Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

An improvement on genetic-based learning method for fuzzy artificial neural networks

M. Reza Mashinchi [*], Ali Selamat

Faculty of Computer Science and Information System, Universiti Teknologi Malaysia, Johor, Malaysia

ARTICLE INFO

Article history:
Received 8 August 2008
Received in revised form 17 February 2009
Accepted 12 March 2009
Available online 9 April 2009

Keywords:
Fuzzy artificial neural network
Genetic algorithm
Two-phase learning
Optimal fuzzy weight

ABSTRACT

Fuzzy artificial neural networks (FANNs), which are the generalizations of artificial neural networks (ANNs), refer to connectionist systems in which all inputs, outputs, weights and biases may be fuzzy values. This paper proposes a two-phase learning method for FANNs, which reduces the generated error based on genetic algorithms (GAs). The optimization process is held on the alpha cuts of each fuzzy weight. Global optimized values of the alpha cuts at zero and one levels are obtained in the first phase and optimal values of several other alpha cuts are obtained in the second phase. Proposed method is shown to be superior in terms of generated error and executed time when compared with basic GA-based algorithms.

1. Introduction

Inexact data, which have been treated as fuzzy sets by Zadeh [30], are present in most environments. This inexactness, which is represented by a membership function, has been used in many applications in soft computing [25]. Such applications are agriculture [8], market prediction [16], image processing [18], polynomial fuzzy functions [2,25] etc. Although ANNs, which are well known as soft computing techniques, have been used to find and tune membership functions and to construct different fuzzy inference approaches [26], it is known that using hybrid techniques of soft computing gives the result that are more promising. Genetic computing (GC), fuzzy logic (FL) and neuro computing (NC) are among the principal techniques used in soft computing [26].

Because of the proliferation of inexact data, there is a great need for finding a fitting function for such data [3]. Since FANNs are techniques that integrate the ability to simultaneously receive these fuzzy data and to study input–output relationships, they can be used to find such a fitting function. So far, two major learning methods have been proposed for FANNs [19]. One approach applies a back-propagation (BP) algorithm, while the other utilizes genetic algorithms (GAs). In a rudimentary study using the BP algorithm a direct fuzzification, which fuzzifies the delta-rule learning for the feed-forward ANNs [3,4,6,11], was considered, but this has been rejected from the theoretical point of view [26]. The learning methods for triangular symmetric fuzzy values proposed in which FANNs are restricted to have triangular and symmetric fuzzy weights [14,15]. Recently an approach based on derivation of min-max function [31,32], which is able to deal with all types of bounded convex fuzzy values, has been discussed [19,20]. However, since the BP algorithm considers local-optimization then it does not guarantee to avoid being trapped in local minima. Another learning approach is based on GAs, were first introduced in [1,5]. This method has the ability of dealing with all types of bounded convex fuzzy data. However, the low speed of learning convergence is an issue that remains to be resolved. Therefore, a new approach is required which:

(a) reduces the possibility of being trapped in local minima,
(b) increases the speed of learning convergence and
(c) processes any bounded convex fuzzy data.

The issues (a) and (b) are more critical when more alpha cuts (α-cuts), which are defined in (2)–(4), are used to identify a fuzzy value. This is due to using more variables need to be optimized, which makes more possibility of being trapped in local minima and takes more processing time to converge. It is worth mentioning that theoretically, the principal role of α-cuts in fuzzy set theory is their capability to represent fuzzy sets. Indeed, each fuzzy set can uniquely be represented by the family of all its α-cuts. In fact, knowing all α-cuts of a fuzzy set is the same as knowing uniquely this fuzzy set itself. This fact is called decomposition or resolution of identity that is studied vastly in the literature by many authors and it is used in theory and applications [13,17]. Therefore, the

* Corresponding author.
E-mail addresses: r_mashinchi@yahoo.com (M. Reza Mashinchi), aselamat@utm.my (A. Selamat).

1568-4946/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2009.03.011

motivation of proposing an approach, which is able to enhance the learning method by increasing the number of computed α-cuts, is worth acceptable idea.

To the aforementioned, in this paper a two-phased GA-based learning method is proposed. The first phase roughly estimates the optimal fuzzy weights, and the second phase provides better estimates for the shape of the membership function.

The rest of paper is organized as follows. In Section 2 some basic notions related to FANNs are reviewed. In Section 3, GA-based learning approach for FANNs is discussed. There, the newly proposed two-phased GA-based learning method is presented and discussed in Section 3.1. Simulations carry out the comparisons of this new method and the GA-based learning approach, based on two famous datasets in the literature, in Section of 3.2. Finally, summary and discussion for future work is provided in Section 4. In addition, Appendix A is given for the datasets used in this paper.

2. Fuzzy neural networks

Current studies on neural computations are closely related to operational aspects that deal with the uncertainties. In this regard, fuzzy logic is integrated with ordinary ANNs to endow them with uncertain information processing capabilities [1]. This integration provides a new alternative class of ANNs called FANNs, the development of which has been arousing with increasing interest [7]. The main idea of FANNs is to use fuzzy set theory to generalize ANNs. In fact, FANNs are the techniques have the ability of learning in qualitative environments by fitting a function to a given set of inexact data [12,24]. However, their low speed of convergence and poor accuracy of available learning methods makes them to be unsuitable for most problems [21,22]. These are due to the complexity of fuzzy value abstracted from some exact values and, thus, learning such data are more difficult. Therefore, there is a need for a better learning method, one of which is proposed in this paper. We recall that there are three different types of FANNs depending on the types of inputs and weight coefficients [1,10]:

- FANN_1: fuzzy weights and crisp inputs;
- FANN_2: crisp weights and fuzzy inputs;
- FANN_3: fuzzy weights and fuzzy inputs.

This paper deals with the most generalized case, FANN_3, and we first discuss some basic notions which will be needed later.

A general structure of a feed-forward FANN_3 with three layers is illustrated in Fig. 1. A fuzzy number $\tilde{A}$ is defined below:

$$\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) | x \in \Re, \quad \mu_{\tilde{A}} : \Re \rightarrow [0, 1]\}, \tag{1}$$

where $\mu_{\tilde{A}}$ is a continuous membership function and $\Re$ is the set of all real numbers. The definitions for α-cuts as core, support and

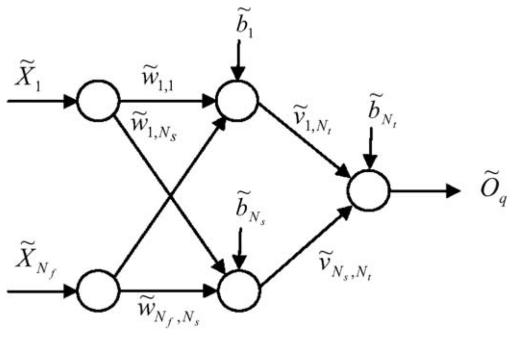

Fig. 1. A three-layer fuzzy neural network architecture.

middle α-cut of a fuzzy number $\tilde{A}$ are as follows:

$$\tilde{A}_1 = Core(\tilde{A}) = \{x \in \Re | \mu_{\tilde{A}}(x) = 1\}, \tag{2}$$

$$\tilde{A}_0 = Support(\tilde{A}) = \{x \in \Re | \mu_{\tilde{A}}(x) > 0\}, \tag{3}$$

$$\tilde{A}_\alpha = middle_\alpha(\tilde{A}) = \{x \in \Re | \mu_{\tilde{A}}(x) \geq \alpha\}, \quad \alpha \in (0, 1). \tag{4}$$

All weights, biases, inputs and consequently outputs, are assumed to be fuzzy numbers. The first layer is input layer and it does not have any computational unit or synaptic weight. In the second layer, the matrix of fuzzy weights, $\tilde{w}_{N_f,N_s}$, shows fuzzy weights connecting neuron N_f in the first layer to neuron N_s in the second layer. The vector of fuzzy biases, $\tilde{b}_{N_s}$, in the second layer shows fuzzy bias of neuron, N_s, in this layer. Similarly, in the third layer, fuzzy weight matrix, $\tilde{v}_{N_s,N_t}$, shows fuzzy weights connecting neuron N_s in the second layer to neuron N_t in the third layer.

The form of activation function of neurons in the first and second layers is sigmoid function given below:

$$f(x) = \frac{1}{1 + e^{-x}}, \quad x \in \Re. \tag{5}$$

Fuzzy output of $\tilde{Inter}_{N_s}$, for the second layer of this architecture is defined as follows:

$$\tilde{Inter}_{N_s} = f(\tilde{Agg}_{N_s}), \quad N_s = 1, 2 \ldots, n, \tag{6}$$

where N_s is the number of neurons in the second layer and $\tilde{Agg}_{N_s}$ is defined as follows:

$$\tilde{Agg}_{N_s} = \sum_{i=1}^{N_f} \tilde{X}_i \tilde{w}_{ij} + \tilde{b}_j, \quad j = 1, 2 \ldots, N_s, \tag{7}$$

where N_f is the number of neurons in the first layer and $\tilde{X}_{N_f}$ is fuzzy input. The third layer receives the $\tilde{Agg}$ values from neurons in the second layer through their fuzzy weight $\tilde{v}$. Therefore, the output is given by

$$\tilde{O}_q = \sum_{j=1}^{Ns} \tilde{Agg}_j \tilde{v}_{jq}, \quad q = 1, 2 \ldots, N_t, \tag{8}$$

where N_f is the number of neurons in the third layer, and $\tilde{O}_q$, is fuzzy output. Note that in this paper, fuzzy multiplication and fuzzy addition of two fuzzy values are defined in (12) and (13).

3. GA-based learning approach for FANNs

GA is a field of study for optimization which was pioneered in the early 1970s for multi-dimensional continuous optimization of multi-modal functions [7,29]. It provides a global optimization technique based on the principles of natural evolution [9,12,23,29], in which chromosomes are simulated for implementation of solution candidates and the notion of fitness is used to measure the goodness of a candidate for an optimized solution [28]. Selection, crossover, and mutation as genetic operators are repeatedly applied to the population to increase the fitness of chromosomes [29]. A selection operator randomly chooses chromosomes from the population to make a new generation with probabilities proportional to the values of the fitness of chromosomes. The crossover operator randomly picks two chromosomes as parents and mates them in order to produce two offspring chromosomes. Mutation is a process, in which a single component of a chromosome is changed randomly [27]. Creating mutation offspring by small random changes in the population individually, will provide genetic diversity and enable the genetic algorithm to search in a broader space and thus avoid local minima [28].

Using the characteristics of GAs, as the basis for FANNs, leads to GA-based FANNs learning methods proposed in [1,5]. This approach uses GAs to improve the accuracy of learning through better adjustment of the weights. In order to do this, GA operates on generated random numbers producing sequences of the weights. The weights so produced are evaluated by a fitness function that reproduces the next generation. F_A the fitness function in (9), which is proposed by Aliev [1], is used to measure the fitness of the produced weights using a training set of pairs of given fuzzy inputs/targets; $\tilde{g}$ defined in (10) is a measure of distance of the target $\tilde{D}$ and actual output $\tilde{O}$.

$$\tilde{F}_A = \frac{1}{1+\tilde{g}}, \tag{9}$$

$$\tilde{g} = \sum_{k=1}^{n} |\tilde{D}_\alpha(k) - \tilde{O}_\alpha(k)|, \tag{10}$$

where n is the number of pairs in the training set and $\tilde{D}_\alpha$ and $\tilde{O}_\alpha$ represent the α-cut of $\tilde{D}$ and $\tilde{O}$, respectively, for $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$.

3.1. Improved GA-based learning approach

Henceforth, GBLMs will denote for genetic algorithm-based fuzzy artificial neural networks Learning Methods, and 2P-GBLM for proposed two-phase learning method, since the latter one is also based on GAs. The improvement in accuracy obtained by this method in terms of generated error is because of better adjustment of the weights. This is while the split of fuzzy value into two parts makes learning method to have less process in each generation rather than GBLM and, thus, less time needs to fulfill learning process. Throughout this paper, the error for $\tilde{A}_\alpha$ where $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$, for fuzzy numbers with bounded supports that is obtained from the weights in each generation is studied. The initial random generated population forms the left and right bounds of the support, core and $middle_\alpha$, $\alpha \in \{0.25, 0.5, 0.75\}$, for all fuzzy weights.

Two major steps in proposed algorithm are "place-definition" and "shape-definition" which are either referred as phases I and II, respectively. In place-definition phase, a suitable $\tilde{O}$ is computed in which the error for support and core would ideally be zero. In this phase, place and general shape of the output is obtained using the support and core of data values. Then shape-definition is applied to obtain an output closer to target shape using several other α-cuts defined in (4). Ideally, this output would also be error free; here the error is defined as the distance between output and target shape. Clearly, the more α-cuts we use, the closer the output shape will be to the target.

The whole process of 2P-GBLM viewed as a one learning block, consisting of two sub-processes, is illustrated in Fig. 2. The difference between GBLM and 2P-GBLM is the view to the relations of α-cuts in a fuzzy value, which it can facilitate to satisfy the factors of finding optimal solution. These factors are length of the chromosome and searching space for optimal solution, in GAs. Long length of the chromosome causes to have more variables that should be optimized. The optimizer has less efficiency if the number of variables increases. Therefore, it needs to have chromosomes of shorter length in order to accelerate the convergence to optimal solution. The broadness of search space is another factor of finding optimal solution. Complexity of the problem increases by the increment of search space broadness. Hence, obtaining to optimal solution would be more difficult. Therefore, in order to facilitate finding optimal solution, it needs to prepare a condition somehow that searching process would be done within a shortened space.

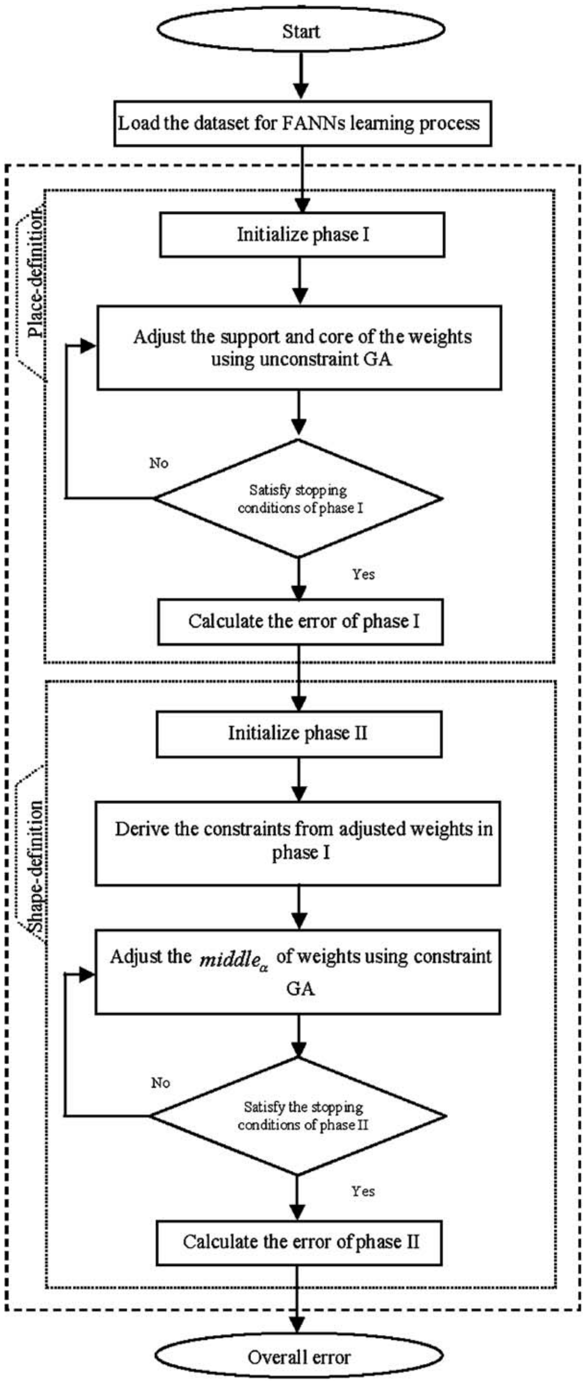
Start
Load the dataset for FANNs learning process
Place-definition
Initialize phase I
Adjust the support and core of the weights using unconstraint GA
No
Satisfy stopping conditions of phase I
Yes
Calculate the error of phase I
Shape-definition
Initialize phase II
Derive the constraints from adjusted weights in phase I
Adjust the $middle_\alpha$ of weights using constraint GA
No
Satisfy the stopping conditions of phase II
Yes
Calculate the error of phase II
Overall error

Fig. 2. 2P-GBLM schema.

A fuzzy value can be viewed physically or logically, regarding the relations between its constructive α-cuts. Each GBLM and 2P-GBLM use one of these two views to the optimization process. Both of the optimization methods are based on GAs and the process holds on a population of the chromosomes. Each chromosome of the population is constructed from boundary values of the α-cuts of a fuzzy value.

The GBLM method uses physical view. In this view, all α-cuts have the same role to construct fuzzy values. Consequently, each potential solution uses one chromosome for finding optimal solution. Increasing the number of α-cuts makes the length of chromosome to be longer, thus, obtaining the optimum would be postponed. In addition, unconstrained search space makes the process to be more complex, since it must consider an extremely broad space. In contrast to GBLM, 2P-GBLM uses the advantages of logical-relation view to constructive α-cuts of a fuzzy value, where the α-cuts categorized based on their features. These features are stemmed from convexity condition emerging the α-cuts to be nested. In this way, each fuzzy value consists of two categories of α-cuts. The first category includes support and core, while the second one includes $middle_\alpha$. Taking an advantage of this categorization, each chromosome as a potential solution splits into two chromosomes. GA uses the values of the first chromosome to constraint the space in order to search the optimum of the second chromosome. In consequence of constraining the second chromosome, search complexity decreases and the replications of searching process to find the optimum would be decreased.

GA, which is used in 2P-GBLM, utilizes two cases of global and local-optimization techniques through the optimization process. The global-optimization technique is used for the first chromosome, while the local-optimization one is utilized for the second chromosome. As the second chromosome formation depends on the first one, thus, the first and second ones are processed, respectively. Each of them can be processed in $x\,\%$ and $(n-x)\,\%$ of n learning generations, arbitrarily. A suitable x, leads to obtain the optimum in a less n generations. Therefore, an attempt to obtain a suitable x is considered in Section 3.2.

The FANN is trained by a dataset in the form of $((\tilde{X}_1(k), \tilde{X}_2(k)); \tilde{O}(k))$. A measure of distance for obtaining a new fitness function, F_P, is defined below:

$$F_P = g = \sum_{k=1}^{n} [(\tilde{D}^L_\alpha(k) - \tilde{O}^L_\alpha(k))^2 + (\tilde{D}^R_\alpha(k) - \tilde{O}^R_\alpha(k))^2], \tag{11}$$

where

- n is the number of training sets,
- $\tilde{D}^L_\alpha(K)$ ($\tilde{O}^L_\alpha(k)$) and $\tilde{D}^R_\alpha(K)$ ($\tilde{O}^R_\alpha(k)$) are left and right boundaries of target (outcome) intervals of the α-cuts for kth element of training set, respectively.where $\tilde{O}_\alpha(k)$ is outcome from Eqs. (6)–(8). Note that fuzzy multiplication and fuzzy addition of two fuzzy values $\tilde{A}$ and $\tilde{B}$ are used as follows:

$$\begin{aligned}\tilde{A}_\alpha(k)\cdot\tilde{B}_\alpha(k) &\equiv [\tilde{A}^L_\alpha(k), \tilde{A}^R_\alpha(k)]\cdot[\tilde{B}^L_\alpha(k)\cdot\tilde{B}^R_\alpha(k)] \\ &= [\min((\tilde{A}^L_\alpha(k)\cdot\tilde{B}^L_\alpha(k), \tilde{A}^L_\alpha(k)\cdot\tilde{B}^R_\alpha(k)), \max(\tilde{A}^R_\alpha(k)\cdot\tilde{B}^L_\alpha(k), \tilde{A}^R_\alpha(k)\cdot\tilde{B}^R_\alpha(k))]\end{aligned} \tag{12}$$

$$\begin{aligned}\tilde{A}_\alpha(k)+\tilde{B}_\alpha(k) &\equiv [\tilde{A}^L_\alpha(k), \tilde{A}^R_\alpha(k)] + [\tilde{B}^L_\alpha(k), \tilde{B}^R_\alpha(k)] \\ &= [\tilde{A}^L_\alpha(k)+\tilde{B}^L_\alpha(k), \tilde{A}^R_\alpha(k)+\tilde{B}^R_\alpha(k)]\end{aligned} \tag{13}$$

- $\tilde{A}^L_\alpha(k)$ ($\tilde{B}^L_\alpha(k)$) and $\tilde{A}^R_\alpha(k)$ ($\tilde{B}^R_\alpha(k)$) are left and right boundaries of the α-cuts of fuzzy value $\tilde{A}_\alpha(k)$ ($\tilde{B}_\alpha(k)$) for the kth element of training set, respectively.

To minimize fitness function F_P, first GAs are applied on the support and cores to be globally optimized using crossover and mutation functions. The results obtained are rough estimates for the optimized weights. Then, in order to find better estimates for the shape of these optimized weights, the obtained results are considered for applying GAs on $middle_\alpha$ with the same fitness function in a manner to be locally optimized using crossover and mutation functions. Fitness function returns a generated error based on α-cuts in each generation for each two phase separately. This process is repeated until $g < \varepsilon$, where ε is an extremely small positive number. Overall error is the addition of the first and the second phases errors, defined as the distance between target and outcome of 2P-GBLM. Naturally, the worse (better) the fitness values, the larger (smaller) the distance between the target and outcome of FANN will be.

3.2. Simulation

This subsection discusses the analysis of the simulation carried out in this research and is reported in four parts as follows:

- In Section 3.2.1, the effect of different allotments of percentages of each first and second phase for 2P-GBLM is analyzed.
- In Section 3.2.2, the superiority of proposed method is shown in terms of generated error and executed time. This is done by analyzing the results of four known methods using fitness functions, F_A, F_P and their corresponding distance measures utilized in each two GBLM and 2P-GBLM method.
- In Section 3.2.3, the accuracy of proposed method is illustrated in comparisons using mean squared error (MSE) for generated error and relative CPU time for executed time analysis.
- In Section 3.2.4, the trained FANN is tested to show the behavior of trained networks for new unknown inputs.

In these subsections, the comparisons have been done based on two terms of generated error and executed time. The time, which is required to fulfill maximum number of the iterations, shows algorithm speed. However, algorithm speed concept is strictly dependant on computer's speed and its specifications [33]. Thus, in Section 3.2.3 the relative CPU time of GBLM and 2P-GBLM has been computed to show the superiority of proposed method in terms of time. The computed times, for each learning process, are obtained using MATLAB 7.1 and CPU Intel 1.70 GHz. In order to validate proposed learning method, two well-known datasets are utilized that have been used by other researchers [20,1]. It needs a dataset of curved fuzzy values to make datasets applicable for 2P-GBLM. To this end, an estimate of $middle_\alpha$ has been generated in order to curving the triangular natured values of one of the referred datasets [1]. Finally, two used datasets are shown in Tables 6 and 7 in Appendix A.

3.2.1. Finding the best percentages for the first and second phases

At the beginning of simulation, rough assignment of different percentages to each phase of 2P-GBLM is analyzed and an approximation of best percentages assignment was obtained so that the least overall error is achieved. Two datasets, shown in Tables 6 and 7, are chosen from [20,1] to train the networks. To insure the reliability of training results, same initial population is used for each time of training process replication for all nine cases of the allotments. The depicted results were obtained from the average of MSEs from 1000 replications of 100 training generations.

In order to compare different sets of percentages assigned to the first and second phases, two networks were constructed, using two different datasets. The training sets were fed to the networks by assigning x% to the first phase and $(100-x)$% to the second phase of complete training generations, where x runs from 10 to 90 by increments of 10. The results of trained networks based on their related datasets are shown in Figs. 3 and 4. The overall bars of final error rates, depicted in Figs. 3 and 4, are based on the first and second phases corresponding to their datasets. The results show that we have the best accuracy in terms of overall error when 90% of complete generation is assigned to the first phase and 10% to second phase. From Figs. 3 and 4, it seems that the overall error decreases (increases) as the percentage of generations in the first phase increase (decrease) and the second phase one decrease (increase). Therefore, it seems that it is better to allot more generations to the first phase convergence.

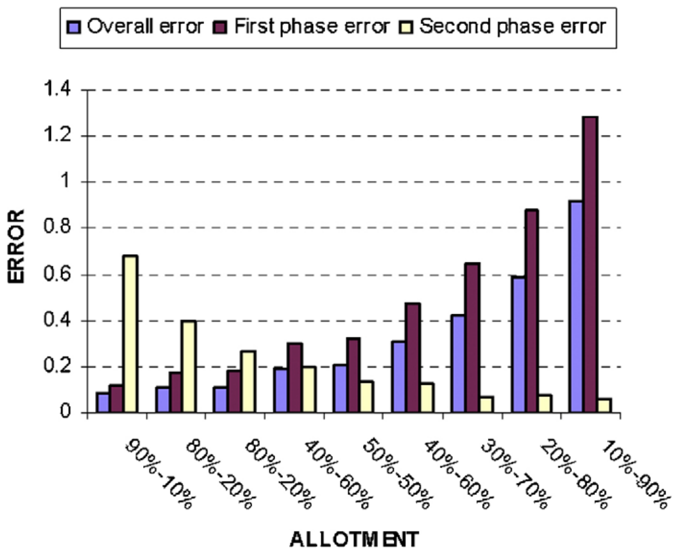

Fig. 3. Training of 2P-GBLM by different percentages (first phase%–second phase%) using dataset of Table 6.

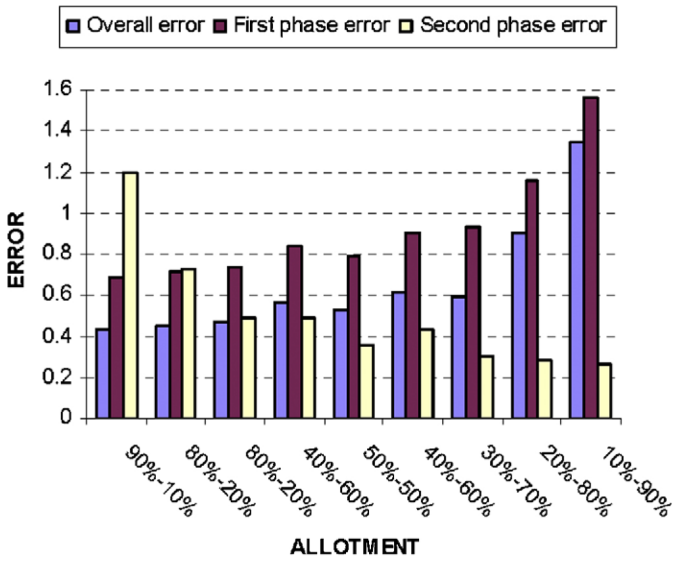

Fig. 4. Training of 2P-GBLM by different percentages (first phase%–second phase%) using dataset of Table 7.

3.2.2. *Superiority of proposed method*

In this part, the superiority of proposed method, using two datasets from [20,1], is shown. Tables 6 and 7 exhibit the datasets that are used to train the networks. The designs of membership

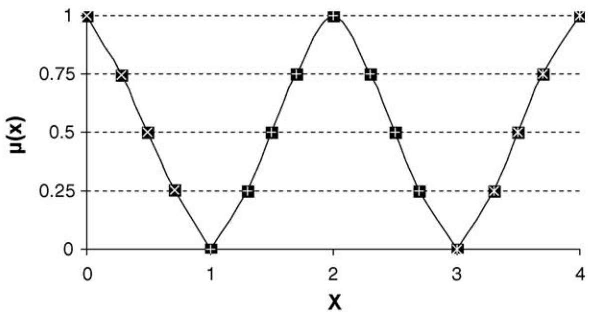

Fig. 5. Membership functions of Small, Medium and Large.

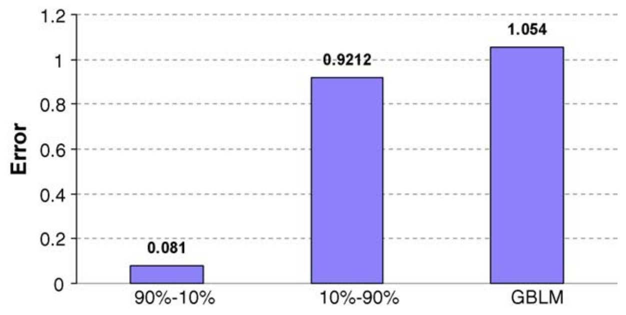

Fig. 6. Training results of GBLM and 2P-GBLM in case of its best and worst accuracies using dataset of Table 6.

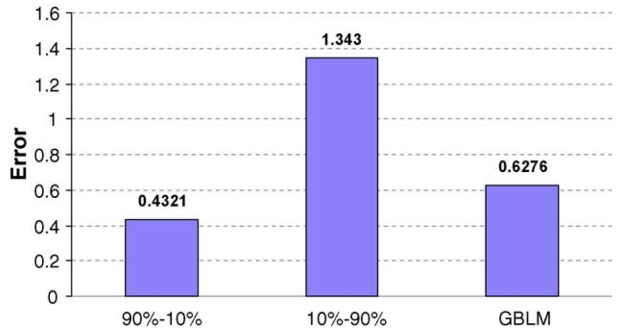

Fig. 7. Training results of GBLM and 2P-GBLM in cases of its best and worst accuracies using dataset of Table 7.

functions of Small, Medium and Large of Table 6 are shown in Fig. 5. The FANNs of both GBLM and proposed 2P-GBLM are trained using datasets of Tables 6 and 7. To insure the reliability of comparison results, all trainings are carried out using the same initial population for both methods. The results from Section 3.2.1 are used to assign the percentages of the first and second phases in 2P-GBLM. Figs. 6 and 7 show the errors for GBLM and 2P-GBLM, for both the best and the worst accuracy cases of the latter method. All training results for generated error in this part are given in terms of the average MSE in 1000 replications.

For further comparison, both datasets of Tables 6 and 7 were trained using fitness function F_A and distance measure given in (10). Similarly, a network of 2P-GBLM is trained by assigning percentages of the best accuracy case to the first and the second phases. The results are reported in Tables 1 and 2.

Error rates of learning process after 2000 generations for both GBLM and 2P-GBLM have been shown in Table 1. It shows a better accuracy for 2P-GBLM compared with GBLM. Here, fitness function F_P and measure of distance given in (11) have been utilized to train both datasets of Tables 6 and 7.

Comparison of training results of GBLM and 2P-GBLM by using both fitness functions of F_A and F_P and their corresponding measure of distances is contained in Table 2. It is noticeable that the proposed 2P-GBLM has better accuracy when compared with GBLM in both terms of generated error and executed time. While the fitness function F_A always has a greater error than F_P, this is

Table 1
Comparison of 2P-GBLM using fitness function (9).

Dataset	Average of actual errors			
	GBLM	2P-GBLM	Phase I	Phase II
Table 6	4.4587	0.7889	0.3911	1.0541
Table 7	1.9319	1.3046	0.3225	1.9594

Table 2
Comparison of the best result of four methods.

Method used	Average of actual errors											
	Dataset of Table 6						Dataset of Table 7					
	GBLM	Time	2P-GBLM	Time overall	Time phase I	Time phase II	GBLM	Time	2P-GBLM	Time	Time phase I	Time phase II
F_A	4.4587	03:05	0.7889	2:04	1:52	0:12	1.5319	06:10	1.3046	4:13	3:46	0:29
F_P	1.0036	02:35	0.07	01:53	1:41	0:12	0.6276	05:05	0.4321	03:44	3:19	0:25

Table 3
Comparison results of 2P-GBLM and GBLM trainings, using dataset of Table 6.

GA options		2P-GBLM					GBLM	Relative CPU time
Generation	Population	Overall two phase error	Phase I	Assigned percentage	Phase II	Assigned percentage	Error	GBLM/2P-GBLM
2000	100	0.00859	0.0160	90%	0.0036	10%	0.4541	1.3714
2000	100	0.2123	0.2563	10%	0.1829	90%		1.2433

Table 4
Comparison results of 2P-GBLM and GBLM training, using dataset of Table 7.

GA options		2P-GBLM					GBLM	Relative CPU time
Generation	Population	Overall two phase error	Phase I	Assigned percentage	Phase II	Assigned percentage	Error	GBLM/2P-GBLM
2000	400	0.3763	0.667	90%	0.182	10%	0.5820	1.3636
2000	400	0.6250	0.861	10%	0.467	90%		1.2503

more obvious in the case of dataset of Table 6. It is because F_P is more suited to symmetrical datasets, and the dataset of Table 6 is more symmetrical than that of Table 7.

While this, it is.

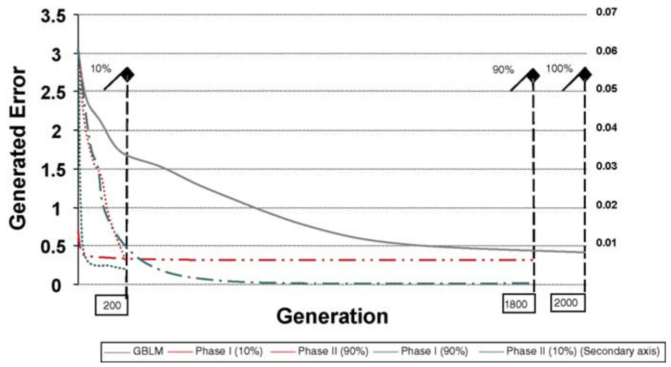

Fig. 8. Training convergence of 2P-GBLM using dataset in Table 6.

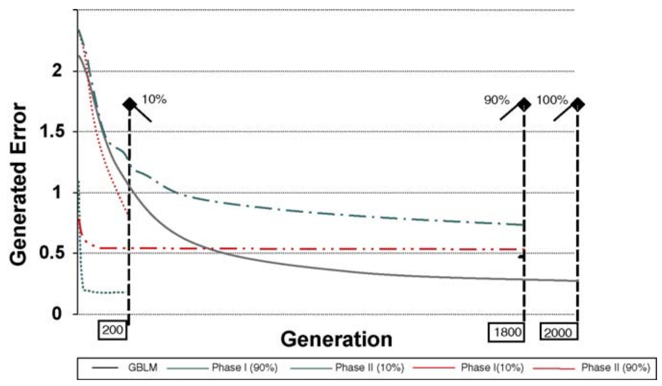

Fig. 9. Training convergence of 2P-GBLM using dataset in Table 7.

3.2.3. *Accuracy of proposed method*

In order to show the accuracy of proposed superior method resulted from Section 3.2.2, again the network is trained using the fitness function F_P with datasets of Tables 6 and 7. The training results are obtained by averaging the MSE in 100 replications and using the same initial population for each replicate. The related parameters are illustrated in Tables 3 and 4. Percentage assignments for the cases of best and the worst accuracies obtained from Section 3.2.1, are used for 2P-GBLM. The comparison with the results obtained by GBLM is shown in Tables 3 and 4. Table 3 shows that 2P-GBLM has been improved the accuracy in both terms of generated error and the time. In Table 4, the accuracy of 2P-GBLM has been improved in terms of the time for both cases, while it has lost the superiority in terms of generated error for the second case.

Figs. 8 and 9 correspond to Tables 3 and 4, respectively. They provide an analysis of learning convergence in terms of generated error in each generation. In these figures, it is noticeable that when the error of first phase is small, the error of second phase is also small. In other words, the rate of convergence in the first phase has a significant effect on the error generated in the second phase. Therefore, a smaller distance between the target and actual output in place-definition phase, leads to the reduction of error in

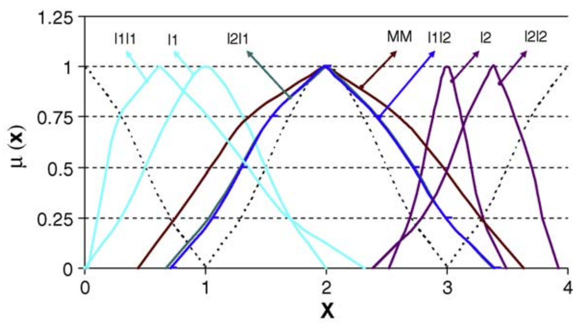

Fig. 10. The outcomes from trained network on new data based on dataset in Table 6.

Table 5
Predicted outcomes based on dataset of Table 7 using 2P-GBLM trained network.

	$\tilde{X}_0^L$	$\tilde{X}_{0.25}^L$	$\tilde{X}_{0.5}^L$	$\tilde{X}_{0.75}^L$	$\tilde{X}_1^L$	$\tilde{X}_0^R$	$\tilde{X}_{0.25}^R$	$\tilde{X}_{0.5}^R$	$\tilde{X}_{0.75}^R$	$\tilde{X}_1^R$
$\tilde{X}1$	2	2.3	2.5	2.7	3	4	3.7	3.5	3.3	3
$\tilde{X}2$	0	0.1	0.2	0.4	0.5	1	0.9	0.8	0.6	0.5
	$\tilde{O}_0^L$	$\tilde{O}_{0.25}^L$	$\tilde{O}_{0.5}^L$	$\tilde{O}_{0.75}^L$	$\tilde{O}_1^L$	$\tilde{O}_0^R$	$\tilde{O}_{0.25}^R$	$\tilde{O}_{0.5}^R$	$\tilde{O}_{0.75}^R$	$\tilde{O}_1^R$
$f_{net}(\tilde{X}1;\tilde{X}1)$	−0.59	4.9	5	5.1	5.57	6.092	6.0904	6.0901	6.03	5.6
$f_{net}(\tilde{X}2;\tilde{X}2)$	−0.506	−0.169	−0.014	0.5222	0.9297	4.6568	3.9528	3.1648	1.6291	0.9351
$f_{net}(\tilde{X}1;\tilde{X}2)$	−0.593	−0.132	0.1138	1.92	2.7	5.9	5.77	5.20	4.03	2.702
$f_{net}(\tilde{X}2;\tilde{X}1)$	0.43	5.9	6.04	6.07	6.082	6.092	6.0903	6.0901	6.089	6.085

shape-definition phase, thence achieving less overall two phases error.

3.2.4. Testing and prediction

This subsection follows Section 3.2.3 by testing the trained networks, $f_{net}(\tilde{X}i,\tilde{X}j)$, through two datasets of Tables 6 and 7. Error testing of trained networks was done for both GBLM and 2P-GBLM and each individual trained set of the datasets. Training tests were followed by prediction of the result of inputting (Medium, Medium) using 2P-GBLM. See Fig. 10.

Here, predicted value was *MM* that is the outcome of trained network based on the best case obtained from Table 3 of Section 3.2.3; we note that this outcome is which is closed to the value *Medium*. For other data, predicted outcomes for (I1,I1), (I1,I2), (I2,I1) and (I2,I2), are plotted as I1I1, I1I2, I2I1 and I2I2, respectively. It is worth noting that because of symmetry, I2I1 and I1I2, which are almost identical. Table 5 shows predicted outcomes by trained networks based on dataset of Table 7.

Note that, $\tilde{X}1$ and $\tilde{X}2$ are inputs and $\tilde{X}_\alpha^L\ \tilde{X}_\alpha^R$ are the left and right boundaries of fuzzy number where $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$. Similarly, $\tilde{O}_\alpha^L$ and $\tilde{O}_\alpha^R$ are predicted outcome through the trained network $f_{net}(\tilde{X}i,\tilde{X}j)$ for $i, j = 1, 2$.

4. Summary and discussion

This paper proposed an improvement on genetic-based learning methods, called a two-phase genetic-based method or 2P-GBLM, for fuzzy artificial neural networks (FANNs). The aim was to use trained FANNs to obtain an outcome closer to the target by computing more α-cuts. The method was explained by describing the place-definition and shape-definition as two major steps of finding optimal weights for FANNs in 2P-GBLM and it compared with GBLM in two terms of generated error and executed time. It also obtained the allotment of percentages for the best and worst accuracy cases for 2P-GBLM. The results overall showed better accuracy of learning for 2P-GBLM compared with the GBLM. The better results were due to splitting the variables of the problem as one chromosome into two ones, which each of them corresponds to one phase of the learning method. Therefore, overall error is dependant on the convergence of first phase error in which the place of fuzzy value is found within unconstraint search space of real numbers using just support and core as the least required α-cuts. Having two phases, cause each learning generation of a complete training process to be engaged with less α-cuts compared with GBLM and, thus, it takes less execution time. According to the result, the learning generation allotment of the first phase is more important than the second phase on decreasing the overall error. The results in Figs. 3 and 4, illustrated the effect of increase and decrease of different percentage of allotments for each first and second phase. Consequently, Figs. 6 and 7 show the best and worst cases of accuracies belong to most and least allotments to the first phase. In addition, Figs. 8 and 9 show the dependency of learning convergence on the first phase allotment. These are all due to the constraints derived from the first phase to be used in the second phase, which cause to achieve a better convergence. Finally, the training tests were used to demonstrate the results of this research and based on these it concludes that effect of using two phases is to decrease the overall error. One can attempt to assign percentages that are more precise to the two phases training generations of 2P-GBLM.

Acknowledgments

The authors are grateful to anonymous referees whose suggestions improved the quality of revised version of this paper. This work is supported by the Ministry of Science, Technology & Innovation (MOSTI), Malaysia and Research Management Center, Universiti Teknologi Malaysia (UTM), under the Vot 79200. In addition, the first author is partially supported by Fuzzy Systems and its Applications Center of Excellence at Shahid Bahonar University of Kerman, Iran.

Appendix A

See Tables 6 and 7.

Table 6
Liu's dataset.

*X*1										
No.	$\tilde{X}_\alpha$									
	$\tilde{X}_0^L$	$\tilde{X}_{0.25}^L$	$\tilde{X}_{0.5}^L$	$\tilde{X}_{0.75}^L$	$\tilde{X}_1^L$	$\tilde{X}_0^R$	$\tilde{X}_{0.25}^R$	$\tilde{X}_{0.5}^R$	$\tilde{X}_{0.75}^R$	$\tilde{X}_0^L$
1	0	0	0	0	0	1	0.7	0.5	0.3	0
2	0	0	0	0	0	1	0.7	0.5	0.3	0
3	0	3.3	3.5	3.7	0	4	4	4	4	0
4	0	3.3	3.5	3.7	0	4	4	4	4	0

Table 6 (*Continued*)

*X*2

No.	$\tilde{X}_\alpha$									
	$\tilde{X}_0^L$	$\tilde{X}_{0.25}^L$	$\tilde{X}_{0.5}^L$	$\tilde{X}_{0.75}^L$	$\tilde{X}_1^L$	$\tilde{X}_0^R$	$\tilde{X}_{0.25}^R$	$\tilde{X}_{0.5}^R$	$\tilde{X}_{0.75}^R$	$\tilde{X}_0^L$
1	0	0	0	0	0	1	0.7	0.5	0.3	0
2	0	3.3	3.5	3.7	0	4	4	4	4	0
3	0	0	0	0	0	1	0.7	0.5	0.3	0
4	0	3.3	3.5	3.7	0	4	4	4	4	0

f(*X*1;*X*2)

No.	$\tilde{D}_\alpha$									
	$\tilde{D}_0^L$	$\tilde{D}_{0.25}^L$	$\tilde{D}_{0.5}^L$	$\tilde{D}_{0.75}^L$	$\tilde{D}_1^L$	$\tilde{D}_0^R$	$\tilde{D}_{0.25}^R$	$\tilde{D}_{0.5}^R$	$\tilde{D}_{0.75}^R$	$\tilde{D}_1^R$
1	0	0	0	0	0	1	0.7	0.5	0.3	0
2	1	1.3	1.5	1.7	2	3	2.7	2.5	2.3	2
3	1	1.3	1.5	1.7	2	3	2.7	2.5	2.3	2
4	0	3.3	3.5	3.7	0	4	4	4	4	0

Table 7
Aliev's dataset.

*X*1

No.	$\tilde{X}_\alpha$									
	$\tilde{X}_0^L$	$\tilde{X}_{0.25}^L$	$\tilde{X}_{0.5}^L$	$\tilde{X}_{0.75}^L$	$\tilde{X}_1^L$	$\tilde{X}_0^R$	$\tilde{X}_{0.25}^R$	$\tilde{X}_{0.5}^R$	$\tilde{X}_{0.75}^R$	$\tilde{X}_0^L$
1	0	0.025	0.050	0.0745	0.1	0.2	0.1745	0.150	0.1255	0.1
2	0.3	0.345	0.40	0.455	0.5	0.6	0.57	0.550	0.53	0.5
3	0.2	0.28	0.35	0.42	0.5	0.9	0.73	0.65	0.57	0.5
4	0.9	0.917	0.924	0.932	0.95	1	0.986	0.974	0.967	0.95
5	0.2	0.35	0.4	0.45	0.6	0.9	0.85	0.8	0.75	0.6
6	0.3	0.355	0.40	0.455	0.5	0.6	0.575	0.550	0.525	0.5
7	0.2	0.225	0.30	0.345	0.6	0.9	0.770	0.65	0.630	0.6
8	0.2	0.230	0.250	0.270	0.5	0.9	0.745	0.60	0.555	0.5
9	0.7	0.730	0.750	0.780	0.8	0.9	0.870	0.850	0.825	0.8

*X*2

No.	$\tilde{X}_\alpha$									
	$\tilde{X}_0^L$	$\tilde{X}_{0.25}^L$	$\tilde{X}_{0.5}^L$	$\tilde{X}_{0.75}^L$	$\tilde{X}_1^L$	$\tilde{X}_0^R$	$\tilde{X}_{0.25}^R$	$\tilde{X}_{0.5}^R$	$\tilde{X}_{0.75}^R$	$\tilde{X}_0^L$
1	0	0.025	0.050	0.0745	0.1	0.2	0.1745	0.150	0.1255	0.1
2	0.4	0.455	0.5	0.595	0.6	0.8	0.745	0.70	0.655	0.6
3	0.8	0.830	0.850	0.870	0.9	1.2	1.125	1.05	0.980	0.9
4	0.6	0.630	0.650	0.670	0.7	0.8	0.770	0.750	0.730	0.7
5	0.4	0.430	0.450	0.470	0.5	0.6	0.570	0.550	0.530	0.5
6	0	0.030	0.05	0.070	0.1	0.2	0.170	0.150	0.130	0.1
7	0.6	0.630	0.650	0.0670	0.7	0.8	0.770	0.750	0.725	0.7
8	0.4	0.430	0.450	0.470	0.5	0.6	0.570	0.550	0.530	0.5
9	0	0.030	0.05	0.070	0.1	0.2	0.170	0.150	0.130	0.1

f(*X*1;*X*2)

No.	$\tilde{D}_\alpha$									
	$\tilde{D}_0^L$	$\tilde{D}_{0.25}^L$	$\tilde{D}_{0.5}^L$	$\tilde{D}_{0.75}^L$	$\tilde{D}_1^L$	$\tilde{D}_0^R$	$\tilde{D}_{0.25}^R$	$\tilde{D}_{0.5}^R$	$\tilde{D}_{0.75}^R$	$\tilde{D}_1^R$
1	−0.083	−0.0272	0.0185	0.06425	0.166	0.249	0.2117	0.2045	0.18725	0.166
2	−0.558	−0.046	0.456	0.858	0.912	2.248	2.037	1.535	1.033	0.912
3	−1.665	−1.025	−0.45	0.132	0.729	4.068	3.205	2.38	1.355	0.729
4	2.08	2.41	2.74	3.368	3.405	5.155	4.71	4.27	4.02	3.405
5	−0.945	−0.370	0.201	0.571	1.360	3.428	2.806	2.394	1.872	1.360
6	−0.022	0.203	0.418	0.633	0.861	1.403	1.261	1.131	1.001	0.861
7	−1.247	−0.600	0.035	0.675	1.321	3.825	3.149	2.573	1.952	1.321
8	−0.649	0.248	0.343	0.535	0.937	3.131	2.576	2.033	1.490	0.937
9	1.182	1.428	1.664	1.895	2.149	3.073	2.837	2.611	2.385	2.149

References

[1] R.A. Aliev, B. Fazlollahi, R.M. Vahidov, Genetic algorithm-based learning of fuzzy neural network. Part 1. Feed-forward fuzzy neural networks, Fuzzy Sets and Systems 118 (2001) 351–358.

[2] S. Abbasbandy, J.J. Neirto, M. Amirfagharian, Best approximation of fuzzy functions, Nonlinear Studies 14 (2007) 87–102.

[3] J.J. Buckley, Y. Hayashi, Fuzzy neural nets and applications, Fuzzy Sets and AI 1 (1992) 11–41.

[4] J.J. Buckley, Y. Hayashi, Neural nets for fuzzy systems, Fuzzy Sets and Systems 71 (1995) 265–276.

[5] J.J. Buckley, Y. Hayashi, K. Reilly, R.M. Krishna, Genetic learning algorithms for fuzzy neural nets, in: Proceedings of IEEE International Conference, vol. 3, 1994, pp. 1969–1974.

[6] J.J. Buckley, Y. Hayashi, Direct fuzzification of neural networks, in: Proceedings of First Asian Fuzzy Systems Symposium, vol. 1, 1993, pp. 560–567.
[7] J.J. Buckley, Y. Hayashi, Fuzzy neural networks, in: R. Yager, L. Zadeh (Eds.), Fuzzy Sets, Neural Networks and Soft Computing, Van Nostrand Reinhold, New York, 1994, pp. 233–249.
[8] S. Fukuda, K. Hiramatsu, M. Mori, Fuzzy neural network model for habitat prediction and HEP for habitat quality estimation focusing on Japanese medaka (*Oryais latipes*) in agricultural canals, Paddy and Water Environ 4 (2006) 119–124.
[9] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[10] M. Gupta, H. Ding, in: F. Aminzadeh, M. Jamshidi (Eds.), Foundations of Fuzzy Neural Computations, Soft Computing: Fuzzy Logic, Neural Networks, and Distributed Artificial Intelligence, Prentice-Hall, Englewood Cliffs, NJ, 1994, pp. 165–195.
[11] Y. Hayashi, J.J. Buckley, E. Czogola, Fuzzy neural networks with fuzzy signals and weights, International Journal of Intelligent Systems 8 (1993) 527–537.
[12] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975.
[13] H.T. Nguyen, E.A. Walker, A First Course in Fuzzy Logic, third edition, CRC Press, London, 2006.
[14] H. Ishibuchi, K. Kwon, H. Tanakan, A learning algorithm of fuzzy neural networks with triangular fuzzy weights, Fuzzy Sets and Systems 71 (1995) 277–293.
[15] H. Ishibuchi, M. Nii, Numerical analysis of learning of fuzzified neural networks from fuzzy if–then rules, Fuzzy Sets and Systems 120 (2001) 281–307.
[16] R.J. Kuo, P. Wu, C.P. Wang, An intelligent sales forecasting system through integration of artificial intelligence and fuzzy neural networks with fuzzy weight elimination, Neural Networks 15 (2002) 909–925.
[17] G.J. Klir, B. Yuan, Fuzzy sets and fuzzy logic, in: Theory and Application, Prentice-Hall Inc., New Jersey, 1995.
[18] P. Liu, Representation of digital image by fuzzy neural network, Fuzzy Sets and Systems 130 (2002) 109–123.
[19] P. Liu, H.X. Li, Fuzzy neural network theory and application, World Scientific, River Edge, NJ, 2004.
[20] P. Liu, H.X. Li, Approximation analysis of feed forward regular fuzzy neural network with two hidden layers, Fuzzy Sets and Systems 150 (2005) 373–396.
[21] M. Hadi Mashinchi, M. Reza Mashinchi, S.M.H.J. Shamsuddin, W. Pedrycz, Genetically tuned fuzzy back-propagation learning method based on derivation of min–max function for fuzzy neural networks, in: Proceedings of The International Conference on Genetic and Evolutionary Methods, Worldcomp Congress, USA, (2007), pp. 213–219.
[22] M. Hadi Mashinchi, S. M. HJ. Shamsuddin. Three-term fuzzy back-propagation. In A.-E. Hassanien et al. (Eds.), Foundation of Comput. Intel., Springer-Verlag, Berlin, Germany, 201:1 (2009) 143–158.
[23] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer, Berlin, 1996.
[24] H. Narazaki, A.L. Ralescu, A method to implement qualitative knowledge in multi-layered neural networks, fuzzy logic and fuzzy control, Lecture Notes in Computer Science 833 (1991) 82–95.
[25] S. Abbasbandy, M. Otadi, Numerical solution of fuzzy polynomials by fuzzy neural networks, Applied Mathematics and Computations 181 (2006) 1084–1089.
[26] P. Liu, H. Li, Efficient learning algorithms for three-layer regular feedforward fuzzy neural networks, IEEE Transactions on Neural Networks 15 (2004) 545–558.
[27] L.H. Tsoukalas, R.E. Uhrig, Fuzzy and Neural Approaches in Engineering, John Wiley & Sons Inc., New York, 1997.
[28] J.A. Vasconcelos, J.A. Ramirez, R.H.C. Takahashi, R.R. Saldanha, Improvements in genetic algorithms 37, IEEE Transactions on Magnetics (2001) 3414–3417.
[29] T. Weise, Global Optimization Algorithms—Theory and Application, Thomas Weise, 2007. Online available as e-book at http://www.it-weise.de/.
[30] L.A. Zadeh, Fuzzy sets, Information and Control 8 (1965) 338–353.
[31] X. Zhang, C.C. Hang, P.Z. Wang, The min–max function differentiation and training of fuzzy neural networks, IEEE Transactions on Neural Networks 7 (1996) 1139–1150.
[32] X. Zhang, C. Tan, C.C. Hang, P.Z. Wang, An efficient computational algorithm for min-max operations, Fuzzy Sets and Systems 104 (1999) 297–304.
[33] R.T. Zheng, N.Q. Ngo, P. Shum, S.C. Tjin, L.N. Binh, A staged continuous tabu search algorithm for the global optimization and its applications to the design of fiber Bragg gratings, Computational Optimization and Applications 30 (2005) 319–335.