A Dynamic Hierarchical Fuzzy Neural Network for A General Continuous Function

Wei-Yen Wang, I-Hsum Li, Shu-Chang Li, Men-Shen Tsai, and Shun-Feng Su

Abstract

A serious problem limiting the applicability of the fuzzy neural networks is the "curse of dimensionality", especially for general continuous functions. A way to deal with this problem is to construct a dynamic hierarchical fuzzy neural network. In this paper, we propose a two-stage genetic algorithm to intelligently construct the dynamic hierarchical fuzzy neural network (HFNN) based on the merged-FNN for general continuous functions. First, we use a genetic algorithm which is popular for flowshop scheduling problems (GA_FSP) to construct the HFNN. Then, a reduced-form genetic algorithm (RGA) optimizes the HFNN constructed by GA_FSP. For a real-world application, the presented method is used to approximate the Taiwanese stock market.

Keywords: hierarchical structures, genetic algorithms, Fuzzy neural networks.

1. Introduction

Recently, using fuzzy systems for system identification has become a popular topic [1-7]. In nonlinear system identification, fuzzy systems can effectively fit the nonlinear system by calculating the optimized coefficients of the learning mechanism. The traditional fuzzy systems cannot directly be used when there are a large number of input variables, because there will be too many free parameters to be trained, and hence high computational and memory demands. Bellman [13] called this phenomenon "the curse of dimensionality". To overcome this difficulty, [8]-[10] proposed hierarchical fuzzy systems, which decompose a single large fuzzy system into subsystems. Decomposition leads to a multi-staged structure of several simple, interacting subsystems, each having a reduced number of input variables. Each subsystem thus has a reduced complexity compared to a single fuzzy neural network, measured by the number of rules and of parameters which have to be stored and processed.

Some studies [5,6] combining fuzzy logic with neural networks have been done to improve the efficiency of function approximation. In [7], we proposed a hierarchical fuzzy neural network, called the merged fuzzy neural network (merged-FNN), to build a universal approximator for a battery state-of-charge (BSOC) problem. Because the BSOC problem can be modeled as a continuous function with natural hierarchical structure, there was a natural fit between the problem structure and the merged-FNN constructed in [7]. However, in practice, it is often the case that a system to be modeled has an unknown hierarchical structure. We call this case a general continuous function in this paper. For general continuous functions, although universal representation methods, such as FNNs [4-7] or neural networks [11-12], can be used to approximate such systems, this is often less effective and efficient, as such FNNs or neural network models are less accurate, more complicated, and are unlikely to match physical meaning and mechanism. Therefore, an approximator with the ability to self-construct a hierarchical structure is needed for learning general continuous functions.

In this paper, we propose a two-stage genetic algorithm to intelligently construct the hierarchical fuzzy neural networks (HFNNs) based on the merged-FNN [7] for general continuous functions with training data pairs $[X(k), y(k)]$, where $k = 1,..., p$. First, to construct an HFNN which matches the hierarchical structure of the problem, we use a genetic algorithm, GA_FSP, which is popular in the flowshop problem. In a flowshop scheduling problem (FSP) [14, 15] we have n independent jobs $\{J_1,..., J_n\}$ that have to be processed on m different machines $\{M_1,..., M_m\}$. Every job is composed of m operations, and every operation requires a different machine. We consider the input variables $\{x_1,..., x_n\}$ as n independent jobs and the neurons of the first layer of the merged-FNN as m different machines. Therefore, we can extend the GA_FSP to construct a dynamic hierarchical structure. In the second stage, a reduced-form genetic algorithm (RGA) [4] optimizes the HFNN constructed by the GA_FSP. After the two-stage genetic al-

Corresponding Author: Wei-Yen Wang is with the Department of Applied Electronics Technology, National Taiwan Normal University, 160, He-ping East Rd., Section 1, Taipei 106, Taiwan.
E-mail: wwang@ntnu.edu.tw
Manuscript received 28 Aug. 2008; revised 1 Dec. 2008; accepted 25 Dec. 2008.

gorithm, a recursive learning strategy is used to find the simplest structure of the HFNN for general continuous functions. Finally, the simulation results of estimating the Taiwanese stock market is used to demonstrate the effectiveness of the present method.

2. A Merged Fuzzy Neural Network for General Continuous Functions

A. Hierarchical structure of general continuous functions

A general continuous function is given as follows:

$$y = \overline{G}(X) = \overline{G}(x_1, x_2, ..., x_n) \tag{1}$$

where $X = (x_1, x_2, ..., x_n)$ is the input variable vector, and y is the output variable. According to Kolmogorov's theorem [9, 10], a general continuous function $\overline{G}(X)$ can be represented as $M+1$ continuous functions with a natural hierarchical structure [9]:

$$y = G(X) = G_1(G_2(X), G_3(X), ..., G_{M+1}(X)) \tag{2}$$

where the qth continuous function $G_q\ (q = 1, 2, ..., M+1)$, each of which has a natural hierarchical structure, is

$$y_q^1 = G_q(Y_q^2) = g_q^1(y_q^{2,1}, y_q^{2,2}, ..., y_q^{2,l}) \tag{3}$$

and

$$y_q^{2,j} = g_q^{2,j}(X_q^{2,j}) = g_q^{2,j}(x_1^{q,2,j}, x_2^{q,2,j}, ..., x_{m_{q,2,j}}^{q,2,j}),\ j = 1, 2, ..., l \tag{4}$$

In equations (3) and (4), g_q^1 and $g_q^{2,j}(j = 1, 2, ..., l)$ are lower dimensional functions (i.e., $l < n$ and $m_{q,2,j} < n$).

A natural approach to approximating a hierarchical function $G(X)$ is to use a matching hierarchical structure, that is, to design a function $H(X)$ with the same hierarchical structure as $G(X)$. In the next section, we will explain the structure of the hierarchical fuzzy neural networks used to approximate the hierarchical function $G(X)$.

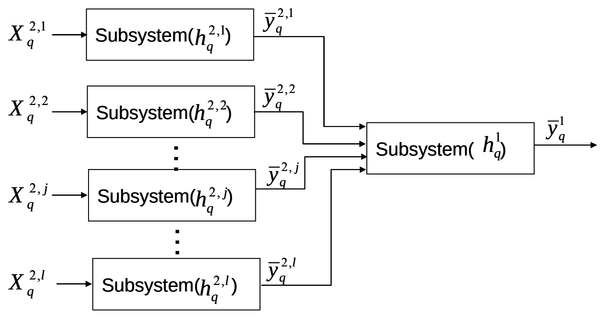


Fig. 1. the configuration of the qth merged-FNN.

B. Constructing a hierarchical fuzzy neural network (HFNN) from the merged fuzzy neural networks (Merged-FNNs)

Let $H(X) = H_1(H_2(X), ..., H_{M+1}(X))$ be the hierarchical fuzzy neural network (HFNN) matching the hierarchical function $G(X)$, where $H(X)$ consists of several merged-FNNs [7]. Each merged-FNN of the HFNN (i.e $H_q(X)$) is used to approximate one of the hierarchical functions $G_q(X)$, where $q = 1, ..., M+1$. Figure 1 is the configuration of the merged-FNN, which is a two level fuzzy neural network with a hierarchical structure. Each subsystem of the merged-FNN is a typical fuzzy-neural network.

For the qth merged-FNN, the ith fuzzy IF-THEN rule is given by equation (5) as follows:

$$R_q^{(i)}: \begin{cases} \text{IF } x_1^{q,2,j} \text{ is } A_1^{q,2,j^{(i)}} \text{ and ... and } x_{m_{q,2,j}}^{q,2,j} \text{ is } A_{m_{q,2,j}}^{q,2,j^{(i)}} \\ \text{then } \overline{y}_q^{2,j} \text{ is } w_{(i)}^{q,2,j}, \quad \text{for FNN}_q^{2,j} \\ \text{IF } \overline{y}_q^{2,1} \text{ is } A_1^{q,1^{(i)}} \text{ and ... and } \overline{y}_q^{2,l} \text{ is } A_l^{q,1^{(i)}} \\ \text{then } \overline{y}_q^1 \text{ is } w_{(i)}^{q,1}, \quad \text{for FNN}_q^1 \end{cases} \tag{5}$$

where $\text{FNN}_q^{2,j}$ $(q = 1, ...M; j = 1, 2, ..., l)$ represents an FNN in Level 2, FNN_q^1 is the FNN in Level 1, $A_k^{q,2,j}\ (k = 1, ..., m_{q,2,j})$ and $A_k^{q,1}(k = 1, ..., l)$ are fuzzy sets and $w^{q,2,j}$ and $w^{q,1}$ are singleton fuzzy sets. In the rest of this paper, we adopt A to represent $A_k^{q,2,j}$ and $A_k^{q,1}$, which neglects the number of levels and subsystems. That is, A stands for the entire antecedent fuzzy set no matter what the level and subsystem are. By using product inference, center-averaging and singleton fuzzification, the output of FNN$_1$ at Level-1 is

$$\overline{y}_q^1 = \frac{\sum_{i=1}^{\overline{h}_q^1} w_{(i)}^{q,1} \left[\prod_{k=1}^{l} \mu_{A_k^i}(\overline{y}_q^{2,k}) \right]}{\sum_{i=1}^{\overline{h}_q^1} \left[\prod_{k=1}^{l} \mu_{A_j^i}(\overline{y}_q^{2,k}) \right]} = w^{q,1^T} \varphi^{q,1}(\overline{Y}_q^2) \tag{6}$$

where $w^{q,1} = [w_{(1)}^{q,1}\ w_{(2)}^{q,1} ... w_{(\overline{h}_q^1)}^{q,1}]^T$ is the weighting vector for the merged-FNN in Level 1 and $\overline{Y}_q^2 = [\overline{y}_q^{2,1}\ \overline{y}_q^{2,2} ... \overline{y}_q^{2,l}]^T$. The outputs of $\text{FNN}_q^{2,j}$ $(q = 1, ...M; j = 1, 2, ..., l)$, at Level-2, which are the inputs of FNN_q^1 at Level-1, can be expressed as

$$\overline{y}_q^{2,j} = \frac{\sum_{i=1}^{\overline{h}_q^{2,j}} w_{(i)}^{q,2,j} \left[\prod_{k=1}^{m_{q,2,j}} \mu_{A_k^i}(X_q^{2,k}) \right]}{\sum_{i=1}^{\overline{h}_q^{2,j}} \left[\prod_{k=1}^{m_{q,2,j}} \mu_{A_k^i}(X_q^{2,k}) \right]} = w^{q,2,j^T} \varphi^{q,2,j}(X_q^{2,k}) \tag{7}$$

where $w^{q,2,j} = [w_{(1)}^{q,2,j}\ w_{(2)}^{q,2,j} ... w_{(\overline{h}_q^{2,j})}^{q,2,j}]^T$ is the weighting vector for the merged-FNN in Level 2, $\overline{h}_q^1$ and $\overline{h}_q^{2,j}$ are the number of fuzzy rules in Level-1 and Level-2, respectively, $m_{q,2,j}$ represents the number of input variables for the jth FNN at Level-2, $w^q = [w^{q,1^T}\ w^{q,2,1^T}\ w^{q,2,2^T} ... w^{q,2,l^T}]^T$ is an adjustable parameter vector for the qth merged-FNN, and $\varphi^{q,1}(\cdot)$ and $\varphi^{q,2,j}(\cdot)$ are fuzzy basis functions.

In the merged-FNN for the continuous function $G_q(X)\ (q=1,\ldots,M+1)$, the subsystem $h_q^{2,j}$ can be represented is

$$\bar{y}_q^{2,j} = h_q^{2,j}(X_q^{2,j}) = h_q^{2,j}(x_1^{q,2,j}, x_2^{q,2,j},\ldots,x_{m_{q,2,j}}^{q,2,j}),\ j=1,\ldots,l \tag{8}$$

where $\bar{y}_q^{2,j}$ and $X_q^{2,j}$ respectively represent the output and input of the jth subsystem at Level-2. The subsystem h_q^1 is

$$\bar{y}_q^1 = h_q^1(\bar{Y}_q^2) = h_q^1(\bar{y}_q^{2,1},\bar{y}_q^{2,2},\ldots,\bar{y}_q^{2,l}) \tag{9}$$

where $\bar{Y}_q^2 = [\bar{y}_q^{2,1}\ \bar{y}_q^{2,2},\ldots,\bar{y}_q^{2,l}]^T$ is the input vector at Level-1 and $\bar{y}_q^{2,j}$ is the output variable of jth subsystem $h_q^{2,j}$ at the Level-2. Finally, the mathematical formula of the merged-FNN is

$$\begin{aligned}\bar{y}_q^1 &= h_q^1(\bar{Y}_q^2) = h_q^1(h_q^{2,1}(X_q^{2,1}), h_q^{2,2}(X_q^{2,2}),\ldots,h_q^{2,l}(X_q^{2,l}))\\ &= h_q^1(\bar{y}_q^{2,1},\bar{y}_q^{2,1},\ldots,\bar{y}_q^{2,l})\end{aligned} \tag{10}$$

From equations (2) and (10), it is easy to define the mathematical formula of the HFNN as

$$\bar{y}_1^1 = h_1^1(h_2^1(\bar{Y}_2^2),\ldots,h_{M+1}^1(\bar{Y}_{M+1}^2)) \tag{11}$$

and the whole adjustable parameter vector is $w=[w^1\ w^2\ldots w^q\ldots w^{M+1}]$. The objective of the learning algorithm is to minimize the error function:

$$E=\sum_{k=1}^{p}(y(k)-\bar{y}_1^1(k))^2 \tag{12}$$

Remark 1: Kolmogorov's theorem [10] guarantees that the HFNN constructed by the present learning method can approximate the general continuous function $G(X)$ to any desired arbitrary error ε if the number M is finite. This HFNN is called the universal approximator for the general continuous function $G(X)$.

3. Constructing the HFNN for A General Continuous Function with Two-Stage GA

In this section, we propose a two-stage genetic algorithm to intelligently construct the HFNN for general continuous functions $\overline{G}(X)$ with training data pairs $[X(k), y(k)]$, where $k=1,\ldots,p$. First, to construct an HFNN which matches the hierarchical structure of the problem, we use a genetic algorithm, GA_FSP, which is popular in the flowshop problem. Subsequently, a reduced-form genetic algorithm RGA [4] optimizes the merged-FNN constructed by GA_FSP. In section III-C, a recursive learning strategy is proposed to find the minimum M in (11) for the general continuous function (2) (i.e. the simplest structure of the HFNN that models the general continuous function).

A. Constructing the HFNN using GA_FSP

For a flowshop scheduling problem (FSP) one usually has n independent jobs $\{J_1,\ldots,J_n\}$ that have to be processed on m different machines $\{M_1,\ldots,M_m\}$. Every job is at least composed of one operation, and different operations are done by different machines. The objective is to find an ordering of the jobs on the machines. In this paper, to construct the HFNN for unknown continuous functions with the sense of the flowshop scheduling problem, the following assumptions are needed: 1. the input variables $\{x_1,\ldots,x_n\}$ are considered as n independent *one-operation* jobs; 2. each neuron of the first layer of the HFNN is viewed as n different machines. These machines only handle one job. Note that the number of machines is the same as the number of jobs in our case. Then we use the GA_FSP, which will be introduced later to optimize the flowshop scheduling problem, to construct a dynamic hierarchical structure. In this section, first we examine various genetic operators of the GA_FSP such as coding, crossover, mutation and random grouping.

1) Coding

A sequence of input variables is handled as a string in this paper. For example, the string "$x_1\ x_2 \ldots x_n$" represents a sequence of the input variables of the qth merged-FNN. If the string "$x_1\ x_2\ x_1 \ldots x_n$" is generated by genetic operators (i.e. crossover and mutation), this string is not a feasible solution of the flowshop scheduling problem because the gene "x_1" appear twice in the string and one of the genes $x_i\ (i=2,3,\ldots n)$ does not appear. In this paper, we denote the sequence of n input variables by an n-dimensional vector $\bar{s}^\gamma=[s_1^\gamma\ s_2^\gamma \ldots s_i^\gamma \ldots s_n^\gamma]$, where s_i^γ denote the ith input variable x_i in the γth chromosome. The population can be represented as

$$\bar{S}=\begin{bmatrix}\bar{s}^1\\ \bar{s}^2\\ \vdots\\ \bar{s}^\gamma\\ \vdots\\ \bar{s}^{\kappa_2}\end{bmatrix}=\begin{bmatrix}s_1^1 & s_2^1 & \cdots & s_n^1\\ s_1^2 & s_2^2 & \cdots & s_n^2\\ \vdots & \vdots & \ddots & \vdots\\ s_1^\gamma & s_2^\gamma & \cdots & s_n^\gamma\\ \vdots & \vdots & \ddots & \vdots\\ s_1^{\kappa_2} & s_2^{\kappa_2} & \cdots & s_n^{\kappa_2}\end{bmatrix} \tag{13}$$

where κ_2 is the number of chromosomes.

2) Selection

Selection is an operation to select parent chromosomes for generating a new chromosome (i.e. child). From the parent chromosomes, we select the fittest α chromosomes to produce the next generation.

3) Crossover

Crossover is an operation to generate a new chromosome (i.e., child) from two parent chromosomes. The one-point crossover used in our paper is illustrated in Fig.

2. One point is randomly selected for dividing one parent. The set of genes on one side (each side is chosen with the same probability) is passed from one parent to the child, and the other genes are placed in the order of their appearance in the other parent. In Fig. 2, genes x_1, x_2 and x_3 are inherited from Parent 1, and the other genes (i.e. genes $x_4, x_5,..., x_n$) are placed in the order of their appearance in Parent 2.

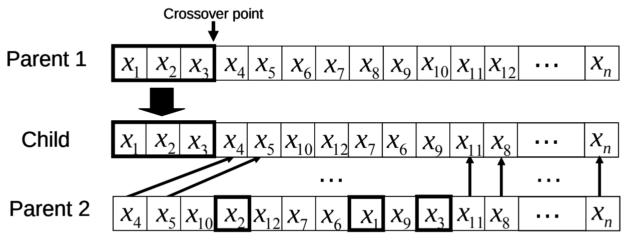


Fig. 2. One-point crossover operator.

4) Mutation

Mutation is an operation to change the order of n genes in each chromosome generated by a crossover operator. Such a mutation operation can be viewed as a transition from a current solution to a neighboring solution in the local search space. We adopt the adjacent two-gene change shown in Fig. 3. The adjacent three genes to be changed are randomly selected.

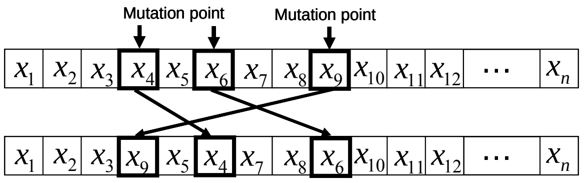


Fig. 3. A mutation operator: Arbitrary three-gene change change.

5) Random grouping

After mutating, a random process is used to construct the structure of the qth merged-FNN. As Fig. 4(a) shows, we randomly divide l groups form the γth chromosome, where the genes in group j means the input variables $X_q^{2,j}$ for the jth subsystem in Level-2 of the qth merged-FNN. The corresponding merged-FNN is shown in Fig. 4(b).

6) Fitness function

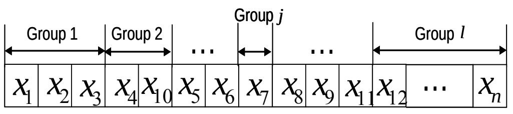
The fitness function is defined as follows:

$$fitness_\gamma = \frac{1}{1+E(\bar{s}^\gamma)}, \gamma = 1,..., \kappa_2 \quad (14)$$

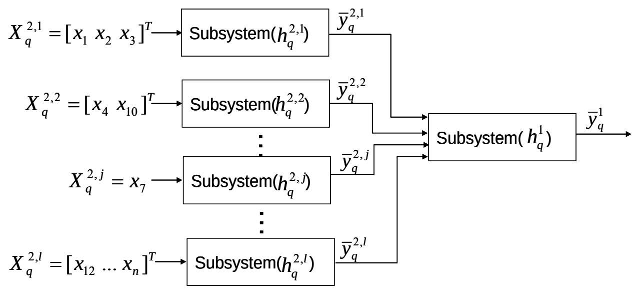
$E(\bar{s}^\gamma) = \sum_{k=1}^{p}(y(k) - \bar{y}_1^1(k))^2$ is the γth error function, where p is the number of training data pairs.

7) Sorting

The newly generated population is sorted by ranking the fitness of chromosomes within the population, resulting in $E(\bar{s}^1) \le E(\bar{s}^2) \cdots \le E(\bar{s}^{\kappa_2})$. The first chromosome of the sorted population $\bar{S} = [\bar{s}^1\ \bar{s}^2 \cdots \bar{s}^{\kappa_2}]^T$ has the highest fitness value (or smallest error).



(a)



(b)

Fig. 4. (a) the random grouping. (b) the corresponding structure of the merged-FNN.

B. Optimizing the HFNN with RGA

The reduced-form genetic algorithm (RGA) [4] was proposed to deal with a vast number of adjustable parameters in the fuzzy-neural networks. The key properties of the RGA are as follows: 1. It uses a sequential-search-based crossover point (SSCP) method to determine a better crossover point is determined. 2. The population size is fixed and can be reduced to a minimum size of 4. 3. The crossover operator is simplified to be a single gene crossover. After GA_FSP, the HFNN usually contents a vast number of adjustable parameters, so we adopt RGA to optimize the HFNN. However, we don't use single gene crossover, because single gene crossover limits the speed of convergence during the learning process. Instead, we use one point crossover.

C. Recursive learning method

Unfortunately, a problem comes along with the hierarchical structure (2). That is, the representation of the two-level hierarchical structure, which is composed of $M+1$ continuous functions $G_q(X)$, is so complicated that the function $H(X)$ matching $G(X)$ will tend to be non-useful. Hence, a recursive learning method (from $q=1$ to $M+1$) is applied to solve this problem. We construct the 1st merged-FNN $H_1(X)$ $(q=1)$ by the present two-stage GA for the hierarchical continuous function $G_1(X)$ $(q=1)$ at first and then go on the next step to build the structure as $H_1(H_2(X))$ $(q=2)$ for the hierarchical continuous function $G_1(G_2(X))$ $(q=2)$. We stop when the approximated error is desirable or $q = M+1$. Note that all continuous functions $G_q(X)$ $(q=1,\dots,M+1)$ are the hierarchical structure shown in (3).

D. Overall system

In order to find a desirable HFNN (i.e. one with a less complexity of the structure or less computational and memory demands) for general continuous functions, a two-stage genetic algorithm with random grouping is proposed in this paper. The two-stage genetic algorithm includes GA_FSP and RGA. GA_FSP constructs the merged-FNN at first and RGA optimizes the merged-FNN constructed by GA_FSP. The flowcharts of the two-stage GA and the overall system are shown in Fig. 5(a) and Fig. 5(b), respectively.

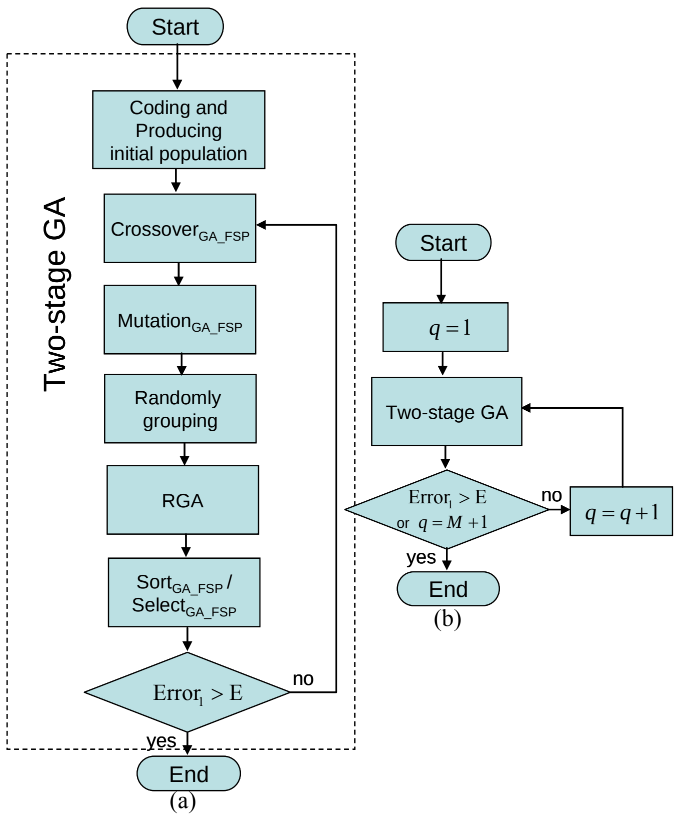


Fig. 5. (a) the flowchart of the two-stage genetic algorithm; (b) the flowchart of the overall system.

4. Simulation Results

In this section, an example is given to show the effectiveness of training of the present hierarchical fuzzy neural network (HFNN). In our example, we approximate the Taiwanese stock market. We choose 12 factors, selected from the fundamental analysis and technical analysis form January 1, 1998 to April 10, 1999, a period of 100 days, to be the input variables. The factors for the fundamental analysis include the leading indicator, the consumer price index, and the prime lending rate. The technical analysis factors are 6-day moving average, 6-day BIAS , 6-day relative strength index, 9-day stochastic index (K), 9-day stochastic index (D), 9-day MACD, 13-day psychological line, trading volume, closing price. The next day's closing price is the output variable.

Some information used for approximating the Taiwanese stock market are as follows: 1. The training data are normalized into the range [0, 1] for convenience; 2. Each input of the FNN (i.e. $h_q^{2,j}$ or h_q^1) has 3 B-spline membership functions (BMFs) [3]; 3. The number of the chromosomes for GA_FSP is 6 and for RGA is 4; The number of learning iteration for GA_FSP is 30 and for RGA is 500; 5. The parameter α for selection is 2. Table 1 shows the simulation results for which the numbers of training data pairs are 100, 90 and 80. Figures 7, 9 and 11 show the simulation results of the present HFNN with two-stage GA when the number of training data pairs is 100, 90 and 80, respectively. The error curves for the various number of training data pairs such as 100, 90 and 80 are shown in Figs. 6, 8 and 10.

Table 1. The simulation results for estimation Taiwanese stock market.

learning structure / The number of training data pair	HFNN with the two-stage GA (MSE)
Training data (1-100 days)	2.5360e-004
Training data (1-90 days)	1.1982e-004
Training data (1-80 days)	1.5764e-006

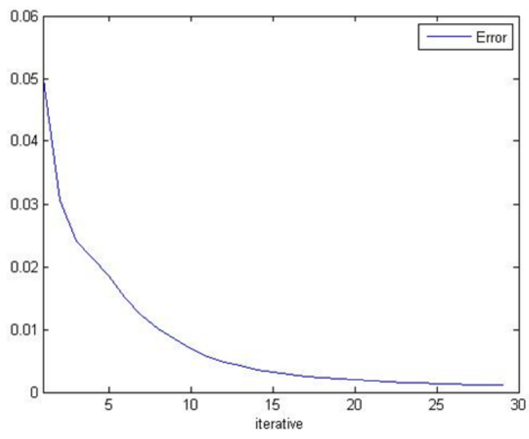


Fig. 6. Error curve of HFNN trained by two-stage GA when training data pair is 100.

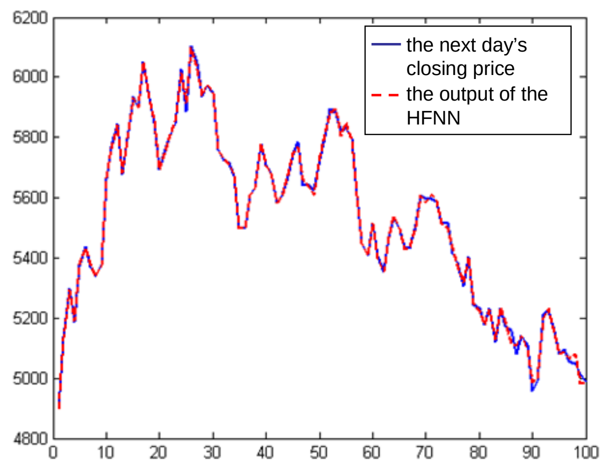


Fig. 7. Output of the HFNN trained by two-stage GA when training data pair is 100.

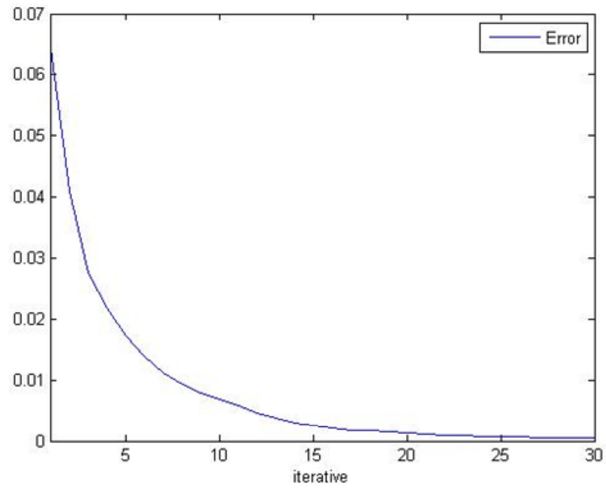


Fig. 8. Error curve of HFNN trained by two-stage GA when training data pair is 90.

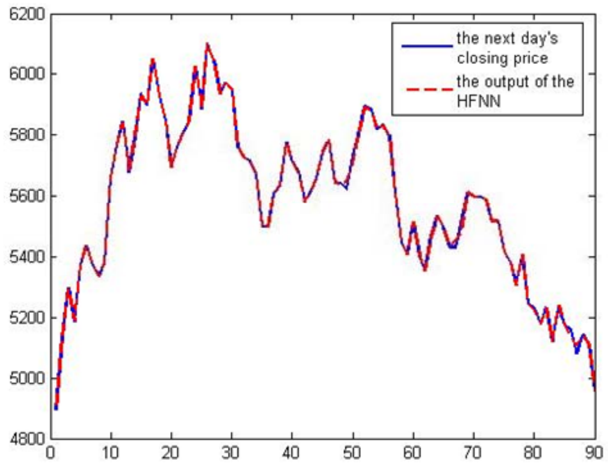


Fig. 9. Output of the HFNN trained by two-stage GA when training data pair is 90.

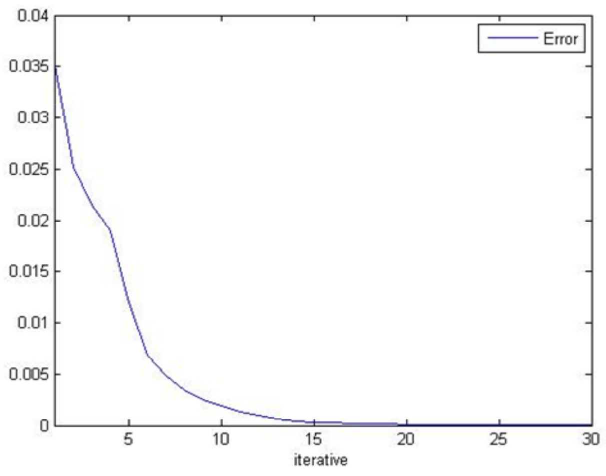


Fig. 10. Error curve of HFNN trained by two-stage GA when training data pair is 80.

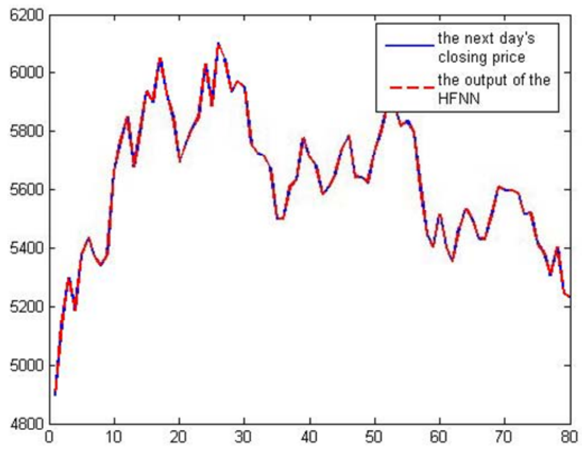


Fig. 11. Output of the HFNN trained by two-stage GA when training data pair is 80.

5. Conclusion

Compared to the tradition fuzzy neural network, the HFNN constructed by the present two-stage genetic algorithm can effectively approximate a general continuous function. In the two-stage genetic algorithm, first we bring the GA_FSP into a new topic of constructing a dynamic hierarchical structure. Second, the RGA is used to handle the vast number of adjustable parameter in the HFNN constructed by GA_FSP. From the simulation results, the present method of this paper provides a well-suited way of learning for Taiwanese stock market.

References

[1] L.X. Wang, Adaptive Fuzzy Systems and Control: Design and Stability Analysis, Prentice Hall International, 1994.

[2] A. Boubakir, F. Boudjema, C. Boubakir, and S. Labiod, "A fuzzy sliding mode controller using nonlinear sliding surface applied to the coupled tanks system," International Journal of Fuzzy Systems, vol. 10, no. 2, pp. 112-118, June 2008.

[3] C.H. Wang, W.Y. Wang, T.T. Lee, and P.S. Tseng, "Fuzzy B-spline membership function (BMF) and its applications in fuzzy-neural control," IEEE Transactions on Systems Man and Cybernetics, vol. 25, no. 5, pp. 841-851, May 1995.

[4] W. Y. Wang and Y. H. Li, "Evolutionary learning of BMF fuzzy-neural networks using a reduced-form genetic algorithm," IEEE Transactions on Systems, Man, And Cybernetics-Part B, vol. 33, no. 6, pp. 966-976, Dec. 2003.

[5] W. Y. Wang, Y. H. Chien, Y. G. Leu, and T. T. Lee, "On-line adaptive T-S fuzzy-neural control for a class of general multi-link robot manipulators," International Journal of Fuzzy Systems, vol. 10, no. 4, pp. 240-249, December 2008.

[6] X. F. Wang, "Fuzzy number intuitionistic fuzzy arithmetic aggregation operators," International Journal of Fuzzy systems, vol. 10, no. 2, pp. 104-111, 2008.

[7] I.H. Li, W.Y. Wang, S.F. Su and Y.S. Lee, "A merged fuzzy neural network and its applications in battery state-of-charge estimation," IEEE Transactions on Energy Conversion, vol. 22, no.3, pp. 697-708, 2007.

[8] M.G. Joo and J.S. Lee, "Universal approximation by hierarchical fuzzy systems with constrains on the fuzzy rules," Fuzzy Sets Systems, vol. 130, pp. 175-188, 2002.

[9] X.J. Zeng and J.A. Keane, "Approximation capabilities of hierarchical fuzzy systems," IEEE Transactions on Fuzzy Systems, vol. 13, no. 5, October, 2005.

[10] L.X. Wang, "Universal approximation by hierarchical fuzzy systems," Fuzzy sets and system, vol. 93, pp. 223-230, 1998.

[11] Z.G. Hou, M.M. Gupta, P.N. Nikiforuk, and M. Tan, and L. Cheng, "A recurrent neural network for hierarchical control of interconnected dynamic systems," IEEE Transactions on Neural Networks, vol. 18, no. 2, pp.466-481, 2007.

[12] R.J. Leduce, B.A. Brandin, M. Lawford, and W.M. Wonham, "Hierarchical interface-based supervisory control-part I: serial case," IEEE Transactions on Automatic Control, vol. 50, no. 9, September 2005.

[13] R. Bellman, Adaptive control processes, Princeton University Press, Princeton, 1966.

[14] R. Ruiz, c. Maroto, and J. Alcaraz, "Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics," European Journal of Operational Research, vol. 165, pp. 34-54, 2005.

[15] Z. Lian, X. Gu, and b. Jiao, "A similar particle swarm optimization algorithm for permutation flowshop scheduling to minimize makespan," Applied Mathematics and Computation, vol. 175, pp. 773-785, 2006.