacher} - X_{i,old}\right) \quad (1)$$

Where $X_{i,teacher}$ is the teacher in X_i neighborhood, $X_{teacher}$ is teacher of whole class, r_2 , r_3 are random numbers in the range of (0, 1). The new position of each learner will be accepted if its fitness value has improved. In the concept of global search, if Pc probability don't meet, each learner chooses a random learner (X_j) from the whole class to provide the learning goal, if X_j is better than X_i, or otherwise, learning occurs according to learner phase in basic TLBO. Therefore, using these operations both local and global search capability will be obtained. All the accepted learners at the end of learner phase are preserved. Due to the enhanced exploitation ability along with the exploration ability, which already existed in the learning phase of the original algorithm, we use the concept of neighborhood in the classroom. For each individual in the population exist a number of neighborhood member that learn from the best one. For maintain of diversity after a number of iterations the neighborhood members of each individual are changed. This issue balance between the exploration and exploiting abilities. Other advantage there is in this algorithm, when a new position is obtained for each member, it may lead to the production of decision variables values that are out of the range of the definition interval. In this case, most researchers use the convergence approach to the upper and lower bound according to algorithm, but this method is Old and disabled method witch cause algorithm to local optima. In the improved teaching-learning based optimization method, we use modified technique to check boundaries of the variables [6]. Its advantage is avoiding equalization of the decision variables.

2.2. Modified MOST algorithm (MMOST)

Determining the architecture of artificial networks has lured many researchers in the field in recent years. We used Multiple Operators using Statistical Tests algorithm MOST [7]. In MOST algorithm, there isn't any controlled method for change struture. This algorithm may have large changes in network structure during the algorithm. Another weakness of this algorithm is the addition of layers frequently without any condition to control. In modified MOST algorithm, the operator pool was removed. For changing structure neurons are added one after another. Selecting the new structures is done more carefully by adding multiple conditions. At the beginning, algorithm starts with a single hidden layer network by the minimum number of neurons. We chose one of popular approach for allowed minimum number that is the average of number of output layer and input layer. Network in the first step has a single hidden layer and neurons are added continually to the hidden layer to obtain a proper structure of the network. To avoid creating very large structures for networks, the neurons are added to single hidden layer of the network until they don't exceed Max-hidden number. In fact, networks with very large structure not only don't have good generalizability, but they also increase the computational time of the algorithm. To eliminate this weakness, we add the second layer to network structure to create proper architecture with a probability less than P. after adding the second layer, the number of neurons in each hidden layer is set by min-hidden. MMOST constructive algorithm chooses the best architecture between constructed structures. So, as noted above, the differences between the MMOST and MOST algorithm are as follows: operator's pool is deleted; neurons are continually added; and there is a more precise choice between the three previous, current and the candidate architectures.

3. THE PROPOSED METHOD

In this paper, we proposed a combination algorithm for producing a neural network with proper structure and weights, to simultaneous optimization of weights and structure. For this purpose, a combination of the modified MOST constructive algorithm with an improved version of the training algorithm was proposed. The role of the constructive algorithm in the proposed algorithm is to construct different structures in order to select the proper one, which is carried out by using a switching systematic approach between the various structures allowed for the neural network. On the other hand, the role of training algorithm is to find optimal weights for the structure that is created by the constructive algorithm. Using constructive algorithms in creating a network architecture reduces computational cost and complexity. But using these algorithms in solving noisy problems [8] has failed, which in combination with other techniques, such as the use of evolutionary algorithms, can be effective in improving the constructive algorithm. In addition, we have made some modifications on the MOST constructive algorithm. For a more detailed description, the pseudo code of the proposed hybrid algorithms is shown in Figure 1. In other words, in order to clarify the combination of evolutionary training algorithms and constructive algorithms, we showed the process in flowchart by Figure 2.

Algorithm 1. Proposed hybrid constructive and training algorithm

1. *Set the parameters: N-hidden-layer = 1, N-hidden = Min-hidden, N-input = (number of features of the problem)*
   N-output = (number of classes of the problem), Iter = 1;
2. *Calculate the dimension of problem* $C_t = dim$*;*
3. *Initializing the weights of neural network that is the same population of evolutionary algorithm;*
4. *p = 0.5; % probability of addition hidden layer into the architecture*
5. ***While** (N-hidden <= Max-hidden || iter <= MaxIter)*
6. *N-hidden = N-hidden+1 and create a new structure →* C_t
7. *%% training algorithm*
8. *train the network with the training algorithm for a certain epoch*
9. *find the best individual*
10. *test the best individual with testing data and gain the testing error*
11. *%% constructive method*
12. ***If** mod(iter,3) ==0*
13. ***If** $testing_{error}(C_t) < testing_{error}(C_{t-1})$ then* $dim = C_t$
14. ***Else if** $testing_{error}(C_t) = testing_{error}(C_{t-1})$ and $testing_{error}(C_t) < testing_{error}(C_{t-2})$ and $Complexity(C_t) < Complexity(C_{t-1})$ then* $dim = C_t$
15. ***Else if** $testing_{error}(C_t) = testing_{error}(C_{t-1})$ and $testing_{error}(C_t) = testing_{error}(C_{t-2})$ and $comp(C_t) < comp(C_{t-2})$ then* $dim = C_t$
16. ***Else if** (N-hidden-layer ==1 && rand<p);*
17. *N-hidden-layer = 2;*
18. *N-hidden = Min-hidden;*
19. ***Else***
20. $Dim = randi(comp(C_{t-1}), comp(C_{t-2}))$
21. ***End if***
22. ***End if***
23. ***End While***

Figure 1. Pseudo code combined algorithm

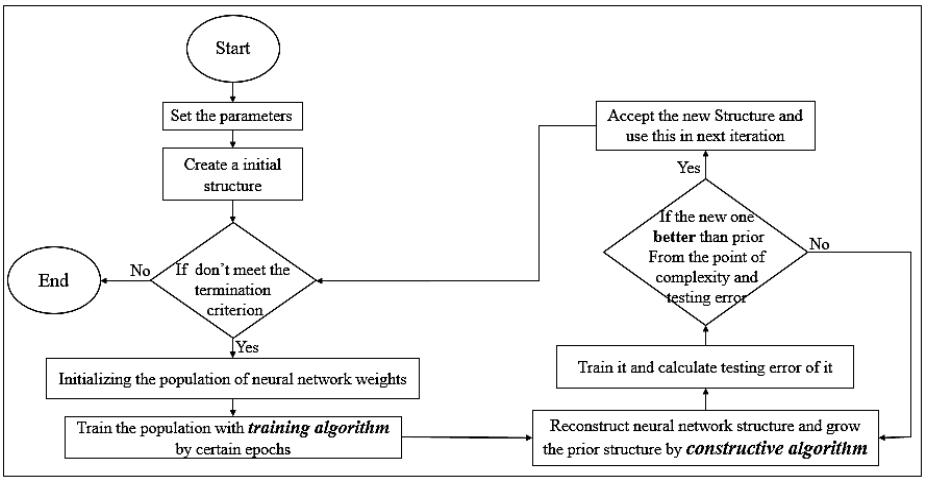

Figure 2. Flowchart of hybrid algorithm

4. COMPARISON RESULTS

In this section, we evaluate the effectiveness of proposed hybrid methods. These algorithms are applied to ten classification problem and two time series prediction problems. We compare the performance of the proposed hybrid algorithms first with each other and then with other available methods.

4.1. Definition of classification and time series prediction problems

The task of assigning a sample to a proper group, based on the characteristics of describing that object in a problem, is defined as classification. The classification problems used in this article include iris, diabetes diagnosis, thyroid, breast cancer, credit card, glass, heart, wine, page blocks, and liver. These classification problems are taken from the UCI machine learning repository [9]. But the time series prediction problems use a specific model to predict future values based on their previous values. The first is the Gas Furnace Dataset [10], which is compiled from Jenkins's Book of Time Series Analysis. It contains gas content and CO2 percentage in gas, and another is a Mackey glass dataset obtained from the below differential equation:

$$\frac{dx(t)}{dt} = -bx(t) + \frac{ax(t-t_d)}{1+x^{10}(t-t_d)} \quad (2)$$

All proposed hybrid algorithms in this article have been implemented using MATLAB software and have used 30 time run to evaluate the performance of these methods. The 4-fold-cross-validation method has been used to divide the original dataset into two training and testing sets. This method can effectively prevent trapping to local minima. Because both the training and testing samples contribute to learning as much as possible, it can provide a satisfactory learning effect. The average error is obtained from the 4-fold-cross-validation which is presented as the final error of the network. In addition, the input dataset to the neural network is normalized using the min-max normalization method to the interval [-1.1]. The results of the comparison are presented in two parts. First, the proposed algorithms are compared with each other, and then the best proposed method is compared with the existing methods.

4.2. Comparing proposed methods with each other

Each of these algorithms has been executed 30 times, and the results of the experiments have been compared with each other according to three criteria: classification error percentage of training and testing data and complexity percentage. The function of error calculation For the Mackey glass is RMSE and for gas furnace is MSE. First, we compare the performance of two kind of training algorithm that consist of classic training algorithm (back- propagation) and evolutionary training algorithm (improved teaching learning-based optimization). The results from Table 1 show that the ITLBO algorithm has a higher efficiency for most data sets. According to Table 1, the ITLBO algorithm for all of classification problems has better performance than the Bp algorithm, then in part2 from Table 1 we showed the results of comparing proposed hybrid algorithms with each other. All the results are based on three characteristics (parameters) of training and testing error for classification, MSE error and complexity. To better demonstrate the superior algorithm, we did rank average test, and the rank average for different data set was presented in Table 2. As can be seen

from Table 2, MCO-ITLBO has gained the first rank for all of characteristic. To evaluate whether the MCO-ITLBO results are significantly better than other approaches, we calculated the p-value test with a significant level of 0.05 for data sets. The calculated P- values for MCO-ITLBO are shown in Table 3 in comparison with other algorithms. The best results are bolded in the tables. In Figures 3 the box plot graphs showed the results of the distribution of training and testing errors for the whole data set for 30 times running. The charts show that MCO-ITLBO is superior in most cases.

Table 1. Average results of 30 runs of two kind training algorithm (part1) and each hybrid algorithm (part2)

Dataset	Criteria	BP	ITLBO (part1)	MCO-TLBO	MCO-ITLBO	MCO-BP (part 2)
1. Iris	Training error (%Class)	5.3125	**0.9516**	0.7819	**0.0138**	7.6061
	Testing error (%Class)	7.5468	**2.1440**	3.2752	**2.2975**	9.8990
	Training error (MSE)	0.0024	**3.5642e-09**	6.7324e-05	**6.2480e-07**	1.0290e-02
	Testing error (MSE)	0.4892	**1.7729e-06**	0.0721	**3.3917e-03**	1.2864
	Connection	fix	fix	26.4667	**21.5806**	23.4667
2. Diabetes	Training error (%Class)	31.2322	**18.6897**	19.5282	**18.8775**	28.0176
	Testing error (%Class)	35.2887	**27.8473**	29.3795	**23.3906**	36.8359
	Training error (MSE)	0.6013	**1.1637e-06**	1.529e-08	**2.3468e-10**	6.3375e-04
	Testing error (MSE)	0.6933	**7.1247e-05**	0.00179	**4.1822e-8**	1.3349
	Connection	fix	fix	109.1333	**58.4194**	129.2333
3. Thyroid	Training error (%Class)	17.8340	**6.7798**	8.2215	**5.6059**	18.3961
	Testing error (%Class)	16.5672	**12.5903**	11.1318	**6.5800**	22.3837
	Training error (MSE)	0.1399	**1.3987e-05**	4.8976e-06	**5.6589e-09**	3.8605e-03
	Testing error (MSE)	1.5968	**2.5786e-04**	0.00402	**2.6158e-07**	1.4095
	Connection	fix	fix	521.1	**133.6452**	168.2235
4. Cancer	Training error (%Class)	20.9419	**1.9672**	1.7213	**1.0843**	21.5350
	Testing error (%Class)	16.3510	**4.5018**	8.594	**2.0729**	29.5131
	Training error (MSE)	0.1437	**3.0600e-04**	2.3604e-07	**2.5695e-11**	1.2626e-03
	Testing error (MSE)	0.8224	**0.0220**	0.00336	**7.3679e-6**	7.4285
	Connection	fix	fix	257.9333	**67.4194**	77.4545
5. Card	Training error (%Class)	22.5247	**15.7492**	12.6111	**12.2379**	19.9432
	Testing error (%Class)	23.8126	**17.3196**	24.3478	**13.5889**	30.4480
	Training error (MSE)	0.0258	**3.3272e-09**	2.2402e-04	**7.2066e-08**	3.4985e-05
	Testing error (MSE)	1.1437	**3.0939e-05**	0.0184	**1.9381e-03**	3.0699
	Connection	fix	fix	452.6667	**127.4194**	183.5455
6. Glass	Training error (%Class)	36.7770	**18.3169**	18.2252	**17.0881**	66.8305
	Testing error (%Class)	54.8387	**36.9146**	31.7172	**22.0733**	67.9063
	Training error (MSE)	0.4235	**1.0244e-06**	9.2081e-04	**2.3845e-10**	1.4604e-04
	Testing error (MSE)	0.8631	**0.1431**	0.0964	**3.269e-06**	3.7141
	Connection	fix	fix	392.6667	123.3548	**104.7273**
7. Heart	Training error (%Class)	21.2208	**10.4473**	12.1587	**11.2647**	20.2982
	Testing error (%Class)	23.6746	**20.9877**	20	**13.3325**	23.4254
	Training error (MSE)	0.0022	**7.4441e-11**	9.4259e-05	**8.2271e-12**	2.4283e-06
	Testing error (MSE)	0.8086	**0.0156**	8.7661e-03	**1.0260e-07**	0.2213
	Connection	fix	fix	198.7333	**93.3548**	100.9091
8. Wine	Training error (%Class)	17.0641	**4.9978**	0.51491	**0.4687**	21.0421
	Testing error (%Class)	23.1063	**13.1856**	3.8182	**3.8094**	23.5537
	Training error (MSE)	0.0215	**1.6924e-09**	3.9983e-08	**1.9516e-11**	1.5403e-06
	Testing error (MSE)	1.0307	**0.0089**	1.2784e-03	**5.9690e-06**	5.1870
	Connection	fix	fix	286.9333	117.6129	**98.9091**
9. Page-blocks	Training error (%Class)	10.0000	**6.1260**	6.918	**6.3471**	12.8043
	Testing error (%Class)	12.2151	**7.2960**	13.309	**6.4575**	16.6030
	Training error (MSE)	0.0993	**3.3001e-07**	2.6707e-09	**2.1143e-19**	1.3954e-10
	Testing error (MSE)	0.8666	**5.1312e-04**	0.02848	**1.0861e-08**	0.0584
	Connection	fix	fix	328.9333	107.7097	**64.0909**
10. Liver	Training error (%Class)	37.6344	**24.3154**	23.9419	**22.9577**	45.2282
	Testing error (%Class)	38.5543	**25.8622**	38.75	**27.9210**	43.8811
	Training error (MSE)	0.0919	**7.0024e-09**	2.3477e-05	**5.4317e-07**	9.2163e-03
	Testing error (MSE)	0.5596	**0.0016**	0.0267	**2.419e-03**	0.3244
	Connection	fix	fix	86.4667	47.2903	**31.2727**
11. Mackey-glass	Training error (%Class)	**8.3054e-06**	3.7775e-04	2.2489e-04	1.5475e-05	**3.8468e-06**
	Testing error (%Class)	**0.0070**	0.0280	2.3505e-04	**2.0770e-04**	0.0560
	Connection	fix	fix	47.75	40.8710	**16.4839**
12. Gas furnace	Training error (%Class)	**3.5329e-04**	0.0071	1.5932e-3	**1.2101e-04**	6.846e-04
	Testing error (%Class)	4.6352	**0.1987**	**1.6671e-3**	3.2001e-03	0.0228
	Connection	fix	fix	62.6	47.1935	**15.4516**

Table 2. Average of the ranks for proposed algorithms

Training error (Classification)	Algorithm	MCO-TLBO	MCO-ITLBO	MCO-Bp
	Rank	2.0000	**1.0000**	3.0000
Testing error (Classification)	Algorithm	MCO-TLBO	MCO-ITLBO	MCO-Bp
	Rank	2.0000	**1.0000**	3.0000
Training error (Mse)	Algorithm	MCO-TLBO	MCO-ITLBO	MCO-Bp
	Rank	2.4000	**1.0000**	2.6000
Testing error (Mse)	Algorithm	MCO-TLBO	MCO-ITLBO	MCO-Bp
	Rank	2.0000	**1.0000**	3.0000
Number of connections	Algorithm	MCO-TLBO	MCO-ITLBO	MCO-Bp
	Rank	2.8000	**1.4000**	1.8000

Table 3. P-value results for pairwise comparison of MCO-ITLBO versus other algorithms by wilcoxon test

Dataset	Criteria	MCO-TLBO	MCO-Bp
1. Iris	testing error	**0.65641**	9.0733e-12
	connection	**0.1258**	1.2196e-11
2. Diabetes	testing error	0.093779	2.9174e-11
	connection	**0.30617**	9.8375e-09
3. Thyroid	testing error	1.2369e-05	5.9941e-07
	connection	0.00010012	7.0643e-09
4. Cancer	testing error	**0.40898**	2.6757e-11
	connection	**0.31604**	5.0422e-11
5. Card	testing error	**0.77239**	5.5893e-11
	connection	**0.1433**	2.9321e-11
6. Glass	testing error	0.03935	2.8502e-11
	connection	0.006635	3.9787e-10
7. Heart	testing error	**0.3416**	2.7722e-11
	connection	**0.51018**	1.497e-10
8. Wine	testing error	0.043924	2.3481e-11
	connection	0.02634	3.1372e-11
9. Page-blockes	testing error	0.00338	3.0123e-11
	connection	**0.04819**	2.9991e-11
10. liver	testing error	**0.74965**	2.8991e-11
	connection	**0.69925**	3.6305e-09
11. Mackey-Glass	testing error	0.0011143	0.007959
	connection	**0.19073**	0.00058737
12. Gas Furnace	testing error	**0.14532**	5.462e-06
	connection	3.352e-08	0.0050842

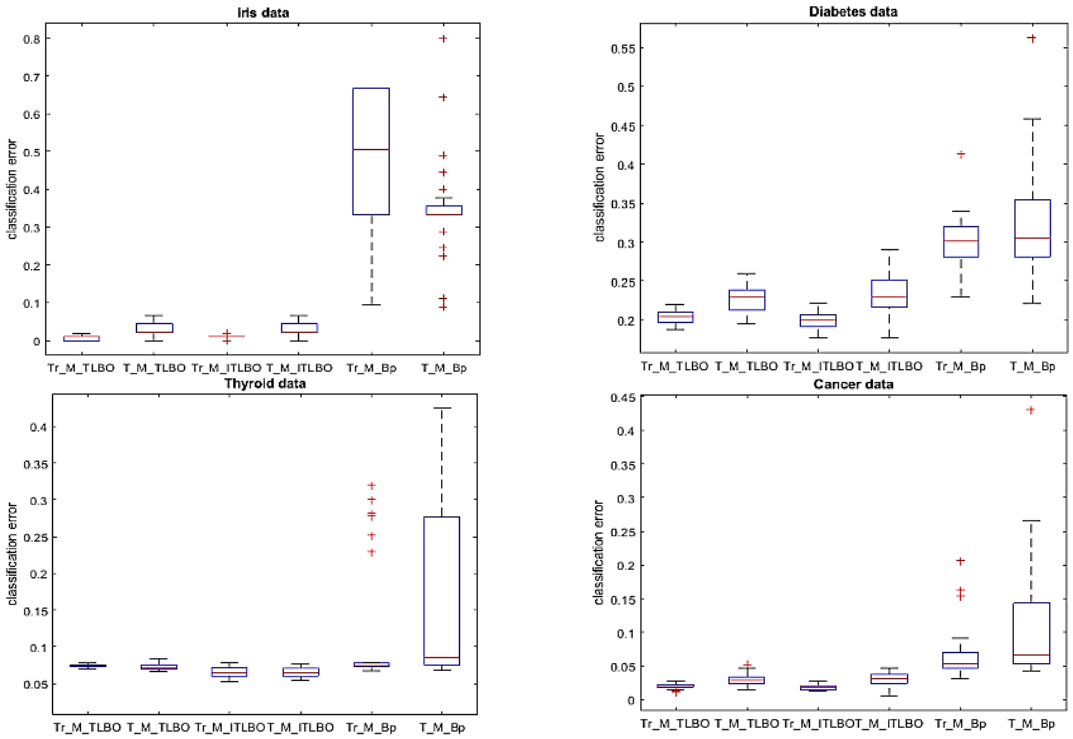

Figure 3. Box plots of training and testing errors for all datasets

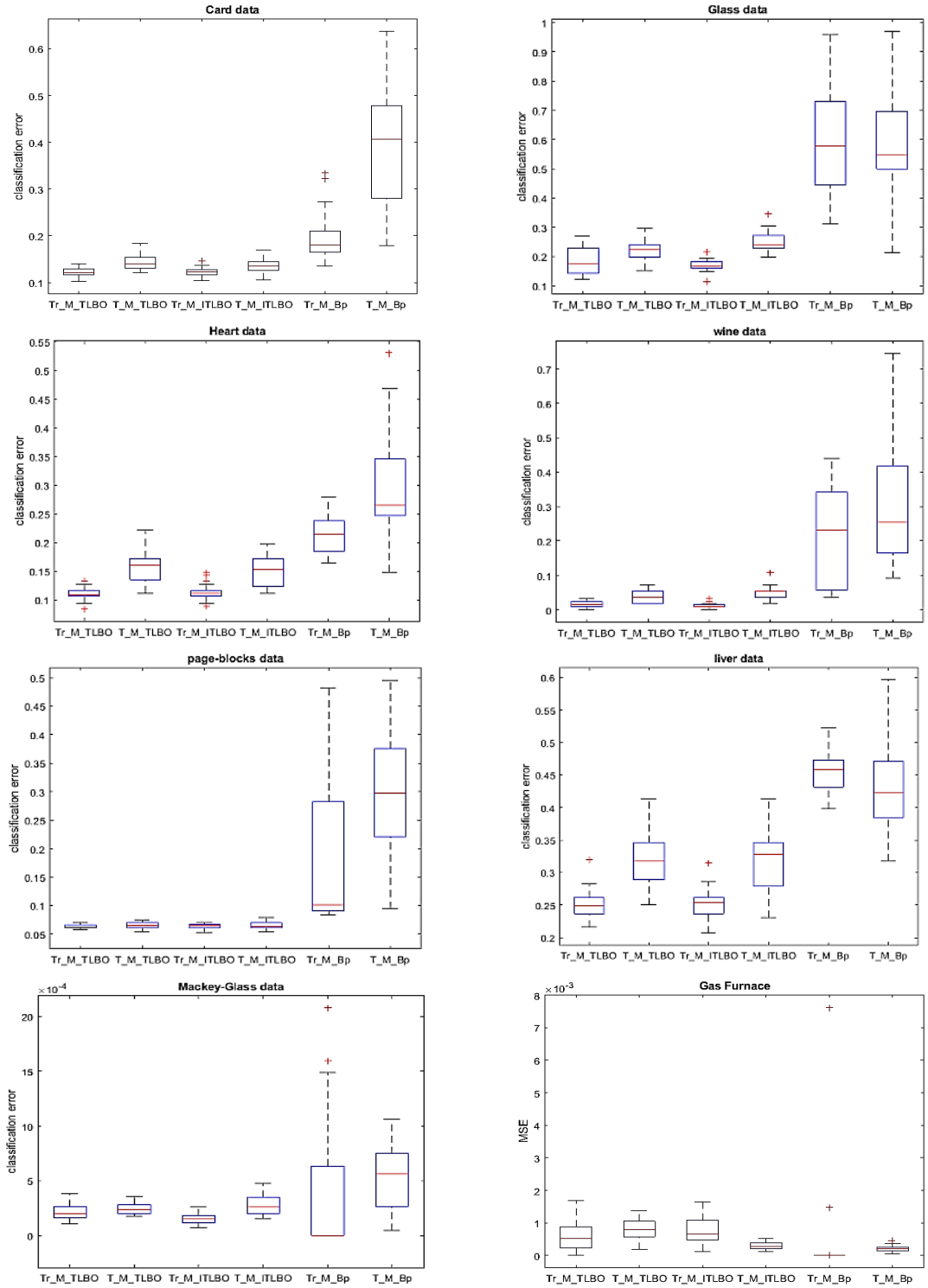

Figure 3. Box plots of training and testing errors for all datasets (*continue*)

4.3. Results of comparing the best proposed hybrid method with other methods

In this section, we compare our hybrid algorithms with other literature methods in Table 4. The percentage of training error and testing error collection in this table. Each article works on a batch of datasets. The cells of this table that don't have any value (that indicate with an - icon) shows that these values are missing data or belong to a dataset that articles don't work on this. We give a brief description of the comparative approaches as follows. We reference all the approaches that we compared our best proposed method with them.

Table 4. Comparing the results of best algorithm with other methods in literature

Dataset	Ceretia	Other approach																Proposed algorithm
Iris	Training error (%)	[3] 6.000	[11] 3.73	[15] -	[16] 6.67	[18] 1.7978	[19] -	[20] -	[21] 3.4	[22] 1.5311	[23] 2	[25] 2.3	[26] -	[27] 2.7	[28] 0.88	[7] -	[29] 5.67	**0.0138**
	Testing error (%)			3.33		1.9573	2.00	3.73	-	1.6906		2.2	2.67	4.87	4.37	3.33		2.2975
Diabetes	Training error (%)	-	[14] -	[15] -	[17] 29.87	[18] 14.9689	-	[20] -	[21] 22.91	[22] **13.968**	-	[25] 21.6	[26] -	-	-			18.8775
	Testing error (%)		21.66	21.63		16.635		23.09	-	**15.635**		21.8	23.9583					23.3906
Thyroid	Training error (%)		- -	[15] -	[17] 9.72	[18] **5.2770**	-	-	[21] 6.89	[22] 5.477	-	-	-	-	-	[7] -	[29] 5.85	5.6059
	Testing error (%)			0.38		5.4104			-	**5.1104**						6.05		6.5800
Cancer	Training error (%)	[11] 26.97	[14] -	[15] -	[16] 7.14	[18] 2.9596	-	[20] -	[21] 26.84	[22] 2.3596	[24] 5.34	[25] 2.6	[26] -	[27] 2.27	-	-	-	**1.0843**
	Testing error (%)		2.59	27		3.0263		3.13		2.4596	-	2.3	2.365	2.39				**2.0729**
Card	Training error (%)	-	-	-		[18] 12.9080	-	-	-	[22] 12.574	-	-	-	-	-	[7] -	-	**12.2379**
	Testing error (%)					13.1498				12.583						**7.97**		13.5889
Wine	Training error (%)	[12] 41.48	[14] -	[15] -	[16] 4.75	-	[19] -	[20] -	- -	-	[24] 3.95	[25] 1.6	[26] -	[27] 1.55	[28] 3.92	[7] -	[29] 3.38	**0.4687**
	Testing error (%)		**0.48**	1.12	-		0.56	2.81			-	**0.3**	1.11	6.3	4.66	2.81		3.8094
Heart	Training error (%)	-	[14] -	[15] -	-	-	[19] -	[20] -	-	-	[23] 11.41	[25] -	-	-	-	[7] 15.74	-	**11.2647**
	Testing error (%)		13.56	15.55			13.59	40.88			-	16.6						**13.3325**
Liver	Training error (%)	-	[14] -	-	-	-	-	-	-	-	-	-	[26] 24.28	-	-	[7] **20.87**	-	22.9577
	Testing error (%)		27.98															**27.9210**
page-blocks	Training error (%)	-	-	[15] -	[17] 5.56	-	[19] -	-	-	-	[24] **3.92**	-	-	-	-	-	-	6.3471
	Testing error (%)			**3.02**			3.58											6.4575
Glass	Training error (%)	-	[14] -	[15] -	-	[18] 33.0586	[19] -	[20] 38.94	-	[22] 26.058	[24] 26.55	[23] 29.48	[26] -	-	-	[7] 26.17	[29] 35.05	**17.0881**
	Testing error (%)		31.23	20.74		34.0919	26.69	-		28.091		-	25.6					**22.0733**
Mackey-Glass	Training error (%)	[13] 5.5E-04	-	-	-	[18] 1.6E-05		-	-	[22] 1.4E-03	-	-	-	-	-	-		**1.5475e-05**
	Testing error (%)	4.7E-04				**1.7E-05**				1.4E-03								2.0770e-04
Gas Furnace	Training error (%)	[13] 0.003	-	-	-	[18] 0.2637		-	-	[22] 0.1837	-	-	-	-	-	-	-	**1.2101e-04**
	Testing error (%)	0.004				0.2870				0.1903								**3.2001e-03**

5. CONCLUSION

In this paper, we proposed a hybridization of training algorithms and constructive algorithms to simultaneously determine the weight and structure of the neural network. The goal is to examine hybridization of a deterministic and systematic procedure (constructive algorithm) with random search (evolutionary algorithm) for neural network optimization. Combined methods include the base and improved version of the TLBO algorithm with the MMOST algorithms. Then we compared hybrid algorithms, and selected the superior algorithm in classification and time series prediction problems. The results of the comparison illustrate the superior performance belongs to the MCO-ITLBO algorithm. This version has a powerful training algorithm against early convergence, and balances between exploitation and exploration. This algorithm in combination with the MMOST constructive algorithm, more effectively selects the optimal network structure. We have also verified these results with statistical tests, and finally this algorithm was compared with other methods in literature and it has been proven that it is more convenient than other algorithms for classification and time series prediction error. These promising results motivate us to find ways to change our path to future work. This development can be using chaotic (disorder) mappings in this method.

REFERENCES

[1] N. Zhang, "An online gradient method with momentum for two-layer feedforward neural networks," *Applied Mathematics and Computation*, vol. 212, no. 2, pp.488–498, 2009.
[2] M. Gori, A. Tesi, "On the problem of local minima in back-propagation," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 14, no. 1, pp. 76–86, 1992.
[3] H. Melo, J. Watada, "Gaussian-PSO with fuzzy reasoning based on structural learning," *Neurocomputing*, vol. 172, pp. 405-412, 2016.
[4] S. Yang, Y. Chen, "An evolutionary constructive and pruning algorithm for artificial neural," *Neurocomputing*, vol. 86, pp. 140–149, 2012.

[5] R. V. Rao, V. J. Savsani, D. P. Vakharia, "Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems," *Computer-Aided Design*, vol. 43, no. 3, pp. 303-315, 2011.
[6] V. Ho-Huu, T. Nguyen-Thoi, T. Vo-Duy, T. Nguyen-Trang, "An adaptive elitist differential evolution for optimization of truss structures with discrete design variables," *Computers and Structures*, vol. 165, pp. 59–75, 2016.
[7] O. Aran, E. Alpaydın, "An incremental neural network construction algorithm for training multilayer perceptrons," *Artificial Neural Networks and Neural Information Processing*, 2003.
[8] M. Azevedo, Braga, A. Pádua, de Menezes, B. Rodrigues, "Improving neural networks generalization with new constructive and pruning methods," *Journal of Intelligent & Fuzzy Systems*, vol. 13, no. 2, pp. 75-83, 2002.
[9] C. L. Blake, C. J. Merz, "UCI Repository of Machine Learning Databases, University of California at Irvine," 1998.
[10] Available at http://datasets.connectmv.com/datasets/
[11] M. Khishe, M. R. Mosavi, M. Kaveh, "Improved migration models of biogeography-based optimization for sonar dataset classification by using neural network," *Applied Acoustics*, vol. 118, pp. 15–29, 2017.
[12] Q. Fan, Z. Wang, H. Zha, D. Gao, "MREKLM: A fast multiple empirical kernel learning machine," *Pattern Recognition*, vol. 61, pp. 197-209 2017.
[13] N. S. Jaddi, S, Abdullah, A. R. Hamdan, "A solution representation of genetic algorithm for neural network weights and structure,". *Information Processing Letters*, vol. 116, no. 1, pp.22-25, 2016.
[14] R. M. Cruz, R. Sabourin, G. D. Cavalcanti, "META-DES. Oracle: Meta-learning and feature selection for dynamic ensemble selection," *Information Fusion*, vol. 38, pp. 84-103, 2017.
[15] H. Badem, A. Basturk, A. Caliskan, M. E. Yuksel, "A new efficient training strategy for deep neural networks by hybridization of artificial bee colony and limited--memory BFGS optimization algorithms," Neurocomputing, vol. 266, pp. 506–526, 2017.
[16] M. De Gregorio M. Giordano, "An experimental evaluation of weightless neural networks for multi-class classification," *Appl. Soft Comput.*, vol. 72, pp. 338-354, 2018.
[17] C. Zhang, C. Liu, X. Zhang, G. Almpanidis, "An up-to-date comparison of state-of-the-art classification algorithms," *Expert Systems with Applications*, vol. *82*, pp.128-150, 2017.
[18] N. S. Jaddi, S. Abdullah, "A cooperative-competitive master-slave global-best harmony search for ANN optimization and water-quality prediction," *Applied Soft Computing*, vol. *51*, pp.209-224, 2017.
[19] J. Vashishtha, P. Goyal, J. Ahuja, and others, "A Novel Fitness Computation Framework for Nature Inspired Classification Algorithms," *Procedia Comput. Sci.*, vol. 132, pp. 208–217, 2018.
[20] D. P. F. Cruz, R. D. Maia, L. A. da Silva, L. N. de Castro," BeeRBF: a bee-inspired data clustering approach to design RBF neural network classifiers," *Neurocomputing*, vol. *172*, pp. 427-437, 2016.
[21] Z. Xie, Y. Xu, Q. Hu, "Uncertain data classification with additive kernel support vector machine," *Data Knowl. Eng.* vol. 117, pp. 87-97, 2018.
[22] N. S. Jaddi, S. Abdullah, M. Abdul Malek, "Master-Leader-Slave Cuckoo Search with Parameter Control for ann Optimization and Its Real-World Application to Water Quality Prediction," *PLoS ONE. vol.12(1)*, 2017.
[23] A. Caliskan, M. E. Yuksel, H. Badem, A. Basturk, "Performance improvement of deep neural network classifiers by a simple training strategy," *Engineering Applications of Artificial Intelligence.* vol. 67, pp. 14–23, 2018.
[24] M. F. Mohammed, C. P. Lim, "Improving the Fuzzy Min-Max neural network with a K-nearest hyperbox expansion rule for pattern classification," *Applied Soft Computing*, vol. 52, pp. 135-145, 2017.
[25] B. Y. Hiew, S. C. Tan, W. S. Lim, "A double-elimination-tournament-based competitive co-evolutionary artificial neural network classifier," *Neurocomputing*, vol. *249*, pp. 345-356, 2017.
[26] K. Sun, J. Zhang, C. Zhang, J. Hu, "Generalized extreme learning machine autoencoder and a new deep neural network," *Neurocomputing*, vol. *230*, pp. 374-381, 2017.
[27] Z. Gao, C. Ma, D. Song, Y. Liu, "Deep quantum inspired neural network with application to aircraft fuel system fault diagnosis," *Neurocomputing*, vol. *238*, pp. 13-23, 2017.
[28] J. Wang, Q. Cai, Q. Chang, J. M. Zurada, "Convergence analyses on sparse feedforward neural networks via group lasso regularization," *Information Sciences*, vol. 381, pp. 250-269, 2017.
[29] S. Shinde, U. Kulkarni, "Extended fuzzy hyperline-segment neural network with classification rule extraction," *Neurocomputing*, vol. 260, pp. 79–91, 2017.