World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:4, No:7, 2010

Computational Intelligence Hybrid Learning Approach to Time Series Forecasting

Chunshien Li, Jhao-Wun Hu, Tai-Wei Chiang and Tsunghan Wu

***Abstract*—**Time series forecasting is an important and widely popular topic in the research of system modeling. This paper describes how to use the hybrid PSO-RLSE neuro-fuzzy learning approach to the problem of time series forecasting. The PSO algorithm is used to update the premise parameters of the proposed prediction system, and the RLSE is used to update the consequence parameters. Thanks to the hybrid learning (HL) approach for the neuro-fuzzy system, the prediction performance is excellent and the speed of learning convergence is much faster than other compared approaches. In the experiments, we use the well-known Mackey-Glass chaos time series. According to the experimental results, the prediction performance and accuracy in time series forecasting by the proposed approach is much better than other compared approaches, as shown in Table IV. Excellent prediction performance by the proposed approach has been observed.

***Keywords*—**forecasting, hybrid learning (HL), Neuro-Fuzzy System (NFS), particle swarm optimization (PSO), recursive least-squares estimator (RLSE), time series

I. INTRODUCTION

SYSTEM modeling for function approximation, optimization and forecasting has been widely investigated for years. Time series forecasting is one of the momentous applications in system modeling. In time series, time is usually a very important factor to make decision or prediction. Data in past history recorded in time sequence is called time series. Managers usually use historical data to forecast various types of variables such as changes in stock, sales of products, population growth, and many others. The accurate and valuable prediction of these variables can assist managers to make a decision. Time series is also a group of statistics, according to the order which events are occurred in time sequence. For instance, the daily average temperature or monthly rainfall at a place, daily stock market closing price, company's turnover, unemployment rate, economic growth rate and total amount of national income and export. Modern business and economic activities, in essence, are dynamic, and they changes frequently. How to make a reliable forecast is one of the most important issues for modern enterprises and organizations. The usage of time series to forecast the future tendency has to make use of detailed data which were generated for some time past in the cause of understanding the trend of changes.

Many previous researches had used many kinds of methods in time series forecasting [8]-[16]. Neuro-fuzzy system (NFS) is one of the most frequently used methods. Many kinds of optimization algorithms had been used for NFSs. The design of NFS to forecasting had been also proposed in previous researches. However, the forecasting performance and accuracy were not good enough.

Authors are with Department of Information Management .National Central University.Taiwan, ROC. e-mail: jamesli@mgt.ncu.edu.tw

We propose neuro- fuzzy system with hybrid learning (HL) approach in this research to make prediction for time series as accurate as possible. We use few past data as inputs to the neuro-fuzzy predictor to forecast one-step ahead result.

NFSs are useful to represent and process linguistic information and to deal with uncertainty and imprecision [6]. In our study, we use Takagi-Sugeno (T-S) neuro-fuzzy model [6]-[7] for time series forecasting. The reasoning process in T-S neuro-fuzzy model is very similar to that in a traditional fuzzy inference system. An appropriate neuro-fuzzy model for prediction is very important to forecast time series accurately. How to design and adjust fuzzy sets and fuzzy rules in a NFS is critically significant on forecasting results. Furthermore, because there are usually many unknown parameters in a NFS, the selection of learning algorithm to adapt the neuro-fuzzy predictor plays an extremely pivotal position. In order to adapt the free parameters of the proposed neuro-fuzzy predictor, we propose a hybrid learning approach using both the particle swarm optimization (PSO) [4] and the recursive least-squares estimator (RLSE) [6] to solve the problem. The PSO is used to update premise parameters of the NFS predictor, and the RLSE is used to update the consequent parameters. In this paper, we specify the proposed hybrid learning approach for NFS to the problem of time series forecasting. The main goal of the research is to find the optimal solution for the neuro-fuzzy predictor so that the prediction performance by the proposed approach can be as good as possible. The PSO is a useful method for solving global optimization problems, and the RLSE can solve linear model question in very efficient way. The hybrid learning approach for the NFS predictor can make forecasting result with great accuracy.

In Section II, the methodology of NFS is specified. In Section III, we describe the hybrid learning approach with PSO and RLSE to the proposed neuro-fuzzy predictor. In Section IV, the Mackey-Glass chaos time series is used as an illustrated example to demonstrate the proposed neuro-fuzzy prediction approach. Finally, a discussion for the experimental results is given and the paper is concluded.

II. NEURO-FUZZY SYSTEM AS A PREDICTOR

To design a multiple-input-single-output neuro-fuzzy system as a predictor, we use T-S fuzzy model. T-S fuzzy model was first proposed by Takagi and Sugeno to develop a systematic approach to generate fuzzy rules from a given input-output data set [5]-[7].

In the first-order T-S model, the consequent of a fuzzy rule is a linear combination of crisp inputs. Let us consider a *M*-input-one-output fuzzy system with *K* fuzzy rules in the rule base. The form of a T-S fuzzy rule can be given as follows.

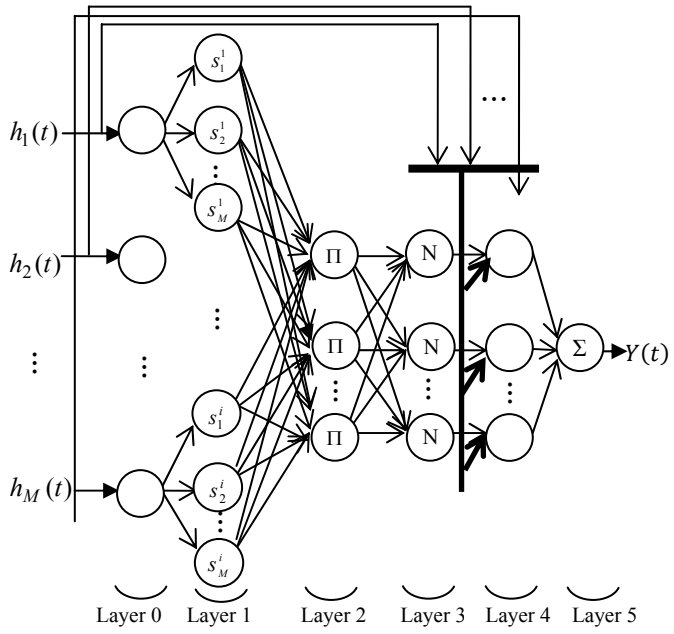

Fig. 1. Structure of neuro-fuzzy system.

Rule i: IF x_1 is $s_1^i(h_1(t))$ and x_2 is $s_2^i(h_2(t))$ ⋯ and x_M is $s_M^i(h_M(t))$

Then $y^i(t) = a_0^i + a_1^i h_1(t) + \cdots + a_M^i h_M(t)$

$i = 1,2,\cdots,K$ (1)

where $\{h_1(t), h_2(t), \ldots, h_M(t)\}$ are the inputs to the neuro-fuzzy system at time t ; $y^i(t)$ is the i-th rule output, $\{s_1^i, s_2^i, \cdots s_M^i\}$ are the fuzzy sets of the i-th rule, and $\{a_1^i, a_2^i, \cdots a_M^i\}$ are the consequent parameters of the i-th rule. The fuzzy model can be cast into neural structure to be a neuro-fuzzy system (NFS). The proposed approach is based on the NFS methodology. In the study, input data are from historical data in time series.

The NFS structure is shown in Fig. 1. There are six layers in the proposed neuro-fuzzy system. The explanation for the six layers is specified as follows.

Layer 0: This layer is called the input layer. Each node in this layer corresponds to an input crisp variable $H(t) = [h_1(t), h_2(t), \ldots, h_M(t)]$.

Layer 1: The layer is called the fuzzy-set layer. Each node of the layer represents a linguistic value characterized by a fuzzy set. Each node output indicates a membership degree. To design fuzzy sets, we use Gaussian membership functions. The general description of fuzzy sets using Gaussian membership function is given as follows.

$$\text{gaussmf}(h) = \exp(-0.5((h-m)/\sigma)^2) \quad (2)$$

where m and σ are the mean and the spread of the Gaussian-type fuzzy set. The parameters of mean and spread for all the fuzzy sets in the premise-parts of the proposed NFS are called the premise parameters.

Layer 2: The layer is for the firing strengths of the K if-then rules. The nodes perform the *fuzzy-and* operations for the premise parts of the fuzzy rules. Each node output indicates a firing strength of a corresponding fuzzy rule. The firing strength for the i-th rule is denoted as β^i, and it is defined as.

$$\beta^i(t) = \text{product}(s_1^i(h_1(t)), s_2^i(h_2(t)), \cdots, s_M^i(h_M(t))) \quad (3)$$

Layer 3: The process of normalization for the firing strengths of the fuzzy rules is performed in this layer. The normalized firing strength for the i-th fuzzy rule is given as follows.

$$r^i(t) = \frac{\beta^i(t)}{\sum_{i=1}^{K} \beta^i(t)} \quad (4)$$

Layer 4: The layer is called the consequent layer. The nodes in the layer perform the normalized consequents of all the fuzzy rules. Then the output of the i-th fuzzy rule is given as follows.

$$r^i(t)y^i(t) = r^i(t)(a_0^i + a_1^i h_1(t) + \cdots a_M^i h_M(t)) \quad (5)$$

where $\{a_1^i, a_2^i, \cdots a_M^i\}$ are the consequent parameters of the i-th fuzzy rule.

Layer 5: The layer is called the output layer. There is only one node in the layer for single output. The node in this layer combines all the outputs from Layer 4 to produce the system output, given as follows.

$$Y(t) = \sum_{i=1}^{K} r^i(t)y^i(t) = \frac{\sum_{i=1}^{K} \beta^i(t)y^i(t)}{\sum_{i=1}^{K} \beta^i(t)} \quad (6)$$

Once the consequent parameters are determined, the system output of the NFS can be expressed as follows.

$$Y(t) = \sum_{i=1}^{K} r^i(t) \times (a_0^i + a_1^i h_1(t) + \cdots + a_M^i h_M(t)) \quad (7)$$
$$t = 1,2,\cdots,N$$

Assume that there are N samples observed from an unknown system of interest. The observed samples are collected to be used as training data for the proposed neuro-fuzzy predictor. The training data (TD) is denoted as follows.

$$\text{TD} = \{d(t), t = 1,2,\cdots,N\} \quad (8)$$

To overview the approach structure for the prediction of the output of an unknown system, the approach diagram for prediction is given in Fig. 2.

In Fig.2, the input vector $\boldsymbol{H}(t)$ is obtained from the TD in (8) which are observed from the unknown system. The purpose of the neuro-fuzzy predictor is to generate an

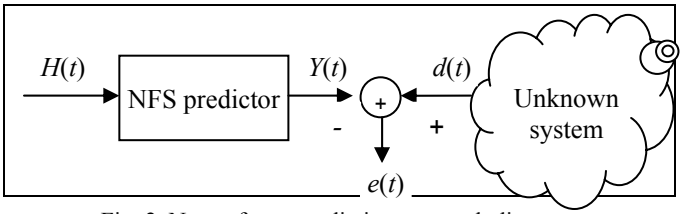

Fig. 2. Neuro-fuzzy prediction approach diagram.

output, $Y(t)$, and the error between the unknown system output and the NFS output is minimized. The prediction error is defined as follows.

$$e(t) = d(t) - Y(t), t = 1,2,\cdots,N \tag{9}$$

With the N pairs of input-output training patterns, the concept of root mean square error (RMSE) is used as error norm, given as follows.

$$\text{RMSE} = \sqrt{\frac{\sum_{t=1}^{N} e(t)^2}{N}} \tag{10}$$

The premise parameters in Layer 1 and the consequent parameters in Layer 4 can be viewed as the two subsets of the free parameters for the proposed NFS. The parameters are sometimes called the system parameters for the NFS. They can be adapted using machine learning algorithms such as PSO and RLSE. In the following section, we specify the PSO-RLSE hybrid learning method for the neuro-fuzzy predictor.

III. HYBRID PSO-RLSE LEARNING APPROACH

A. Particle Swarm Optimization

PSO was proposed by Eberhart and Kennedy in the mid 1990s [4]. PSO is an excellent approach having collective wisdom concept to the evolution of the optimization search in research areas. PSO has been applied successfully to a wide variety of search and optimization problems [1]-[3]. According to the PSO algorithm, a particle is viewed as a bird searching for food (a better solution), and all the particles comprise a population, which is called a "swarm" of the birds. Each particle moves toward its own best position and the swarm-best position. All particles competes each other to be the swarm best. In this way, PSO combines the search behaviors of both individual search and swarm search. There exists a subtle relationship for competition and cooperation in the search process. All particles shall remember their own best positions during the searching process. Besides, all particles have their own velocities in order to determine their search movements.

Particles search iteratively for the optimum solution using the concept of fitness function. A fitness function sometimes is called a cost function. Particles change their search direction by means of two search memories, which are given as follows.

Pbest : the best location of individual particle, and

Gbest : the best location of the swarm.

A search process by PSO is shown conceptually in Fig. 3.

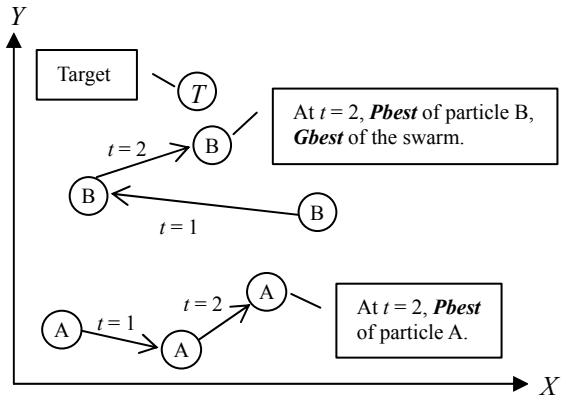

Fig. 3. Search process by PSO.

In Fig. 3, the target T is the location of the optimum solution. In the swarm, there are two particles, which are labeled with A and B, respectively. Both particles are updated for their locations at time t. When $t = 2$, it is obvious that particle B is the closest particle to the target. This means that particle B is the ***Gbest*** of the swarm.

Assume the problem space is with Q dimensions. The particles' velocities are updated as follows.

$$\begin{aligned} \boldsymbol{V}_i(t+1) = \boldsymbol{V}_i(t) &+ c_1 \times \xi_1 \times (\boldsymbol{Pbest}_i(t) - \boldsymbol{L}_i(t)) \\ &+ c_2 \times \xi_2 \times (\boldsymbol{Gbest}_i(t) - \boldsymbol{L}_i(t)) \end{aligned} \tag{11}$$

$$\boldsymbol{V}_i(t) = [v_{i,1}(t), v_{i,2}(t), \cdots, v_{i,Q}(t)] \tag{12}$$

where $\boldsymbol{V}_i(t)$ is the velocity of i-th particle at time t, $\{c_1, c_2\}$ are the parameters for PSO, and $\{\xi_1, \xi_2\}$ are random numbers in [0, 1].

The particles locations are updated as follows.

$$\boldsymbol{L}_i(t+1) = \boldsymbol{L}_i(t) + \boldsymbol{V}_i(t) \tag{13}$$

$$\boldsymbol{L}_i(t) = [l_{i,1}(t), l_{i,2}(t), \cdots, l_{i,Q}(t)] \tag{14}$$

where $\boldsymbol{L}_i(t)$ is the location of the i-th particle at time t.

The procedure of PSO is given in Fig. 4.

In Fig. 4, $f(\boldsymbol{L}_i)$ indicates the cost in RMSE for the current location of the i-th particle, $f(\boldsymbol{Pbest}_i)$ indicates the cost for $\boldsymbol{Pbest}_i$, and $f(\boldsymbol{Gbest})$ represents the cost for ***Gbest***.

Step 1.	Decided the swarm size, dimensions of particles, and the maximal number of iterations.
Step 2.	Initialize the positions and the velocities of the particles for the swarm randomly.
Step 3.	Calculate fitness for each particle according to fitness function.
Step 4.	Update position and velocity for each particle in the swarm according to (11) and (13).
Step 5.	If $f(\boldsymbol{L}_i) < f(\boldsymbol{Pbest}_i)$, then update the $\boldsymbol{Pbest}_i$ of the swarm. If $f(\boldsymbol{Pbest}_i) < f(\boldsymbol{Gbest})$, update the ***Gbest*** of the swarm.
Step 6.	If stopping criteria satisfied, ***Gbest*** is the optimum solution. Otherwise, go back to **Step 3** to continue the PSO procedure.

Fig. 4. Implementation procedure of PSO.

B. Recursive Least-Squares Estimator

In general, the least square estimation (LSE) problem [6] can be specified below. The output of a linear model y is given as follows.

$$y = \theta_1 f_1(u) + \theta_2 f_2(u) + \cdots + \theta_n f_n(u) \tag{15}$$

where u is the model's input, $f_i(.)$ is known function of u, and θ_i , $i=1,2,\ldots, n$ represents unknown parameters to be estimated. The problem of LSE can be written in a concise form, as follows.

$$\boldsymbol{A\theta} = \boldsymbol{y} \tag{16}$$

where $\boldsymbol{A}$ is a input matrix, $\boldsymbol{\theta}$ is a parameter vector to be estimate and $\boldsymbol{y}$ is an output vector.

The recursive least-squares estimator (RLSE) for the problem of (16), where the k-th row of $[\boldsymbol{A}, \boldsymbol{y}]$, denoted by $[\boldsymbol{b}_k^T, \boldsymbol{y}_k]$, is sequentially obtained, can be calculated as follows.

$$\boldsymbol{P}_{k+1} = \boldsymbol{P}_k - \frac{\boldsymbol{P}_k \boldsymbol{b}_{k+1} \boldsymbol{b}_{k+1}^T \boldsymbol{P}_K}{1 + \boldsymbol{b}_{k+1}^T \boldsymbol{P}_k \boldsymbol{b}_{k+1}} \tag{17a}$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \boldsymbol{P}_{k+1} \boldsymbol{b}_{k+1} (\boldsymbol{y}_{k+1} - \boldsymbol{b}_{k+1}^T \boldsymbol{\theta}_k) \tag{17b}$$

where k ranges from 0 to N -1 and the final RLSE $\hat{\boldsymbol{\theta}}$ is equal to $\boldsymbol{\theta}_N$, the estimator using all N data pairs. There are N training data in (8) to be involved for the adaption of the neuro-fuzzy predictor.

To start the algorithm in (17), we need to select the initial values of $\boldsymbol{\theta}_0$ and $\boldsymbol{P}_0$ which is given as follows.

$$\boldsymbol{P}_0 = \alpha \boldsymbol{I} \tag{18}$$

where α is a large value and $\boldsymbol{I}$ is the identity matrix. $\boldsymbol{\theta}_0$ can be initially set to zeros.

C. Hybrid learning algorithm

To train the proposed hybrid-learning-based neuro-fuzzy system (HL-NFS) for time series forecasting, the PSO in (11) and (13) is used together with the RLSE in (17) for fast convergence of learning. The PSO is used to update the premise parameters and the RLSE is used to update the consequence parameters. Fig. 5 shows the implementation flowchart. The training procedure for HL-NFS is shown as follows.

Step 1. Collect sample data. The first half of data is used for training, and the other half is for testing.

Step 2. Update the premise parameters by the PSO in (11) and (13).

Step 3. Update the consequent parameters by the RLSE in (17). According to (17), we have to design the vector $\boldsymbol{b}_{k+1}$ as follows.

$$\boldsymbol{b}_{k+1} = [\boldsymbol{bb}^1(k+1) \quad \boldsymbol{bb}^2(k+1) \cdots \boldsymbol{bb}^K(k+1)] \tag{19a}$$

$$\boldsymbol{bb}^i(k+1) = [\gamma^i \quad h_1(k+1)\gamma^i \cdots h_M(k+1)\gamma^i] \tag{19b}$$

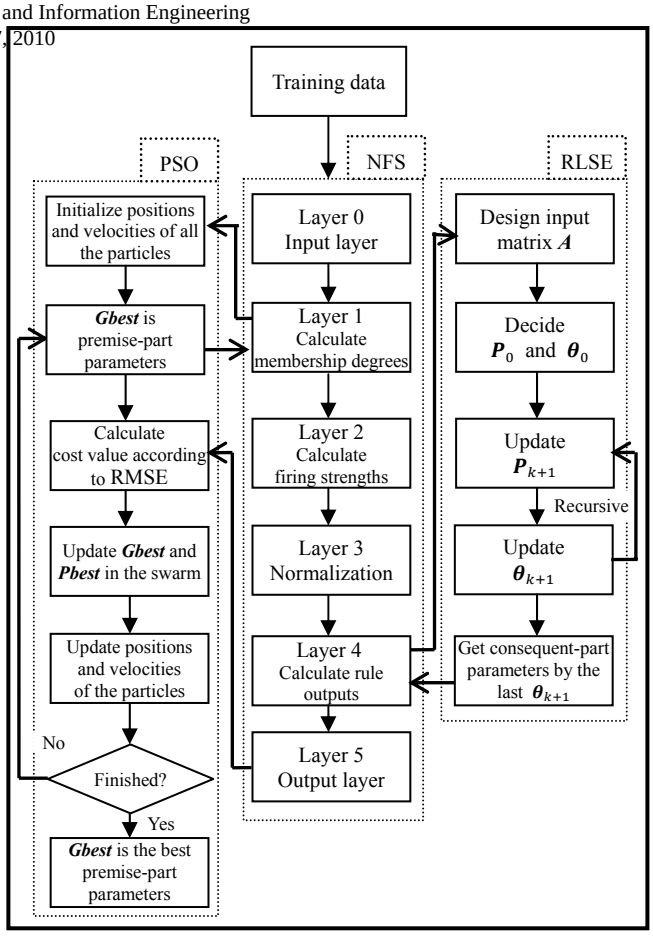

Fig. 5 Flowchart for the training process of the proposed approach.

$$k = 0,1,\cdots, N-1$$

$$i = 1,2,\cdots K$$

The parameter vector $\boldsymbol{\theta}$ is given as follows.

$$\boldsymbol{\theta} = [\boldsymbol{\omega}^1 \quad \boldsymbol{\omega}^2 \cdots \boldsymbol{\omega}^K]^T \tag{20a}$$

$$\boldsymbol{\omega}^i = [a_0^i \quad a_1^i \quad \cdots \quad a_M^K]^T \tag{20b}$$

$$i = 1,2,\cdots K$$

Step 4. After all parameters of the NFS predictor are determined, calculate system output in (7).

Step 5. Return to the PSO procedure in Fig. 4. Calculate cost in (10) for each particle.

Step 6. Compare the costs of all particles. Update ***Pbest*** and ***Gbest*** in the swarm.

Step 7. If stopping criteria satisfied, ***Gbest*** is the optimum premise parameters for the NFS. Otherwise, go back to **Step 2** to continue the procedure.

D. Testing procedure of HL-NFS

In order to verify and test the accuracy and prediction performance of the HL-NFS after learning, the remaining half of sample data is used for testing. The testing procedure is given as follows.

Step 1. After the training procedure, the premise parameters have been determined by PSO and the consequent parameters have been updated by

RLSE.

Step 2. Testing data are used as input to the trained HL-NFS predictor.

Step 3. Generate prediction output in (7).

Step 4. The output of the HL-NFS predictor is compared to the corresponding target to produce prediction error in (9).

IV. EXPERIMENTS FOR THE APPROACH

A. Settings of system and algorithm implementation

In the experiments, the well-known Mackey-Glass chaos time series is used and it is defined as follows.

$$\dot{x}(t) = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{21}$$

where τ= 17. The time step is given as 0.1 s. The initial condition is given as $x(0) = 1.2$ and $x(t) = 0$ for $t < 0$. The mapping relationship of the HL-NFS to predict the chaos time series in (21) is given as follows:

$$x(t+\overline{P}) = f(x(t), x(t-\Delta), \cdots, x(t-(\overline{D}-1)\Delta)) \tag{22}$$

where t is the time index, $\overline{P} = \Delta = 6$, and $\overline{D} = 4$. From the Mackey-Glass time series $x(t)$,the data pairs ($H(t)$, $d(t)$) extracted from the series are given as follows.

$$H(t) = [x(t-18), x(t-12), x(t-6), x(t)] \tag{23}$$

$$d(t) = x(t+6) \tag{24}$$

for t = 118 to 1117. The objective of the model is to exploit the obtained samples to predict the output of the time series in the future. There are 1000 data pairs generated. The first 500 data pairs are used for system training and the remaining 500 data pairs are used for testing. For the NFS predictor design, each input variable has two fuzzy sets, and there are 2^4 = 16 fuzzy rules in the neuro-fuzzy system, in which there are 16 premise parameters and 80 consequent parameters. All of the fuzzy sets use the form of Gaussian membership function in (2), which has two free parameters. The proposed HL-NFS approach is compared to the NFS using the PSO alone (denoted as NFS-PSO). The settings of the PSO-RLSE hybrid learning method for the proposed HL-NFS are given in Table I. And the settings of the PSO for the NFS-PSO approach are given in Table II.

TABLE I
SETTINGS OF PSO-RLSE FOR HL-NFS

PSO	
Dimensions of particle	16
Swarm size	100
Initialization of particle position	Random in [0, 1]
Initialization of particle velocity	Random in [0, 1]
Learning rate (c_1,c_2)	2
Maximum iterations	1000
RLSE	
Number of consequent-part parameters	80
$\boldsymbol{\theta}_0$	80×1 zero vector
$\boldsymbol{P}_0$	$\boldsymbol{P}_0 = \alpha \boldsymbol{I}$
α	10^8
$\boldsymbol{I}$	80×80 identity matrix

TABLE II
SETTINGS FOR PSO IN NFS

Dimensions of particle	96
Swarm size	100
Initialization of particle position	Random in [0, 1]
Initialization of particle velocity	Random in [0, 1]
Learning rate (c_1,c_2)	2
Maximum iterations	1000

B. Experimental results

After 1000 training iterations, the parameters of the proposed HL-NFS are shown in Table III. The performance by the proposed HL-NFS and the NFS-PSO are shown in Table IV. The prediction performance of the HL-NFS learning is 0.0014 in RMSE and the RMSE for testing is 0.0013. In the NFS using the PSO alone (NFS-PSO), there are 96 unknown parameters which are determined by the PSO. And the performances for training and testing are 0.0113 and 0.0113 in RMSE, respectively. The learning curves are shown in Figs. 6 and 7. The training and testing results by the NFS-PSO and the proposed HL-NFS are shown in Figs. 8 to 15. The blue line represents the target, and the red and dotted line represents the prediction output by the proposed HL-NFS and the compared NFS-PSO approach. Figs. 16 and 17 show the fuzzy sets after learning. The performance comparison to other works is given in Table IV.

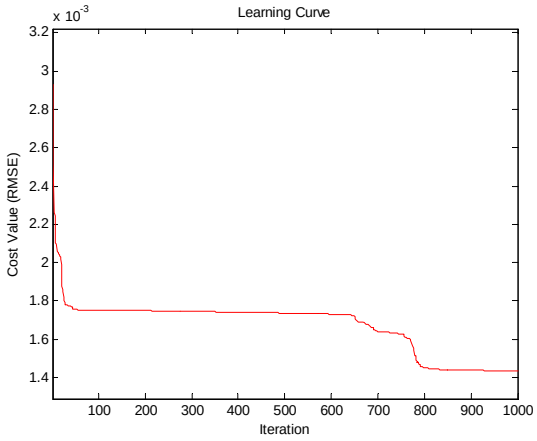

Fig. 6 Learning curve by the proposed HL-NFS approach.

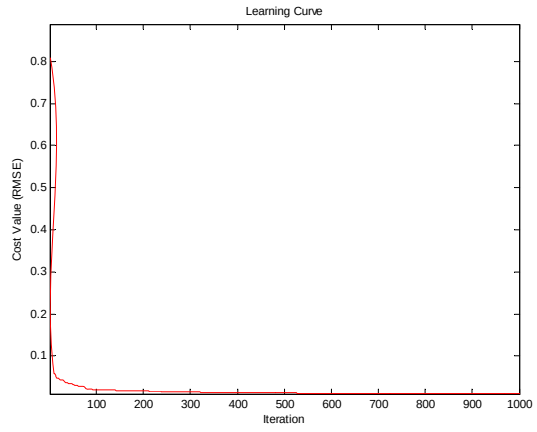

Fig. 7 Learning curve by the compared NFS-PSO approach.

TABLE III
PARAMETERS AFETER LEARNING BY THE HL-NFS

Premise-part $h_1(t)=x(t-18)$		$h_2(t)=x(t-12)$		$h_3(t)=x(t-6)$		$h_4(t)=x(t)$	
	m	σ	*m*	σ	*m*	σ	*m*	σ
s_1	1.6587	0.3486	0.5931	0.1981	0.9384	0.6394	0.9691	0.1480
s_2	0.4829	0.2694	0.9736	0.7136	0.6528	0.2360	1.3003	0.1509

Consequent-part = $a_0+a_1h_1(t)+a_2h_2(t)+a_3h_3(t)+a_4h_4(t)$ Parameters	a_0	a_1	a_2	a_3	a_4
Rule 1	4.1048	-10.4386	14.9053	-3.4357	-1.7852
Rule 2	-18.8653	1.1165	-18.6503	44.9217	-13.1210
Rule 3	-2.5798	8.0324	-12.1059	4.7486	2.8750
Rule 4	27.0955	4.4071	2.4483	-39.8492	8.0871
Rule 5	0.3566	0.3959	-0.2958	-0.5022	0.8181
Rule 6	-2.8194	0.6550	1.0849	-1.3495	3.1039
Rule 7	2.7680	-1.2446	-1.1585	1.5481	-1.6064
Rule 8	25.9175	-0.7634	-31.8984	18.0425	-11.6680
Rule 9	0.7910	2.3342	-0.8341	0.1243	0.0521
Rule 10	3.2043	-2.0677	-0.7358	-3.9836	2.8566
Rule 11	-3.1840	-0.8984	3.2290	-1.9703	3.5241
Rule 12	-21.0412	-13.9039	17.7967	-28.8458	37.0513
Rule 13	-1.2293	-0.2933	-0.0885	2.4198	-0.0846
Rule 14	2.8909	-1.2871	-1.6488	-2.5176	3.2357
Rule 15	3.6915	-0.4888	0.6932	-1.2407	-1.0618
Rule 16	15.0918	20.4092	-19.9707	26.8166	-31.8162

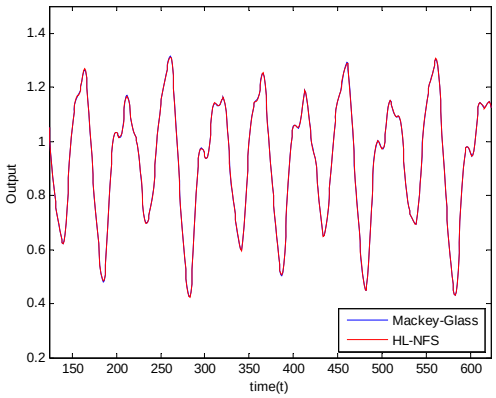

Fig. 8 Training results by the proposed HL-NFS approach. Training data are sampled from the Mackey-Glass chaos time series from *t*=124 to 623.

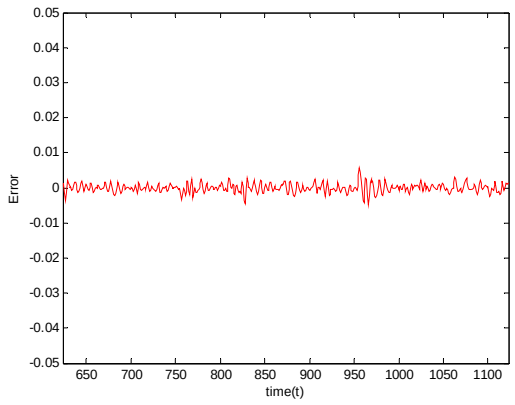

Fig. 9 Prediction error by the proposed HL-NFS approach in training phase.

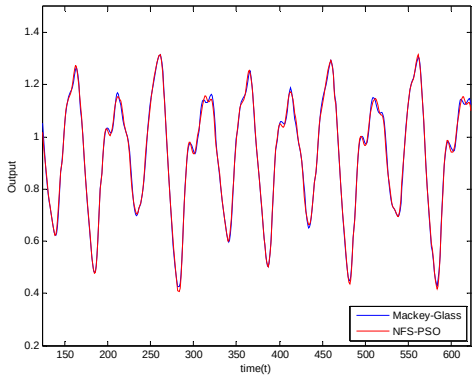

Fig. 10 Training results by the NFS-PSO approach. Training data are sampled from the Mackey-Glass chaos time series from *t*=124 to 623.

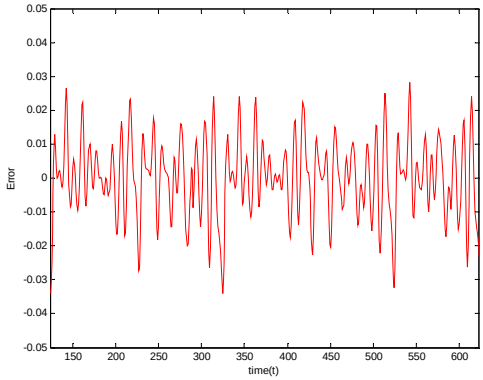

Fig. 11 Prediction error by the NFS-PSO approach in training phase.

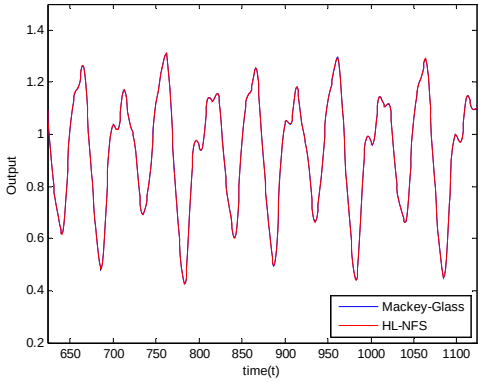

Fig. 12 Testing results by the proposed HL-NFS approach. Training data are sampled from the Mackey-Glass chaos time series from *t*=624 to 1123.

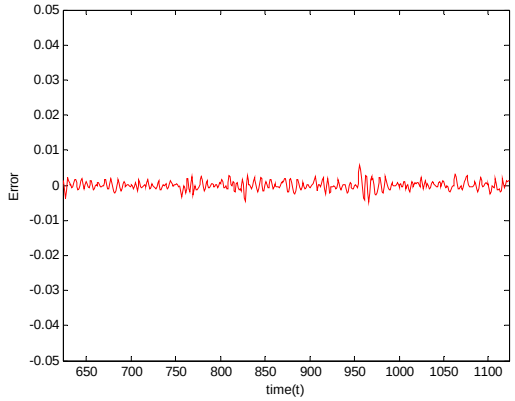

Fig. 13 Prediction error by the proposed HL-NFS approach in testing phase.

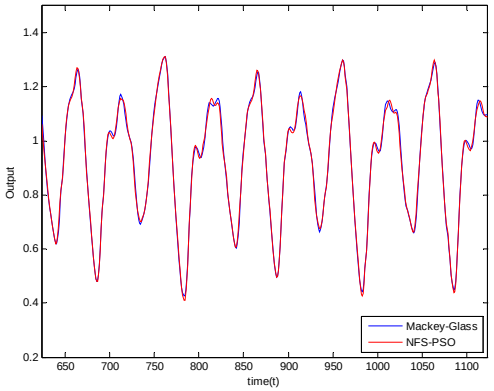

Fig. 14 Testing results by the NFS-PSO approach. Training data are sampled from the Mackey-Glass chaos time series from t=624 to 1123.

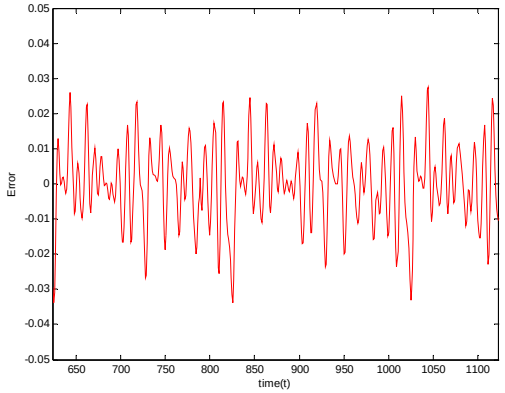

Fig. 15 Prediction error by the NFS-PSO approach in testing phase.

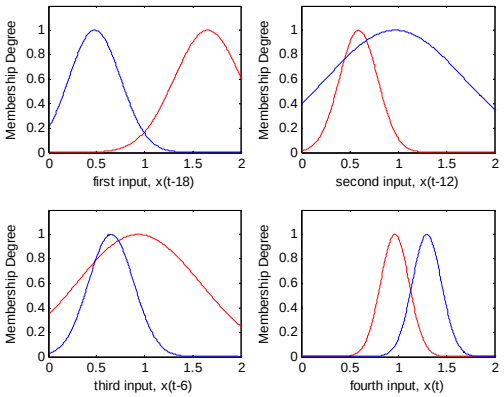

Fig. 16 Fuzzy sets of the proposed HL-NFS after training.

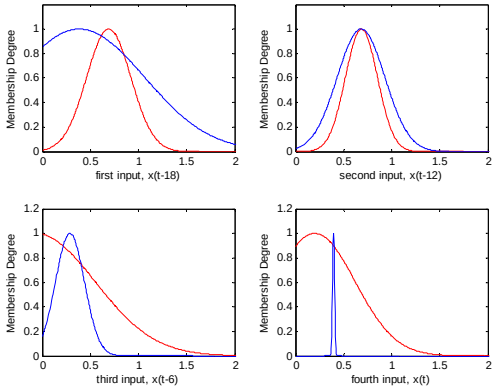

Fig. 17 Fuzzy sets of the NFS-PSO after training.

TABLE IV
PERFORMANCE COMPARISON

Method	**RMSE (training)**	**RMSE (testing)**	**Rules**
Chen et al. [10]	0.0158	0.0163	13
Cho and Wang [11] (Table 1)	0.0096	0.0114	23
Cho and Wang [11] (Table 2)	0.0107	0.0128	21
Cho and Wang [11] (Table 3)	0.0119	0.0131	13
Jang [4]	0.0016	0.0015	16
Nauck and Kruse [12]	0.1070	0.1080	26
Paul and Kumar [13]	0.0053	0.0055	9
Paul and Kumar [13]	0.0056	0.0057	10
WNN + gradient [14]	0.0067	0.0071	N/A
WNN + hybrid [14]	0.0056	0.0059	N/A
LLWNN + gradient [14]	0.0038	0.0041	N/A
LLWNN + hybrid [14]	0.0033	0.0036	N/A
HyFIS [15]	0.0021	0.0021	16
FNT model (case 1) [16]	0.0069	0.0071	N/A
NFS-PSO	0.0113	0.0113	16
HL-NFS (proposed approach)	0.0014	0.0013	16

V. Discussion and Conclusion

The proposed hybrid learning neuro-fuzzy system (NFS) approach to the problem of time series forecasting has been presented in the paper. The NFS is used as a predictor for time series forecasting. The new hybrid-learning method using both the well-known PSO and the RLSE has been applied for the adaption of the NFS in the proposed approach. The PSO is used to update the premise parameters and the RLSE is for the consequent parameters. The two methods are incorporated in hybrid way for the learning of the HL-NFS. The popularly famous

Mackey-Glass chaos time series is used to test the proposed approach. The proposed approach is then compared to several other approaches for performance comparison.

To illustrate and contrast the forecasting performance by the proposed approach, the NFS is first trained using the PSO alone (denoted as NFS-PSO approach) to adapt the system parameters of the neuro-fuzzy predictor. And then, the NFS is trained using the proposed PSO-RLSE hybrid learning method (denoted as HL-NFS approach). The forecasting performances for the two learning approaches are shown in Table IV. In the compared NFS-PSO approach, the performance shows that it is kind of hard for the PSO to find the optimal solution. The performance is not good enough with only RMSE=0.0113. This happens because there are too many parameters for the PSO to find the optimal solution. In this case, it is more likely for the PSO to fall into a local minimum during the learning process of the NFS. In contrast, the proposed HL-NFS approach can find the optimum solution easily and rapidly. The system parameters are divided into two subsets, which are called the premise subset and the consequent subset. The PSO is responsible for the premise subset of parameters whose amount is much reduced. The RLSE is used to update the consequent subset of parameters. Thus, it is obvious that the proposed HL-NFS approach can reach to the optimal solution for the NFS in fast way. With the learning curves in Figs. 6 and 7, the learning convergence of the proposed HL-NFS approach is much faster than the NFS-PSO approach. The experimental results by the HL-NFS approach are shown in Figs. 8, 9, 12 and 13, in which the Mackey-Glass time series is almost coincident with the predicted curves in the phases of training and testing. The performances in RMSE for training and testing are 0.0014 and 0.0013, respectively.

The proposed HL-NFS approach has shown excellent performance in prediction of the Mackey-Glass chaos time series. As shown in Table IV, the prediction performance by the proposed HL-NFS for the Mackey-Glass chaos time series is superior to other compared approaches.

ACKNOWLEDGMENT

This research work is supported by the National Science Council, Taiwan, ROC, under the Grant contract no. NSC98-2221-E-008-086.

REFERENCES

[1] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimization method for constrained optimization problems," *Intelligent Technologies–Theory and Application: New Trends in Intelligent Technologies*, pp. 214-220, 2002.

[2] K. E. Parsopoulos and M. N. Vrahatis, "Recent approaches to global optimization problems through particle swarm optimization," *Natural Computing*, vol. 1, pp. 235-306, 2002.

[3] Y. Shi, R. C. Eberhart, E. Center, and I. N. Carmel, "Empirical study of particle swarm optimization," *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 3 pp.1945-1950,1999.

[4] J. Kennedy and R. Eberhart, "Particle swarm optimization," *IEEE International Conference on Neuro Network, 1995,* vol. 4, pp. 1942-1948, 1995.

[5] J. S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE transactions on systems, man, and cybernetics*, vol. 23, pp.665-685, 1993.

[6] J. S. R. Jang, C. T. Sun, E. Mizutani, "Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence," *Prentice Hall*, 1997.

[7] M. Sugeno and G. T. Kang, "Structure identification of fuzzy model," *Fuzzy sets and systems*, vol. 28, pp. 15-33, 1988.

[8] I. Sugiarto and S. Natarajan, "Parameter estimation using least square method for MIMO Takagi-Sugeno neuro-fuzzy in time series forecasting," *Jurnal Teknik Elektro*, pp. 82-87, vol. 7, 2008.

[9] T. A. Jilani, S. M. A. Burney, and C. Ardil, "Fuzzy metric approach for fuzzy time series forecasting based on frequency density based partitioning," *International Journal of Computational Intelligence*, vol.4, pp. 112-117, 2007.

[10] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Transactions on neural networks*, vol. 2, pp. 302-309, 1991.

[11] 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.

[12] D. Nauck and R. Kruse, "Neuro-fuzzy systems for function approximation," *Fuzzy Sets and Systems*, vol. 101, pp. 261-272, 1999.

[13] S. Paul and S. Kumar, "Subsethood-product fuzzy neural inference system (SuPFuNIS)," *IEEE Transactions on Neural Networks*, vol. 13, pp. 578-599, 2002.

[14] Y. Chen, B. Yang, and J. Dong, "Time-series prediction using a local linear wavelet neural network," *Neurocomputing*, vol. 69, pp. 449-465, 2006.

[15] 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.

[16] Y. Chen, B. Yang, J. Dong, and A. Abraham, "Time-series forecasting using flexible neural tree model," *Information sciences*, vol. 174, pp. 219-235, 2005.