NARMAX-Model-Based Time Series Modeling and Prediction: Feedforward and Recurrent Fuzzy Neural Network Approaches

Yang Gao, *Member, IEEE* and Meng Joo Er, *Member, IEEE*

School of Electrical and Electronic Engineering

Nanyang Technological University, Singapore

Abstract

The nonlinear autoregressive moving average with exogenous inputs (NARMAX) model provides a powerful representation for time series analysis, modeling and prediction due to its strength to accommodate the dynamic, complex and nonlinear nature of real time series applications. This paper focuses on the modeling and prediction of NARMAX-model-based time series using the fuzzy neural network (FNN) methodology with an extention of the model represention include feedforward and recurrent FNNs. This paper introduces and develops a efficient algorithm, namely generalized fuzzy neural network (G-FNN) learning algorithm, for model structure determination and parameter identification with the aim of producing improved predictive performance for NARMAX time series models. Experiments and comparisons demonstrate that the proposed G-FNN approaches can effectively learn complex temporal sequences in an adaptive way and outperform some well-known existing methods.

Keyword

Time series prediction, fuzzy neural networks, NARMAX models.

1 Introduction

Time series prediction is an important practical problem with variety of applications in business and economic planning, inventory and production control, weather forecasting, signal processing, and many other fields. In the last decade, neural networks (NNs) have been extensively applied for complex time series processing tasks [3, 4, 8, 9, 10]. This is due to their strength in handling nonlinear functional dependencies between past time series values and the estimate of the value to be forecast. More recently, fuzzy logic has been incorporated with the neural models for time series prediction [1, 2, 5, 6, 11], which are generally known as fuzzy neural networks (FNNs) or NN-based fuzzy inference systems (FISs) approaches. The FNN possesses both the advantages of FISs, such as human-like thinking and ease of incorporating expert knowledge, and NNs, such as learning abilities, optimization abilities and connectionist structures. By virtue of this, low-level learning and computational power of NNs can be incorporated into the FISs on one hand and high-level human-like thinking and reasoning of FISs can be incorporated into NNs on the other hand.

In this paper, NARMAX time series models are investigated using FNN approaches in both feedforward and recurrent model representations. The proposed FNN predictors have ability to model complex time series with on-line adjustment, fast learning speed, self-organizing FNN topology, good generalization and computational efficiency. This is achieved by applying a sequential and hybrid (supervised/unsupervised) learning algorithm, namely generalized fuzzy neural network (G-FNN) learning algorithm, to form the FNN prediction model. Various comparative analysis are developed to show the advancing performance of the proposed approaches over some existing methods.

The rest of the paper is organized as follows. Section 2 reviews the NARMAX models as a point of departure for the use of feedforward and recurrent G-FNNs. The concept of optimal predictors is also introduced in this section. Section 3 provides a description of the G-FNN architecture and learning algorithm. In Section 4, one feedforward and two recurrent G-FNN predictors are proposed for nonlinear time series modeling and prediction. Two experiments are presented in Section 5 to demonstrate the predictive ability of the proposed methods. The results are compared with those using a few existing methods. Finally, conclusions and directions for future research are given in Section 6.

2 NARMAX Model and Optimal Predictors

2.1 General NARMAX(n_y,n_e,n_x) Model

The statistical approach for forecasting involves the construction of stochastic models to predict the value of an observation $y(t)$ using previous observations. A very general class of such models used for forecasting purpose is the nonlinear autoregressive moving average with exogenous inputs (NARMAX) models given by

$$y(t) = \mathbf{F}[y(t-1),\ldots,y(t-n_y),e(t-1),\ldots,e(t-n_e),x(t-1),\ldots,x(t-n_x)] + e(t) \quad (1)$$

where y, e and x are output, noise and external input of the system model respectively, n_y, n_e and n_x are the maximum lags in the output, noise and input respectively, and $\mathbf{F}$ is an unknown smooth function. It is assumed that $e(t)$ is zero mean, independent and identically distributed, independent of past y and x, and has a finite variance σ^2.

Several special cases of the general NARMAX(n_y,n_e,n_x) model are frequently seen, which are represented as

NAR(n_y) Model:

$$y(t) = \mathbf{F}[y(t-1),\ldots,y(t-n_y)] + e(t) \quad (2)$$

NARMA(n_y,n_e) Model:

$$y(t) = \mathbf{F}[y(t-1),\ldots,y(t-n_y),e(t-1),\ldots,e(t-n_e)] + e(t) \quad (3)$$

NARX(n_y,n_x) Model:

$$y(t) = \mathbf{F}[y(t-1),\ldots,y(t-n_y),x(t-1),\ldots,x(t-n_x)] + e(t) \quad (4)$$

2.2 Optimal Predictors

Optimum prediction theory focuses on the sense of minimizing mean squared error (MSE). It is known from [14] that the minimum MSE predictor is the conditional mean as follows, given the

infinite past and provided the conditional mean exists.

$$\hat{y}(t) = \mathbf{E}[y(t)|y(t-1), y(t-2), \ldots] \tag{5}$$

In practice, one has only the finite past commencing with the first observation. In this case, the minimum MSE predictor is [4]

$$\hat{y}(t) = \mathbf{E}[y(t)|y(t-1), y(t-2), \ldots, y(1)] \tag{6}$$

2.2.1 Approximately Optimal NAR(X) Predictors

As it is assumed that $e(t)$ is zero mean and has finite variance σ^2 in (1), the optimal predictor (6) for the NAR(n_y) or NARX(n_y,n_x) model in (2) or (4) is approximately given by

$$\begin{aligned} \hat{y}(t) &= \mathbf{E}[y(t)|y(t-1), \ldots, y(t-n_y)] \\ &= \mathbf{F}[y(t-1), \ldots, y(t-n_y)] \qquad \text{(NAR model)} \qquad (7) \\ &\text{or} \\ &= \mathbf{F}[y(t-1), \ldots, y(t-n_y), x(t-1), \ldots, x(t-n_x)] \qquad \text{(NARX model)} \qquad (8) \end{aligned}$$

where optimal predictors (7) and (8) have MSE σ^2.

2.2.2 Approximately Optimal NARMA(X) Predictors

For the NARMA(n_y,n_e) or NARMAX(n_y,n_x,n_e) model in (3) or (1), the optimal predictor (6) is approximately given by

$$\begin{aligned} \hat{y}(t) &= \mathbf{E}[y(t)|y(t-1), \ldots, y(t-n_y)] \\ &= \mathbf{F}[y(t-1), \ldots, y(t-n_y), \hat{e}(t-1), \ldots, \hat{e}(t-n_e)] \qquad \text{NARMA model} \qquad (9) \\ &\text{or} \\ &= \mathbf{F}[y(t-1), \ldots, y(t-n_y), \hat{e}(t-1), \ldots, \hat{e}(t-n_e), x(t-1), \ldots, x(t-n_x)] \\ & \qquad \text{NARMAX model} \qquad (10) \end{aligned}$$

where $\hat{e} = y - \hat{y}$.

In this case, the following initial conditions are often used

$$\hat{y}(t) = \hat{e}(t) = 0 \quad t \leq 0 \tag{11}$$

3 Basic G-FNN = Takagi-Sugeno-Kan-Type FIS

In this section, basic G-FNN architecture and learning algorithm are introduced as the preliminary knowledge for FNN predictor designs in the next section.

3.1 G-FNN Architecture

G-FNN is a newly developed neural-network-based TSK-type FIS whose architecture is given by Figure 1.

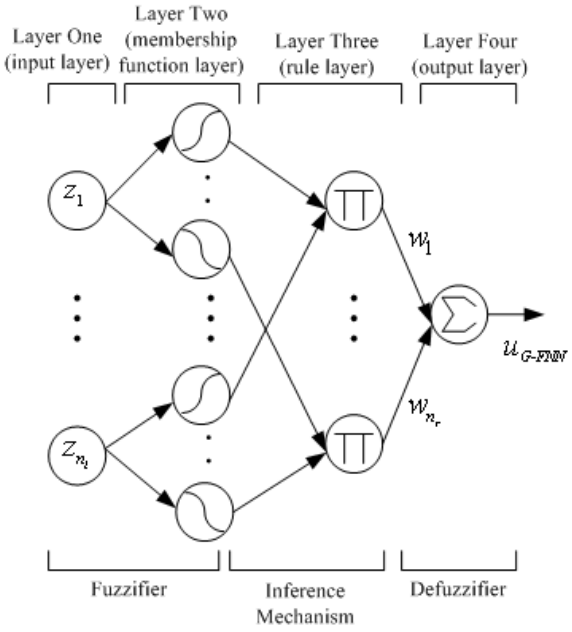

Figure 1: G-FNN architecture.

Output of the G-FNN is given by

$$\begin{aligned} u_{G-FNN}(t) &= \mathbf{F}_{G-FNN}[\mathbf{z}(t)] = \sum_{j=1}^{n_r} \phi_j(\mathbf{z}) w_j \\ &= \sum_{j=1}^{n_r} \exp[-(\mathbf{z}-\mathbf{c}_j)^T \Sigma_j (\mathbf{z}-\mathbf{c}_j)] w_j \end{aligned} \tag{12}$$

where input vector $\mathbf{z} = [z_1 \dots z_{n_i}]^T$, center vector $\mathbf{c}_j = [c_{1j} \dots c_{n_ij}]^T$ and width matrix $\mathbf{\Sigma}_j = diag(\frac{1}{\sigma_{1j}^2} \dots \frac{1}{\sigma_{n_ij}^2})$ for Gaussian membership function of jth rule, TSK-type weight vector $w_j = k_{0j} + k_{1j}z_1 + \dots + k_{n_ij}z_{n_i}$ of jth rule, and k_{ij}s are real-valued parameters.

Eq. (12) can be represented in matrix form as

$$u_{G-FNN}(t) \quad = \quad \mathbf{\Phi}^T(t)\mathbf{w} \tag{13}$$

where regression vector $\mathbf{\Phi} = [\phi_1 \quad \phi_1 z_1 \dots \phi_1 z_{n_i} \dots\dots \phi_{n_r} \quad \phi_{n_r} z_1 \dots \phi_{n_r} z_{n_i}]^T$ and weight vector $\mathbf{w} = [k_{01} \quad k_{11} \dots k_{n_i 1} \dots\dots k_{0n_r} \quad k_{1n_r} \dots k_{n_i n_r}]^T$

3.2 G-FNN Learning Algorithm

In this paper, structure and parameters of the time series model is formed automatically and dynamically using G-FNN learning algorithm. The G-FNN learning algorithm is a sequential and hybrid (supervised/unsupervised) learning algorithm, which provides an efficient way to construct the G-FNN model online and combine structure and parameter learning simultaneously within the G-FNN. The structure learning includes determining the proper number of membership function n_r, i.e. the coarse of input fuzzy partitions and the finding of correct fuzzy logic rules. The learning of network parameters corresponds to the learning of the premise and consequent parameters of the G-FNN. Premise parameters include membership function parameters $\mathbf{c}_j$ and $\mathbf{\Sigma}_j$, and consequent parameter refers to the weight w_j of the G-FNN.

Parameter learning is performed by combining the semi-closed fuzzy set concept for the membership learning and the linear least square (LLS) method for weight learning. For structure learning, two criteria are proposed for fuzzy rule generation and an error reduction ratio (ERR) concept is proposed for rule pruning. Given the supervised training data, the proposed learning algorithm first decides whether or not to perform the fuzzy rule generation based on two proposed criteria. If structure learning is necessary, premise parameters of a new fuzzy rule will be obtained using semi-closed fuzzy set concept. The G-FNN will further decide whether there are redundant rules to be deleted based on the ERR, and it will also change the consequences of all the fuzzy rules properly. If no structure learning is necessary, the parameter learning will be performed to adjust the current premise and consequent parameters. This structure/parameter learning will be repeated for each online incoming training input-output pair. The flow chart of this learning

algorithm is shown in Figure 2.

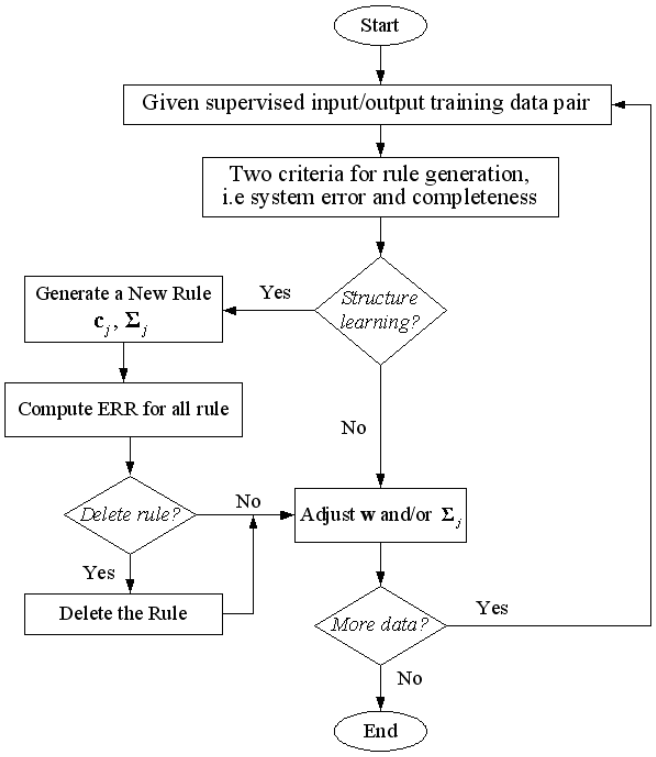

Figure 2: Flow chart of the G-FNN learning algorithm.

3.2.1 Structure Learning

Two Criteria of Fuzzy Rule Generation

The output error of G-FNN with respect to the teaching signal is an important factor in determining whether a new rule should be added. For each training data pair $[\mathbf{z}(t), u(t)] : t = 1 \dots n_d$, where $u(t)$ is the desired output or the supervised teaching signal and n_d is the total number of training data, first compute the G-FNN output $u_{G-FNN}(t)$ for the existing structure using (12). System error can then be defined as $e(t) = ||u(t) - u_{G-FNN}(t)||$. If $e(t)$ is bigger than a designed threshold K_e, a new fuzzy rule should be considered.

A fuzzy rule is a local representation over the region of input space. If a new pattern does not fall within the accommodation boundary of the existing rules, G-FNN will consider to generate a new rule and vice versa. For each observation $[\mathbf{z}(t), u(t)]$ at sample time t, calculate regularized Mahalanobis distance $md_j(t) = \sqrt{[\mathbf{z}(t) - \mathbf{c}_j]^T \mathbf{\Sigma}_j [\mathbf{z}(t) - \mathbf{c}_j]}, j = 1 \dots n_r$. Accommodation factor can then be defined as $d(t) = \min md_j(t)$. If $d(t)$ is bigger than $K_d = \sqrt{\ln \frac{1}{\epsilon}}$, a new rule should

be considered because the existing fuzzy system does not satisfy ϵ-completeness[12]. Otherwise, the new input data can be represented by the nearest existing rule.

Remark 3.1 An idea of hierarchical learning is adopted, where the thresholds K_e and K_d decay during the on-line learning as follows:

$$K_e = \begin{cases} e_{\max} & 1 < t < \frac{n_d}{3} \\ \max[e_{\max}(\frac{e_{\min}}{e_{\max}})^{\frac{3m}{n_d}-1}, \ e_{\min}] & \frac{n_d}{3} \leq t \leq \frac{2n_d}{3} \\ e_{\min} & \frac{2n_d}{3} \leq t \leq n_d \end{cases} \tag{14}$$

$$K_d = \begin{cases} d_{\max} = \sqrt{\ln \frac{1}{\epsilon_{\min}}} & 1 < t < \frac{n_d}{3} \\ \max[d_{\max}(\frac{d_{\min}}{d_{\max}})^{\frac{3m}{n_d}-1}, \ d_{\min}] & \frac{n_d}{3} \leq t \leq \frac{2n_d}{3} \\ d_{\min} = \sqrt{\ln \frac{1}{\epsilon_{\max}}} & \frac{2n_d}{3} \leq t \leq n_d \end{cases} \tag{15}$$

Pruning of Fuzzy Rules

In this learning algorithm, ERR concept proposed in [3] is utilized for rule and parameter sensitivity measurement. It is further adopted for rule pruning. At sample time t, total training data of the G-FNN are $(\mathbf{Z}, \mathbf{u})$ where $\mathbf{Z} = [\mathbf{z}(1) \ \ \mathbf{z}(2) \dots \mathbf{z}(t)]^T$ and $\mathbf{u} = [u(1) \ \ u(2) \dots u(t)]^T$. From (13), we have

$$\mathbf{u} = \mathbf{\Theta}\mathbf{w} + \mathbf{e} \tag{16}$$

and

$$\begin{aligned} \mathbf{\Theta} &= [\mathbf{\Phi}(1) \dots \mathbf{\Phi}(t)]^T \\ &= \begin{bmatrix} \phi_1(1) & \phi_1 z_1(1) \dots \phi_1 z_{n_i}(1) & \dots\dots & \phi_{n_r}(1) & \phi_{n_r} z_1(1) \dots \phi_{n_r} z_{n_i}(1) \\ & \vdots & \vdots & & \vdots \\ \phi_1(t) & \phi_1 z_1(t) \dots \phi_1 z_{n_i}(t) & \dots\dots & \phi_{n_r}(t) & \phi_{n_r} z_1(t) \dots \phi_{n_r} z_{n_i}(t) \end{bmatrix} \end{aligned}$$

where $\mathbf{u} \in \Re^t$ is the teaching signal, $\mathbf{w} \in \Re^v$ is real-valued weight vector, $\mathbf{\Theta} \in \Re^{t \times v}$ is known as the regressor, $\mathbf{e} = [e(1) \ \ e(2) \dots e(t)]^T \in \Re^t$ is system error vector that is assumed to be uncorrelated with the regressor $\mathbf{\Theta}$, and $v = n_r(n_i + 1)$.

For any matrix $\mathbf{\Theta}$, if its row number is larger than the column number, i.e. $t \geq v$, it can be

transformed into a set of orthogonal basis vectors by QR decomposition [13]. Thus

$$\mathbf{\Theta} = \mathbf{QR} \tag{17}$$

where $\mathbf{Q} = [\mathbf{q}_1 \;\; \mathbf{q}_2 \;\; \ldots \;\; \mathbf{q}_v] \in \Re^{t \times v}$ has orthogonal columns, and $\mathbf{R} \in \Re^{v \times v}$ is upper triangular matrix. Therefore, (16) can then be written as

$$\mathbf{u} = \mathbf{QRw} + \mathbf{e} = \mathbf{Qg} + \mathbf{e} \tag{18}$$

where $\mathbf{g} = [g_1 \;\; g_2 \;\; \ldots \;\; g_v]^T \in \Re^v$ and the LLS method gives

$$\begin{aligned} \mathbf{g} &= (\mathbf{Q}^T\mathbf{Q})^{-1}\mathbf{Q}^T\mathbf{u} \\ g_\gamma &= (\mathbf{q}_\gamma^T\mathbf{q}_\gamma)^{-1}\mathbf{q}_\gamma^T\mathbf{u} : \gamma = 1 \ldots v \end{aligned} \tag{19}$$

As $\mathbf{Q}$ has orthogonal columns, the energy function of $\mathbf{u}$ can be represented as [3]

$$\mathbf{u}^T\mathbf{u} = \sum_{\gamma=1}^{v} g_\gamma^2 \mathbf{q}_\gamma^T\mathbf{q}_\gamma + \mathbf{e}^T\mathbf{e} \tag{20}$$

If $\mathbf{u}$ is the desired output after its mean has been removed, the variance of $\mathbf{u}$ would be

$$t^{-1}\mathbf{u}^T\mathbf{u} = t^{-1}\sum_{\gamma=1}^{v} g_\gamma^2 \mathbf{q}_\gamma^T\mathbf{q}_\gamma + t^{-1}\mathbf{e}^T\mathbf{e} \tag{21}$$

where $t^{-1}\sum g_\gamma^2 \mathbf{q}_\gamma^T\mathbf{q}_\gamma$ is the variance that can be explained by the regressor $\mathbf{q}_\gamma$, while $t^{-1}\mathbf{e}^T\mathbf{e}$ is the unexplained variance of $\mathbf{u}$. Thus, $t^{-1}\sum g_\gamma^2 \mathbf{q}_\gamma^T\mathbf{q}_\gamma$ is the increment to the explained desired output variance introduced by $\mathbf{q}_\gamma$. Thus, an ERR due to $\mathbf{q}_\gamma$ can be defined as [3]

$$err_\gamma = \frac{g_\gamma^2 \mathbf{q}_\gamma^T\mathbf{q}_\gamma}{\mathbf{u}^T\mathbf{u}} \tag{22}$$

Substituting g_γ by (19), we have

$$err_\gamma = \frac{(\mathbf{q}_\gamma^T\mathbf{u})^2}{\mathbf{q}_\gamma^T\mathbf{q}_\gamma\mathbf{u}^T\mathbf{u}} \tag{23}$$

The ERR offers a simple and effective means of seeking a subset of significant regressors. The

term err_γ in (23) reflects the similarity of $\mathbf{q}_\gamma$ and $\mathbf{u}$ or the inner product of $\mathbf{q}_\gamma$ and $\mathbf{u}$. Larger err_γ implies that $\mathbf{q}_\gamma$ is more significant to the output $\mathbf{u}$.

The ERR matrix of the G-FNN is defined as

$$\mathbf{ERR} = \begin{bmatrix} err_1 & err_2 & \dots & err_{n_r} \\ err_{n_r+1} & err_{n_r+2} & \dots & err_{n_r+n_r} \\ \vdots & \vdots & \dots & \vdots \\ err_{n_i \times n_r+1} & err_{n_i \times n_r+2} & \dots & err_{n_i \times n_r+n_r} \end{bmatrix} \tag{24}$$

$$= \begin{bmatrix} \mathbf{err}_1 & \mathbf{err}_2 & \dots & \mathbf{err}_{n_r} \end{bmatrix} \tag{25}$$

We can further define the total ERR $Terr_j, j = 1 \dots n_r$ corresponding to the jth rule as

$$Terr_j = \sqrt{\frac{(\mathbf{err}_j)^T \mathbf{err}_j}{n_i + 1}} \tag{26}$$

If $Terr_j$ is smaller than a designed threshold $0 < K_{err}¡1$, the jth fuzzy rule should be deleted, and vice versa.

Remark 3.2 K_{err} plays a very important role on system performance, which directly affect rule generation and modeling error. Generally speaking, the bigger the value of K_{err}, the higher the chance of pruning a rule, and vice versa. If K_{err} is set too big, rule generation will be insufficient. Since the lease square method is used to determine the weight vectors of the G-FNN, insufficient rule generation will lead to big modeling error.

3.2.2 Parameter Learning

Determination of Premise Parameters

Premise parameters or fuzzy Gaussian membership functions of the G-FNN are allocated to be semi-closed fuzzy sets and therefore satisfies the ϵ-completeness of fuzzy rules. The following three cases are considered while determining the premise parameters:

1. $e(t) > K_e$ and $d(t) > K_d$

   First compute the Euclidean distance $ed_{ij_n}(t) = ||z_i(t) - b_{ij_n}||$ between $z_i(t)$ and the

boundary point $b_{ij_n} \in \{c_{i1}, c_{i2}, \ldots, c_{iN_r}, z_{i,\min}, z_{i,\max}\}$. Find

$$\tilde{j}_n = \arg\min ed_{ij_n}(t) \tag{27}$$

If $ed_{i\tilde{j}_n}(t)$ is less than a threshold or a dissimilarity ratio of neighboring membership function K_{mf}, we choose

$$c_{i(n_r+1)} = b_{i\tilde{j}_n}, \quad \sigma_{i(n_r+1)} = \sigma_{i\tilde{j}_n} \tag{28}$$

Otherwise, we choose

$$c_{i(n_r+1)} = z_i(t) \tag{29}$$

$$\sigma_{i(n_r+1)} = \frac{\max\left(|c_{i(n_r+1)} - c_{i(n_r+1)_a}|, |c_{i(n_r+1)} - c_{i(n_r+1)_b}|\right)}{\sqrt{\ln \frac{1}{\epsilon}}} \tag{30}$$

2. $e(t) > K_e$ but $d(t) \leq K_d$

$\mathbf{z}(t)$ can be clustered by the adjacent fuzzy rule, however, the rule is not significant enough to accommodate all the patterns covered by its ellipsoidal field. Therefore, the ellipsoidal field needs to be decreased to obtain a better local approximation. A simple method to reduce the Gaussian width is as follows

$$\sigma_{i\tilde{j}_{new}} = K_s \times \sigma_{i\tilde{j}_{old}} \tag{31}$$

where K_s is a reduction factor which depends on the sensitivity of the input variables. The sensitivity measure of the ith input variable in the jth fuzzy rule is denoted as s_{ij}, which is calculated using the ERR matrix in (24), i.e.

$$s_{ij} = \frac{err_{i \times n_r + j}}{\sum_{i=1}^{n_i} err_{i \times n_r + j}} \tag{32}$$

The threshold K_s could be designed with a minimum value of $K_{s,\min}$ when $s_{ij} = 0$, and a maximum value of 1 when s_{ij} is greater than or equal to the average sensitivity level of

the input variables in the jth rule, i.e.

$$K_s = \begin{cases} \frac{1}{1+\frac{(1-K_{s,\min})n_r{}^2}{K_{s,\min}}(s_{ij}-\frac{1}{n_i})^2} & s_{ij} < \frac{1}{n_i} \\ 1 & \frac{1}{n_i} \leq s_{ij} \end{cases} \tag{33}$$

3. $e(t) \leq K_e$ but $d(t) > K_d$ or $e(t) \leq K_e$ and $d(t) \leq K_d$

The system has good generalization and nothing need to be done except adjusting weight.

Remark 3.3 K_{mf} implies the minimum required similarity level of neighboring membership function. It is therefore recommended to set K_{mf} as 20% to 40% of the range value of the corresponding input variable. $K_{s,\min} \leq K_s \leq 1$ is used for the width adjustment, i.e. to increase the significance of the corresponding Gaussian membership function. Not to wipe out the major information carried by the Gaussian function, it is recommended to choose $0.8 \leq K_{s,\min} < 1$.

Determination of Consequent Parameters

As aforementioned, TSK-type consequent parameters are determined using LLS method. The LLS method is employed to find the weight vector $\mathbf{w}$ such that the error energy ($\mathbf{e}^T\mathbf{e}$) is minimized in (16) [13]. Furthermore, the LLS method provides a computationally simple but efficient procedure for determining the weight so that it can be computed very quickly and used for real-time control. The weight vector is calculated as

$$\mathbf{w} = \mathbf{\Theta}^{\dagger}\mathbf{u} \tag{34}$$

where $\mathbf{\Theta}^{\dagger}$ is the pseudoinverse of $\mathbf{\Theta}$

$$\mathbf{\Theta}^{\dagger} = (\mathbf{\Theta}^T\mathbf{\Theta})^{-1}\mathbf{\Theta}^T \tag{35}$$

4 Feedforward and Recurrent G-FNN Predictors

4.1 Feedforward G-FNN NAR(X) Predictors

With proper choice of their inputs, feedforward G-FNNs can be used to resemble NAR(n_y) or NARX(n_y,n_x) models for time series prediction (see Figure 3). A feedforward G-FNN is a nonlinear approximation to function $\mathbf{F}$ that is equivalent to optimal predictors in (7) and (8) as

follows

$$\begin{aligned}\hat{y}(t) &= \hat{\mathbf{F}}[y(t-1),\ldots,y(t-n_y)] \\ &= \mathbf{F}_{G-FNN}\{[y(t-1),\ldots,y(t-n_y)]^T\} \quad \text{(NAR model)} \quad (36)\end{aligned}$$

or

$$\begin{aligned}&= \hat{\mathbf{F}}[y(t-1),\ldots,y(t-n_y),x(t-1),\ldots,x(t-n_x)] \\ &= \mathbf{F}_{G-FNN}\{[y(t-1),\ldots,y(t-n_y),x(t-1),\ldots,x(t-n_x)]^T\} \quad \text{(NARX model)} (37)\end{aligned}$$

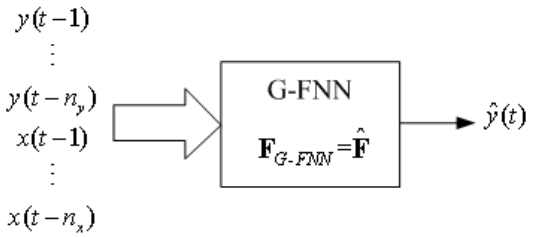

Figure 3: Feedforward G-FNN NARX.

As mentioned in Section 3.2, the structure and parameters, i.e. n_r, $\mathbf{c}_j$, $\mathbf{\Sigma}_j$ and w_j, of the feedforward G-FNN are estimated from n_d input-output training pairs using G-FNN learning algorithm, thereby obtaining an estimate $\hat{\mathbf{F}}$ of $\mathbf{F}$. Estimate are obtained by minimizing the sum of the squared residuals $\sum_{t=1}^{n_d}[y(t)-\hat{y}(t)]^2$. This is done by LLS method.

4.2 Recurrent G-FNN I NAR(X) Predictors

Unlike feedforward G-FNN, recurrent G-FNN I uses feedback signals while approximating NAR(X) models (see Figure 4). By imposing G-FNN learning algorithm on the feedback connections, the convergence can be guaranteed. To resemble NAR(X) models, output of recurrent G-FNN I is represented as

$$\begin{aligned}\hat{y}(t) &= \hat{\mathbf{F}}[\hat{y}(t-1),\ldots,\hat{y}(t-n_y)] \\ &= \mathbf{F}_{G-FNN}\{[\hat{y}(t-1),\ldots,\hat{y}(t-n_y)]^T\} \quad \text{(NAR model)} \quad (38)\end{aligned}$$

or

$$= \hat{\mathbf{F}}[\hat{y}(t-1),\ldots,\hat{y}(t-n_y),x(t-1),\ldots,x(t-n_x)]$$

$$= \mathbf{F}_{G-FNN}\{[\hat{y}(t-1),\ldots,\hat{y}(t-n_y),x(t-1),\ldots,x(t-n_x)]^T\} \quad \text{(NARX model)} \quad (39)$$

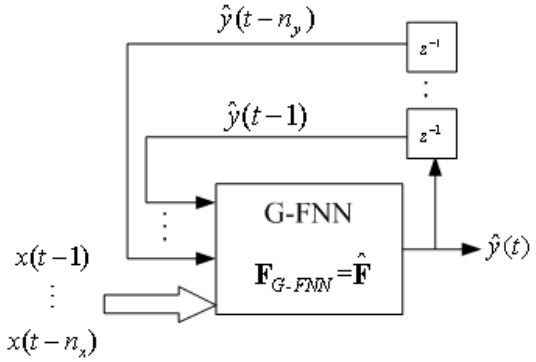

Figure 4: Recurrent G-FNN I NARX.

A recurrent G-FNN I NAR model is very close to a relaxation network, e.g. recurrent G-FNN I NAR(1) model is functionally equivalent to a SISO Hopfield network (a typical relaxation network). Relaxation networks start from a given state and settle to a fixed point, typically denoting a class [4]. Such behavior is generally considered more desirable for classification rather than modeling. In order to use the recurrent G-FNN I to model and predict a trajectory rather than reach a fixed point, it is advisable to use $NARX$ model while applying recurrent G-FNN I predictors.

4.3 Recurrent G-FNN II NARMA(X) Predictors

The recurrent G-FNN II presented in this section is G-FNN with feedback signals (see Figure 5) that can be used to estimate optimal NARMA(X) predictors in (9) and (10) as follows

$$\begin{aligned}
\hat{y}(t) &= \hat{\mathbf{F}}[y(t-1),\ldots,y(t-n_y),\hat{e}(t-1),\ldots,\hat{e}(t-n_e)] \\
&= \mathbf{F}_{G-FNN}\{[y(t-1),\ldots,y(t-n_y),\hat{e}(t-1),\ldots,\hat{e}(t-n_e)]^T\} \quad \text{(NARMA model)} \quad (40)
\end{aligned}$$

or

$$\begin{aligned}
&= \hat{\mathbf{F}}[y(t-1),\ldots,y(t-n_y),\hat{e}(t-1),\ldots,\hat{e}(t-n_e),x(t-1),\ldots,x(t-n_x)] \\
&= \mathbf{F}_{G-FNN}\{[y(t-1),\ldots,y(t-n_y),\hat{e}(t-1),\ldots,\hat{e}(t-n_e),x(t-1),\ldots,x(t-n_x)]^T\} \\
&\qquad \text{(NARMAX model)} \quad (41)
\end{aligned}$$

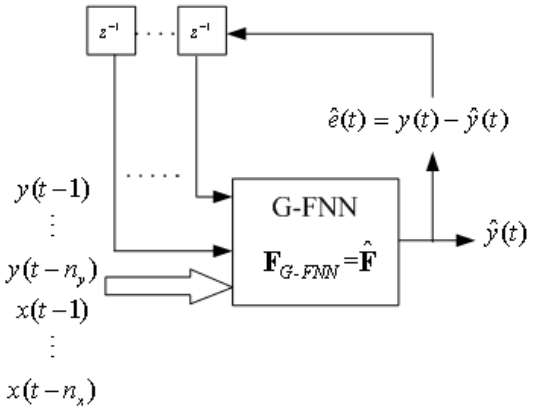

Figure 5: Recurrent G-FNN II NARMAX.

Similarly, topology of the recurrent G-FNN II is formed based on the G-FNN learning algorithm to minimize $\sum[y(t) - \hat{y}(t)]^2$.

5 Experiments and Comparisons

To demonstrate the validity of our proposed predictors, the feedforward G-FNN, recurrent G-FNN I and II are trained and tested on a NARX(2,1) dynamic process and chaotic Mackey-Glass time series. In addition, performance of the proposed strategies are compared with some recently developed fuzzy neural methods, including methods with only parameter learning such as adaptive network-based fuzzy inference system (ANFIS) [5] and hybrid neural fuzzy inference system (HyFIS) [6], and methods with both structure and parameter learnings such as RBF-based adaptive fuzzy system (RBF-AFS) [1], fuzzy neural network systems using similarity analysis (FNNS) [2], orthogonal least squares based RBF network (OLS-RBFN) [3], resource-allocating network (RAN) [8], resource-allocating network using orthogonal techniques (RANO) [10], growing neural gas network (GNGN) [9] and dynamic evolving neural-fuzzy inference system (DENFIS) [11] etc.

5.1 NARX(2,1) Dynamic Process

The following dynamic NARX(2,1) process is generated

$$y(t) = \mathbf{F}[y(t-1), y(t-2), x(t-1)] + e(t)$$

$$= \frac{y(t-1)y(t-2)[y(t-1)+2.5]}{1+y^2(t-1)+y^2(t-2)} + x(t-1) + e(t) \quad (42)$$

where the input has the form $x(t) = \sin(2\pi t/25)$. In this example, thresholds of the G-FNN learning algorithm are predefined in a systematic manner based on the desired accuracy level of the system (refer to Appendix), i.e. $e_{\max} = 0.1$, $e_{\min} = 0.02$, $d_{\max} = \sqrt{\ln(\frac{1}{0.5})}$, $d_{\min} = \sqrt{\ln(\frac{1}{0.8})}$, $K_{s,\min} = 0.9$, $K_{mf} = 0.5$ and $K_{err} = 0.002$. The topology of all feedforward and recurrent G-FNNs are estimated using the first 200 observations of a time series generated from NARX(2,1) model in (42). The feedforward and recurrent G-FNN predictors are then tested on the following 200 observations from the model.

Two experiments are performed to verify the capabilities of the proposed G-FNN predictors for both noise-free and noisy situations. The disturbance signal is set to be $N(0, 4e-2)$ white noise in noisy condition. The results are shown in Tables 1 and 2 respectively. The testing set error is the basis of model comparison. Training set error is indicative of whether the model has overfitted the data.

Table 1: G-FNN Performance on NARX(2,1) Time Series (noise-free)

Model	n_y	n_e	n_x	n_r	n_p[1]	Training MSE	Testing MSE
Feedforward G-FNN	1	0	0	2	8	2.1e-1	2.1e-1
	2	0	0	7	45	4.0e-4	3.9e-4
	1	0	1	6	40	2.8e-3	2.8e-3
	2	0	1	5	48	1.2e-4	1.1e-4
Recurrent G-FNN I	1	0	0	2	8	2.38	2.53
	2	0	0	10	56	1.35	1.95
	1	0	1	7	45	7.9e-3	8.0e-3
	2	0	1	5	46	1.5e-4	3.2e-4
Recurrent G-FNN II	1	1	0	3	17	10.2e-2	6.6e-2
	2	1	0	3	26	2.7e-2	2.8e-2
	1	1	1	7	60	2.3e-3	6.4e-3
	2	1	1	4	52	2.3e-4	4.5e-4

[1] n_p denotes number of parameters of the G-FNN and is calculated as $\sum_{i=1}^{n_i} n_{MF_i} \times 2 + n_r(n_i+1)$, where $n_i = n_y + n_e + n_x$ is total number of input variables of the G-FNN both feedforward and recurrent, n_{MF_i} is the number of Gaussian membership functions of the ith input variable.

A few observations and comments can be made based on the above two tables as follow:

- As closer to the real time series model, G-FNN prediction models with $n_y = 2$ provides better fitting results than those with $n_y = 1$. Similarly, NARX prediction models perform better than NAR prediction models.

Table 2: G-FNN Performance on NARX(2,1) Time Series (noisy)

Model	n_y	n_e	n_x	n_r	n_p	Training MSE	Testing MSE
Feedforward	1	0	1	7	47	7.0e-2	8.9e-2
G-FNN	2	0	1	5	48	4.7e-2	5.7e-2
Recurrent	1	0	1	6	40	7.1e-2	10.0e-2
G-FNN I	2	0	1	6	58	5.9e-2	9.4e-2
Recurrent	1	1	1	6	52	8.0e-2	11.0e-2
G-FNN II	2	1	1	5	63	5.2e-2	9.2e-2

- Using G-FNN learning algorithm, structure and parameters of NARMAX time series model are formed automatically and systematically. In particular, fuzzy rules of the G-FNN predictors are recruited or pruned dynamically according to their significance to the system performance and the complexity of the mapped system. Computational complexity of the G-FNN predictors does not increase exponentially with the increase in system dimensionality. In other words, the G-FNN predictors are able to generate a very parsimonious rule base.

- The recurrent G-FNN I NAR models perform poorly that verifies our analysis in Section 4.2. The recurrent NAR models aim to reach a fixed point rather than predict a continuous range of values.

- As aforementioned in Section 2.2.1, training MSE of the G-FNN NARX predictors in noisy condition should be at best equal to $4e-2$, i.e. variance of the white noise. This is verified because the G-FNNs are not able to model the noise, which is good from a prediction perspective. So in practice, it is highly recommended to use preprocessing techniques to filter out the noise from the signal before using G-FNN predictors, such as the LPF.

- Modeling and predicting NARX(n_y,n_x) time series, performance sequence of the proposed prediction models can be concluded as follows: feedforward NARX $>$ recurrent II NARX $\approx$ recurrent I NARX $>$ feedforward NAR $\approx$ recurrent II NAR $>$ recurrent I NAR, where $>$ denotes "performs better than" and $\approx$ denotes "is comparable with". Therefore, the recurrent G-FNN I and II have no much advantage over the feedforward G-FNN in both noise-free and noisy conditions.

To further demonstrate superior performance of the G-FNN predictors, comparisons with three other dynamic-topology fuzzy neural methods, i.e. RBF-AFS, FNNS and OLS-RBFN, are

shown in Table 3. In this comparison, noise-free signal and fitting model with $n_y = 2$ and $n_x = 1$ are used for all the methods. It can be seen that the G-FNN provides better generalization as well as greater parsimony. The proposed G-FNN predictors are fast in learning speed because no iterative learning loops are needed. Generating less number of rules and parameters leads to better computational efficiency using G-FNN learning algorithm.

Table 3: Performance Comparisons with FNNS, RBF-AFS and OLS-RBFN

Model	n_r	n_p	Testing MSE	Training Method
RBF-AFS	35	280	1.9e-2	iterative loops
FNNS	22	84	5.0e-3	300 iterative loops
OLS-RBFN	65	326	8.3e-4	one pass
Feedforward G-FNN	5	48	1.1e-4	one pass
Recurrent G-FNN I	5	46	3.2e-4	one pass
Recurrent G-FNN II	4	52	4.5e-4	one pass

5.2 Chaotic Mackey-Glass Time Series

Chaotic Mackey-Glass time series is a benchmark problem that has been considered by a number of researchers [1, 3, 5, 8, 10]. This time series is generated from the following equation

$$y(t) = (1-a)y(t-1) + \frac{by(t-\tau)}{1+y^{10}(t-\tau)} \tag{43}$$

where $\tau > 17$ gives chaotic behavior. Higher value of τ yields higher dimensional chaos. For the ease of comparison, parameters are selected as: $a = 0.1$, $b = 0.2$ and $\tau = 17$.

The fitting model of (43) is chosen to be

$$y(t) = \mathbf{F}[y(t-p), y(t-p-\triangle t), y(t-p-2\triangle t), y(t-p-3\triangle t)] \tag{44}$$

For simulation purpose, it is assumed that $y(t) = 0$, $\forall t < 0$ and $y(0) = 1.2$. Since shown in first experiment that the feedforward G-FNN provides advantages over the recurrent G-FNNs, only feedforward G-FNN is used in this experiment. In this example, thresholds of the G-FNN learning algorithm are predefined in a systematic manner based on the desired accuracy level of the system (refer to Appendix), i.e. $e_{\max} = 0.1$, $e_{\min} = 0.01$, $d_{\max} = \sqrt{\ln(\frac{1}{0.5})}$, $d_{\min} = \sqrt{\ln(\frac{1}{0.8})}$, $K_{s,\min} = 0.9$, $K_{mf} = 0.25$ and $K_{err} = 0.0005$.

In this experiment, it is chosen that $p = \triangle t = 6$ and $119 \leq t \leq 1140$. The first 500 in-

put/output data pairs of Mackey-Glass time series generated from (43) are used for training the feedforward G-FNN while the following 500 data pairs are used for validating the identified model. Using G-FNN learning algorithm, totally 8 fuzzy rules are generated for the feedforward G-FNN predictor during training as shown in Figure 6(a). The corresponding Gaussian membership functions are illustrated with respect to the input variables as Figure 7. It can be seen that the membership functions are evenly distributed over the training interval. This is in line with the aspiration of "local representation" in fuzzy logic. The root mean square error (RMSE) is able to converge very quickly during training as illustrated in Figure 6(b). As the G-FNN algorithm uses one-pass learning method to avoid iterative learning loops, fast learning speed can be achieved. Figure 8 shows that the actual and predicted values for both training and testing data are essentially the same and their differences can only be seen on a finer scale.

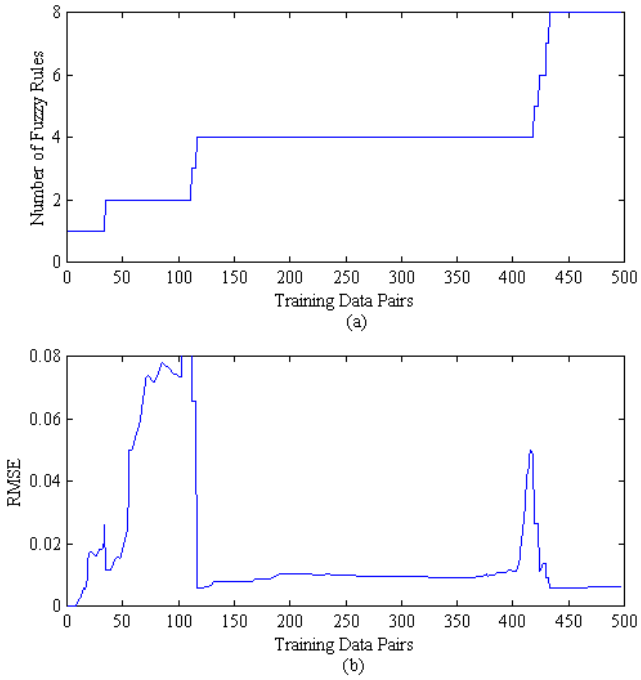

Figure 6: Training performances: (a) Fuzzy rule generation; (b) RMSE during training process.

Performance comparisons of the feedforward G-FNN with RBF-AFS, OLS-RBFN, HyFIS and ANFIS are shown in Table 4. All the methods in this comparison provide structure learning except for HyFIS and ANFIS which only provide parameter learning. Similar to example 1, feedforward G-FNN is shown to be better performed than RBF-AFS and OLS. G-FNN predictor has bigger training and testing RMSEs compared with ANFIS while compacter rule base. ANFIS

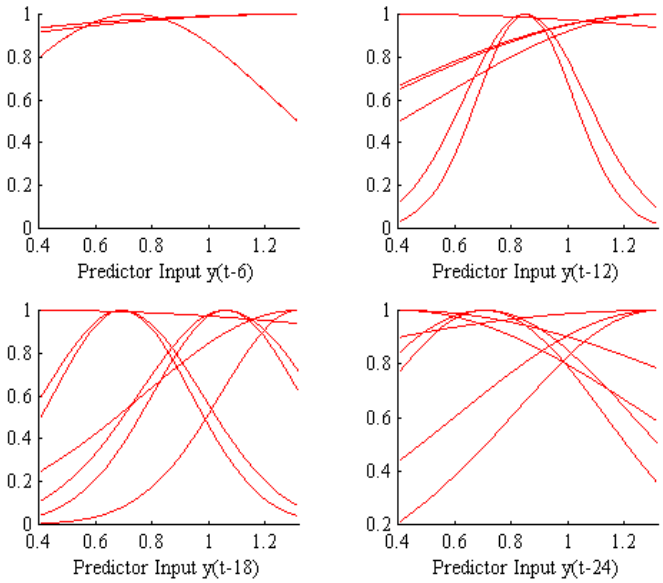

Figure 7: Gaussian membership functions w.r.t input variables.

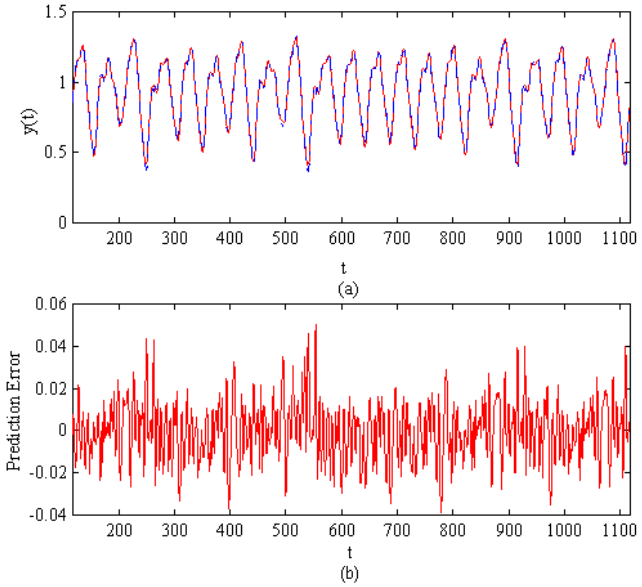

Figure 8: Prediction results: (a) Mackey-Glass time series from $t = 118$ to 1140 and six-step ahead prediction; (b) Prediction error.

is trained with hundreds of iterative learning loops that may lead to global optimal solution, whereas feedforward G-FNN in this case only acquires sub-optimal solution. Another comparison is performed for feedforward G-FNN with some existing one-pass also known as on-line learning models, i.e. RAN, RANO, GNGN, and DENFIS, applied on the same task (see Table 5). Feedforward G-FNN predictor outperforms the rest methods in terms of predictive accuracy as well as computational efficency.

Table 4: Performance Comparisons with RBF-AFS, OLS-RBFN, HyFIS and ANFIS

Model	n_r	n_p	Training RMSE	Testing RMSE[1]	Training Method
RBF-AFS	21	210	0.0107	0.0128	iterative loops
OLS-RBFN	35	211	0.0087	0.0089	one pass
HyFIS	16	78	0.0021	0.0100	500 iterative loops
ANFIS	16	104	0.0016	0.0030	500 iterative loops
Feedforward G-FNN	8	86	0.0062	0.0052	one pass

[1]Testing RMSEs are calculated based on the identified models obtained from the first 500 training pairs by each method. No further tuning of parameters is applied while testing the identified models.

Table 5: Performance Comparisons with RAN, RANO, GNGN and DENFIS

Model	n_r	Testing NDEI[1]
RAN	40	0.1642
RANO	18	0.1492
GNGN	1000	0.062
DENFIS	883	0.042
Feedforward G-FNN	8	0.0226

[1]Nondimensional error index (NDEI), also known as normalized RMSE, is defined as the RMSE divided by the standard deviation of the target time series.

6 Conclusions

In this paper, one feedforward and two recurrent G-FNN predictors are proposed, tested and compared. The proposed G-FNN predictors provide a sequential and hybrid learning method for model structure determination and parameter identification which greatly improves predictive performance. Experiments and comparisons demonstrate the superior performance of the proposed approaches over some well-known existing methods. Further study is necessary to investigate the robustness of the proposed methods.

References

[1] K. B. Cho and B. H. Wang, "Radial Basis Function Based Adaptive Fuzzy Systems and their Applications to System Identification and Prediction," *Fuzzy Sets and Systems*, Vol. 83, pp.325-339, 1996.

[2] C. T. Chao, Y. J. Chen and C. C. Teng, "Simplification of Fuzzy-Neural Systems Using Similarity Analysis," *IEEE Trans. System, Man and Cybernetic*, Vol. 26, pp.344-354, 1996.

[3] S. Chen, C. F. N. Cowan and P. M. Grant, "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Network," *IEEE Trans. Neural Networks*, Vol. 2, No. 2, pp. 302-309, 1991.

[4] J. T. Connor, R. D. Martin and L. E. Atlas, "Recurrent Neural Networks and Robust Time Series Prediction," *IEEE Trans. Neural Networks*, Vol. 5, No. 2, pp. 240-254, 1994.

[5] J. S. R. Jang, "ANFIS: Adaptive-Network-Based Fuzzy Inference System," *IEEE Trans. System, Man and Cybernetic*, Vol. 23, pp. 665-684, 1993.

[6] J. Kim and N. Kasabov, "HyFIS: Adaptive Neuro-Fuzzy Inference Systems and Their Application to Nonlinear Dynamical Systems," *Neural Networks*, Vol. 12, pp. 1301-1319, 1999.

[7] B. Fritzke, "A Growing Neural Gas Network Learns Topologies," *Adv. Neural Inform. Processing Syst.*, Vol. 7, 1995.

[8] J. Platt, "A Resource-Allocating Network for Function Interpolation," *Neural Computation*, Vol. 3, pp. 213-225, 1991.

[9] M. Salmeron, J. Ortega, C. G. Puntonet and A. Prieto, "Improved RAN Sequential Prediction Using Orthogonal Techniques," *Neurocomputing*, Vol. 41, pp. 153-172, 2001.

[10] N. K. Kasabov and Q. Song, "DENFIS: Dynamic Evolving Neural-Fuzzy Inference System and Its Application for Time-Series Prediction," *IEEE Trans. Fuzzy Systems*, Vol. 10, No. 2, pp. 144-154, 2002.

[11] L. X. Wang, *A Course in Fuzzy Systems and Control*, New Jersey: Prentice Hall, 1997.

[12] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, 1992.

[13] A. Jazwinski, Stochastic Processes and Filtering Theory, New York: Academic Press, 1970.