International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 3, No.3 (Sep-2014)

Evolving Fuzzy Neural Networks by Particle Swarm Optimization with Fuzzy Genotype Values

Hidehiko Okada

Department of Intelligent Systems, Faculty of Computer Science and Engineering, Kyoto Sangyo University, Kyoto, Japan

Received 25 Apr. 2014, Revised 14 May 2014, Accepted 1 Jun. 2014, Published 1 Sep. 2014

Abstract: Particle swarm optimization (PSO) is a well-known instance of swarm intelligence algorithms and there have been many researches on PSO. In this paper, the author proposes an extension of PSO for solving fuzzy-valued optimization problems. In the proposed extension, genotype values (i.e. values in particle position vectors) are not real numbers but fuzzy numbers. Search processes in PSO are extended so that PSO can handle genotype instances with fuzzy numbers. The proposed method is experimentally applied to evolution of neural networks with fuzzy weights and biases. Experimental results showed that fuzzy neural networks evolved by the proposed method could model hidden target fuzzy functions despite the fact that no training data was explicitly provided.

Keywords: Evolutionary algorithm, Swarm intelligence, Particle swarm optimization, Fuzzy number, Feedforward neural network, Neuroevolution

1. Introduction

A multi-layered feed forward neural network (NN) with fuzzy-valued weights and biases was proposed in literature [1]. The fuzzy NN (FNN) approximately models a fuzzy function $Y = F(\boldsymbol{x})$, where Y is a fuzzy number and $\boldsymbol{x}$ is a real vector, by learning given data $(\boldsymbol{x}_1, Y_1)$, $(\boldsymbol{x}_2, Y_2)$, …. The FNN can learn the data in which Y_1, Y_2, … include both of real numbers and fuzzy numbers, because a real number can be specified as a fuzzy number with zero width (i.e., with the same value of upper and lower limits). As the learning method for the FNNs, a supervised learning method was also proposed [1] which is an extension of the traditional back propagation (BP), but a method that does not require training data has yet not been proposed.

Besides, evolutionary algorithms have recently been applied to the reinforcement training of NNs, known as neuroevolution (NE) [2-5]. In NE, weights and biases are tuned by evolutionary operations, not by the BP algorithm. Because NE does not utilize BP, NE does not require errors between NN output values and their target signals but only require each NN to be ranked based on the performance of the NN for a given task. Thus, NE is applicable to problems in which the error function is difficult or impossible to be determined, such as controlling autonomous robots. EAs have been applied to NE of traditional NNs with real-valued weights and biases, where the genotypes (chromosomes) consist of real numbers or bit strings that encode real numbers. The ordinary EAs have not employed fuzzy numbers as their genotype values because their evolutionary operations are designed to handle genotypes with crisp values and thus the operations cannot handle genotypes with fuzzy values.

The author previously proposed an extension of genetic algorithm which can handle fuzzy-valued genotypes [6]. In this paper, the author proposes a similar extension of another EA, particle swarm optimization (PSO). PSO [7,8] is a well-known instance of the EAs (more specifically, an instance of swarm intelligence algorithms [9]). Researchers have applied PSO to the training of NNs [10-19], but the NNs are traditional ones with real-valued weights and biases. On the contrary, the extended PSO proposed in this paper can be applied directly to fuzzy optimization problems by employing fuzzy variables in a fuzzy optimization problem as genotype values. The author experimentally applies the proposed method (fuzzy-valued PSO: FPSO) to reinforcement training of FNNs and compares the experimental result with the result by the previously proposed fuzzy-valued GA [6].

E-mail: hidehiko@cc.kyoto-su.ac.jp

2. Neural Networks with Fuzzy Weights and Biases

The FNN employed in this research is the same as in the literature [1], which is a three-layered feed forward NN with fuzzy weights and biases. Fig.1 shows its structure. An FNN receives an input real vector $\boldsymbol{x}$ and calculates its output fuzzy value O (for simplicity, the output layer includes a single unit) as follows [1]:

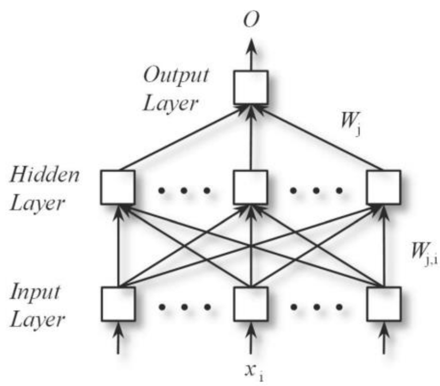

Figure 1. Neural network with fuzzy weights and biases [1].

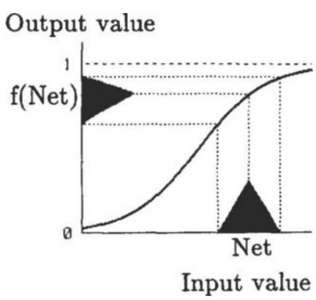

Figure 2. Input-output relation of each unit in the hidden and output layers [1].

Input Layer:

$$o_i = x_i. \tag{1}$$

Hidden Layer:

$$Net_j = \sum_i W_{j,i} o_i + \Theta_j, \tag{2}$$

$$O_j = f(Net_j). \tag{3}$$

Output Layer:

$$Net = \sum_j W_j O_j + \Theta, \tag{4}$$

$$O = f(Net). \tag{5}$$

In (1)-(5), x_i and o_i are real values, while Net_j, Net, $W_{j,i}$, W_j, Θ_j, Θ, O_j and O are fuzzy values. $f(x)$ is the unit activation function which is typically the sigmoidal one: $f(x) = 1/(1 + e^{-x})$. $f(x)$ maps a fuzzy input number to a fuzzy output number as illustrated in Fig.2.

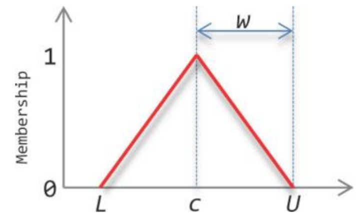

Figure 3. Symmetric triangular fuzzy number and its real-valued parameters [6].

The feed-forward calculation of the FNN is based on the extension principle [20] and the interval arithmetic [21] (for more detail, see the literature [1]). Let us denote two closed intervals as A and B, where $A = [a^L, a^U]$ and $B = [b^L, b^U]$. In this case,

$$\begin{aligned} A + B &= [a^L, a^U] + [b^L, b^U] \\ &= [a^L + b^L, a^U + b^U]. \end{aligned} \tag{6}$$

$$\begin{aligned} k \cdot A &= k \cdot [a^L, a^U] \\ &= [ka^L, ka^U], \text{if } k \geq 0, \text{else } [ka^U, ka^L]. \end{aligned} \tag{7}$$

$$\begin{aligned} A \cdot B &= [a^L, a^U] \cdot [b^L, b^U] \\ &= [min(a^L b^L, a^L b^U, a^U b^L, a^U b^U), \\ &\quad max(a^L b^L, a^L b^U, a^U b^L, a^U b^U)]. \end{aligned} \tag{8}$$

The FNN includes $mn + m$ weights (i.e., mn weights between n input units and m hidden units, and m weights between m hidden units and an output unit) and $m + 1$ biases (= the total number of units in the hidden and output layers). Thus, the FNN includes $mn + 2m + 1$ fuzzy variables in total. The FPSO handles these fuzzy variables as a genotype $\boldsymbol{X} = (X_1, X_2, \ldots, X_D)$ where X_i is a fuzzy number and $D = mn + 2m + 1$.

Suppose each X_i is a symmetric triangular fuzzy number (Fig.3) as in [6]. In this case, X_i can be specified by its upper and lower limits or by its center and width (radius): $X_i = [x_i^L, x_i^U]$ or $X_i = (x_i^c, x_i^w)$ where x_i^L, x_i^U, x_i^c, x_i^w denote the upper, lower, center and width of X_i respectively.

3. Fuzzy PSO: PSO with Fuzzy-Valued Genotypes

The proposed FPSO consists of the same processes as those in the ordinary PSO with real-valued genotypes. Processes of initialization of population, fitness evaluation and updates of particles are extended so that these processes can handle fuzzy-valued genotypes.

A. Initialization of Population

In the initialization process, $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_P$ are randomly initialized where $\boldsymbol{P}$ is the population size. Because the elements in $\boldsymbol{X}_a$ (i.e., $X_{a,1}, X_{a,2}, \ldots, X_{a,D}$) are weights and biases in an FNN in this research, smaller absolute values of $X_{a,i}$ are preferable as initial values. Thus, the initial values for $X_{a,i}$ are randomly sampled from the normal distribution $N(0,\varepsilon)$ or uniformly from an interval $[-\varepsilon,\varepsilon]$ where ε is a small positive number. In the case of employing the [lower, upper] model (the LU model), two values are sampled per $X_{a,i}=[x_{a,i}^L,x_{a,i}^U]$: the smaller (larger) one is set to $x_{a,i}^L$ ($x_{a,i}^U$). In the case of employing the (center, width) model (the CW model), two values are sampled per $X_{a,i}=(x_{a,i}^c,x_{a,i}^w)$: one of the two values is set to $x_{a,i}^c$ and the absolute value of the other is set to $x_{a,i}^w$.

B. Fitness Evaluation

To evaluate fitness of an FNN as a phenotype instance of the corresponding genotype instance $\boldsymbol{X}_a=[X_{a,1},X_{a,2},\dots,X_{a,D}]$ where $\boldsymbol{X}_a\in\{\boldsymbol{X}_1,\boldsymbol{X}_2,\dots,\boldsymbol{X}_P\}$, the FNN is supplied with several samples of input real vectors and calculates output values. The input values are sampled within the variable domain of application problem. Fitness of the genotype instance $\boldsymbol{X}_a$ is evaluated based on the output values. The method for scoring the fitness based on the output values depends on the problem to which the FNN is applied. For example, in a case where the FNN is applied to controlling an automated system, some performance measure of the system can be used as the fitness score of the genotype instance corresponding to the FNN.

C. Updates of Particles

Let the position vector of a particle, its personal best and the global (or its local) best be denoted as $\boldsymbol{X}_r$, $\boldsymbol{Pbest}_r$, $\boldsymbol{Gbest}$ (or $\boldsymbol{Lbest}_r$). In the case of using the LU model, $\boldsymbol{X}_r=[X_{r,1},X_{r,2},\dots,X_{r,D}]$ and $X_{r,i}=[x_{r,i}^L,x_{r,i}^U]$. Let the velocity for $x_{r,i}^L$ and $x_{r,i}^U$ be denoted as $v1_{r,i}$ and $v2_{r,i}$ respectively. Note that $v1_{r,i}$ ($v2_{r,i}$) is not the lower (upper) limit of an interval so that $v2_{r,i}$ can be smaller than $v1_{r,i}$. $v1_{r,i}$ and $v2_{r,i}$ are updated as:

$$v1_{r,i}=wv1_{r,i}+c_1r_1(pbest_{r,i}^L-x_{r,i}^L)+c_2r_2(gbest_i^L-x_{r,i}^L), \tag{9}$$

$$v2_{r,i}=wv2_{r,i}+c_1r_1(pbest_{r,i}^U-x_{r,i}^U)+c_2r_2(gbest_i^U-x_{r,i}^U), \tag{10}$$

employing the global best model, or as:

$$v1_{r,i}=wv1_{r,i}+c_1r_1(pbest_{r,i}^L-x_{r,i}^L)+c_2r_2(lbest_{r,i}^L-x_{r,i}^L), \tag{11}$$

$$v2_{r,i}=wv2_{r,i}+c_1r_1(pbest_{r,i}^U-x_{r,i}^U)+c_2r_2(lbest_{r,i}^U-x_{r,i}^U), \tag{12}$$

employing the local best model. The constant values w, c_1, c_2 and the random values r_1, r_2 are the same as those in the ordinary PSO with the real-valued genotypes.

Similarly, in the case of using the CW model, $X_{r,i}=(x_{r,i}^c,x_{r,i}^w)$ and $v1_{r,i}$ and $v2_{r,i}$ are the velocity for $x_{r,i}^c$ and $x_{r,i}^w$ respectively. $v1_{r,i}$ and $v2_{r,i}$ are updated as:

$$v1_{r,i}=wv1_{r,i}+c_1r_1(pbest_{r,i}^c-x_{r,i}^c)+c_2r_2(gbest_i^c-x_{r,i}^c), \tag{13}$$

$$v2_{r,i}=wv2_{r,i}+c_1r_1(pbest_{r,i}^w-x_{r,i}^w)+c_2r_2(gbest_i^w-x_{r,i}^w), \tag{14}$$

employing the global best model, or as:

$$v1_{r,i}=wv1_{r,i}+c_1r_1(pbest_{r,i}^c-x_{r,i}^c)+c_2r_2(lbest_{r,i}^c-x_{r,i}^c), \tag{15}$$

$$v2_{r,i}=wv2_{r,i}+c_1r_1(pbest_{r,i}^w-x_{r,i}^w)+c_2r_2(lbest_{r,i}^w-x_{r,i}^w), \tag{16}$$

employing the local best model.

By using the updated $v1_{r,i}$ and $v2_{r,i}$, $X_{r,i}$ is updated as:

$$x_{r,i}^L=x_{r,i}^L+v1_{r,i}, \tag{17}$$

$$x_{r,i}^U=x_{r,i}^U+v2_{r,i}, \tag{18}$$

or as:

$$x_{r,i}^c=x_{r,i}^c+v1_{r,i}, \tag{19}$$

$$x_{r,i}^w=x_{r,i}^w+v2_{r,i}. \tag{20}$$

Note that $x_{r,i}^L$ must not be larger than $x_{r,i}^U$ because $x_{r,i}^L$ and $x_{r,i}^U$ are the lower and upper limits of the fuzzy number $X_{r,i}$. Similarly, $x_{r,i}^w$ must not be negative because $x_{r,i}^w$ is the width of $X_{r,i}$. If the value of $x_{r,i}^L$ becomes larger than the value of $x_{r,i}^U$ after the updates by (17) and (18), these values must be repaired to meet the constraint. The repair method can be as follows:

- the value of $x_{r,i}^U$ is assigned to $x_{r,i}^L$,
- the value of $x_{r,i}^L$ is assigned to $x_{r,i}^U$,
- the mean value of $x_{r,i}^L$ and $x_{r,i}^U$ is calculated and assigned to both of $x_{r,i}^L$ and $x_{r,i}^U$, or
- the two values for $x_{r,i}^L$ and $x_{r,i}^U$ are switched.

Similarly, if the value of $x_{r,i}^w$ becomes negative after the updates by (20), the value must be repaired to meet the constraint. The repair method can be as follows:

- the value of $x_{r,i}^w$ is assigned to 0, or
- the absolute value of $x_{r,i}^w$ is assigned to $x_{r,i}^w$.

4. Application to Evolving Fuzzy Neural Networks

The author experimentally evaluates the ability of the proposed FPSO by applying it to evolution of FNNs, in the same manner as in [6]. The FNNs are challenged to model hidden fuzzy functions. The author adopts the same two functions [6] as the targets for FNNs to model so that the author can compare the experimental result with that by the fuzzy GA [6]. For simplicity, the input x of the target functions is not a real vector but a real scalar (so that the FNN includes only a single input unit) and $0 \le x \le 1$, as in the literature [1]. The outputs of the target functions are symmetric triangular fuzzy numbers. The functions $F_1(x) = [F_1(x)^L, F_1(x)^U]$ and $F_2(x) = [F_2(x)^L, F_2(x)^U]$ are as follows:

$$F_1(x)^L = 0.2sin(2\pi x) - 0.1x^2 + 0.4, \quad (21)$$

$$F_1(x)^U = 0.2sin(2\pi x) + 0.1x^2 + 0.6. \quad (22)$$

$$F_2(x)^L = 0.2sin(2\pi x) + 0.2x^2 + 0.2, \quad (23)$$

$$F_2(x)^U = 0.2sin(2\pi x) - 0.2x^2 + 0.7. \quad (24)$$

Figs.4 and 5 show these two functions, where:

- F0.0L and F0.0U denote $F(x)^L$ and $F(x)^U$, i.e., the lower and upper limits of the support interval of $F(x)$,
- F0.5L and F0.5U denote the lower and upper limits of the 0.5-level interval of $F(x)$, i.e., $F(x)|_{0.5}$, and
- F1.0 denotes the peak of $F(x)$, i.e., $F(x)|_{1.0}$.

The FNN is designed as follows [6]:

- Number of units: 1 input, 10 hidden, 1 output.
- $-10.0 \le x_{r,i}^L, x_{r,i}^U, x_{r,i}^c \le 10.0$.

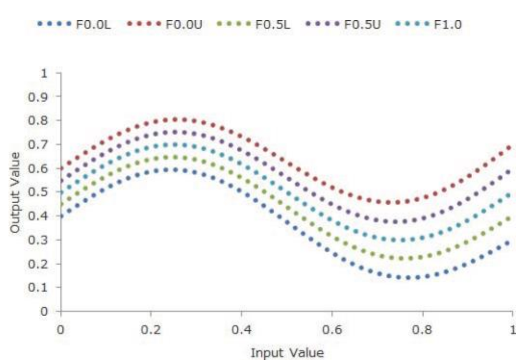

Figure 4. Target Function $F_1(x)$ [6].

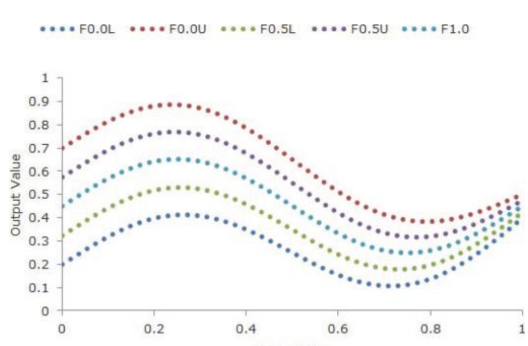

Figure 5. Target Function $F_2(x)$ [6].

- $0.0 \le x_{r,i}^w \le 10.0$.

The FPSO is designed as follows:

- Total number of FNNs evolved in a single run: 1,000,000.
- Population size and number of iterations: (100, 10,000), (500, 2,000)
- $w = 0.9$, $c_1 = 1.4$, $c_2 = 1.4$.
- Initial values of $x_{r,i}^L, x_{r,i}^U, x_{r,i}^c$ for the fuzzy weights and biases: uniformly random within $[-1.0,1.0]$.
- Initial values for $x_{r,i}^w$: uniformly random within $[0.0,1.0]$.
- Initial values of $v1_{r,i}$, $v2_{r,i}$: 0.0.
- $-0.1 \le v1_{r,i}, v2_{r,i} \le 0.1$

The number of iterations is 10,000 (or 2,000) for the FPSO with 100 (or 500) particles so that the total number of FNNs evaluated in a single run is consistently 1,000,000 (= 100×10,000 = 500×2,000).

Particles $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_P$ are ranked by utilizing the same error function as that in literature [1,6]. As the values for the h-level intervals of fuzzy numbers, the author employs h = $0.2, 0.4, \ldots, 1.0$ in this experiment. A phenotype instance FNN which corresponds to a genotype instance $\boldsymbol{X}_i$ is supplied with a real input value x_r and calculates its output fuzzy number O_r. x_r is sampled within the input domain [0,1] as $x_r = 0.0, 0.01, 0.02, \ldots, 1.0$. Besides, each value of x_r is supplied to the target function $F(x)$ and the output fuzzy number $F(x_r)$ is obtained. Then, the cost e_r for the input x_r is calculated as:

$$e_r = \sum_h h((o_{r,h}^L - f_{r,h}^L)^2 + (o_{r,h}^U - f_{r,h}^U)^2), \quad (25)$$

where,

- $o_{r,h}^L$ and $o_{r,h}^U$ are the lower and upper limits of the h-level interval of O_r, i.e., $O_r|_h = [o_{r,h}^L, o_{r,h}^U]$, and
- $f_{r,h}^L$ and $f_{r,h}^U$ are the lower and upper limits of the h-level interval of $F(x_r)$, i.e., $F(x_r)|_h = [f_{r,h}^L, f_{r,h}^U]$.

For each genotype instance $\boldsymbol{X}_i$, e_r is calculated 101 times ($e_0, e_1, \ldots, e_{100}$) for the 101 input values $x_r = 0.0, 0.01, 0.02, \ldots, 1.0$, and the sum of e_r is used for ranking $\boldsymbol{X}_i$. An instance with a smaller sum of e_r is ranked better. Note that e_r scores are not utilized for calculating the values of updating the weights and biases but only for determining $\boldsymbol{Pbest}_r$ and $\boldsymbol{Gbest}$ (or $\boldsymbol{Pbest}_r$ and $\boldsymbol{Lbest}_r$).

Figs.6 and 7 show the results of this experiment. Fig.6 shows the output fuzzy function of the best FNN among the total 20,000,000 FNNs (= [1,000,000 FNNs in each run] × [five runs] × [two variations for population

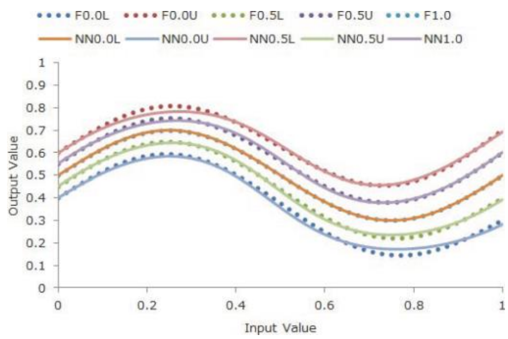

Figure 6. Output fuzzy function of the best FNN evolved by FPSO for modeling $F_1(x)$.

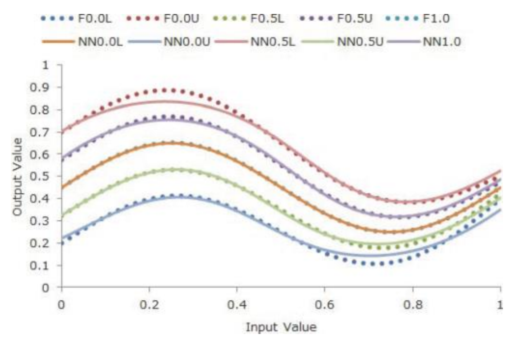

Figure 7. Output fuzzy function of the best FNN evolved by FPSO for modeling $F_2(x)$.

sizes]×[two variations for the interval model]) evolved by the FES for modeling $F_1(x)$. Fig.7 shows those for modeling $F_2(x)$ in the same manner as Fig.6. In Figs.6 and 7,

- F0.0L, F0.0U, F0.5L, F0.5U and F1.0 are the same as those in Figs.4 and 5,
- NN0.0L and NN0.0U denote the lower and upper limits of the support interval of the FNN output fuzzy number,
- NN0.5L and NN0.5U denote the lower and upper limits of the 0.5-level interval of the FNN output fuzzy number, and
- NN1.0 denotes the peak of the FNN output fuzzy number.

Fig.8 shows the membership functions of O and $F_1(x)$ for the input values $x = 0.3$ and $x = 0.7$, where O is the output fuzzy number of the best FNN. In this figure,

- NN(0.3) and NN(0.7) show the membership functions of the output fuzzy number of the best FNN for the input values 0.3 and 0.7, while
- F(0.3) and F(0.7) show the membership functions of $F_1(x)$ for the input values 0.3 and 0.7.

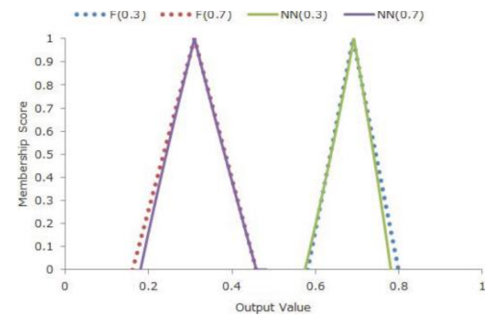

Figure 8. Output fuzzy numbers of the best FNN evolved by FPSO and target fuzzy numbers $F_1(x)$ for the inputs values of 0.3 and 0.7.

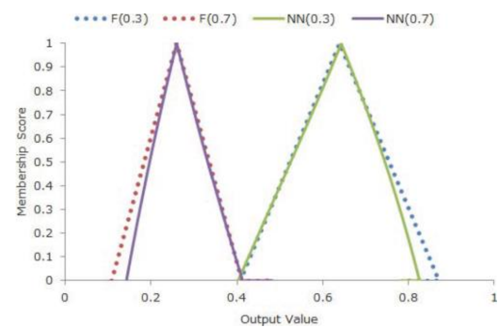

Figure 9. Output fuzzy numbers of the best FNN evolved by FPSO and target fuzzy numbers $F_2(x)$ for the inputs values of 0.3 and 0.7.

Fig.9 shows those for $F_2(x)$ in the same manner as Fig.8. The shapes of the FNN output fuzzy numbers (the solid curves in Figs.8 and 9) are similar to those of the target fuzzy numbers (the dotted lines in the same figures) for larger values of the membership score. These results shown in Figs.6-9 reveal that the best FNNs evolved by the FPSO approximate their target functions (especially for larger membership scores because the error is weighted more for the larger scores, see (25)), despite the fact that no training data is explicitly provided.

Fig.10 shows the error values of the best FNN for $F_1(x)$ among each number of FNNs evolved (e.g., 500,000 FNNs are evolved in total at the 5,000th generation by the FPSO with 100 particles). In this figure, "FPSO" shows the result by FPSO proposed in this paper, and "FGA" shows the result by FGA [6]. The error values are the averaged ones over five runs. Fig.11 shows the error values for $F_2(x)$ in the same manner as Fig.10. Figs.10 and 11 reveal that, for both of the two target functions, FGA contributed better than FPSO in evolving better FNNs: after the evolution of 1,000,000 FNNs, the dotted curves for FGA went below the solid curves for FPSO. This result will be because PSO tends to prematurely converge particles into a local minimum while GA can explorer the search space well by the crossover and mutation operations. Although the result indicate FGA is superior to FPSO in evolving neural networks with fuzzy weights, several researchers have reported that PSO can outperform GA [22-26]. The author will further compare FPSO with FGA by applying them to

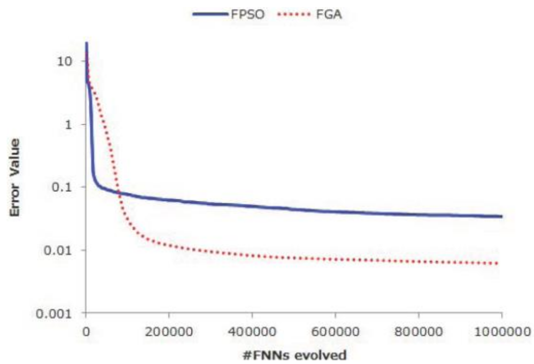

Figure 10. Error value of the best FNN at each number of FNNs evolved for modeling $F_1(x)$.

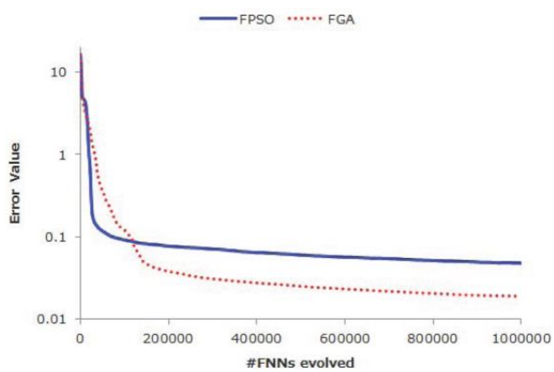

Figure 11. Error value of the best FNN at each number of FNNs evolved for modeling $F_2(x)$.

other fuzzy optimization problems, e.g., optimizing fuzzy if-then rules for fuzzy inference systems.

Besides, several methods have been proposed [27-32] for improving the traditional PSO. These improvements can be adopted to our FPSO. The author will evaluate how well these methods can improve our FPSO for solving fuzzy optimization problems.

5. CONCLUSION

In this paper, the author proposed the fuzzy-valued extension of PSO, and applied it to the evolution of neural networks with fuzzy weights and biases. In the proposed FPSO, genotype values are not real numbers but fuzzy numbers. To handle the fuzzy genotype values, the FPSO extends its processes of updating particles. The FPSO was challenged to evolve FNNs which model each of the two fuzzy functions. The experimental results showed that the best FNNs evolved by the FPSO approximated the target functions (especially for larger membership scores) despite the fact that no training data was explicitly provided.

In the future work, the author will further evaluate the ability of the FPSO by applying it to problems other than neuroevolution, e.g., evolving fuzzy if-then rules for fuzzy inference systems.

ACKNOWLEDGMENT

This research was supported by Kyoto Sangyo University Research Grant.

REFERENCES

[1] H. Ishibuchi, H. Tanaka and H. Okada, Fuzzy neural networks with fuzzy weights and fuzzy biases, IEEE International Conferences on Neural Networks, 1650-1655 (1993).

[2] D.B. Fogel, L.J. Fogel and V.W. Porto, Evolving neural networks, Biological Cybernetics, **63**, 6, 487-493 (1990).

[3] X. Yao, Evolving artificial neural networks, Proceedings of the IEEE, **87**, 9, 1423-1447 (1999).

[4] K.O. Stanley and R. Miikkulainen, Evolving neural networks through augmenting topologies, Evolutionary Computation, **10**, 2, 99-127 (2002).

[5] D. Floreano, P. Durr and C. Mattiussi, Neuroevolution: from architectures to learning, Evolutionary Intelligence, **1**, 1, 47-62 (2008).

[6] H. Okada, Genetic algorithm with fuzzy genotype values and its application to neuroevolution, International Journal of Computer, Information Science and Engineering, **8**, 1, 1-7 (2014).

[7] J. Kennedy and R. Eberhart, Particle swarm optimization, IEEE International Conference on Neural Networks, **IV**, 1942-1948 (1995).

[8] R. Eberhart and J. Kennedy, A new optimizer using particle swarm theory, 6th International Symposium on Micro Machine and Human Science, 39-43 (1995).

[9] J. Kennedy and R. Eberhart, Swarm Intelligence, Morgan Kaufmann (2001).

[10] H. Akkar, Optimization of artificial neural networks by using swarm intelligent, 6th International Conference on Networked Computing, 1-5 (2010).

[11] R. Mendes, P. Cortez, M. Rocha and J. Neves, Particle swarm for feedforward neural network training, IEEE International Joint Conference on Neural Networks, 1895-1899 (2002).

[12] K. Serkan, I. Turker, Y. Alper and G. Moncef, Evolutionary artificial neural networks by multi-dimensional particle swarm optimization, Neural Networks, **22**, 10, 1448-1462 (2009).

[13] J. Yu, L. Xi and S. Wang, An improved particle swarm optimization for evolving feedforward artificial neural networks, Neural Processing Letters, **26**, 3, 217-231 (2007).

[14] M. Carvalho and T.B. Ludermir, Particle swarm optimization of neural network architectures and weights, 7th International Conference on Hybrid Intelligent Systems, 336-339 (2007).

[15] B. Al-Kazemi and C.K. Mohan, Training feedforward neural networks using multi-phase particle swarm optimization, 9th International Conference on Neural Information Processing, **5**, 2615-2619 (2002).

[16] F. Zaho, Z. Ren, D. Yu and Y. Yang, Application of an improved particle swarm optimization algorithm for neural network training, International Conference on Neural Networks and Brain, **3**, 1639-1698 (2005).

[17] J. Salerno, Using the particle swarm optimization technique to train a recurrent neural model, IEEE International Conference on Tools with Artificial Intelligence, 45-49 (1997).

[18] M. Zamani and A. Sadeghian, A variation of particle swarm optimization for training of artificial neural networks, chapter 9, 131-144, in A-D. Ali (ed), Computational intelligence and modern heuristics, InTech (2010)

[19] M. Meissner, M. Schmuker and G. Schneider, Optimized particle swarm optimization (OPSO) and its application to artificial neural network training, BMC Bioinformatics, **7**, 125 (2006).

[20] L.A. Zadeh, The concept of a linguistic variable and its application to approximate reasoning - I, II, and III, Information Sciences, **8**, 199-249, 301-357, and **9**, 43-80 (1975).

[21] G. Alefeld and J. Herzberger, Introduction to Interval Computation, Academic Press (1983).

[22] C. Ou and W. Lin, Comparison between PSO and GA for parameters optimization of PID controller, IEEE International Conference on Mechatronics and Automation, 2471-2475 (2006).

[23] Y. Duan, R.G. Harley and T.G. Habetler, Comparison of particle swarm optimization and genetic algorithm in the design of permanent magnet motors, IEEE 6th International Power Electronics and Motion Control Conference, 822-825, (2009).

[24] D. cada, A. Rosa, L.C. Duarte and V.V. Lopes, Comparison of GA and PSO performance in parameter estimation of microbial growth models: a case-study using experimental data, IEEE Congress on Evolutionary Computation, 1-8 (2010).

[25] K.V.S.R. Murthy, M. Ramalinga Raju and G.G. Rao, Comparison between conventional, GA and PSO with respect to optimal capacitor placement in agricultural distribution system, Annual IEEE India Conference, 1-4 (2010).

[26] I. Kecskes, L. Szekacs, J.C. Fodor and P. Odry, PSO and GA optimization methods comparison on simulation model of a real hexapod robot, IEEE 9th International Conference on Computational Cybernetics (ICCC), 125-130 (2013).

[27] F. Zhao, Z. Ren, D. Yu and Y. Yang, Application of an improved particle swarm optimization algorithm for neural network training, International Conference on Neural Networks and Brain, \textbf{3}, 1693-1698 (2005).

[28] F. Nian, W. Li, X. Sun and M. Li, An improved particle swarm optimization application to independent component analysis, International Conference on Information Engineering and Computer Science, 1-4 (2009).

[29] J. Guo and T. Sheng-jing, An improved particle swarm optimization with re-initialization mechanism, International Conference on Intelligent Human-Machine Systems and Cybernetics, 437-441 (2009).

[30] X. Xiao, C. Mei and G. Liu, Improved particle swarm optimization algorithm based on random perturbations, Third International Joint Conference on Computational Science and Optimization, **1**, 404-408 (2010).

[31] Y.V. Pehlivanoglu and O. Baysal, Improved particle swarm optimization: catching the big wave on the surf, IEEE Congress on Evolutionary Computation, 1-8 (2012).

[32] Y. Cai, Z. Chen and H. Min, Improving particle swarm optimization algorithm for distributed sensing and search, Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 373-379 (2013).

Hidehiko Okada is currently a Professor with the Department of Computer Science and Engineering, Kyoto Sangyo University, Kyoto, Japan. He received the B.S. degree in industrial engineering and the Ph.D. degree in engineering from Osaka Prefecture University in 1992 and 2003, respectively. He had been a researcher with NEC Corporation from 1992 to 2003, and since 2004 he has been with the university. His current research interests include computational intelligence and human-computer interaction. He is a member of Information Processing Society of Japan, Institute of Electronics, Information and Communication Engineers, Society of Instrument and Control Engineers, Japanese Society for Artificial Intelligence, Japan Society for Fuzzy Theory and Intelligent Informatics and Human Interface Society. He received the best paper award in the 1st International Conference on Industrial Application Engineering 2013.